UNIVERSITÀ DEGLI STUDI
DI GENOVA

Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi
Corso di Laurea Magistrale
in INGEGNERIA INFORMATICA

Tesi di Laurea

TECNICHE DI INTELLIGENZA ARTIFICIALE APPLICATE ALL' ANALISI DEL MOVIMENTO E ALLA PIANIFICAZIONE PER IL SUPPORTO AD ATTIVITÀ DI FITNESS

Relatori
Prof. Gualtiero Volpe
Prof. Marco Maratea

Laureandi
Gabriele Guo
Luca Rabagliati

Anno accademico 2021/2022

Indice

Introduzione **4**

Obiettivi **6**

Stato dell'arte **7**

Metodologia **9**
- Movement analysis 10
  - Motion Capture 10
  - Markerless MoCap 11
  - Human Pose Estimation 12
    - Tipi di modelli 13
    - Frameworks 14
  - Scelta framework MediaPipe 16
    - Modelli esistenti 16
    - Architettura del framework 18
  - Produzione dataset 20
  - Approcci di Learning 21
    - Heuristics 21
    - Shallow learning 21
    - LSTM 22
- Planning 24
  - Answer set programming 24
    - Sintassi 25
    - Metodologia di risoluzione 27
  - Definizione del problema 28

Architettura **30**
- Integrazione tra Movement analysis e Planning 32

Implementazione **34**
- Movement analysis 34
  - Gestione del dataset 34
  - Feature engineering 37
  - Valutazione performance dei modelli 42
    - Metriche per la valutazione 44
  - Descrizione moduli e training 46
  - Live Webcam Testing 47
- Planning 48
  - Problem instance 48
  - Problem encoding 50
  - Interpretazione Output 55

Risultati **56**
Movement analysis 56
Performance modelli 56
Demo 59
Planning 64
Grafico riassuntivo performance 73
Conclusione **74**
Future works 74
Ringraziamenti 75
Luca. 75
Gabriele. 75
References **76**

Introduzione

Il mondo dell'intelligenza artificiale e del fitness sono in continua espansione, ma non è solito vederli uniti in un unico sistema.
Questa tesi, originariamente nata agli inizi del periodo della pandemia, mira a risolvere il problema delle palestre chiuse con l'idea di avere una sorta di personal trainer virtuale, che tenga in forma l'utente insegnando ed eventualmente correggendo i movimenti dei vari esercizi di fitness a corpo libero, inoltre previene una serie di infortuni comuni che possono portare a lievi o anche gravi problemi fisici.
Alcuni esempi potrebbero essere iniziare l'allenamento senza un corretto riscaldamento, non avere una corretta forma dell'esercizio, non riposare abbastanza tra un allenamento all'altro oppure eseguire un determinato esercizio che è troppo pesante per il livello della persona. [1]
Il tutto utilizzando un dispositivo portatile con videocamera, tramite tecniche di machine learning, computer vision e analisi di movimento.

Il fitness è un campo molto importante per la vita di ogni individuo, per mantenere uno stile di vita sano e longevo che porta molteplici benefici, come il rafforzamento scheletrico e muscolare, il miglioramento dell'umore, la prevenzione di malattie croniche come il diabete e malattie cardiache. [2,3]

Come caso di studio per il nostro progetto, abbiamo preso in considerazione la disciplina chiamata *calisthenics*, che consiste principalmente in allenamenti a corpo libero utilizzando strumenti come sbarre per trazioni e parallele al fine di migliorare la forza, la coordinazione e il controllo del proprio corpo.

Data la situazione pandemica del 2020 e con l'avvento del lockdown, molte persone si sono trovate costrette a rimanere in casa senza un personal trainer e hanno deciso di continuare o cominciare ad allenarsi.
A questo punto veniva a mancare la figura del personal trainer, importante soprattutto per gli amatori e principianti che si sono appena avvicinati alla disciplina.
I suoi principali compiti sono due:

1. supportare l'utente durante l'esecuzione di un determinato movimento, correggere eventuali errori e dettagliare lo svolgimento dell'esercizio.
2. consigliare esercizi propedeutici per il raggiungimento degli obiettivi stabiliti dall'utente, quali gruppo muscolare da allenare, la categoria desiderata (skills, endurance o weighted). Il tutto compatibilmente alle disponibilità e al livello attuale della persona.

Lo sviluppo di un sistema per il personal training richiede perciò di integrare due componenti, la prima che si occupi di analizzare il movimento dell'utente e la seconda che si occupi di pianificarne l'attività tenendo in considerazione le esigenze e le abilità dell'utente.

L’importanza dell’analisi della qualità del movimento è grande, perché può essere utilizzata nel contesto di differenti attività, come nello sport (arti marziali, ginnastica, pattinaggio) e nella danza (classica, contemporanea). Inoltre nelle competizioni sportive, può dare supporto ai giudici per la valutazione delle performance. [4,26]

Per quanto riguarda la pianificazione, il planner è in grado di fornire un piano specifico per la persona che approccia per la prima volta questa disciplina e che non conosce in dettaglio alcuni aspetti degli esercizi come le propedeuticità e le regole.

Il documento è così strutturato, nella prima parte vengono stabiliti gli obiettivi che si vogliono raggiungere e come raggiungerli. Successivamente viene discussa l’analisi dello stato dell’arte in cui valutiamo le applicazioni più diffuse e già disponibili per gli utente.
A seguire la sezione della metodologia dove presentiamo come abbiamo organizzato il lavoro e le scelte relative alle tecnologie utilizzate. Dopodiché illustreremo l’architettura del sistema, approfondendo i vari blocchi. Nella sezione successiva verrà discussa l’implementazione in cui sono descritte le fasi più tecniche, come lo sviluppo del codice.
In seguito verranno commentati i risultati e infine presentate le nostre conclusioni.

Obiettivi

Lo scopo del presente elaborato è quello di progettare un sistema prototipale che si dimostri essere un punto di partenza per future applicazioni che utilizzino l'intelligenza artificiale nel campo del fitness.

Il prototipo che verrà realizzato dovrà simulare lo svolgimento dei compiti di un personal trainer, più in dettaglio valutare la qualità del movimento di uno specifico esercizio eseguito dall'utente e in base a ciò pianificare una scheda di allenamento adatta alle necessità della persona.

Per spiegare in dettaglio gli obiettivi del nostro progetto, prendiamo in esame una tipica user story dove l'utente specifica i dati relativi alle sue esigenze, obiettivi e disponibilità, così da poter dare la possibilità al nostro sistema di elaborare un piano di allenamento personalizzato.

Successivamente l'utente inizia la sessione di allenamento giornaliero dove, tramite un'interfaccia grafica, vengono mostrati i disclaimers relativi alla sicurezza e a prestare attenzione all'ambiente circostante, dopodiché viene effettuata la fase di riscaldamento dove per ogni esercizio presentato, sarà mostrata una descrizione dello svolgimento del movimento tramite un breve video saltabile se l'utente è già a conoscenza della corretta esecuzione. A questo punto l'allenamento inizia con la presentazione, tramite un video esplicativo, dell'esercizio da eseguire. Successivamente l'utente viene guidato con indicazioni presenti su schermo che mostrano i passaggi da eseguire.

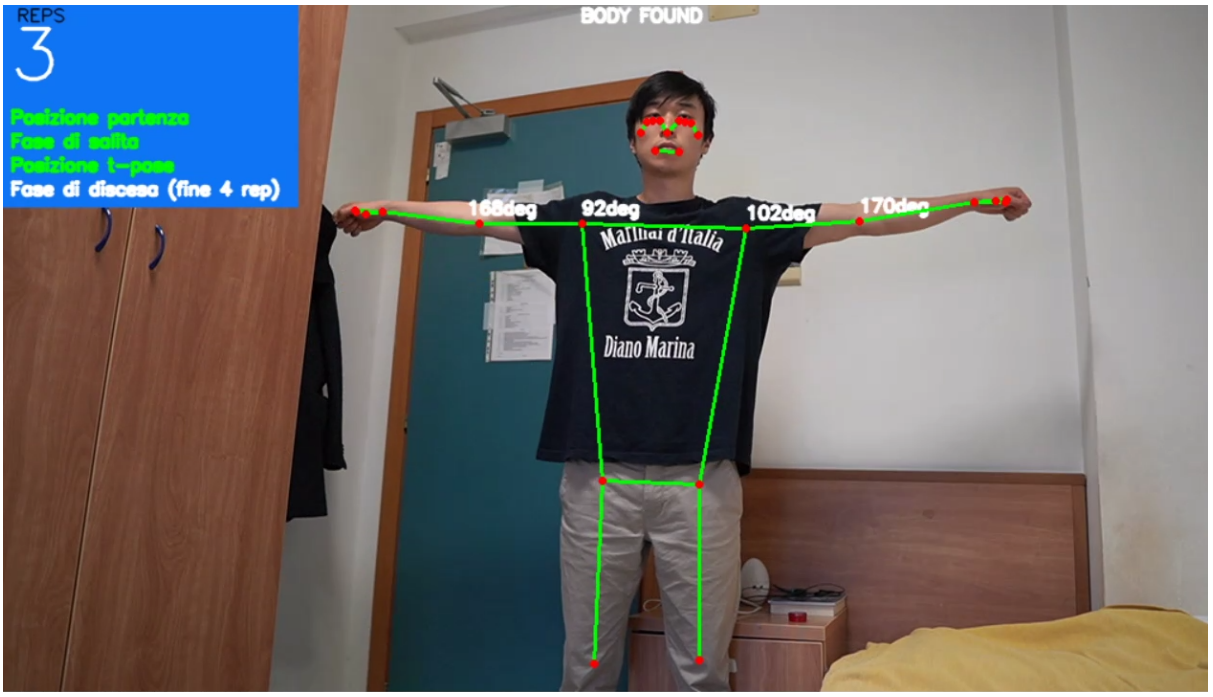


Fig. 1 - Presentazione del prototipo

Una volta concluso l'esercizio, si passa al successivo e, una volta concluso l'allenamento verrà presentato un report della performance dei singoli esercizi.

Stato dell'arte

Nel mercato vi sono molteplici applicazioni di fitness che tengono traccia degli allenamenti e mirano al benessere della persona, ma nessuna ha la funzionalità di analizzare effettivamente la qualità del movimento eseguito dalla persona e di offrire un feedback all'utente.
Per esempio tra le applicazioni di fitness più diffuse si possono citare: [5,6]

- **Nike Training Club**[1], offre un range di tipi di esercizi che comprendono allenamenti high intensity interval training (HIIT), yoga, cardio, core e strength. Si possono selezionare gli allenamenti per gruppo muscolare, attrezzature e durata in base alle preferenze. L'app è gratuita. Uno degli aspetti negativi più importanti è che le schede di allenamento non sono efficacemente tarate al livello dell'utente e quindi potrebbero presentarsi esercizi troppo complicati per un principiante.
- **Peloton**[2], app principalmente per classi di cycling, ma offre anche allenamenti di strength e HIIT. Si possono effettuare allenamenti in tempo reale con istruttori oppure selezionare corsi on-demand. Le note negative sono che per l'utilizzo dell'app è necessario un abbonamento mensile e alcuni corsi possono presentarsi troppo impegnativi per principianti.
- **Fit Body**[3], app fitness progettata per donne, caratterizzata da vari programmi di allenamento predefiniti da allenatori esperti e con tempi preimpostati. Gli aspetti negativi sono che l'app è a pagamento mensile e i programmi potrebbero non essere abbastanza motivanti.
- **Centr**[4], app ideata da Chris Hemsworth, presenta schede di allenamenti settimanali, vari metodi e stili di allenamenti. L'app è scaricabile gratuitamente ma bisogna sottoscrivere un abbonamento per usufruire a pieno delle sue funzionalità.
- **Le Sweat TV**[5], app di fitness che offre allenamenti in base alla parte del corpo che si intende allenare, agli strumenti posseduti e durata desiderata (massimo 30 minuti). È necessario una sottoscrizione ad un abbonamento per l'utilizzo e presenta poche varietà di esercizi.
- **Jefit Workout Planner Gym Log**[6], designata per weightlifters, quindi si concentra esclusivamente su allenamenti di forza che richiedono pesi e bilancieri. I dati del post-workout sono calcolati in base alle informazioni immesse in input dall'utente.
- **Apple Fitness+**[7], presenta allenamenti in base a diverse categorie tra cui fascia d'età, allenamenti per principianti e specifici per sport selezionati. Integrazione con Apple Watch per il monitoraggio dell'ossigenazione sanguigna e frequenza cardiaca, dati

[1] https://www.nike.com/it/ntc-app
[2] https://www.onepeloton.com/app
[3] https://fitbodyapp.com/
[4] https://centr.com/
[5] https://tv.le-sweat.com/
[6] https://www.jefit.com/
[7] https://www.apple.com/it/apple-fitness-plus/

utili per eventuali analisi sulle prestazioni della persona. L'utilizzo dell'app ha bisogno di una sottoscrizione mensile.

- **Thenx**[8], app di fitness incentrata sul calisthenics, propone vari tipi di allenamenti basati al raggiungimento di certi obiettivi come lo sviluppo di un determinato gruppo muscolare, oppure progressioni mirate all'ottenimento o miglioramento di una skill, con tanti video tutorial di istruttori professionisti che forniscono dettagli esercizio per esercizio. L'app è gratuita con molte limitazioni, sottoscrivendo l'abbonamento si possono usufruire di tutte le sue funzionalità.
- **Heria Pro**[9], app di fitness dove è centrale la personalizzazione degli allenamenti. Tramite le preferenze e lo stile di allenamento dell'utente, l'applicazione crea programmi di allenamento adatti alla situazione. Bisogna sottoscrivere un abbonamento per accedere a tutte le funzionalità dell'app.

Notiamo come molte di queste applicazioni analizzate abbiano più o meno le stesse funzionalità e che comunque aiutano l'utente ad allenarsi. Un punto fortemente negativo è il fatto di avere schede di allenamento precompilate senza effettuare alcun adattamento per l'utente, che nel nostro caso avviene, in quanto il sistema si adatta alle esigenze della persona dopo una valutazione del suo livello di abilità. Per quanto riguarda la scelta del gruppo muscolare e le preferenze per gli strumenti necessari allo svolgimento di un determinato esercizio, sono presenti in alcune applicazioni sopra citate e vengono mantenute anche nel nostro progetto. Un altro punto da notare è la durata degli allenamenti. Molte applicazioni sopra citate propongo allenamenti con durata preimpostata, mentre la nostra applicazione è in grado di generare allenamenti di durata desiderata dall'utente. Una delle funzioni che manca totalmente alle applicazioni prese in considerazione sono l'analisi del movimento, la valutazione e la guida durante l'esecuzione.

Questa tesi è mirata a dimostrare che queste funzionalità sono possibili e realizzabili, valutando e dettagliando ogni aspetto.

[8] https://thenx.com/
[9] https://heriapro.com/

Metodologia

Nel 2020 l'inizio della pandemia ha costretto molti centri fitness a chiudere, impedendo alle persone di potersi allenare. Questo avvenimento è stato per noi una forte sorgente di ispirazione, per identificare e formulare il problema che questa tesi affronta.

Il primo passo è stato l'analisi dello stato dell'arte, in cui abbiamo valutato le varie soluzioni offerte dalle varie aziende, concentrandoci sulle applicazioni mobile designate per l'utente finale.

Quasi la totalità delle applicazioni offerte propongono schede di allenamento in base al livello della persona, tracking dei progressi, timer e strumenti simili al supporto degli allenamenti. Quindi la nostra idea è stata di portarlo ad uno step successivo, cioè simulare l'azione di un allenatore che osserva la persona mentre esegue il movimento dell'esercizio e corregge eventuali esecuzioni errate, dopodiché date le prestazioni, generare una scheda di allenamento settimanale in base al livello, la disponibilità e le preferenze dell'utente.

Come realizzare tutto ciò? Dopo aver definito il problema e analizzato lo stato dell'arte, siamo partiti cercando varie metodologie di body tracking e valutando gli aspetti positivi e negativi che esse offrono.
Le principali sono il motion capture e il motion capture senza markers (markerless).
Effettuate le varie analisi e scelte di quale metodologia usare, si passa a valutare quali strumenti o framework siano più adatti al nostro obiettivo.

Quindi si procede con le varie implementazioni arrivando alla produzione dei risultati.

Si effettua la stessa metodologia per la parte di planning e infine vengono integrati questi due sistemi.

Movement analysis

Le due principali opzioni per acquisire dati [7] utili per l'analisi della qualità del movimento sono:

- Motion Capture (MoCap)
- Markerless Motion Capture

Motion Capture

Il motion capture è un metodo che mediante l'equipaggiamento di un set di sensori e videocamere di precisione, permette di ottenere un modello di movimento del corpo umano. In fase di produzione, una persona indossa una particolare tuta dotata di markers. I markers sono riferimenti posizionati su punti di rilievo anatomici come giunture, mani, piedi, muscoli espressivi del viso. Le informazioni prodotte dal movimento verranno poi utilizzate per svariate applicazioni, dove la più popolare è nel campo cinematografico e videoludico, in cui si implementano visual effects (VFX) ed animazioni fluide e realistiche in 3d.

Fig. 2 - Esempio sessione di MoCap

Tuttavia il costo delle attrezzature, la necessità di uno spazio dedicato e di una calibrazione che richiede tecnici specializzati per effettuarne il setup (preparazione dello studio, installazione dei sistemi, cioè il posizionamento delle videocamere per ricoprire l'area di lavoro e minimizzare le occlusioni) e di conseguenza rende impossibile l'utilizzo di questo approccio per una applicazione che dovrebbe essere usata da un vasto pubblico. Un altro problema è dovuto all'utilizzo di tute con *marker* che richiedono un certo tempo per la preparazione e può risultare scomoda dal punto di vista dell'utente.

Grazie alla costante innovazione tecnologica che sta semplificando l'elaborazione dei modelli di motion capture, sono state sviluppate applicazioni capaci di catturare un modello di movimento senza la necessità del costoso equipaggiamento al prezzo di una rilevazione meno accurata. Questa tecnica viene denominata *markerless mocap*.

Markerless MoCap

Il *markerless mocap* è una tecnica di motion capture senza l'utilizzo di *marker*.
Questo comporta una libertà maggiore del luogo di ripresa senza dover effettuare lunghi setup e un costo delle attrezzature molto contenuto. Questo però implica una rilevazione meno accurata.

Sono presenti in commercio alcuni tipi di dispositivi che sono basati su tecnologie laser come il *lidar* e luce strutturata come il *Kinect* di Microsoft, *RealSense* di Intel.

Fig. 3 - Prodotto Kinect e il suo funzionamento

Questi prodotti funzionano tramite sensori, laser e videocamere per registrare i movimenti della persona e trasformarli in dati che possono essere analizzati successivamente [28].

Alla fine il progetto si focalizza a minimizzare il costo all'utente finale, quindi la soluzione più adatta è utilizzare le tecniche di human pose estimation che è un modo di realizzare markerless mocap.

Human Pose Estimation

La human pose estimation (HPE) è una tecnica di computer vision che consiste nell'identificare e classificare la posa della persona in una specifica scena tracciando i punti importanti, denominati *keypoints*, del corpo umano, come citato precedentemente nel mocap, quindi giunture, mani, piedi, punti facciali, ecc..
In generale il modello è usato per rappresentare ed inferire pose in 2D e 3D, formando una rappresentazione di uno "scheletro" della persona e successivamente vengono elaborate le informazioni ottenute per la realizzazione di applicazioni specifiche.
Le tecniche di pose estimation possono essere usate in congiunzione con altre tecnologie di computer vision per fitness e riabilitazione, applicazioni di realtà aumentata, di videosorveglianza e linguaggio dei gesti.

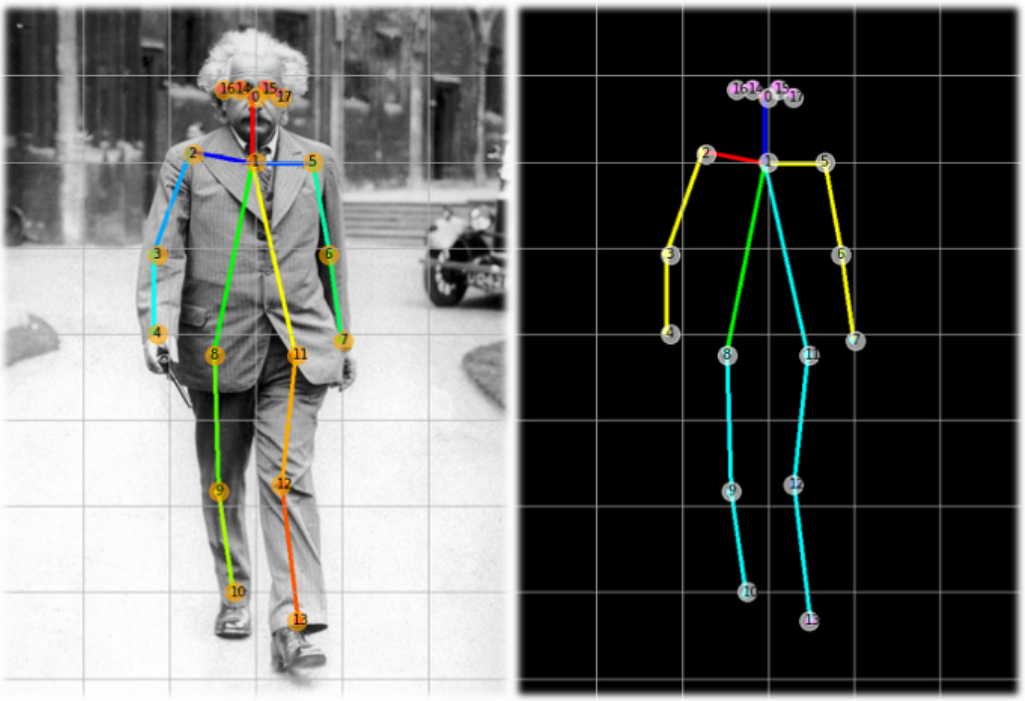


Fig. 4 - Esempio di human pose estimation

Per analizzare il movimento del corpo umano, occorre analizzare una sequenza di immagini, anziché una sola foto, così da capire in che direzione vanno i *keypoints*.
Una volta dato in input l'immagine (frame), attraverso un determinato framework di HPE, si ottengono una lista di coordinate per l'analisi, x, y e z, dove quest'ultima coordinata è un valore stimato siccome non si ha a disposizione un'altra immagine dello stesso momento da un'angolazione differente. Un esempio di riferimento è la nostra vista grazie a cui è possibile percepire la profondità attraverso l'uso degli occhi.

Tipi di modelli

Vi sono tre tipi di modelli di human pose estimation usati per rappresentare il corpo umano in 2D e 3D.

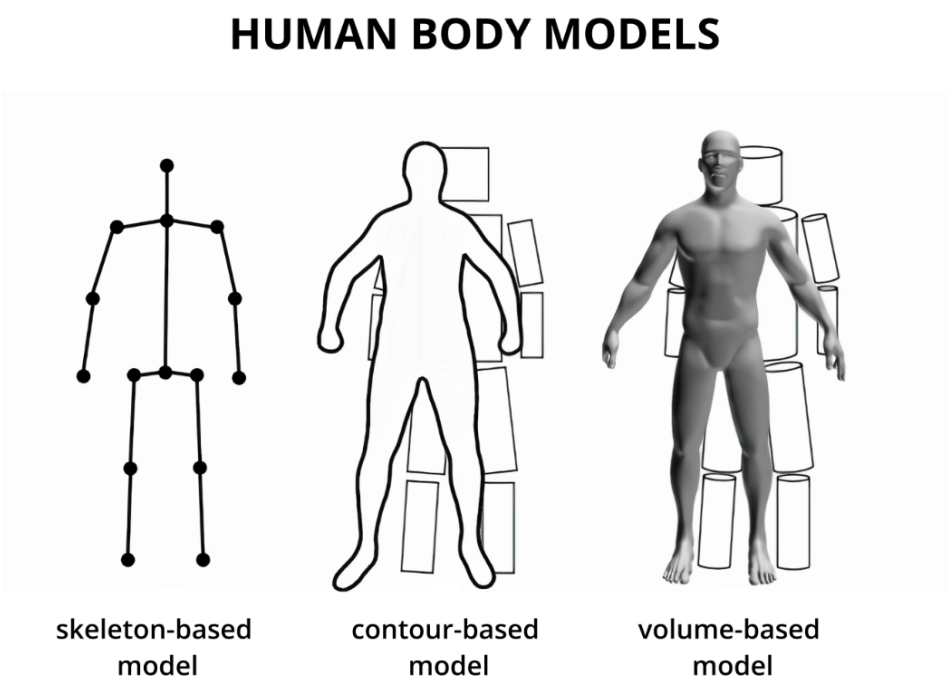


Fig. 5 - Tipi di modelli HPE

a. Skeleton-based model, chiamato anche “kinematic model”, come si deduce dal nome, presenta punti sulle giunture ed è frequentemente utilizzato per relazionare parti differenti del corpo umano
b. Contour-based model, chiamato anche “planar model”, è usato per le pose in 2D e consiste nel tracciare il contorno della persona, così definendo la forma del corpo umano. Una sua applicazione molto nota è la segmentazione tra il soggetto e il background, molto usata nelle applicazioni di meeting come Teams e Zoom.
c. Volume-based model, chiamato anche “volumetric model”, è usato per le pose in 3D e consiste nel ricostruire il modello del corpo umano nello spazio, molto dispendioso dal punto di vista computazionale e non adatto al nostro caso.

I metodi principali per ricavare la human pose estimation sono principalmente due, l’approccio top-down e bottom-up.

- Top-down, rileva la persona per intero e successivamente cerca le singole parti del corpo
- Bottom-up, cerca prima ogni giuntura del corpo e poi li organizza fino a formare la posa

Frameworks

Vi sono differenti framework per l'human pose estimation e in questa sezione presentiamo quelli più noti e ad uso pubblico.

OpenPose

OpenPose è un framework open-source di human pose estimation che funziona in real-time e rileva keypoints per il corpo, viso, mani e piedi. Può rilevare molteplici persone e fino a 135 keypoints all'interno di un singolo frame. Questo framework adotta l'approccio bottom-up per la costruzione dello scheletro umano. [8]
Esso supporta architetture hardware differenti come CUDA GPU, OpenCL GPU e sistemi CPU. Il framework è ampiamente usato per 2D pose estimation, 3D pose estimation, 3D pose reconstruction e come plugin in Unity.
Dal punto di vista computazionale è molto dispendioso per cui richiede componenti hardware abbastanza performanti per raggiungere i 24-30 fps necessari per fornire feedback in tempo reale.

TF Pose Detection

Il pacchetto Pose Detection di TensorFlow è un framework open-source di real-time pose detection basato su un'architettura costruita con tensorflow.js.
È progettato per essere molto leggero e quindi per essere utilizzato su dispositivi come smartphones o direttamente da un browser. Fornisce 3 tipi di modelli allo stato dell'arte. [9]

- **MoveNet**, tecnica che rileva 17 keypoints del corpo

- **MediaPipe BlazePose**, tecnica che rileva 33 keypoints, in aggiunta ai 17 di COCO (Common Objects in Context), fornisce keypoints aggiuntivi per il viso, mani e piedi. È la tecnica che adottiamo in questo progetto.

- **PoseNet**, simile a MoveNet

DensePose
È un framework open-source che può mappare tutti i pixels appartenenti alla persona 2D ad un modello 3D surface-based in real-time. Esso è implementato con Caffe2 (deep learning framework) e può essere utilizzata per problemi di pose estimation singole o multiple. [10]

AlphaPose

AlphaPose è un framework open-source di human pose estimation che rileva molteplici persone in real-time e adotta un approccio top-down. Una sua funzionalità particolare è il pose flow che confronta la posa della stessa persona su più frames. [11]

HRNet (High-Resolution Net)

HRNet è un'architettura usata per human pose estimation per rilevare i keypoints rispetto ad un oggetto o una persona nel frame dato.
La sua peculiarità è di mantenere le rappresentazioni ad alta risoluzione durante l'intero processo di rilevamento. [12]
Questa architettura HRNet viene applicata anche in altri campi come la semantic segmentation, face alignment, image classification e object detection.

Scelta framework MediaPipe

Dopo un'accurata analisi dei vari frameworks, si è deciso di utilizzare **MediaPipe BlazePose** in quanto presenta caratteristiche molto interessanti per il nostro caso come il supporto per dispositivi di bassa gamma e per le ottime prestazioni anche in ambienti con scarsa luminosità.
Ora andremo a dettagliare questo framework sotto ogni aspetto.

Modelli esistenti

MediaPipe è un progetto sviluppato da Google che offre soluzioni personalizzabili nell'ambito delle dirette e streaming media. Esso fornisce accesso ad una vasta varietà di modelli di Machine Learning progettati in modo da poter essere utilizzati su dispositivi mobili iOS e Android, oppure sistemi embedded come il Raspberry Pi e Jetson Nano, o semplicemente su un computer portatile.

Questi modelli includono:

- Modelli anatomici
    - Hand tracking
    - Pose tracking
    - Face detection
    - Holistic tracking (combinazione delle 3 precedenti)
    - Face mesh tracking
    - Iris tracking
    -
- Modelli di segmentazione
    - Selfie segmentation
    - Hair segmentation
    -
- Modelli per oggetti
    - 2D object detection/tracking
    - 3D object detection e pose estimation

A seguire viene rappresentata una tabella di compatibilità con i vari linguaggi di programmazione e target di sviluppo/piattaforma.

	Android	iOS	C++	Python	JS	Coral
Face Detection	✅	✅	✅	✅	✅	✅
Face Mesh	✅	✅	✅	✅	✅	
Iris	✅	✅	✅			
Hands	✅	✅	✅	✅	✅	
Pose	✅	✅	✅	✅	✅	
Holistic	✅	✅	✅	✅	✅	
Selfie Segmentation	✅	✅	✅	✅	✅	
Hair Segmentation	✅		✅			
Object Detection	✅	✅	✅			✅
Box Tracking	✅	✅	✅			
Instant Motion Tracking	✅					
Objectron	✅		✅	✅	✅	
KNIFT	✅					
AutoFlip			✅			
MediaSequence			✅			
YouTube 8M			✅			

Fig. 6 - Tabella di compatibilità con i vari linguaggi per ogni modello

Nel progetto adottiamo il modello di *Pose*. Ulteriori informazioni vengono dettagliate nella sezione di implementazione.
È stato utilizzato come linea guida un video tutorial di Nicholas Renotte. [13]

Architettura del framework

Il framework è sviluppato principalmente in linguaggio C++, utilizzando dipendenze come Tensorflow, OpenCV, OpenGL/Metal, Eigen, ed è composto da componenti API chiamati *Graphs* e *Calculators*.[10]

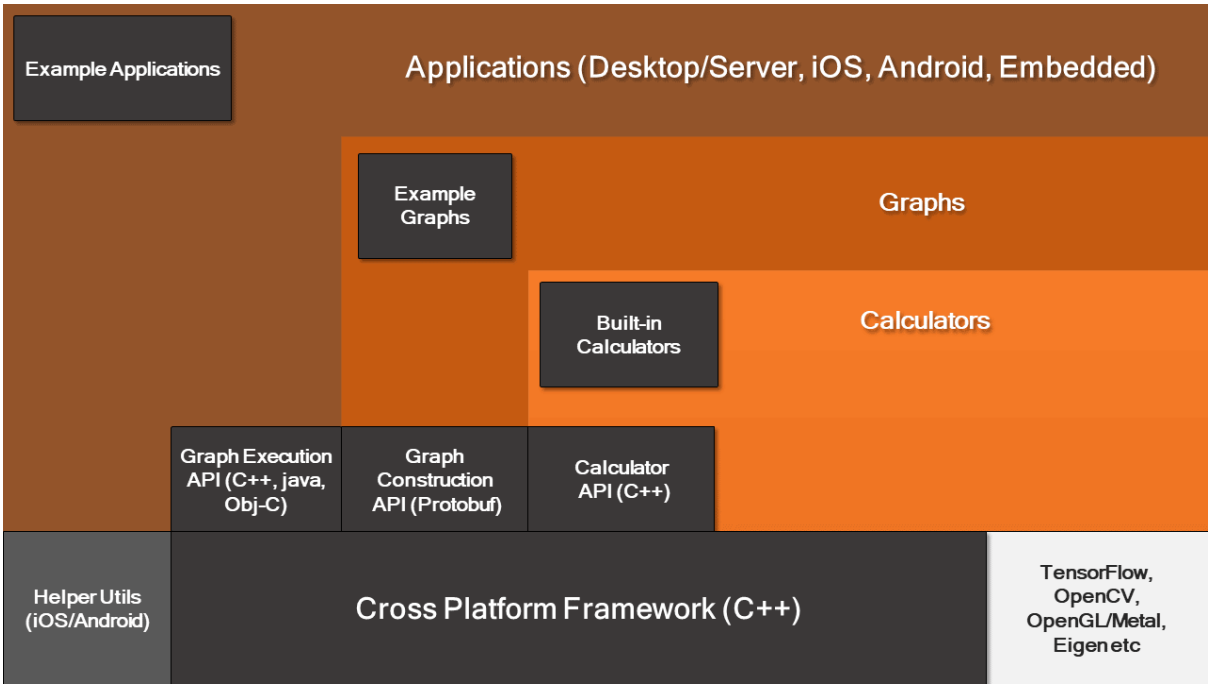


Fig. 7 - Architettura di MediaPipe

La pipeline che rileva la persona/oggetto è rappresentata per mezzo di un grafo.
Per esempio il grafo per il rilevamento della posizione delle mani si presenta nel seguente modo.

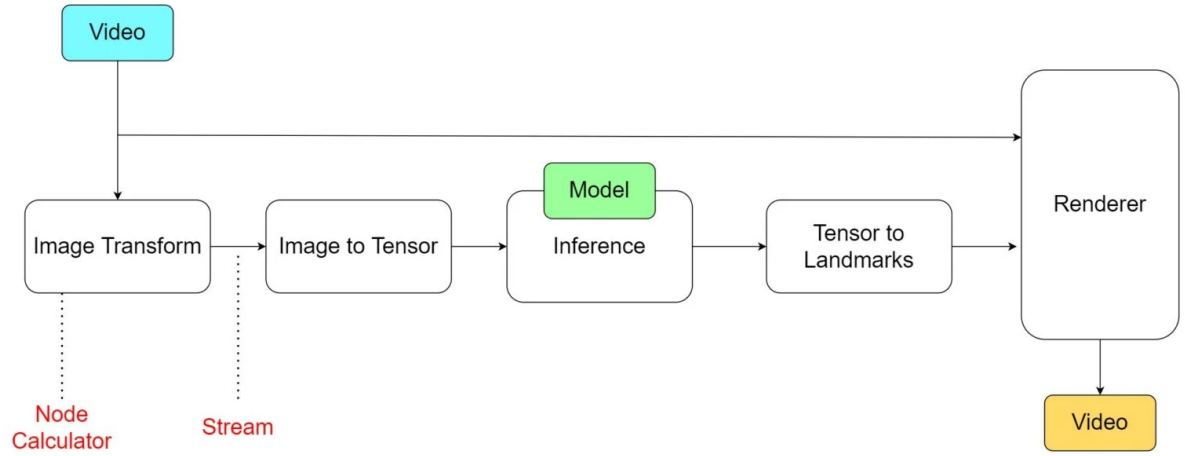


Fig. 8 - Pipeline per il modello Hands

[10] https://learnopencv.com/introduction-to-mediapipe/

Generalmente ogni grafo presenta nodi e archi. Nel sistema MediaPipe, i nodi sono chiamati *Calculators* e gli archi *Streams*. Ogni *Stream* contiene una sequenza di *Packets* che a sua volta contengono *timestamps*.
Nell'immagine soprastante i *Calculators* sono rappresentati dai blocchi rettangolari e gli *Streams* rappresentati come frecce.

Calculators
Sono componenti sviluppati in C++ che processano e risolvono specifici task. I *packets* di dati (che possono essere frames di video o segmenti audio), entrano ed escono attraverso i *ports* di un *calculator*. Quando esso viene inizializzato, vengono dichiarati i tipi di *packets* che andranno ad attraversare il *calculator* e ogni volta che avviamo il *graph*, MediaPipe implementa i metodi *Open, Process* e *Close* all'interno dei *calculators*.
Open inizializza il *calculator*, il metodo *Process* viene eseguito ogni volta che un *packet* entra nel componente e infine viene invocato *Close* quando non ci sono più *packets* da processare.
Per esempio nel *graph* precedentemente presentato, il *calculator* ImageTransform prende in input un frame del video e restituisce in output il frame trasformato. Stessa cosa per il *calculator ImageToTensor*, la sua funzione è di prendere un frame in input e di restituire in output un *tensor*.

Vi sono quattro categorie di *calculators* all'interno di MediaPipe.

- *Pre-processing calculators* sono una famiglia di *calculators* che processano immagini e media. Per esempio *ImageTransform* e *ImageToTensor* appartengono a questa categoria.
- *Inference calculators* permette l'integrazione con Tensorflow e Tensorflow Lite per l'inferenza
- *Post-processing calculators* effettua task di post-processing di machine learning come il rilevamento, segmentazione e classificazione. Per esempio *TensorToLandmark* è un post-processing calculator
- *Utility calculators* sono una famiglia di *calculators* che effettuano dei task finali come annotare il frame

Le API dei *calculators* permettono di creare calculators personalizzati e di adattarsi a seconda dell'esigenza.

Per velocizzare i vari processi, i *calculators* sono eseguiti in threads separati. Ulteriormente per ottimizzare le performance, molti *calculators* supportano l'accelerazione GPU.
Un altro aspetto fondamentale è la sincronizzazione quando si lavora con dati legati alle serie temporali. I *graph* assicurano il corretto data flow in accordo ai timestamps dei *packets*.
Inoltre, MediaPipe gestisce la sincronizzazione e l'interoperabilità con i *calculators CPU*.

Produzione dataset

Per la produzione del nostro dataset, inizialmente si è analizzato quali movimenti tenere in considerazione e tutte le problematiche che possono insorgere.

Per semplicità si considera l'esercizio *alzate laterali*, un movimento classico per l'allenamento delle spalle.

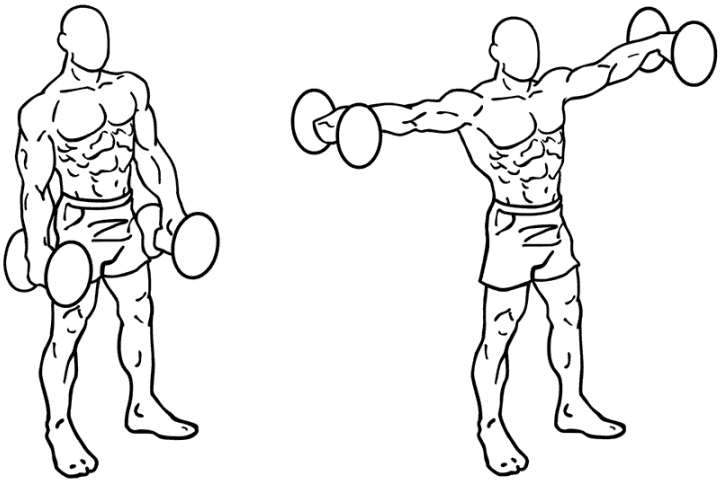


Fig. 9 - Esecuzione delle alzate laterali

Questo esercizio consiste nell'alzare le braccia lateralmente in modo sincrono fino a raggiungere un angolo di 90° rispetto al corpo, mantenendo il busto eretto, testa dritta con un buon sostegno lombare e scapole attivate, quindi ad arrivare ad assumere una posizione di "T" con le braccia, fermandosi più o meno in parallelo alle clavicole.

Il dataset per questo movimento è strutturato in quattro classi, di cui:

- 53 campioni per i quali il movimento non è corretto a causa delle braccia piegate
- 43 campioni per i quali il movimento non è corretto a causa di asimmetria nella postura delle braccia
- 46 campioni per i quali il movimento non è corretto a causa della mancanza del raggiungimento di 90 gradi delle braccia rispetto al corpo
- 46 campioni per i quali il movimento è stato eseguito correttamente

Ogni campione è un video della durata di 2-3 secondi.
I filmati sono stati registrati utilizzando una fotocamera mirrorless e una compact point-and-shoot in 1080p a 30fps, ma si possono utilizzare anche smartphone dotate di funzionalità per registrare video ad almeno 720p.

Approcci di Learning

Il problema da trattare nel nostro caso è un problema di classificazione, cioè di classificare la qualità del determinato movimento in base ai dati in input.
In questa sezione si discutono le possibili metodologie adottate per risolvere il problema.

Heuristics

Questo è un approccio molto basilare che consiste nel calcolare le features del movimento e stabilire intervalli di valori con cui classificare la qualità del movimento. In altre parole si definiscono gli stati delle posizioni del corpo.
Questa metodologia è stata utilizzata principalmente per comprendere le potenzialità che il framework offre e successivamente per implementare una linea guida di supporto in real-time per le varie fasi dell'esecuzione del movimento. Questo approccio è la base su cui ci appoggiamo per la produzione del vero modello di machine learning.

Shallow learning

Sono stati testati molteplici modelli di classificazione, tra cui ***DecisionTree***, ***RandomForest***, ***SupportVectorMachine***, ***GradientBoosting*** e ***ExtremeGradientBoosting***.
I risultati verranno dettagliati e discussi nei capitoli successivi.

Lo shallow learning si contrappone al deep learning, dove la macchina impara su più livelli, sempre dagli esempi ma in modo indiretto.

Nel deep learning la rete neurale (*neural network*) è composta da almeno due livelli intermedi nascosti (*hidden layers*). Quindi, l'output non è definito in modo diretto dall'input della rete perché l'ultimo livello prende in input l'output del livello precedente.

Invece nello shallow learning, la rete neurale è composta da un solo strato intermedio nascosto. Quindi, l'output della rete è definito direttamente dall'input della rete.
Il vantaggio dello shallow learning è soprattutto la minore complessità computazionale.
A parità di altri fattori, l'addestramento della macchina consuma meno risorse e il tempo di elaborazione è minore.
Lo svantaggio è la semplicità dell'apprendimento e la difficoltà ad apprendere concetti più astratti.
Nel deep learning il passaggio delle informazioni su più livelli consente di generalizzare l'analisi e permette un ragionamento artificiale più astratto.

LSTM

Long-short term memory, in acronimo LSTM, è una versione modificata di RNN (*recurrent neural network*) che risolve alcuni problemi che RNN presenta (*vanishing gradient problem*). LSTM è utilizzato per classificare, elaborare e predire le serie temporali, cioè è capace di elaborare una sequenza di informazioni (nel nostro caso una sequenza di frames), anziché il singolo frame.

Sappiamo che la metodologia delle reti neurali richiede un dataset molto grande per performare al meglio. Nel nostro caso abbiamo un dataset relativamente piccolo, quindi sapevamo a priori che le performance probabilmente sarebbero state insufficienti, ma si è voluto comunque provare la metodologia LSTM pur consapevoli delle limitazioni.

Dopo aver effettuato vari test con questa metodologia, abbiamo capito che nel nostro caso, dobbiamo valutare esercizi molto simili tra di loro e trovare piccole variazioni che determinano l'errore di esecuzione.

Come si vede dalla figura sottostante, questi test sono risultati molto scarsi.

Fig. 10 - LSTM andamento della fase di training utilizzando il dataset con 4 classi

Per confermare questa nostra ipotesi, abbiamo utilizzato lo stesso algoritmo per classificare **due esercizi completamente diversi** (alzateLaterali e push-ups) e come si vedono dagli screenshots sottostanti, abbiamo ottenuto ottimi risultati per lo split 10% test (circa 80% accuracy), mentre con split 5% abbiamo ottenuto un risultato **poco rappresentativo** in quanto la quantità di campioni per effettuare la fase di testing è piccolissima ottenendo 100% di accuracy.

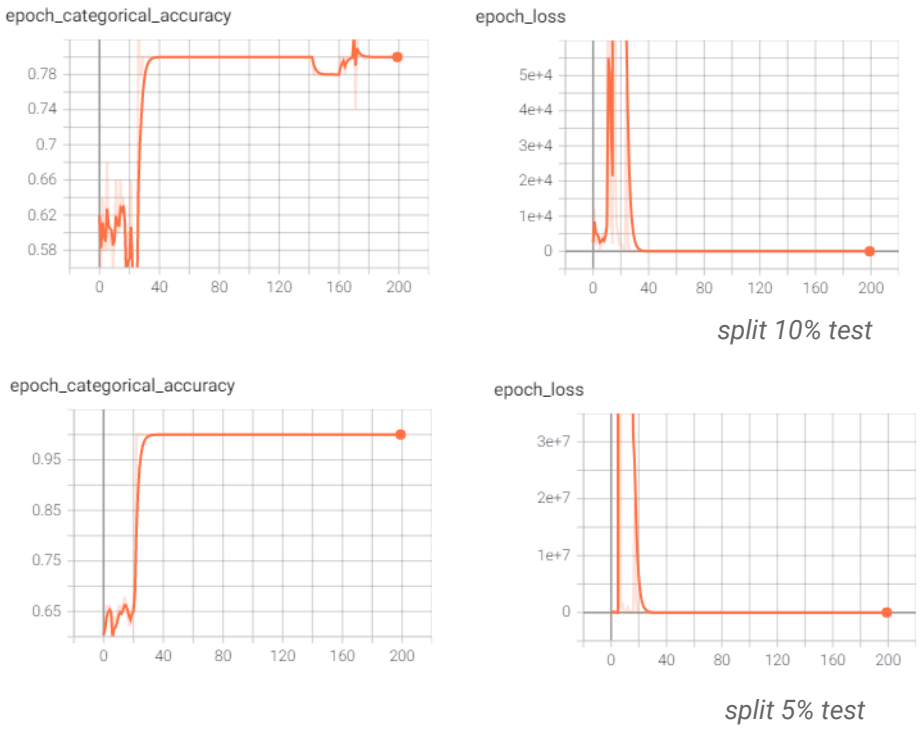


Fig 11 - LSTM andamento della fase di training facendo la prova con due esercizi differenti

Questi risultati sono probabilmente dovuti al fatto che il nostro dataset è troppo piccolo per utilizzare un approccio di tipo *deep learning* e conseguentemente utilizzeremo le metodologie descritte nello *shallow learning*.

Planning

In questa sezione vengono dettagliati gli aspetti per la parte di *planning* per la generazione della scheda di allenamento.

Answer set programming

Il problema del *workout scheduling problem*, dettagliato nel capitolo successivo, viene modellato nel linguaggio Answer Set Programming (ASP) che è una forma di programmazione logica di tipo dichiarativo utilizzato per problemi di ricerca complessi. [15,16,17]
Come altri linguaggi di programmazione dichiarativi, il programmatore specifica qual è il problema da risolvere, rispetto al come risolverlo.
In ASP i problemi di ricerca sono ridotti al calcolo di answer sets e per generarli vengono utilizzati programmi appositi noti come answer set solvers. Vi sono due linguaggi tipici di questo modello di programmazione, l'*Answer Set Programming in Logic* (AnsProlog) e *Datalog*, che appartengono ad un sottoinsieme di Prolog.

$$ASP = DB + LP + KR + SAT$$

In poche parole, ASP è una combinazione di concetti di database (DB), logic programming (LP), knowledge representation (KR) e satisfiability (SAT).

ASP, oltre ad avere riscontro nella rappresentazione della conoscenza e di pianificazione (planning), può essere anche utilizzato per risolvere problemi computazionalmente difficili, come il problema della colorazione dei grafi, del cammino hamiltoniano o di soddisfacibilità booleana.

Il tool che utilizziamo per risolvere il problema di pianificazione è *clingo*, sviluppato dall'Università di Potsdam. [18]

Sintassi

Un *atomo standard* è un'espressione $p(t_1,...,t_n)$, dove *p* è un *predicato* con *n* argomenti e $t_1,...,t_n$ sono i termini. Le variabili sono stringhe con la prima lettera maiuscola e le costanti sono interi non negativi o stringhe con la prima lettera minuscola. Un *termine* è una variabile o una costante. Un atomo $p(t_1,...,t_n)$ è ground se $t_1,...,t_n$ sono costanti. Un *set ground* è un set di coppie nella forma *<consts : conj>*, dove *consts* è una lista di costanti e *conj* è una congiunzione di atomi ground. Un *set di simboli* è un set definito sintatticamente come $\{Terms_1 : Conj_1; \dots ; Terms_t : Conj_t\}$, dove $t>0$, e per tutte le $i \in [1,t]$, ogni $Terms_i$ è una lista di termini tale che $|Terms_i| = k > 0$, e ogni $Conj_i$ è una congiunzione di atomi standard.
Un *set di termini* è o un ground set o un set di simboli. Intuitivamente, un set di termini *{X : a(X, c), p(X); Y : b(Y,m)}* rappresenta l'unione di due set: il primo contiene i valori *X* che rendono la congiunzione *a(X,c)*, *p(X)* vera, e il secondo contiene i valori *Y* che rendono la congiunzione *b(Y,m)* vera. Una *funzione aggregato* è della forma *f(S)*, dove *S* è un set di termini, e *f* è un simbolo di una *funzione aggregato*.
Le *funzioni aggregato* mappano vari set di costanti su una costante. Le più comuni funzioni implementate nei sistemi ASP sono:

- *#count*, numero di termini
- *#sum*, somma di interi
- *#max*, valore massimo tra interi
- *#min*, valore minimo tra interi

Un *atomo aggregato* è della forma $f(S) \prec T$, dove *f(S)* è una funzione aggregato, $\prec \in \{<,\leq,>,\geq,\neq,=\}$ è un operatore di confronto, e T è un termine chiamato guardia. Un atomo aggregato $f(S) \prec T$ è ground se *T* è una costante e *S* è un set ground.
Un *atomo* è un atomo standard o un atomo aggregato. Una *regola r* ha la seguente struttura:

$$a_1 \vee \dots \vee a_n :- b_1, \dots b_k, not\ b_{k+1}, \dots, not\ b_m.$$

dove $a_1, \dots, a_n$ sono atomi standard, $b_1, \dots, b_k$ sono atomi, $b_{k+1}, \dots, b_m$ sono atomi standard, e $n, k, m \geq 0$. Un *literal* è o un atomo standard *a* oppure la sua negazione *not a*.
La disgiunzione $a_1 \vee \dots \vee a_n$ è la *testa* di *r*, mentre la congiunzione $b_1, \dots, b_k$, *not* $b_{k+1}, \dots,$ *not* b_m è il suo *corpo*. Le regole senza corpo sono dette *fatti*. Le regole senza testa sono dette *vincoli*. Una variabile che compare solo nei termini di una regola *r* è detta *locale* in *r*, altrimenti è una variabile *globale* di *r*. Un programma ASP è un insieme di regole *safe*, dove una regola *r* è *safe* se rispetta le seguenti condizioni:

- per ogni variabile globale *X* di *r* c'è un atomo standard *l* nel corpo di *r* tale che *X* appare in *l*
- ogni variabile locale di *r* che appare in un set di simboli {*Terms : Conj*} appare anche in un atomo positivo in *Conj*

Un *weak constraint* è nella forma:

$$:\sim\ b_1,\ \ldots\ ,\ b_k\ ,\ not\ b_{k+1}\ ,\ \ldots\ ,\ not\ b_m.\ [w@l]$$

dove *w* e *l* sono rispettivamente il peso e la priorità del *weak constraint*.
Un atomo standard, un literal, una regola, un programma o un weak constraint è *ground* se non vi appaiono variabili in esso.

Nel nostro modello utilizziamo anche *#minimize{p : x(p)}* e *#maximize{p : x(p)}* che servono per minimizzare e massimizzare la somma di tutti i valori *p* che rendono vera *x(p)*, è quindi interpretabile come un weak constraint, ma con una migliore leggibilità.

Metodologia di risoluzione

La metodologia di risoluzione di ASP viene presentata nel seguente schema a blocchi:

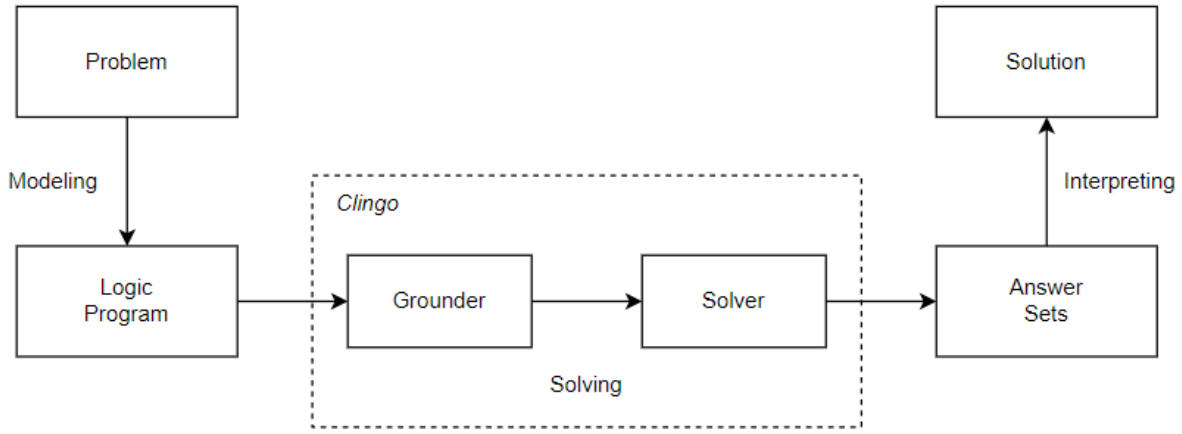


Fig. 12 - Metodologia di risoluzione di ASP

Il processo parte dal problema che viene modellato in un programma logico. Poi vi è la parte di solving gestita da *clingo*, che si suddivide in due parti.
Per primo, il grounder genera una rappresentazione proposizionale finita dato l'input del logic program, dopodiché, il solver processa e stampa gli answer sets (o stable models) del programma proposizionale.

Infine, si ottiene la soluzione tramite l'interpretazione corretta degli answer sets.

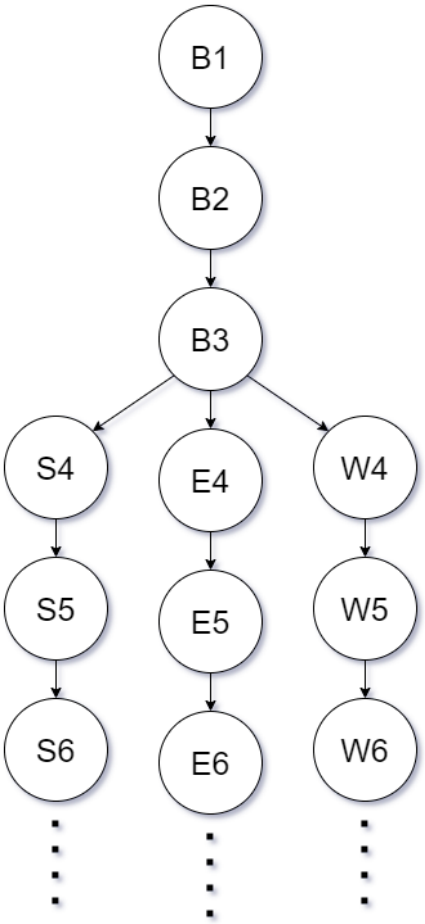
Definizione del problema

Il problema del *workout scheduling problem* consiste nel generare una scheda di allenamento settimanale in base ad alcuni criteri che devono essere soddisfatti.
Avere un'efficiente scheda di allenamento è importante per migliorare la salute o la performance atletica di una persona. Ognuno può avere obiettivi differenti, come sviluppare forza, perdere peso, aumentare la resistenza, allenare determinati gruppi muscolari o la combinazione di questi. Ci sono anche altri requisiti e limitazioni che devono essere presi in considerazione quando si crea una scheda. Per esempio le disponibilità di tempo e il livello attuale della persona.

In questa sezione descriviamo in modo informale i vari requisiti che teniamo in considerazione.

Requisiti dell'allenamento.

Il livello di un allenamento è così astratto e strutturato, per ogni livello vi sono proposte di esercizi dedicati e tarati per quel livello.
B1, B2, B3 sono 3 livelli base (B) in cui saranno presenti esercizi di base del calisthenics scalati in 3 difficoltà. Superato il livello B3 si ha la possibilità di scegliere di specializzarsi in tre rami differenti.



Skills (S), un ramo dove prevalentemente vengono proposti esercizi propedeutici per ottenere skills come le verticali, front lever, back lever, ecc…

Endurance (E), un ramo dove prevalentemente vengono proposti esercizi del calisthenics ad alte ripetizioni e bassi tempi di riposo.

Weighted (W), un ramo dove prevalentemente vengono proposti esercizi di calisthenics a basse ripetizioni ma con pesi aggiuntivi specifici per migliorare la forza basato sulla teoria del "*progressive overload*[11]".

Nel nostro caso, stabiliamo che una scheda di allenamento conterrà esercizi del livello attuale dell'utente e del livello precedente. Invece se il livello attuale è B1 avremo esercizi solo del livello B1 essendo il livello iniziale e di conseguenza non ha livelli precedenti.

Una volta ottenuti gli answer sets che rispettano tutti i vincoli, scegliamo l'answer set con la maggior varietà degli esercizi.

[11] https://www.healthline.com/health/progressive-overload

Nella fase di ottimizzazione, minimizziamo gli esercizi del livello precedente, quindi saranno presenti più esercizi del livello attuale.

Alla fine della settimana, il sistema produrrà un report della performance dell'utente in cui stabilirà se mantenere, aumentare o diminuire il livello di difficoltà.

Requisiti dell'utente.

Per quanto riguarda l'utente, avrà la possibilità di inserire una serie di parametri, tra cui il livello attuale, la categoria di allenamento e gruppo muscolare in cui vuole migliorare, i giorni, il tempo e gli strumenti disponibili da dedicare all'allenamento.
Vi è comunque un bilanciamento degli esercizi proposti, cioè se l'utente preferisce allenare la parte superiore del corpo, la maggior parte degli esercizi proposte dallo schedulatore sarà mirata al potenziamento del gruppo muscolare scelto dall'utente ma manterrà una piccola parte degli esercizi relativi allo sviluppo dei gruppi muscolari restanti.

Architettura

L'architettura del nostro sistema si presenta nel seguente modo divisa in due parti, una parte costituita dal planning e l'altra parte dal movement analysis.

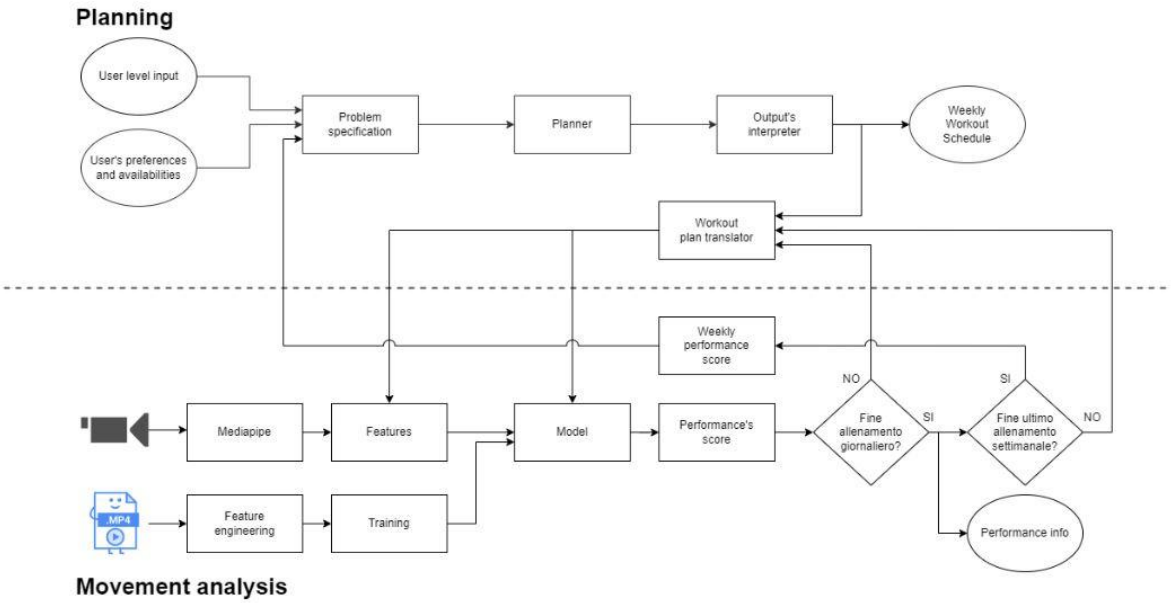


Fig. 13 - Architettura del sistema

Dal punto di vista dell'utente, lo schema parte dagli user input dove l'utente specifica le richieste, disponibilità e preferenze. Nel nostro caso possiamo immaginarlo come un form con eventuali controlli di validità degli input e ci assicuriamo di avere in input dati non contrastanti e "sensati" (per esempio non accetteremo 10 secondi di allenamento) così da rendere impossibile la mancanza di soluzioni.
Successivamente i dati verranno elaborati e trasformati in un file in formato testo che sarà dato in input al nostro planner.
Il planner è un componente basato sul Answer Set Programming che tramite la metodologia *Guess & Check & Optimize* ci restituisce una eventuale soluzione ottima, sub-ottima oppure nessuna in base agli input dati. Generalmente si attendono risultati con soluzione ottimali o sub-ottimali.

Ottenuta la soluzione, essa viene processata dal nostro interprete per essere stampata su schermo in un formato comprensibile all'utente.
Viene anche generato un output in formato json per essere utilizzato da eventuali applicazioni front-end e dal nostro workout plan translator che caricherà e assegnerà i vari modelli degli esercizi che saranno eseguiti dall'utente in quel giorno.

Fatto ciò si passa alla parte di movement analysis.
In questa parte, inizialmente è stato registrato e prodotto un dataset di video per effettuare il training del modello di ML del relativo esercizio, ottenuto ciò, possiamo utilizzarlo per dati nuovi immessi dall'utente.
Caricata la lista degli esercizi da eseguire per il determinato giorno, l'utente verrà guidato ad impostare correttamente il suo dispositivo per la rilevazione del corpo e procederà con l'inizio dell'allenamento.
Il movimento viene rilevato, tracciato e analizzato da *MediaPipe*, restituendo le coordinate di una lista di punti (*keypoints*), a partire dalle quali vengono calcolati i valori di un insieme di feature identificate attraverso un processo di *feature engineering*, che spiegheremo in dettaglio nella sezione riguardante l'implementazione.
Estratte le features, esse vengono trasmesse in input al modello che classificherà la qualità del movimento eseguito dall'utente.

Data la lista degli esercizi da eseguire, presenti nel workout plan helper, l'utente sarà guidato a completare la lista. A fine allenamento, quindi esauriti gli esercizi nella lista, l'utente otterrà un report in cui saranno presenti le informazioni delle sue performance compresi gli eventuali errori commessi, così da capire se l'andamento dell'allenamento è stato positivo o negativo.

L'esito degli allenamenti settimanali andrà ad influenzare non solo l'eventuale passaggio di livello ma andrà anche ad influire sul numero di serie, numero di ripetizioni e tempo di riposo che saranno aumentati o diminuiti negli allenamenti futuri. Questo comportamento viene deciso nel blocco del weekly performance score che viene dettagliato nella sezione successiva. Se la performance settimanale è stata scarsa, allora il sistema produrrà un output in cui consiglierà di scalare la difficoltà. Invece se la performance è stata buona, allora il sistema produrrà un output in cui consiglierà di passare al livello successivo e sarà l'utente a prendere la decisione finale se mantenere il livello attuale oppure effettuare il salto, così da adattarsi alle esigenze di ognuno.

Detto ciò, il ciclo settimanale riparte dagli input dell'utente con parametri nuovi.

Integrazione tra Movement analysis e Planning

Per poter integrare le due componenti, bisogna introdurre dei valori che andranno ad influenzare la scelta del passaggio di livello.

- *Score singolo esercizio*, questo valore è la media calcolata dai risultati della classificazione prodotta dal modello. Più dettagli tecnici saranno descritti nella sezione *training* in implementazione.
- *Score fine allenamento giornaliero* (*ScoreDay)*, questo valore viene elaborato a fine allenamento della giornata calcolato da una somma pesata. I pesi in questione sono di fattore 1 per gli esercizi del livello attuale e di fattore 0.3 per gli esercizi del livello precedente.
  La motivazione di questi pesi risiede nel fatto che si consideri gli esercizi del livello attuale più rilevanti per il cambio di livello. Questo *score* viene calcolato alla fine di ogni giorno di allenamento.

La seguente formula rappresenta lo *score di fine allenamento* giornaliero effettivo che ha raggiunto l'utente eseguendo gli esercizi proposti.

$$ScoreDay = \sum_{i}^{n}\left(\frac{ScoreSingoloEs_i}{Nclassi_i} * PesoLivelloAttuale\right) + \sum_{j}^{m}\left(\frac{ScoreSingoloEs_j}{Nclassi_j} * PesoLivelloPrec\right)$$

dove:

- *n* rappresenta il numero totale di esercizi del livello attuale svolti in quella sessione
- *ScoreSingoloEs* (*Score singolo esercizio*) è citato precedentemente
- *Nclassi* rappresenta il numero di categorie che il modello può classificare per il i-esimo e j-esimo esercizio proposto
- *PesoLivelloAttuale* assume il valore 1, ma potrebbe assumere valori diversi in base a quanta importanza si vuol dare agli esercizi del livello attuale
- *m* rappresenta il numero totale di esercizi del livello precedente svolti in questa sessione
- *PesoLivelloPrec* assume il valore 0.3, ma potrebbe assumere valori diversi in base a quanta importanza si vuol dare agli esercizi del livello precedente

La formula sottostante rappresenta lo *score massimo di fine allenamento* ed è lo score che l'utente può raggiungere eseguendo perfettamente tutti gli esercizi proposti.

$$MaxScoreDay = \sum_{i}^{n}\left(\frac{MaxValClass_i}{Nclassi_i} * PesoLivelloAttuale\right) + \sum_{j}^{m}\left(\frac{MaxValClass_j}{Nclassi_j} * PesoLivelloPrec\right)$$

dove:

- *MaxValClass* è il valore massimo della performance che il modello può classificare

Successivamente, per ogni giorno di allenamento si effettua il rapporto tra lo *ScoreDay* e *MaxScoreDay* per ricavare l'*IndiceCambioLivello*.

$$IndiceCambioLivello = \frac{ScoreDay}{MaxScoreDay}$$

Se il risultato è > 0.8 allora la sessione di allenamento verrà considerata positivamente. Effettuando 5 sessioni con questi risultati consecutivi si avanzerà di livello. Invece, se il risultato è compreso tra 0.5 e 0.8 inclusi, la sessione verrà considerata neutra, quindi non influenzerà il cambio di livello. Per ultimo, se il risultato è < 0.5 la sessione è considerata in modo negativo e con 5 di questi risultati consecutivi si retrocede di un livello.

All'ultimo giorno dell'allenamento della settimana, viene effettuato il check degli ultimi 5 *indici di cambio livello* che stabilisce se l'utente può avanzare, mantenere o retrocedere di livello in base a determinati score relativi alle sessioni di allenamento.

Implementazione

In questa sezione vengono presentati gli aspetti tecnici del progetto e i passaggi effettuati per la realizzazione del prototipo.
Nella prima parte vengono trattati i vari step sull'analisi del movimento, più in dettaglio del movimento *"alzate laterali"*.
Nella seconda parte sono spiegati gli step sulla pianificazione e generazione delle schede di allenamento.

Movement analysis

In questa sezione vengono trattati gli aspetti tecnici sull'analisi del movimento.
Abbiamo adottato l'approccio di shallow learning, è stata fatta questa scelta per via della dimensione del dataset molto ristretto e delle risorse computazionali abbastanza limitate.

Gestione del dataset

I video registrati dei vari movimenti, sia corretti che errati, vengono ritagliati in brevi sequenze, dove il punto di partenza è quando la persona assume la posizione di partenza e il punto di fine è quando conclude la ripetizione dell'esercizio.
Fatto ciò, queste sequenze vengono sistemate in apposite cartelle in base a come sono state classificate.
Tramite il nostro script Python ***annotateData_module.py***, vengono elaborati tutti i video presenti in queste cartelle e vengono prodotti file *pickle* (formato standard per salvare dati in Python) contenenti informazioni del movimento effettuato dalla persona.

In questo script si possono individuare tre funzioni:

- ***extract_keypoints***: estrae i *keypoints* dal video e li trasforma in un array Numpy
- ***createAnnotation***: dato il nome della cartella in input alla funzione, si analizza ed elabora ogni sottocartella che contiene video, producendo i file pickle annotati per ogni video
- ***readAnnotation***: dato il nome della cartella in input alla funzione, legge ogni file pickle e restituisce tre dati, una lista di *sequences*, *labels* e *actions*, che successivamente vengono trasformati in *Numpy arrays* per alcuni vantaggi di prestazione

Dopo le varie ricerche e analisi, si è deciso di utilizzare *Mediapipe* in quanto soddisfa pienamente i nostri requisiti come la possibilità di essere utilizzato su più piattaforme con ottime prestazioni.
Per prima cosa, si è studiata la documentazione e si sono testate le varie funzionalità che il framework offre. Dopodiché si è passati alla fase di prototipazione, dove si è sviluppata una soluzione in JavaScript, quindi mirata per applicazioni web, in cui viene rilevato il corpo della persona e viene effettuato un conteggio di quante volte la persona assume la posizione del T-pose, quindi avvicinandoci al nostro movimento delle *"alzate laterali"*. [14]
Tramite questa implementazione, si sono comprese a fondo le caratteristiche del framework e il suo utilizzo.
È stata utilizzata la webcam del laptop come dispositivo di cattura del video che presenta una qualità HD 720p. I video sono registrati in un ambiente con scarsa luminosità (camera da letto) e presentano buone prestazioni di body tracking. Successivamente testando con una fotocamera mirrorless Sony A6400 con obiettivo Sigma 16mm f/1.4, le prestazioni del tracking sono migliorate ulteriormente grazie alla qualità migliore dell'immagine.

Il framework presenta vari modelli come citato nel capitolo relativo alla metodologia e nel nostro caso abbiamo preso in considerazione il modello *Pose* in ambiente Python.

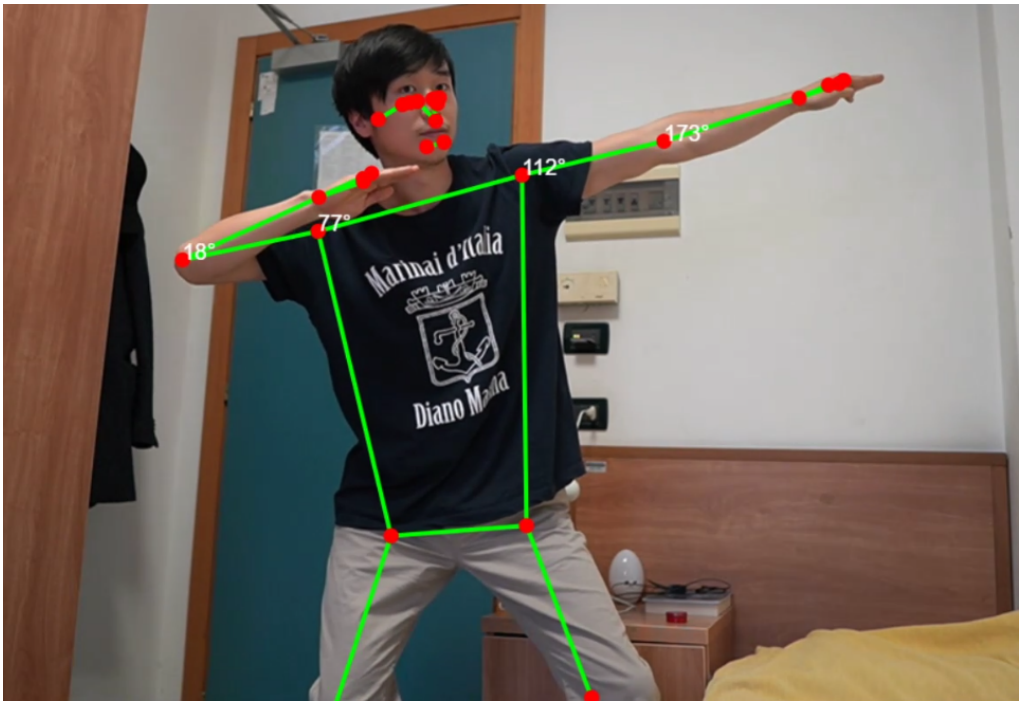
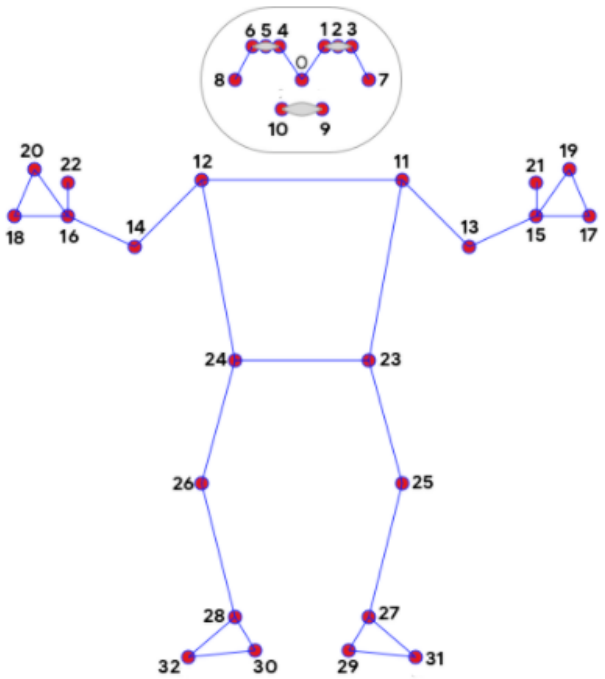


Fig. 14 - Modello Pose di MediaPipe

Esso presenta buone prestazioni in real-time in quanto il componente di rilevamento della persona (*detector*) è invocato solo quando necessario, per esempio al primo frame e quando il tracker non trova più la presenza del corpo nel frame precedente. Per gli altri frame, la pipeline semplicemente deriva la *ROI* (*region of interest*) dai keypoints del frame precedente.

Per ogni frame questo modello restituisce 33 *3D keypoints* con cui lavorare.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Fig. 15 - Rappresentazione dei keypoints del modello Pose

Ogni punto contiene due tipi di informazioni:

- struttura dati *poseLandmarks*
  - *x* : coordinata normalizzata tra 0 (a sinistra) e 1 (a destra)
  - *y* : coordinata normalizzata tra 0 (in alto) e 1 (in basso)
  - *z* : coordinata stimata e normalizzata tra -1 (vicino) e 1 (lontano), rappresenta la profondità avente come origine il punto medio dei fianchi
  - *visibility* : rappresenta la visibilità di quel punto, normalizzata tra 0 (non visibile) e 1 (visibile)

- struttura dati *poseWorldLandmarks*
  - *x*, *y*, *z* : coordinate 3D real-world espresse in metri con l'origine al centro dei fianchi
  - *visibility* : identica a come descritto in *poseLandmarks*

Per nostra preferenza, consideriamo solamente i dati contenuti nella struttura dati *poseLandmarks*.

Feature engineering

Ottenuti i valori dei vari *keypoints*, si procede con il calcolo delle features in cui estraiamo informazioni che caratterizzano determinate pose [4.27].
Questa sezione è implementata nello script ***math_module.py*** in cui sono presenti funzioni utili ai vari calcoli matematici, conversioni, gestione del modello ML e stampe a video di grafici e matrici.

A seguire è rappresentato il feature vector che si passa in input all'algoritmo di ML.

```
[ coeff_schiena,
coeff_braccia,
coeff_angolo_medio_braccio_destro,
coeff_angolo_medio_braccio_sinistro,
coeff_simm_x,
coeff_simm_y,
angolo_spalla_destra,
angolo_spalla_sinistra ]
```

coeff_schiena è un coefficiente che viene calcolato per valutare se l'utente mantiene la schiena dritta durante l'allenamento. Per ogni frame viene calcolata la distanza tra il marker relativo al naso e un punto derivato dalla differenza tra i due punti che rappresentano l'anca sinistra e l'anca destra. Dopodiché i valori vengono normalizzati rispetto al valore massimo ottenuto dall'operazione precedente, sommati e verrà effettuata la media. In questo modo più il valore è vicino ad 1 e più la schiena è rimasta dritta durante l'esercizio (dato che la registrazione dell'esercizio parte solo dopo aver raggiunto la posizione di partenza che implica la schiena dritta).

coeff_braccia è un coefficiente che viene calcolato per valutare se le braccia sono tese durante l'esercizio. Iniziamo calcolando per ogni frame la distanza tra il punto rappresentante la spalla e il punto relativo al polso con la formula della distanza tra due punti. Quindi dati i due punti: $p_1 = (x_1, y_1)$ e $p_2 = (x_2, y_2)$

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

la formula è applicata sia per il braccio sinistro che per il braccio destro.
Si normalizzano i valori in base al valore massimo e si fa la media. Si associa a ***coeff_braccia*** il valore del braccio che ha conseguito il punteggio minore (più il valore è vicino a 1 e più il braccio è teso, quindi prendiamo il valore minore associato al braccio che si è piegato di più durante l'allenamento.

coeff_angolo_medio_braccio_destro e ***coeff_angolo_medio_braccio_sinistro*** sono dei coefficienti che indicano l'angolo medio assunto dal gomito destro e sinistro durante l'esercizio. Per ogni frame viene calcolato l'angolo in radianti utilizzando la seguente formula.

$$rad = arctan(p_3 - p_2) - arctan(p_1 - p_2)$$

dove p_1 rappresenta il punto relativo alla spalla, p_2 il punto relativo al gomito e p_3 il punto relativo al polso.

Dopodiché viene convertito in gradi con la formula

$$deg = | rad * 180/\pi |$$

I valori vengono sommati e viene fatta una media normalizzata rispetto 180°.

Questo è utile per calcolare le features successive.
Di seguito è rappresentata l'implementazione in Python della funzione.

```
def calculate_angle(a, b, c):
    a = np.array(a)  # Punto iniziale
    b = np.array(b)  # Punto centrale
    c = np.array(c)  # Punto finale

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians * 180.0 / np.pi)
    if angle > 180.0:
        angle = 360 - angle
    return angle
```

coeff_simm_x e ***coeff_simm_y*** rappresentano la somma delle distanze tra i polsi per ogni frame. Calcoliamo il punto centrale sull'asse *x* rispetto alla posizione delle 2 spalle

$$spalla\ destra\ = p_1 = (x_1, y_1)$$
$$spalla\ sinistra\ = p_2 = (x_2, y_2)$$
$$centroX = (x_1 + x_2)/2$$

a questo punto dato

$$polso\ destro\ = p_3 = (x_3, y_3)$$
$$polso\ sinistro\ = p_4 = (x_4, y_4)$$

si calcola la differenza tra la distanza del *polso destro* da *centroX* e la distanza tra il *polso sinistro* e *centroX*

$$distanzaX = \left|\left|x_3 - \ centroX\right| - \left|x_4 - \ centroX\right|\right|$$

dopodiché calcoliamo la distanza tra i due polsi sull'asse *y*

$$distanzaY = \left|y_3 - y_4\right|$$

si sommano le distanze di tutti i frame e si ottengono ***coeff_simm_x*** e ***coeff_simm_y***.
Più i valori sono piccoli, più il movimento delle braccia è stato eseguito simmetricamente.

angolo_spalla_destra e ***angolo_spalla_sinistra*** rappresentano l'angolo massimo raggiunto dalle 2 spalle. Per ogni frame e per ogni spalla calcoliamo

$$rad = arctan(p_3 - p_2) - arctan(p_1 - p_2)$$
$$deg = |\, rad * 180/\pi \,|$$

siccome abbiamo a disposizione i punti relativi all'anca, spalla e gomito ma anatomicamente l'anca non ha la stessa *x* della spalla, utilizziamo per l'anca la *x* della spalla per avere una linea verticale da dove misurare l'angolo della spalla, quindi:

$$gomito = p_1 = (x_1, y_1)$$
$$spalla = p_2 = (x_2, y_2)$$
$$anca = p_3 = (x_2, y_3)$$

ovviamente se si calcola l'angolo della spalla destra useremo i punti relativi al gomito destro, spalla destra e anca destra e viceversa.

Di seguito mostriamo alcuni dei grafici più significativi delle misure utilizzate per calcolare le features, dai quali è possibile apprezzare visivamente l'andamento di alcune caratteristiche durante l'intera esecuzione della ripetizione al variare della classe dell'esercizio.
Come mostrato precedentemente ogni feature è composta da un solo valore, spesso normalizzato, per ogni esecuzione dell'esercizio. Quello che mostriamo in questi grafici sono i valori delle misure calcolate per ogni frame dalle quali estraiamo tramite le formule descritte in precedenza il valore della singola feature.

- **esercizio con braccia piegate**

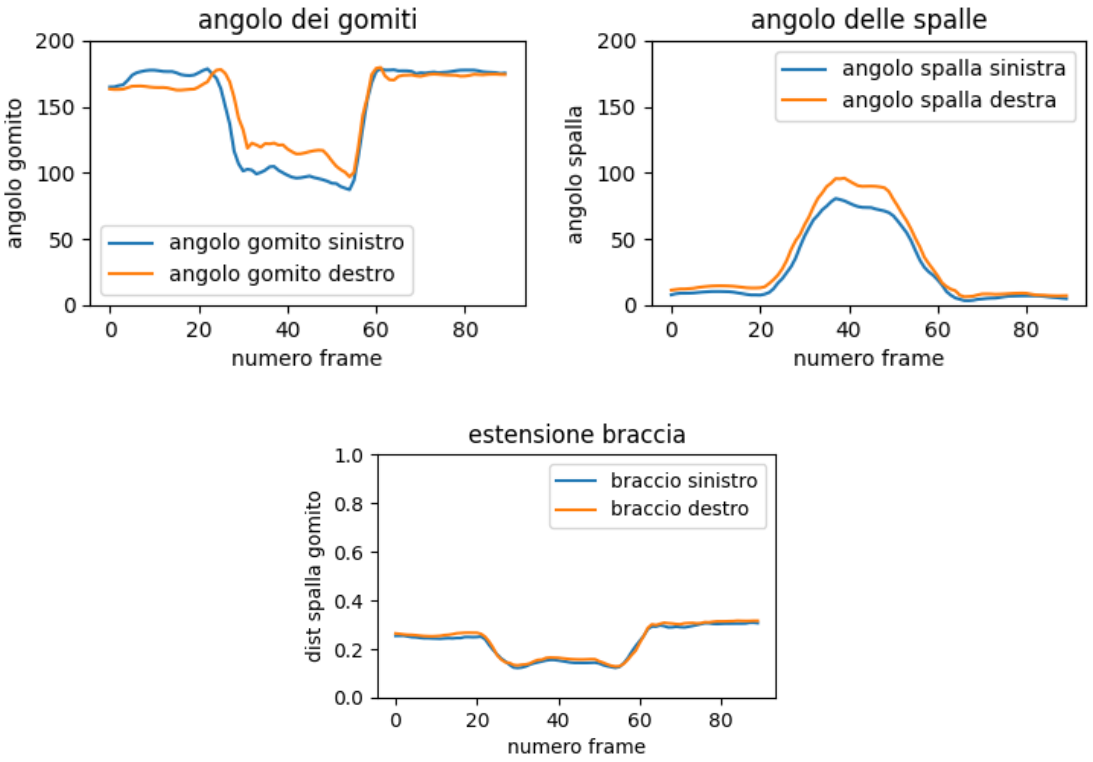


Fig. 16 - Grafici per esercizio di classe 0

- **esercizio con braccia asincrone**

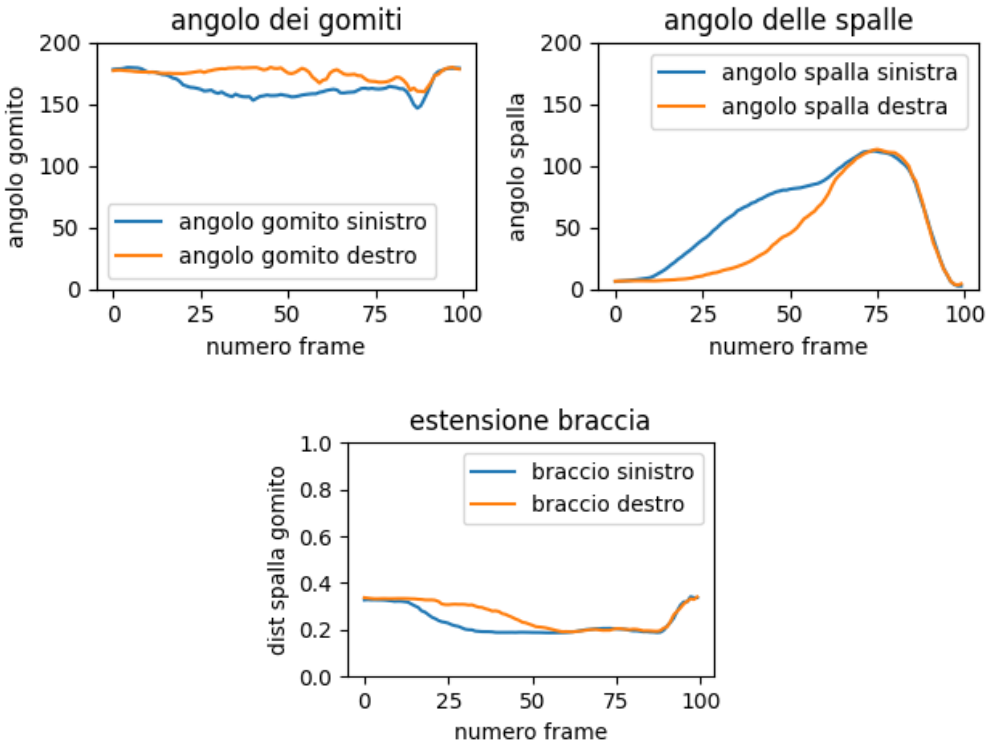


Fig. 17 - Grafici per esercizio di classe 1

- **esercizio con angolo delle braccia che non arriva a 90 gradi**

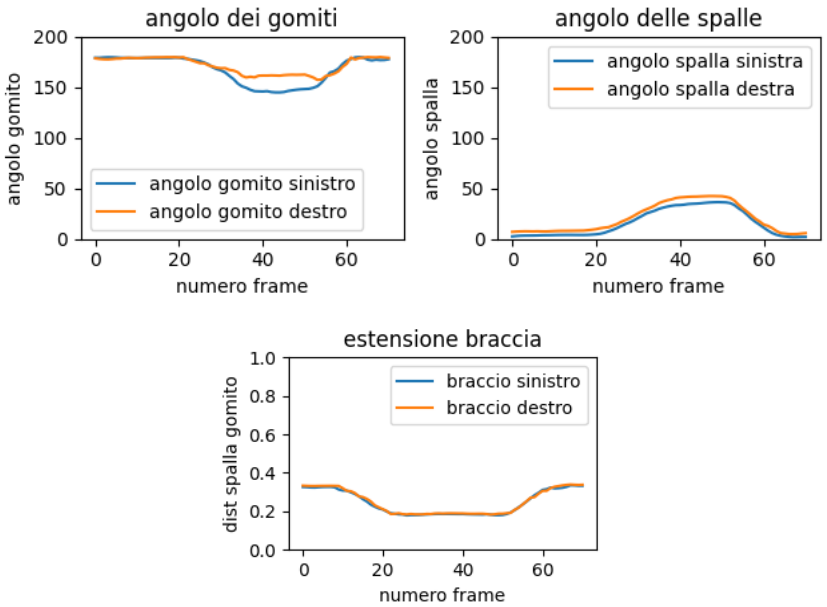


Fig. 18 - Grafici per esercizio di classe 2

- **esercizio eseguito correttamente**

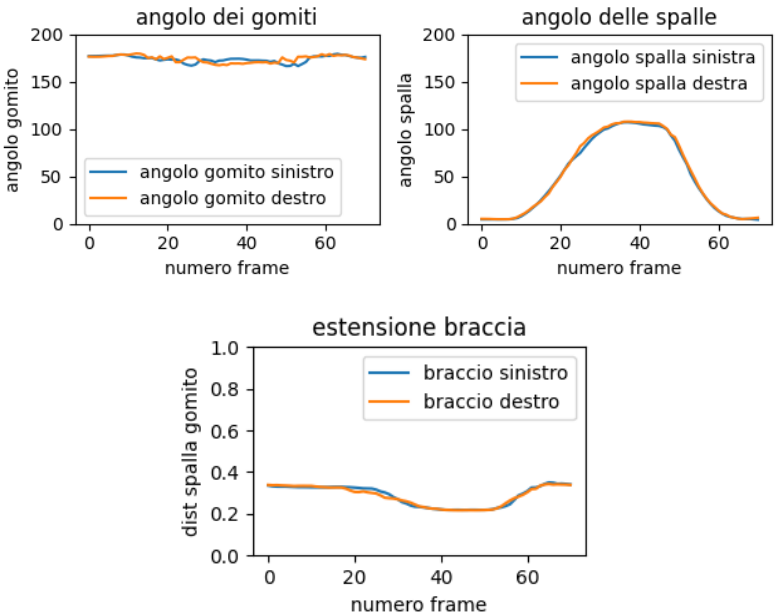


Fig. 19 - Grafici per esercizio di classe 3

Questo metodo di calcolo delle feature ci risolve uno dei problemi più grandi relativi alla valutazione dell'esercizio, cioè il fatto che ogni persona esegue una ripetizione a velocità differente. In questo modo siamo in grado di rimuovere il tempo di esecuzione dell'esercizio dalle variabili a cui prestare attenzione.
Detto questo, possiamo eseguire il training del modello di ML.

Valutazione performance dei modelli

In questa sezione viene presentata la valutazione della generalizzazione dei modelli e gli score che valutano le performance dei modelli sul test set.

Per valutare quale modello performa in modo migliore, utilizziamo la tecnica della *nested cross validation*.

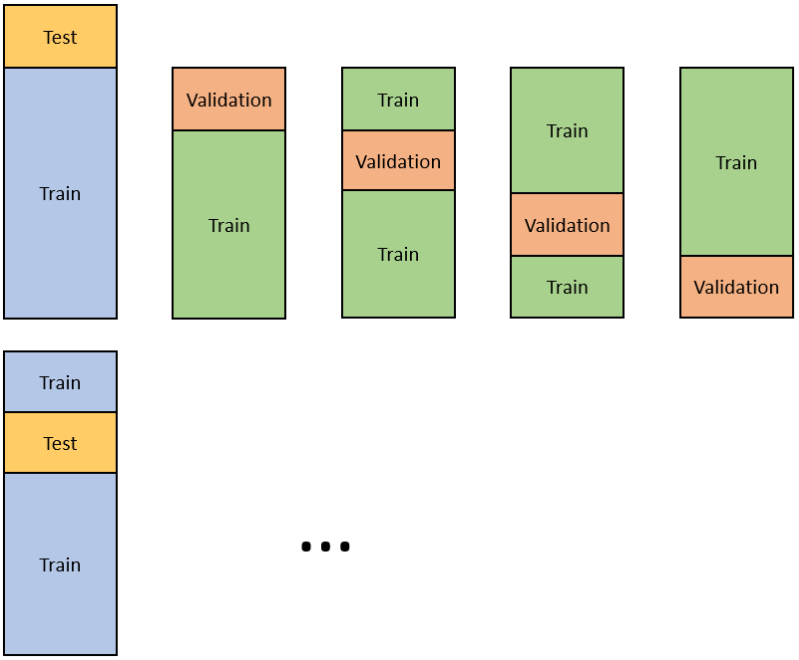


Fig. 20 - Split per nested cross validation

Questa tecnica consiste nell'effettuare un primo split del *training set* in *K fold*, nel nostro caso 5. A turno ogni *fold* viene usato come *test set*, mentre i restanti 4 vengono usati come *train/val set*. Ogni *train/val set* viene nuovamente splittato in *L subfold* che a turno, viene usato come *validation set*, mentre i rimanenti *L-1 subfold* vengono utilizzati come *train set*. Quest'ultima azione viene effettuata dalla funzione *cross_val_score* di *sklearn* mentre vengono testate le varie combinazioni di *hyperparameter* tramite *Optuna* utilizzando come training set il *train set* e viene valutato utilizzando *validation set* tramite *weighted f1 score*. Dopo aver effettuato l'ottimizzazione utilizzando come *validation set* tutti gli *L subfold*, prendiamo gli *hyperparameter* che ci hanno fornito il *weighted f1 score* più alto e li usiamo per effettuare il training del modello con *train/val set* e valutiamo le performance del modello con il *weighted f1 score* sul *test set*. Ripetiamo queste operazioni per tutti i *K fold* e alla fine di ciò, si calcola la media e la deviazione standard dei *weighted f1 score* che abbiamo ottenuto valutando i *K fold*.
Abbiamo scelto di utilizzare *f1 score* e non l'*accuracy score* perché le nostre classi nel *train set* non sono bilanciate, come si vede dalla figura sottostante.

```
classi in ltrain: zero=39 uno=29 due=29 tre=33
classi in ltest: zero=14 uno=14 due=16 tre=13
```

Fig. 21 - Stampa informazioni del dataset che risulta non bilanciato

Di seguito riportiamo lo pseudocodice della tecnica appena descritta.

- Randomly split the dataset into K folds.
- For i from 1 to K:
  - Let `test` be fold i.
  - Let `trainval` be all the data except that which is in `test`.
  - Randomly split `trainval` into L subfolds $(i,1),(i,2),\ldots,(i,L)$. So, each subfold (i,j) has some elements from outer fold 1, some from outer fold 2, ..., but none of them has any from outer fold i.
  - For j from 1 to L:
    - Let `val` be fold (i,j).
    - Let `train` be all the data which is not in either `test` or `val`.
    - Train each proposed parameter set on `train`, and evaluate it on `val`, keeping track of classification accuracy, RMSE, whatever criterion you want to use.
  - Look at the average score for each set of parameters over the L folds, and choose the best one.
  - Train a model with that best set of parameters on `trainval`.
  - Evaluate it on `test`, and save the accuracy / RMSE / whatever.
- Report the mean / mean + std / boxplot / whatever of the K accuracies.

Fig. 22 - Pseudocodice del nested cross validation[12]

Per ogni *fold* della *cross validation* si effettua la fase di *model tuning,* o più precisamente *hyperparameter optimization*, in cui si individuano gli *hyperparameter* ottimali per rendere il modello di classificazione più accurato. Per fare ciò, si è utilizzato il framework *Optuna*.
Un altro strumento simile per l'*hyperparameter optimization* che si potrebbe usare è la funzione *GridSearchCV* implementata da *scikit-learn*.

Per ogni *k-fold* si prendono i migliori *hyperparameter* che sono stati trovati nel ciclo interno della *nested cross validation*, dopodiché si effettua il training con questi *hyperparameter* sul *trainval set*. Il passo successivo è quello di utilizzare il modello per predire le classi di appartenenza degli elementi presenti nel *test set*.
Una volta ottenuto il risultato della predizione (*y_pred*), si comparano con le vere classi di appartenenza (*y_test*), calcolando le varie metriche (*recall score, precision score, f1 score, accuracy score* e *confusion matrix*).

Come mostreremo nel capitolo dei risultati, si sceglie il modello *Random Forest* in quanto presenta l'*f1-score* più alto rispetto agli altri modelli.

[12] https://stats.stackexchange.com/questions/266225/

Metriche per la valutazione

In questa sezione vengono presentate le metriche utilizzate per valutare un modello.

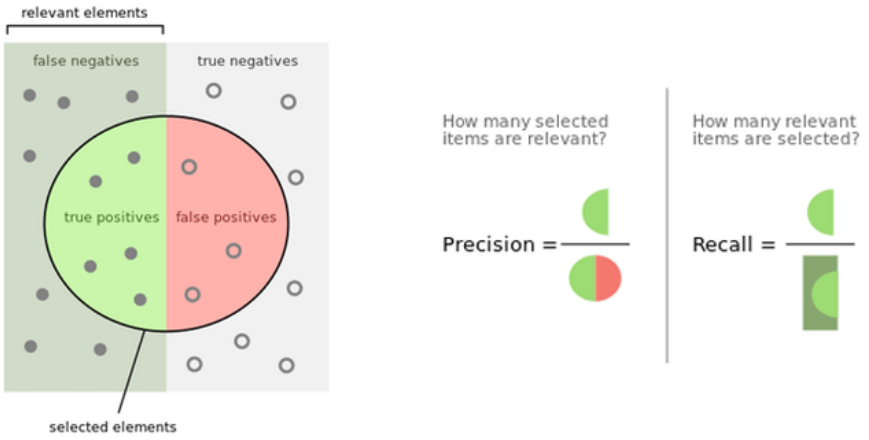


Fig. 23 - Precision score & Recall score

Quando un modello di machine learning cerca di classificare un oggetto tramite una serie di features fa delle prediction che possono ricadere in 4 categorie:

- *True Positive (TP)*: sono gli elementi che facevano parte della classe A e che correttamente sono stati classificati come classe A.
- *True Negative (TN)*: sono gli elementi che non fanno parte della classe A e che correttamente sono stati classificati come non appartenenti alla classe A.
- *False Positive (FP)*: sono gli elementi che non fanno parte della classe A ma che vengono classificati erroneamente come appartenenti della classe A
- *False Negative (FN)*: sono gli elementi che appartengono alla classe A ma che erroneamente vengono classificati come oggetti di classe A.

Una volta chiare queste distinzioni si possono esaminare gli score *Precision* e *Recall*:
Il *Precision score*[13] viene calcolato per ogni classe conosciuta dal modello con la seguente formula

$$Precision(A) = \frac{TP(A)}{TP(A) + FP(A)}$$

Quando questo score è alto indica che il modello è preciso per quella specifica classe quindi se un oggetto viene classificato come appartenente a quella classe, raramente la previsione è sbagliata.

Anche il *Recall score* viene calcolato per ogni classe conosciuta dal modello con la seguente formula

$$Recall(A) = \frac{TP(A)}{TP(A) + FN(A)}$$

Il recall indica l'abilità del classificatore di trovare i campioni positivi e varia da 0 (peggiore) a 1 (migliore).

[13] https://en.wikipedia.org/wiki/Precision_and_recall

Generalmente vogliamo entrambi gli score più alti possibili ma normalmente dobbiamo trovare un trade-off tra le 2 metriche perché all'aumentare di uno score spesso porta all'abbassamento dell'altro.

Utilizzando questi due score si può calcolare F1 score

$$F1\ Score(A) = \frac{2*Precision(A)*Recall(A)}{Precision(A)+Recall(A)}$$

L' *F1 score* viene usato come metodo per "combinare" i 2 score precedenti in una singola metrica dando un peso maggiore allo score più basso che di conseguenza abbasserà *F1 score* rispetto ad una media aritmetica tra i 2 score.

A questo punto, per quanto riguarda il nostro progetto, ci ritroviamo con 4 score F1 (uno per classe) che vogliamo combinare in uno score unico per valutare il modello nel suo complessivo e non solo classe per classe.
Per semplicità, ci riferiamo alle nostre 4 classi (*alzateLaterali0, alzateLaterali1, alzateLaterali2, alzateLaterali3*) tramite *A, B, C, D*.
Per fare ciò utilizziamo il Weighted-F1(e i relativi Weighted-Precision e Weighted-Recall)

$$Weighted_F1 = \frac{(TP(A)+FP(A)) * F1Score(A) + (TP(B)+FP(B)) * F1Score(B) + (TP(C)+FP(C)) * F1Score(C) + (TP(D)+FP(D)) * F1Score(D)}{campioniTotali}$$

Viene utilizzata anche la *confusion matrix*. Essa è uno strumento per analizzare gli errori compiuti da un modello di machine learning.

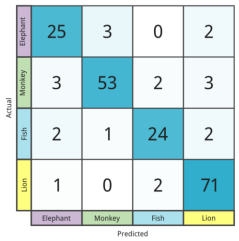


Fig. 24 - Esempio di confusion matrix

Ogni colonna della matrice rappresenta le istanze delle classi predette, mentre ogni riga rappresenta le istanze delle classi reali, data la struttura, sulla diagonale sono presenti il numero di campioni classificati correttamente mentre i valori al di fuori della diagonale rappresentano il numero di campioni classificati erroneamente.

Calcoliamo anche l'*accuracy score* tramite il rapporto tra il numero di campioni classificati correttamente e il numero di campioni totali (anche se utilizzeremo solo *f1 score* per la valutazione del modello in quanto le classi sono sbilanciate).

Descrizione moduli e training

Sono stati scritti 5 moduli per ogni algoritmo di ML applicato e sono:

- ***decisionTree_module.py*** : è il modulo in cui è stato utilizzato un albero decisionale per classificare la qualità del movimento
- ***randomForest_module.py*** : modulo in cui è stato utilizzato una random forest per la classificazione
- ***svm_module.py*** : modulo in cui è stato utilizzato una support vector machine per la classificazione
- ***gradientBoosting_module.py*** : modulo in cui è stato utilizzato il gradient boosting
- ***xgboost_module.py*** : modulo in cui è stato utilizzato extreme gradient boosting

Tutti questi moduli sono simili tra di loro e presentano 3 funzioni che servono per effettuare il training il modello finale:

- ***train_and_score*** : funzione principale utilizzata per invocare il classificatore in questione ed effettuare il fit, predict e score
- ***findBestHyperparameters*** : invoca il framework *Optuna* che effettua la ricerca della migliore combinazione di iperparametri da assegnare al nostro modello. Questa fase si chiama "*Model tuning*" o "*Hyperparameter optimization*"
- ***train*** : funzione utilizzata la prima volta per fare il training con i migliori iperparametri trovati da *Optuna* e successivamente effettua il salvataggio del modello

Tutti i modelli vengono salvati dopo la fase di training, successivamente è possibile caricarli per utilizzarli.

Gli step per utilizzare i modelli prodotti sono:

1. Caricare il modello
   ```
   model = math_module.load_model('nomeModello.sav')
   ```

2. Avviare la predizione passandogli i dati in input
   ```
   y_pred = model.predict(X_feature)
   ```

Il risultato y_pred è una lista contenente i valori che rispecchiano la determinata classe. Per esempio se y_pred = [0 0 2 3 1] significa che sono stati passati 5 dati di test in input, di cui il modello ha classificato il primo dato come classe con identificatore 0, il secondo uguale, il terzo come classe con identificatore 2, il quarto come classe 3 e l'ultimo come classe 1.

Live Webcam Testing

In questa fase carichiamo il modello scelto da utilizzare e ne testiamo le sue capacità predittive.
Questa funzionalità è implementata nel file ***alzateLaterali_live_module.py*** che prende come parametro in input il numero delle ripetizioni da eseguire per le *alzate laterali*.
Successivamente viene effettuata l'apertura del dispositivo di cattura dell'immagine tramite *OpenCV* e attiva *MediaPipe Pose* per il riconoscimento della *pose estimation*, producendo le informazioni necessarie da dare in input al modello caricato. Inoltre è implementata una linea guida che supporta l'utente con suoni e una check list interattiva relativa alle varie fasi del movimento.
A fine ripetizione, viene prodotto il *feature vector* sopra menzionato, utilizzato per classificare la qualità del movimento.

Concluse le ripetizioni, un sistema di *text-to-speech* viene lanciato e riassume le performance relative all'esercizio.

Nella sezione dei risultati viene presentata una **demo**.

Durante la fase di live testing, è stato riscontrato un problema sulla classificazione della qualità del movimento. Si pensava che il sistema saltasse dei frame a causa del tempo computazionale richiesto per svolgere tutte le operazioni. Il tempo impiegato ad eseguire un ciclo della funzione di tracking deve essere inferiore a 33.3 ms (1s / 30fps), quindi 1 frame ogni 33.3 ms.
Si è calcolato il tempo di esecuzione della nostra funzione ed è emerso che
il tempo massimo misurato risultava 24 ms, quindi al di sotto di 33.3 ms.

Alla fine abbiamo constatato che la causa della classificazione incorretta del movimento era dovuta alla qualità dell'immagine che la webcam produceva e al motion blur che si creava durante il movimento in ambienti poco luminosi.

Planning

In questa sezione parleremo dell'aspetto tecnico per la realizzazione del nostro schedulatore implementato in ASP[19,20,21,22,23] utilizzando ***Clingo***[24].
Il problema si presenta in due file, il primo è il *problem instance* dove vengono specificati tutti i predicati che andranno in input al solver; il secondo è il *problem encoding* dove vengono specificati i vincoli del problema e le varie ottimizzazioni delle soluzioni.

Problem instance

L'input è descritto dai seguenti predicati:

- Il predicato ***livello_attuale(categoria, livello)*** indica la categoria e il livello attuale dell'utente. La categoria indicata è una tra le seguenti quattro: B (base), S (skills), E (endurance), W (weighted). Il livello è un numero che varia da 1 a 6, dove il livello 1, 2, 3 può essere abbinato solo alla categoria base mentre il livello 4, 5, 6 può essere abbinato a skills, endurance e weighted.
- Il predicato ***disponibilita(giorno, tempoDiAllenamento)*** indica il giorno disponibile inserito dall'utente che varia da 1 a 7 e il ***tempoDiAllenamento***, espresso in secondi, rappresenta il tempo da dedicare all'allenamento di quel giorno. Questo predicato si ripeterà per il numero di giorni disponibili dell'utente.
- Il predicato ***input_performance_base(n_rep, n_serie, tempoRiposo)*** è generato dalla parte di Multimodal System che deciderà se incrementare, decrementare o mantenere il numero di ripetizioni, il numero di serie e il tempo di riposo agli esercizi della categoria B in base alla performance prodotta dalla settimana precedente. Di default è settato a 0, quindi mantiene l'allenamento invariato.
- Il predicato ***input_performance_skills(n_rep, n_serie, tempoRiposo, pesiExtra)*** ha lo stesso funzionamento di quello base ma per la categoria S.
- Il predicato ***input_performance_endurance(n_rep, n_serie, tempoRiposo, pesiExtra)*** ha lo stesso funzionamento di quello base ma per la categoria E.
- Il predicato ***input_performance_weighted(n_rep, n_serie, tempoRiposo, pesiExtra)*** ha lo stesso funzionamento di quello base ma per la categoria W.
- Il predicato ***warmup_time(tempoRiscaldamento)*** indica il tempo in secondi riservati al riscaldamento.
- Il predicato ***strumento(nomeStrumento)*** indica lo strumento posseduto dall'utente. In base agli strumenti selezionati, verranno scelte le schede di allenamento che contengono gli esercizi svolgibili tramite gli strumenti posseduti.
- Il predicato ***strumento_extra(nomeStrumentoExtra)*** indica se l'utente possiede gli strumenti di supporto come pesi o bande elastiche. In base agli strumenti extra posseduti, verranno scelte le schede di allenamento che contengono gli esercizi svolgibili tramite questi strumenti di supporto.
- Il predicato ***pref_gm(gruppoMuscolare)*** indica quale gruppo muscolare enfatizzare durante la scelta degli esercizi.
- Il predicato ***esercizio(nomeEsercizio, fattoreUtilitaSuperiore, fattoreUtilitaCore, fattoreUtilitaInferiore)*** indica il nome dell'esercizio con i relativi fattori di utilità. Questi ultimi specificano quanto l'esercizio è allenante per ognuno dei tre gruppi muscolari. Il valore di ogni fattore varia da 0 (non allenante) a 10 (allenante).

- Il predicato ***variante_esercizio(idEsercizio, nomeEsercizio, categoriaEsercizio, livelloRamo, n_rep, n_serie, durata_rep, durata_isometria, pesi_extra, strumento, tempoRiposo)*** indica la variante dell'esercizio che specifica:
    - ***idEsercizio***, un identificatore univoco per il predicato.
    - ***nomeEsercizio***, chiave esterna che ci permette di abbinare ogni variante al predicato esercizio visto precedentemente.
    - ***categoriaEsercizio*** e ***livelloRamo*** indicano la categoria di appartenenza della variante specifica e il suo livello di difficoltà.
    - ***n_rep*** e ***n_serie*** indicano il numero di ripetizioni e il numero di serie dell'esercizio da eseguire.
    - ***durata_rep*** indica la durata in secondi dell'esecuzione del determinato esercizio dinamico. Se il suo valore è 0 allora ***durata_isometria*** sarà maggiore di 0. Questo dato è stimato per eccesso basato su esperienza.
    - ***durata_isometria*** indica la durata in secondi dell'esecuzione del determinato esercizio statico. Se il suo valore è 0 allora ***durata_rep*** sarà maggiore di 0.
    - ***pesi_extra*** indica un valore che può essere sia positivo, negativo che nullo.
      Se il valore è positivo indica un'aggiunta di carico espresso in chilogrammi.
      Se il valore è negativo indica l'utilizzo di una banda elastica tarata per facilitare l'esecuzione del determinato esercizio.
      Se nullo allora non impiegheremo alcun peso o banda elastica.
    - ***strumento*** indica lo strumento necessario per eseguire la determinata variante dell'esercizio.
    - ***tempoRiposo*** indica il tempo di riposo espresso in secondi dopo aver concluso la serie.

Problem encoding

Qui descriviamo i vincoli ASP usati per modellare il problema della generazione della scheda di allenamento. La metodologia utilizzata è il ***Guess&Check&Optimize*****,** in cui consiste:

- ***Guess***, genera tutte le possibili soluzioni utilizzando *choice rules*
- ***Check***, effettua vari controlli e filtra quelle non pertinenti utilizzando *vincoli forti*
- ***Optimize***, massimizzare o minimizzare determinati valori in base alla priorità e selezionare la soluzione migliore utilizzando *vincoli deboli*

Di seguito abbiamo il vincolo che genera tutti gli answer set combinando tutte le possibili combinazioni di esercizio, variante esercizio e ogni giorno disponibile, tenendo conto anche dei parametri stabiliti da input performance della determinata categoria.

```
{esegui(NomeES, FU_Sup, FU_Core, FU_Inf, CL, LR, REP+PR, SERIE+PS, DR, DI, PESI,
STR, TR+PTR, DAY) :
esercizio(NomeES, FU_Sup, FU_Core, FU_Inf),
variante_esercizio(_, NomeES, CL, LR, REP, SERIE, DR, DI, PESI, STRUMENTO, TR),
input_performance_base(PR, PS, PTR),
disponibilita(DAY, _)}  :-  variante_esercizio(_,_,CL,_,_,_,_,_,_,_,_,_), CL = b.
```

Questo viene replicato per le altre tre categorie (s, e, w).
Per esempio, nel caso di categoria “Skills”, cambiano l’input del predicato in ***input_performance_skills(PR, PS, PTR)*** nella *testa* della *regola* e selezioniamo la categoria ***CL = s*** nel *corpo* della *regola*.

Successivamente vediamo i vari vincoli che andranno a filtrare tutti gli answer set precedentemente generati.

```
:- #count{STRUMENTO : esegui(NomeES,_,_,_,_,_,_,_,_,_,_,_,STR,_,DAY)} > 1,
   esercizio(NOME_ESERCIZIO,_,_,_),
   disponibilita(DAY, _).
```

Questo vincolo rimuove gli answer set che contengono più di una variante dello stesso esercizio nello stesso giorno.
Poi abbiamo il filtro dei giorni in base alla disponibilità dell’utente.

```
:- NUM_GIORNI_IN_ANSWER = #count{DAY : esegui(_,_,_,_,_,_,_,_,_,_,_,_,_,_,DAY)},
NUM_DAYS = #count{DAY : disponibilita(DAY,_)}, NUM_GIORNI_IN_ANSWER != NUM_DAYS.
```

Rimuove solo gli answer set che non contengono tutti i giorni disponibili.

Di seguito abbiamo il filtro dei livelli e categoria degli esercizi.

```
next_level(X,Z) :- esegui(_,_,_,_,_,X,_,_,_,_,_,_,_,_),
                   X = Z-1,
                   #false : X<Y,
                   esegui(_,_,_,_,_,Y,_,_,_,_,_,_,_,_), Y<Z;
                   esegui(_,_,_,_,_,Z,_,_,_,_,_,_,_,_), X<Z.
```

Generiamo un atomo ausiliario `next_level(X,Z)` che viene usato per stabilire una continuità tra il numero dei livelli. Se troviamo l'atomo `next_level(X,Z)` in un determinato answer set, ci indica che quell'answer set presenta almeno alcuni esercizi con livello adiacente.

```
:- livello_attuale(_,Livello_ramo),
    Livello_ramo != 1,
    #count{X,Y : next_level(X,Y)} = 0.
```

Nel caso del livello attuale sia diverso da 1, questo vincolo cancella gli answer set che non presentano l'atomo `next_level(X,Y)`.

Successivamente filtriamo gli answer set che presentano più di un `next_level(X,Y)` siccome vogliamo risultati che contengano al massimo 2 livelli.

```
:- #count{X,Y: next_level(X,Y)} > 1.
```

Dopodiché cancelliamo tutte le soluzioni che non presentano il livello attuale come secondo termine all'interno dell'atomo `next_level`.

```
:- livello_attuale(_,Livello_ramo),
    next_level(_,X),
    Livello_ramo != X.
```

Il vincolo successivo ci assicura di avere al massimo due livelli all'interno di ogni answer set.

```
:- #count{Ramo,Livello_ramo: esegui(_,_,_,_,Ramo,Livello_ramo,_,_,_,_,_,_,_,_)}>2.
```

Ora tolgo tutti gli answer set che presentano esercizi di livello superiore (quindi più difficili) al livello attuale dell'utente.

```
:- livello_attuale(_, Livello_attuale),
    esegui(_,_,_,_,_,_,Livello_ramo,_,_,_,_,_,_,_,_),
    Livello_ramo > Livello_attuale.
```

E successivamente filtro per la categoria interessata, cancellando ogni answer set che non presenta esercizi della nostra categoria.

```
:- #count{Categoria: esegui(_,_,_,_,_,Categoria,_,_,_,_,_,_,_,_,_),
livello_attuale(Categoria, _)} = 0.
```

Infine, si filtrano gli answer set che presentano esercizi di altre categorie. Per esempio se sono s5, non dobbiamo avere all'interno del nostro answer, esercizi di e4 e w4, ma solo s4.

```
:- livello_attuale(Categoria,_),
    esegui(_,_,_,_,_,Cat_es,_,_,_,_,_,_,_,_,_),
    Categoria != Cat_es,
    Cat_es != b.
```

Ora passiamo ai vincoli che filtrano per strumenti a disposizione.
Generiamo due atomi ausiliari che ci serviranno per applicare il filtro voluto.

```
countStrumenti(CS, Strumento) :- CS = #count{NOME_ESERCIZIO, DAY :
esegui(NOME_ESERCIZIO,_,_,_,_,_,_,_,_,_,_,_,Strumento,_,DAY)}, strumento(Strumento).
```

Il primo atomo `countStrumenti` ha come terms il numero degli esercizi che si svolgono dato uno strumento e lo strumento in questione.

Il secondo atomo `countEs` conta gli esercizi all'interno dell'answer.

```
countEs(CE) :- CE = #count{NOME_ESERCIZIO, Strumento, DAY :
esegui(NOME_ESERCIZIO,_,_,_,_,_,_,_,_,_,_,_,Strumento,_,DAY)}.
```

Infine sommiamo i primi valori di tutti gli atomi di `countStrumenti` e verifichiamo che la somma sia uguale al numero degli esercizi presenti in quel answer, se non è così significa che sono presenti esercizi da svolgere con strumenti che non sono in possesso dell'utente, quindi non fattibili.Quindi eliminiamo questi answer set.

```
:- TOTAL = #sum{CS : countStrumenti(CS, _)},
    countEs(CE),
    TOTAL != CE.
```

Successivamente valutiamo gli strumenti extra di supporto che l'utente possiede.

```
    :- not strumento_extra("weights"),
           strumento_extra("elastic band"),
           esegui(_,_,_,_,_,_,_,_,_,_,SE,_,_,_), SE > 0.

    :- not strumento_extra("elastic band"),
           strumento_extra("weights"),
           esegui(_,_,_,_,_,_,_,_,_,_,SE,_,_,_), SE < 0.
```

Ogni atom ***esegui()*** presenta un valore intero SE (supporto extra) positivo o negativo, se positivo significa che all'esercizio proposto verranno utilizzati pesi extra per aumentare il volume dell'allenamento, se negativo significa che verranno utilizzati bande elastiche per alleggerire l'esercizio. Verranno filtrati i vari answer set che contengono esercizi che richiedono pesi/bande in base al possesso del relativo strumento di supporto extra.

Ora passiamo al calcolo del tempo totale dell'allenamento per il determinato giorno che ci servirà per ottimizzare i tempi di allenamento in base al tempo disponibile dell'utente.

```
tempoTotale(DAY, TOT) :- T = #sum{DURATA_REP*REP*SERIE + DURATA_ISO*SERIE +
TR*SERIE, NE : esegui(NE,_,_,_,_,_,REP,SERIE,DURATA_REP,DURATA_ISO,_,_,TR,DAY)},
    disponibilita(DAY, _),
    warmup_time(WT),
    TOT = T+WT.
```

Il ***tempo totale dell'allenamento*** è data dalla seguente formula:

$$T_{tot\ allenamento} = \sum_{i}^{n}(T_rep_i * N_rep_i * N_serie_i + T_iso_i * N_serie_i + T_riposo_i * N_serie_i)$$

dove:

- ***n***, numero totale degli atom "esegui()" in quel determinato answer set
- ***T_rep***, tempo di esecuzione di una ripetizione del determinato esercizio
- ***N_rep***, numero di ripetizioni del determinato esercizio
- ***N_serie***, numero di serie del determinato esercizio
- ***T_iso***, tempo di isometria del determinato esercizio
- ***T_riposo***, tempo di riposo tra una serie e l'altra

Ottenuto il tempo totale dell'allenamento, come prima ottimizzazione possiamo minimizzare la differenza tra il tempo disponibile dell'utente e il tempo totale dell'allenamento.
In altre parole, lo schedulatore sceglierà l'answer set che si avvicina di più ai tempi disponibili dati in input dall'utente.

```
#minimize { |TEMPO_DISP-TOT|@3,DAY : tempoTotale(DAY, TOT), disponibilita(DAY,
TEMPO_DISP) }.
```

Un'ottimizzazione secondaria è la massimizzazione degli esercizi mirati ad allenare il gruppo muscolare deciso dall'utente che viene presentato sottostante.

```
mediaFU_gm(MEDIA) :- pref_gm("parte_superiore"),
                  SUM = #sum{FU_SUP, NOME_ESERCIZIO, DAY :
                  esegui(NOME_ESERCIZIO,FU_SUP,_,_,_,_,_,_,_,_,_,_,_,DAY)},
                  countEs(CE),
                  MEDIA = SUM/CE,
                  CE != 0.

mediaFU_gm(MEDIA) :- pref_gm("core"),
                  SUM = #sum{FU_CORE, NOME_ESERCIZIO, DAY :
                  esegui(NOME_ESERCIZIO,_,FU_CORE,_,_,_,_,_,_,_,_,_,DAY)},
                  countEs(CE),
                  MEDIA = SUM/CE,
                  CE != 0.

mediaFU_gm(MEDIA) :- pref_gm("parte_inferiore"),
                  SUM = #sum{FU_INF, NOME_ESERCIZIO, DAY :
                  esegui(NOME_ESERCIZIO,_,_,FU_INF,_,_,_,_,_,_,_,_,DAY)},
                  countEs(CE),
                  MEDIA = SUM/CE,
                  CE != 0.

#maximize { MEDIA_FU@2 : mediaFU_gm(MEDIA_FU) }.
```

Per prima cosa produrremo l'atom ***mediaFU_gm(MEDIA)*** in base a ***pref_gm(nomeGruppoMuscolare)*** selezionato dall'utente e calcoliamo la media del fattore utilità per il gruppo muscolare. Successivamente si cerca di massimizzare il suo valore e scegliere l'answer set con la media più alta.

Come ultima preferenza, si cerca di ottimizzare la varietà degli esercizi.

```
:~ CE_NomeEs = #count{ NOME_ES : esegui(NOME_ES,_,_,_,_,_,_,_,_,_,_,_,_,_,_) }.
                                                                    [ -CE_NomeEs@1 ]
```

Quindi tramite questa weak constraint contiamo i nomi degli esercizi univoci e ne massimizziamo il conteggio.

Interpretazione Output

Ottenuta la soluzione dal solver, l'output presenta un formato poco leggibile per l'utente, di conseguenza è stata implementata un interprete che legge la soluzione e la rielabora in un formato più accessibile.

Di seguito viene mostrato uno screenshot del risultato prodotto dal solver.

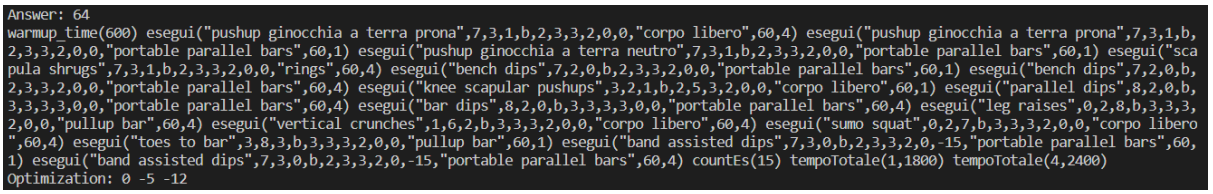


Fig. 25 - Output di clingo

Successivamente tramite lo script *conversione-output.py*, analizza la miglior soluzione prodotta, esporta un file chiamato *workout-plan.json* per interfacciarsi al sistema di *movement analysis* e ad eventuali componenti aggiuntivi.
Lo screenshot sottostante rappresenta un esempio di lettura del file *json* prodotto e stampa il piano di allenamento in un formato leggibile dall'utente finale.

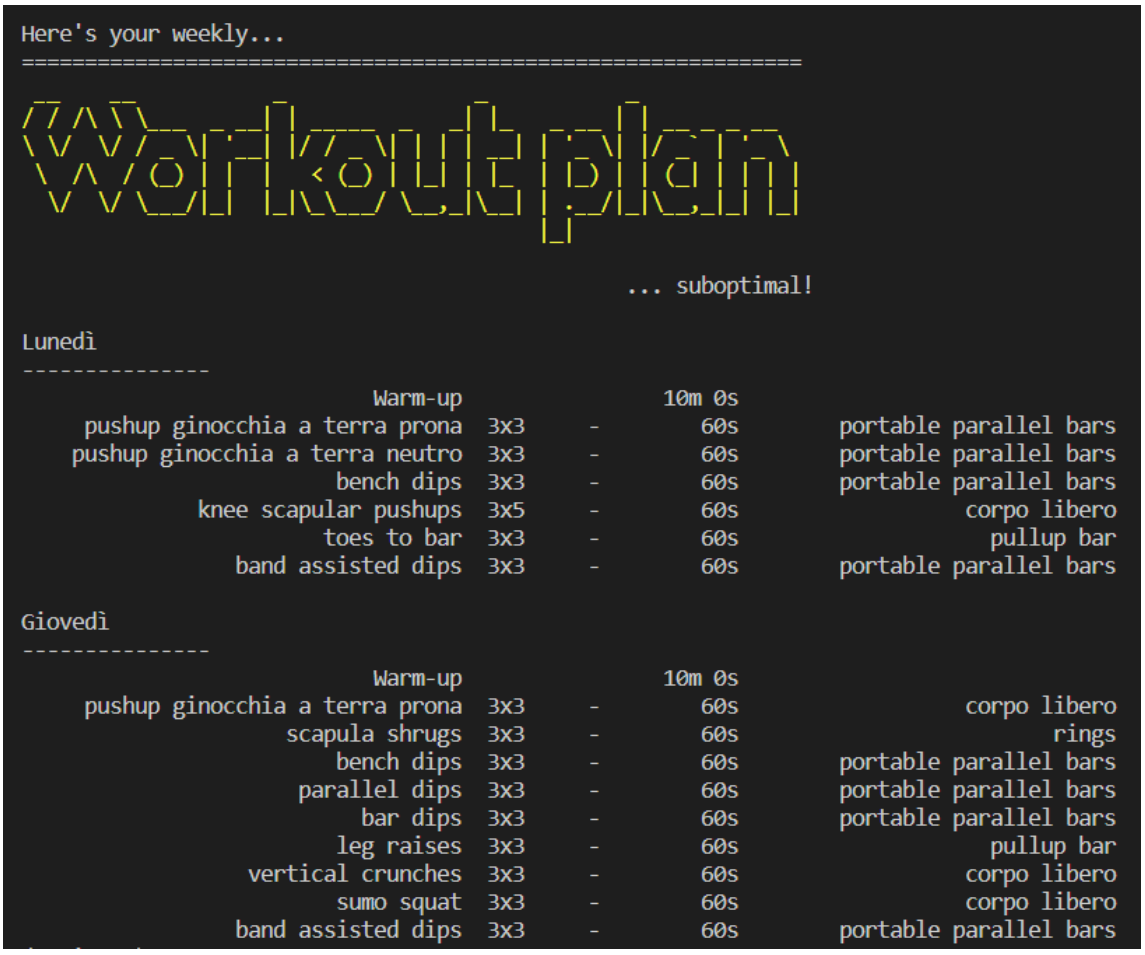


Fig. 26 - Lettura json e stampa della soluzione in un formato leggibile

Risultati

Movement analysis

In questa sezione di movement analysis vengono presentati i risultati e una demo.

Performance modelli

In questa sezione vengono presentati i risultati di ogni modello e viene dettagliata la scelta del migliore.
Nelle figure sono presenti le metriche dell'ultimo *K-fold* della *nested cross validation* e nell'ultima riga vengono riportate la media e la deviazione standard degli *f1-score* di tutti i *K-fold*.

Decision Tree

```
[I 2022-05-30 10:31:44,063] Trial 147 finished with value: 0.04688146711676122 and parameters: {'max_depth': 47, 'mi
n_samples_split': 39, 'max_leaf_nodes': 56, 'criterion': 'gini'}. Best is trial 0 with value: 0.04688146711676122.
f1 weighted per cross val 0.9531185328832388
[I 2022-05-30 10:31:44,096] Trial 148 finished with value: 0.04688146711676122 and parameters: {'max_depth': 33, 'mi
n_samples_split': 58, 'max_leaf_nodes': 50, 'criterion': 'gini'}. Best is trial 0 with value: 0.04688146711676122.
f1 weighted per cross val 0.9531185328832388
[I 2022-05-30 10:31:44,128] Trial 149 finished with value: 0.04688146711676122 and parameters: {'max_depth': 38, 'mi
n_samples_split': 41, 'max_leaf_nodes': 48, 'criterion': 'gini'}. Best is trial 0 with value: 0.04688146711676122.

{'max_depth': 10, 'min_samples_split': 55, 'max_leaf_nodes': 6, 'criterion': 'entropy'}

-------- DECISION TREE --------

Recall score:           [0.84615385 1.         1.         0.875     ]  weighted average:       0.918918918918919
Precision score:        [0.91666667 0.9        1.         0.875     ]  weighted average:       0.9193693693693693
F1 score:               [0.88       0.94736842 1.         0.875     ]  weighted average:       0.91800853485064

--- CONFUSION MATRIX ---

metrics.accuracy - Testing score:       0.918918918918919
Numero campioni di test: 37   Campioni erroneamente classificati: 3   Campioni classificati nulli: 0

                alzateLaterali0  alzateLaterali1  alzateLaterali2  alzateLaterali3
alzateLaterali0               11                1                0                1
alzateLaterali1                0                9                0                0
alzateLaterali2                0                0                7                0
alzateLaterali3                1                0                0                7

DT nested cross validation f-1 score: 0.939 with standard deviation: 0.049
```

Fig. 27 - Stampa delle metriche per il modello Decision Tree

Random Forest

```
[I 2022-05-30 11:20:01,661] Trial 147 finished with value: 0.027198927434221587 and parameters: {'max_depth': 16,
 'min_samples_leaf': 2, 'n_estimators': 205}. Best is trial 14 with value: 0.027198927434221587.
f1 weighted per cross val 0.9656819172113289
[I 2022-05-30 11:20:03,891] Trial 148 finished with value: 0.0343180827886711 and parameters: {'max_depth': 15, 'm
'min_samples_leaf': 3, 'n_estimators': 308}. Best is trial 14 with value: 0.027198927434221587.
f1 weighted per cross val 0.9660970576264694
[I 2022-05-30 11:20:05,272] Trial 149 finished with value: 0.03390294237353064 and parameters: {'max_depth': 17, '
'min_samples_leaf': 1, 'n_estimators': 172}. Best is trial 14 with value: 0.027198927434221587.

{'max_depth': 18, 'min_samples_split': 2, 'min_samples_leaf': 1, 'n_estimators': 270}

-------- RANDOM FOREST --------

Recall score:           [0.84615385 1.         1.         0.875     ]  weighted average:       0.918918918918919
Precision score:        [1.         1.         1.         0.77777778]  weighted average:       0.9519519519519519
F1 score:               [0.91666667 1.         1.         0.82352941]  weighted average:       0.9325649178590354

--- CONFUSION MATRIX ---

metrics.accuracy - Testing score:        0.918918918918919
Numero campioni di test: 37   Campioni erroneamente classificati: 3   Campioni classificati nulli: 1

                alzateLaterali0  alzateLaterali1  alzateLaterali2  alzateLaterali3
alzateLaterali0               11                0                0                2
alzateLaterali1                0                9                0                0
alzateLaterali2                0                0                7                0
alzateLaterali3                0                0                0                7

RF nested cross validation f-1 score: 0.972 with standard deviation: 0.049
```

Fig. 28 - Stampa delle metriche per il modello Random Forest

Support Vector Machine

```
[I 2022-05-30 10:23:49,903] Trial 146 finished with value: 0.041000000000000036 and parameters: {'kernel': 'poly',
, 'degree': 2}. Best is trial 13 with value: 0.041000000000000036.
f1 weighted per cross val 0.959
[I 2022-05-30 10:23:52,645] Trial 147 finished with value: 0.041000000000000036 and parameters: {'kernel': 'poly',
, 'degree': 2}. Best is trial 13 with value: 0.041000000000000036.
f1 weighted per cross val 0.959
[I 2022-05-30 10:23:55,564] Trial 148 finished with value: 0.041000000000000036 and parameters: {'kernel': 'poly',
, 'degree': 2}. Best is trial 13 with value: 0.041000000000000036.
{'kernel': 'poly', 'C': 0.7081329639967373, 'gamma': 'auto', 'degree': 2}
0.959

-------- SVM --------

Recall score:           [0.92307692 1.         1.         1.        ]  weighted average:       0.972972972972973
Precision score:        [1.         1.         1.         0.88888889]  weighted average:       0.9759759759759761
F1 score:               [0.96       1.         1.         0.94117647]  weighted average:       0.9732273449920509

--- CONFUSION MATRIX ---

metrics.accuracy - Testing score:        0.972972972972973
Numero campioni di test: 37   Campioni erroneamente classificati: 1   Campioni classificati nulli: 0

                alzateLaterali0  alzateLaterali1  alzateLaterali2  alzateLaterali3
alzateLaterali0               12                0                0                1
alzateLaterali1                0                9                0                0
alzateLaterali2                0                0                7                0
alzateLaterali3                0                0                0                8

SVM nested cross validation f-1 score: 0.939 with standard deviation: 0.065
```

Fig. 29 - Stampa delle metriche per il modello Support Vector Machine

Gradient Boosting

```
[I 2022-05-30 18:03:43,614] Trial 148 finished with value: 0.033400509821562396 and parameters: {'max_depth': 10, 'l
42414, 'n_estimators': 1379, 'subsample': 0.34972562164963705}. Best is trial 38 with value: 0.033400509821562396.
f1 weighted per cross val 0.9537454092748211
[I 2022-05-30 18:03:56,956] Trial 149 finished with value: 0.0462545907251789 and parameters: {'max_depth': 2, 'lear
917, 'n_estimators': 1200, 'subsample': 0.5747650761436214}. Best is trial 38 with value: 0.033400509821562396.
{'max_depth': 10, 'learning_rate': 0.0001810210243655771, 'n_estimators': 512, 'subsample': 0.4830656479431243}
0.9665994901784376

-------- GRADIENT BOOSTING --------

Recall score:           [0.84615385 1.         1.         1.        ]  weighted average:       0.9459459459459459
Precision score:        [1.  1.  1.  0.8]      weighted average:       0.9567567567567568
F1 score:               [0.91666667 1.         1.         0.88888889]  weighted average:       0.9466966966966968

--- CONFUSION MATRIX ---

metrics.accuracy - Testing score:        0.9459459459459459
Numero campioni di test: 37   Campioni erroneamente classificati: 2   Campioni classificati nulli: 0

                alzateLaterali0  alzateLaterali1  alzateLaterali2  alzateLaterali3
alzateLaterali0               11                0                0                2
alzateLaterali1                0                9                0                0
alzateLaterali2                0                0                7                0
alzateLaterali3                0                0                0                8

GB nested cross validation f-1 score: 0.955 with standard deviation: 0.050
```

Fig. 30 - Stampa delle metriche per il modello Gradient Boosting

Extreme Gradient Boosting

```
[I 2022-05-30 13:09:09,436] Trial 148 finished with value: 0.039507115977704066 and parameters: {'max_depth': 7, 'le
arning_rate': 0.005123547170779848, 'n_estimators': 145, 'min_child_weight': 3, 'colsample_bytree': 0.22901793815014
443, 'subsample': 0.5199280515962258, 'reg_alpha': 0.007309324202320469, 'reg_lambda': 0.04125553809969833, 'gamma':
 15.212467411677094}. Best is trial 90 with value: 0.026580191050779356.
f1 weighted per cross val 0.959933551198257
[I 2022-05-30 13:09:16,058] Trial 149 finished with value: 0.04006644880174304 and parameters: {'max_depth': 7, 'lea
rning_rate': 0.011451740678253396, 'n_estimators': 1086, 'min_child_weight': 3, 'colsample_bytree': 0.20032877104777
383, 'subsample': 0.5533521465964146, 'reg_alpha': 0.004426659141367412, 'reg_lambda': 0.0005966450385435569, 'gamma
': 11.266510195857103}. Best is trial 90 with value: 0.026580191050779356.
{'max_depth': 7, 'learning_rate': 0.0019337382483331405, 'n_estimators': 423, 'min_child_weight': 2, 'colsample_bytr
ee': 0.30775257010390655, 'subsample': 0.2925894240256481, 'reg_alpha': 0.004386560955826888, 'reg_lambda': 0.014735
704358053003, 'gamma': 15.321630531526198}
0.9734198089492206

-------- EXTREME GRADIENT BOOSTING --------

Recall score:           [0.69230769 1.         1.         1.        ]  weighted average:       0.8918918918918919
Precision score:        [1.         1.         1.         0.66666667]  weighted average:       0.927927927927928
F1 score:               [0.81818182 1.         1.         0.8       ]  weighted average:       0.8928746928746928

--- CONFUSION MATRIX ---

metrics.accuracy - Testing score:        0.8918918918918919
Numero campioni di test: 37   Campioni erroneamente classificati: 4   Campioni classificati nulli: 0

                alzateLaterali0  alzateLaterali1  alzateLaterali2  alzateLaterali3
alzateLaterali0               9                0                0                4
alzateLaterali1               0                9                0                0
alzateLaterali2               0                0                7                0
alzateLaterali3               0                0                0                8

XGB nested cross validation f-1 score: 0.946 with standard deviation: 0.067
```

Fig. 31 - Stampa delle metriche per il modello Extreme Gradient Boosting

Come si vede dai risultati, il modello con la media di *f1 score* (derivati dalla *nested cross validation*) più alta risulta il *Random Forest*, con valore 0,972 e una bassa deviazione standard che vale 0,049 (cioè indica una variabilità del *f1 score* molto bassa).

Demo

Di seguito riportiamo una demo, in cui consiste nell'eseguire 5 ripetizioni di *alzate laterali*:

- la prima ripetizione eseguita con braccia piegate
- la seconda eseguita in modo asimmetrica
- la terza che non raggiunge i 90°, quindi ad altezza clavicola
- la quarta eseguita correttamente
- la quinta eseguita correttamente

Avviato il programma, inizia a localizzare la persona e controllare se è nella *posizione di partenza*.

Fig. 32 - Posizione di partenza

Come si può vedere dalla figura, il sistema ha rilevato il corpo della persona e verificato che è in posizione di partenza pronto per iniziare l'esecuzione del movimento. In cima a sinistra della schermata abbiamo una linea guida che si aggiorna in tempo reale durante le varie transizioni tra stati.

Ora si passa all'esecuzione del movimento con braccia piegate.

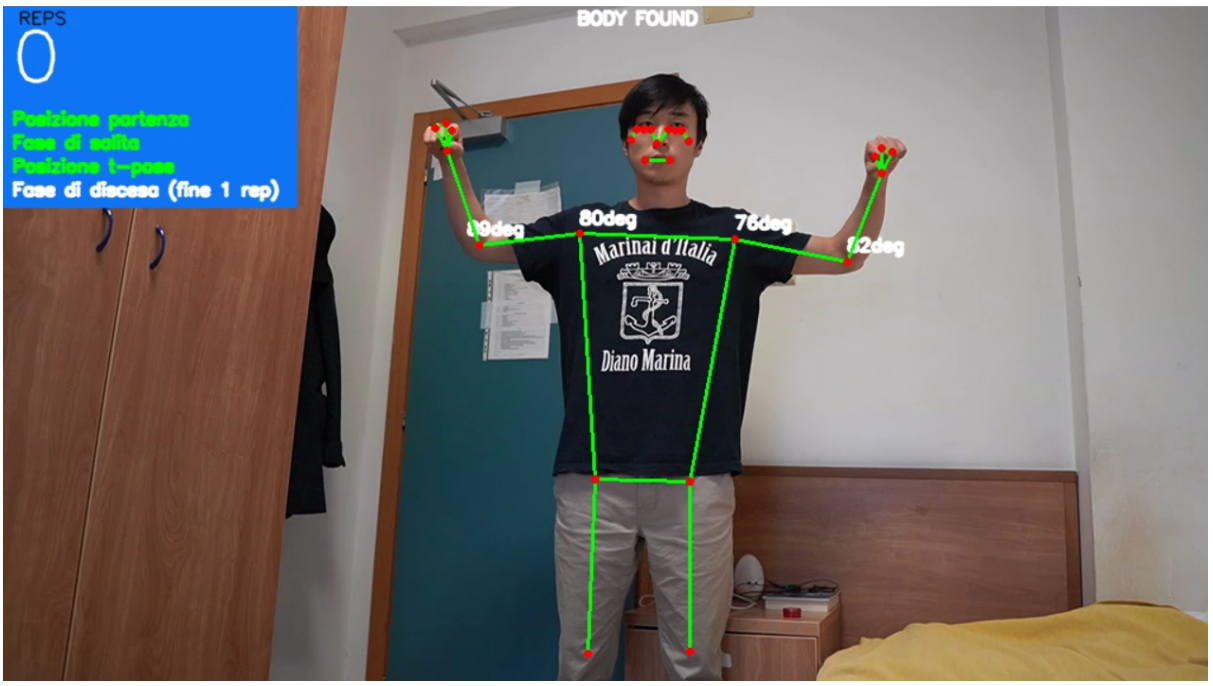


Fig. 33 - Esecuzione errata di alzate laterali con braccia piegate

Come si vede dalla linea guida, il sistema ha rilevato la transizione per la *fase di salita* e il raggiungimento della posizione *t-pose*. Ad ogni transizione di stato viene prodotto un suono acustico, così da poter enfatizzare l'interazione e la trasmissione di informazioni su più canali (vista e udito).

Ritornando alla posizione iniziale, vengono resettati i vari stati.

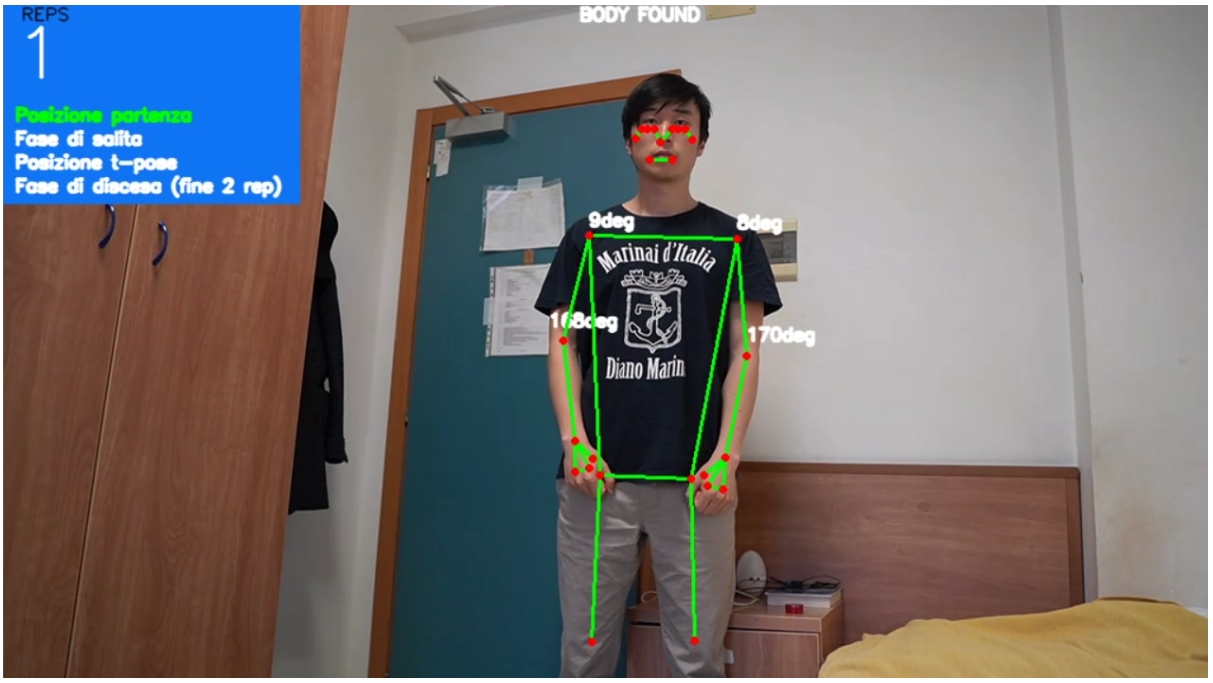


Fig. 34 - Ritorno alla posizione di partenza, fine prima ripetizione

Possiamo notare che il sistema ha rilevato il “ritorno” alla posizione iniziale quindi considerando il movimento come 1 ripetizione.

Successivamente procediamo con l’eseguire la seconda ripetizione dove le braccia salgono in modo asimmetrico.

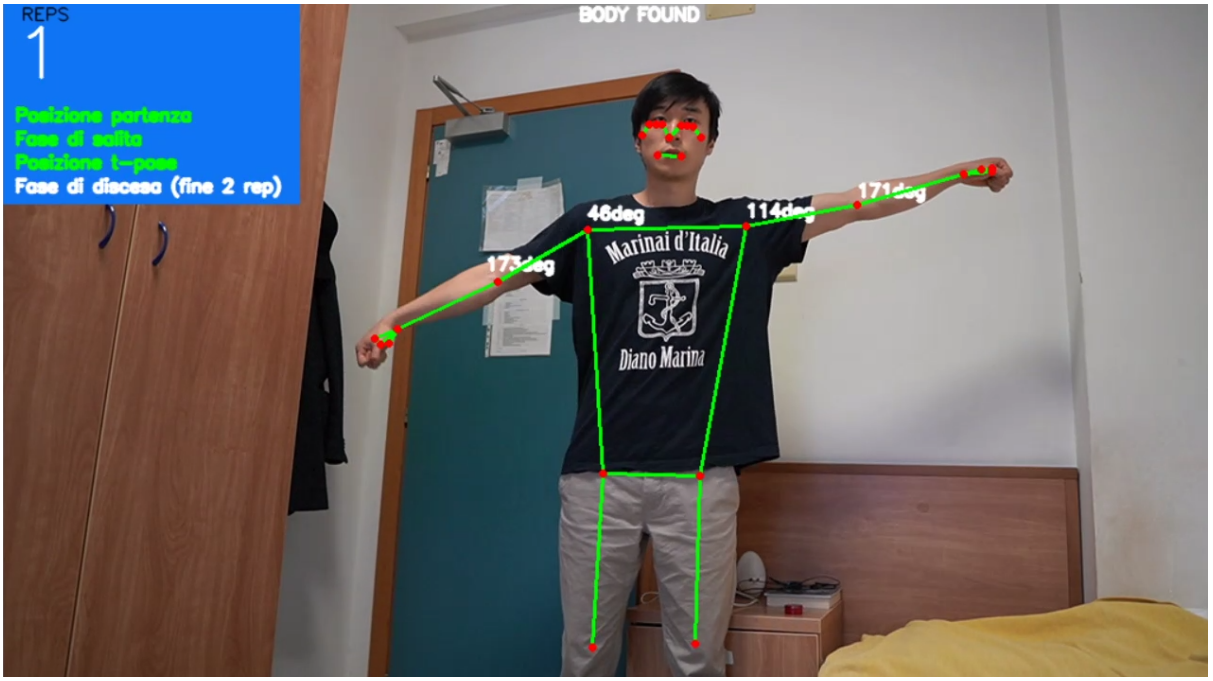


Fig. 35 - Esecuzione errata delle alzate laterali in modo asimmetrico

Effettuata l’*eccentrica* del movimento, il sistema rileva il “ritorno” alla posizione iniziale quindi considerando il movimento completato e conta 1 ripetizione aggiuntiva.

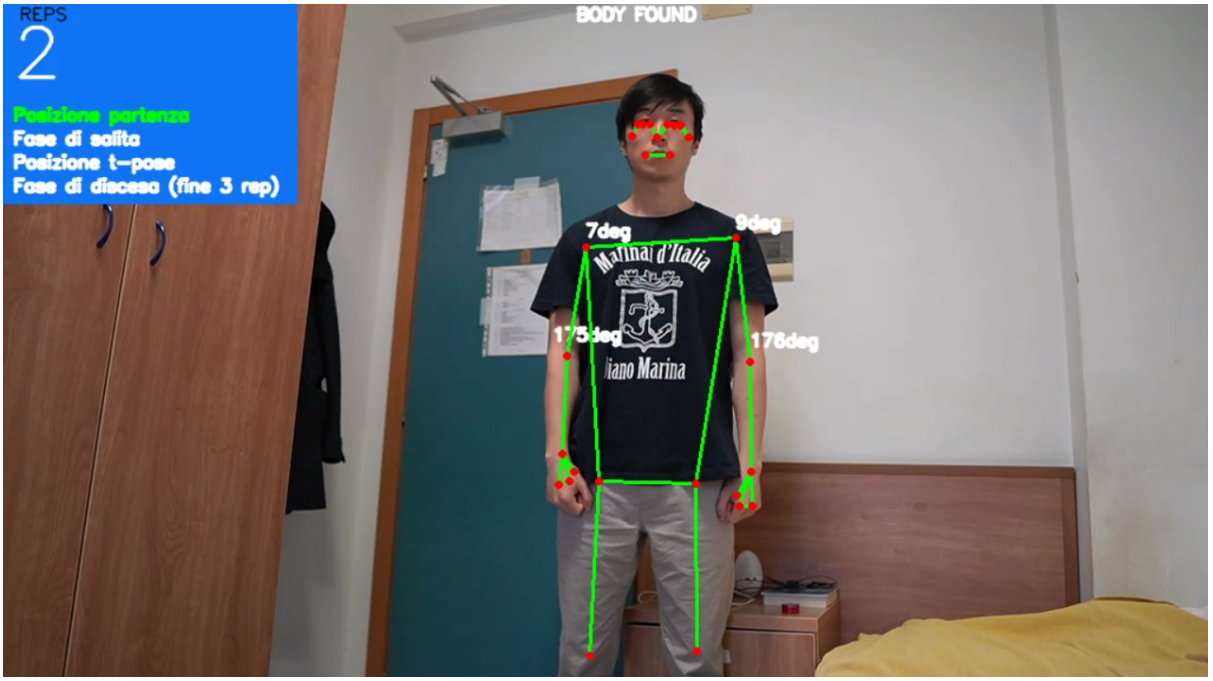


Fig. 36 - Ritorno alla posizione di partenza, fine seconda ripetizione

Successivamente lo si effettua per la terza ripetizione con braccia che non arrivano a 90°.

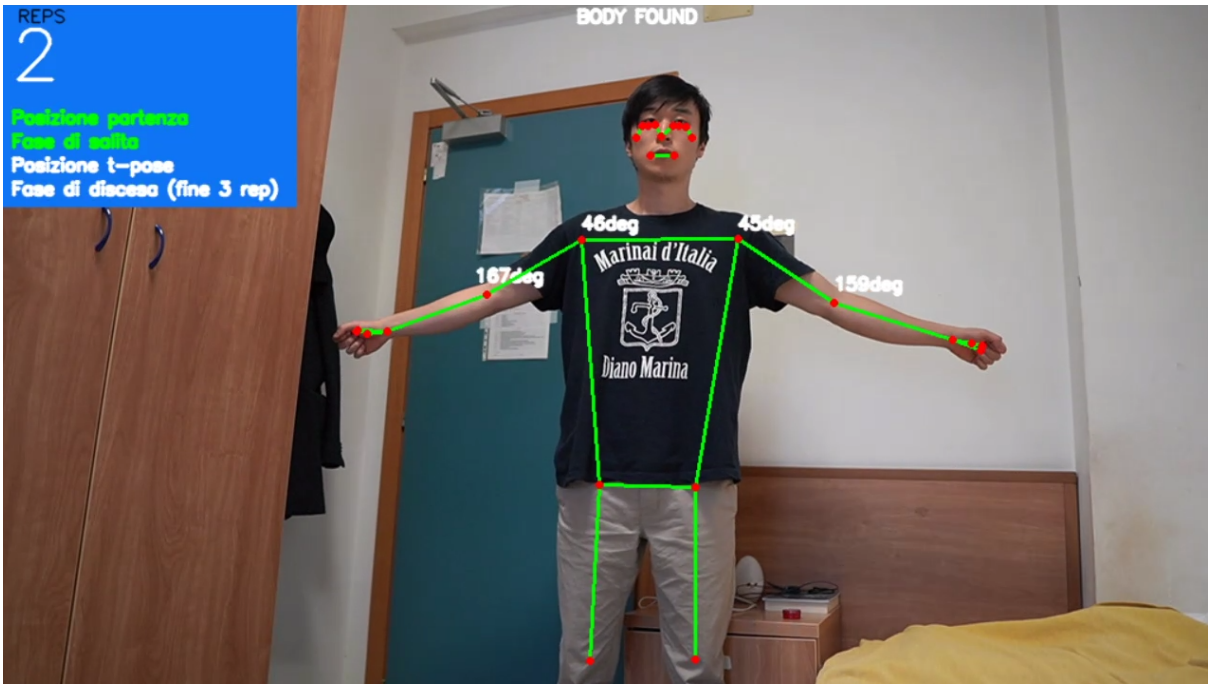


Fig. 37 - Esecuzione errata delle alzate laterali senza raggiungere i 90°

Procediamo con la quarta e quinta ripetizione in cui si eseguono i movimenti correttamente.

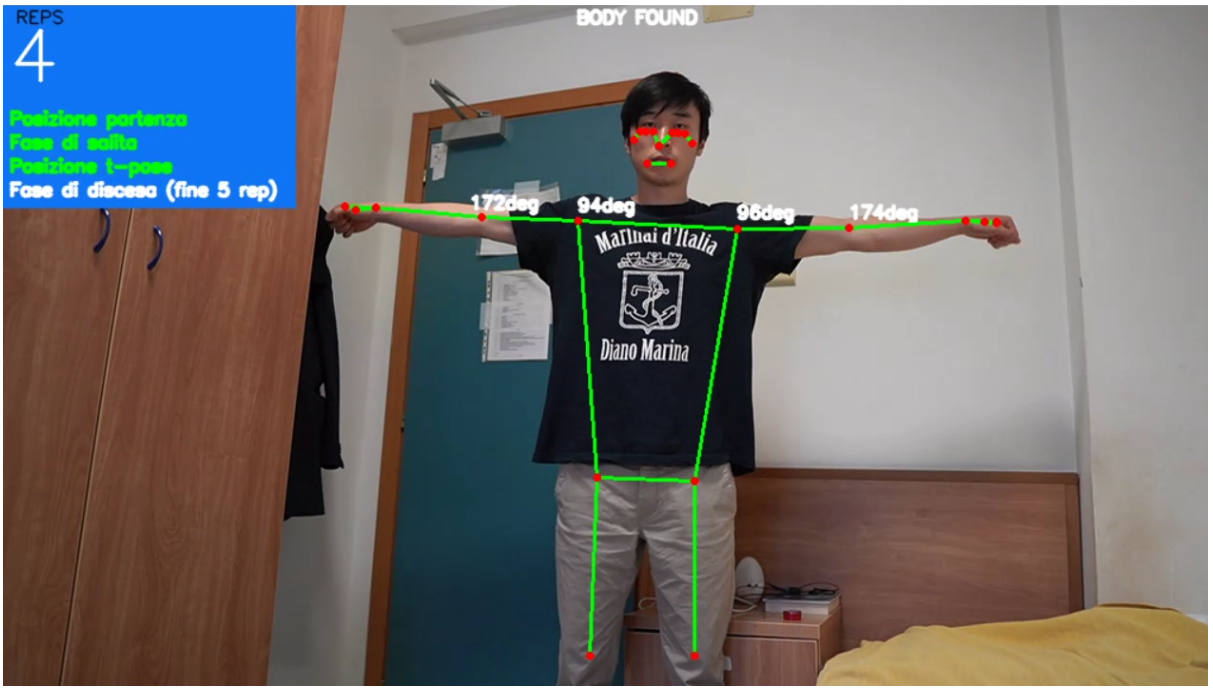


Fig. 38 - Esecuzione corretta delle alzate laterali

Conclusa questa serie di ripetizioni, il sistema elabora i risultati e produce un audio *text-to-speech* in cui riferisce all'utente della performance compiuta e li stampa a video.

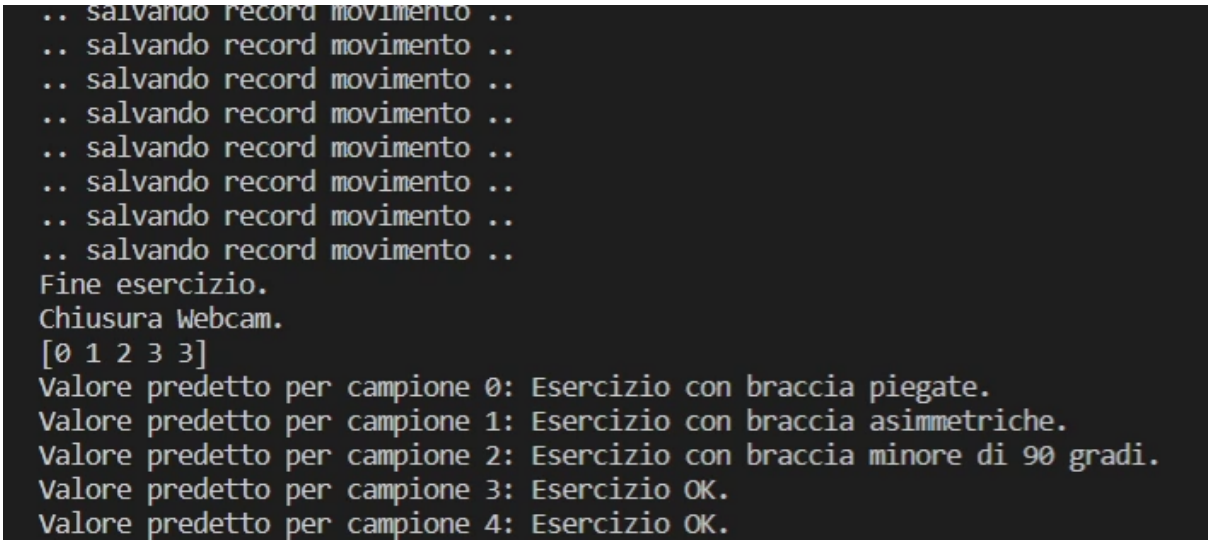


Fig. 39 - Stampa delle predizioni effettuate dal modello

Come possiamo vedere, il sistema ha classificato i vari movimenti con successo.

Planning

In questa sezione vengono presentati i tempi impiegati per generare delle soluzioni in base ai requisiti dell'utente.
Vengono testati vari tipi di scenari utilizzando un processore *Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz*.

Scenario B1 - 1

- Livello utente **B1**
- Disponibilità **2** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

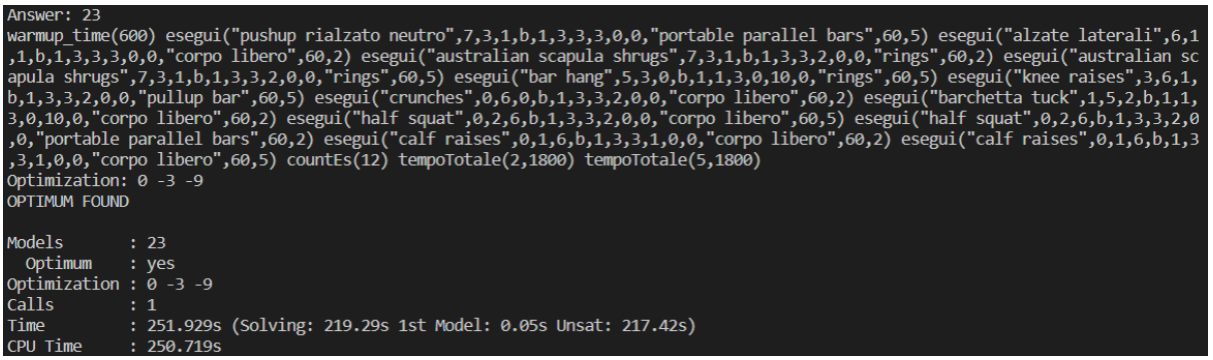


Fig. 40 - Risultato scenario B1 con 2 giorni di allenamento

In questo scenario, sono stati specificati i vari input dell'utente. Essendo livello B1, quindi un livello introduttivo per principianti, lo spazio di ricerca è abbastanza limitato ad esercizi basilari. La disponibilità è di 2 giorni alla settimana con durata di 30 minuti (1800 secondi). Accesso a tutti gli strumenti disponibili, quindi l'utente è in possesso di tutti gli strumenti per poter eseguire il determinato esercizio proposto. Preferenza gruppo muscolare parte superiore, quindi la soluzione che si ottiene, avrà la media del *fattore_utilità_parte_superiore* massimizzata.

La soluzione indica che sono state trovate 23 answer set e che l'ultimo individuato è l'**ottima** e presenta i seguenti valori ottimizzati:

- 0, somma dei moduli delle differenze tra tempo disponibile dato per ogni giorno e la durata dell'allenamento complessiva per ogni giorno, espressa in secondi .
- -3, media del *fattore_utilità_parte_superiore* individuata più alta (in negativo perché si cerca di massimizzare), ricercata tra le soluzioni che presentano 0 del valore precedentemente descritto.
- -9, numeri degli esercizi univoci, quindi massimizzare la varietà degli esercizi tra le soluzioni che presentano i migliori valori delle ottimizzazioni precedenti(0 e -3).

Il tempo impiegato dal solver per trovare la soluzione è di 251 secondi (4 minuti e 11 secondi).

Scenario B1 - 2

- Livello utente B1
- Disponibilità **3** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 16+
  Optimum    : unknown
Optimization : 0 -3 -10
Calls        : 1
Time         : 4923.771s (Solving: 4867.24s 1st Model: 0.13s Unsat: 0.00s)
CPU Time     : 4911.016s
```

Fig. 41 - Ricerca se si trovano soluzioni migliori per scenario B1 con 3 giorni

```
Models       : 16+
  Optimum    : unknown
Optimization : 0 -3 -10
Calls        : 1
Time         : 111.923s (Solving: 17.23s 1st Model: 0.23s Unsat: 0.00s)
CPU Time     : 109.063s
```

Fig. 42 - Risultato scenario B1 con 3 giorni di allenamento

In questo scenario, simile a quello precedente, è stato aggiunto un giorno in più di allenamento da 30 minuti, quindi è aumentato lo spazio di ricerca.

La ricerca è stata interrotta a 4923 secondi (1 ora e 22 minuti), individuando 16 answer set, in quanto fa fatica a trovare l'ottimo.
Si è ripetuto lo stesso test interrompendo nel momento in cui trova lo stesso risultato precedente (sub-ottimo) con i valori ottimizzati a 0, -3, -10, che come viene raffigurato è di 111 secondi (1 minuto e 51 secondi).

Scenario B1 - 3

- Livello utente B1
- Disponibilità **4** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 25+
  Optimum    : unknown
Optimization : 0 -3 -11
Calls        : 1
Time         : 1574.068s (Solving: 1452.68s 1st Model: 0.25s Unsat: 0.00s)
CPU Time     : 1561.328s
```

Fig. 43 - Ricerca se si trovano soluzioni migliori per scenario B1 con 4 giorni

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 25+
  Optimum    : unknown
Optimization : 0 -3 -11
Calls        : 1
Time         : 217.230s (Solving: 94.95s 1st Model: 0.41s Unsat: 0.00s)
CPU Time     : 214.984s
```

Fig. 44 - Risultato per scenario B1 con 4 giorni

In questo terzo scenario, si sono ripetuti gli stessi passaggi, ma impostando 4 giorni di allenamento da 30 minuti.
Il primo test ha individuato 25 modelli sub-ottimi ed è stato interrotto a 1574 secondi.

La soluzione sub-ottima è stata calcolata in 217 secondi (3 minuti e 37 secondi).

Scenario B2 - 1

- Livello utente **B2**
- Disponibilità **2** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 81+
  Optimum    : unknown
Optimization : 0 -5 -11
Calls        : 1
Time         : 3027.469s (Solving: 2986.01s 1st Model: 1.90s Unsat: 0.00s)
CPU Time     : 3015.469s
```

Fig. 45 - Ricerca se si trovano soluzioni migliori per scenario B2 con 2 giorni

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 81+
  Optimum    : unknown
Optimization : 0 -5 -11
Calls        : 1
Time         : 630.622s (Solving: 596.37s 1st Model: 1.31s Unsat: 0.00s)
CPU Time     : 623.734s
```

Fig. 46 - Risultato per scenario B2 con 2 giorni

In questo nuovo scenario, passiamo al livello B2, quindi espandendo la lista degli esercizi proposti. Il solver effettuerà i vari controlli e prenderà esercizi del livello B2 e quello precedente, quindi anche B1 (come stabilito nel *problem encoding*).

Il primo test ha individuato 81 modelli sub-ottimi ed è stato interrotto a 3027 secondi.

Si trova la soluzione sub-ottima a 630 secondi (10 minuti e 30 secondi).

Scenario B2 - 2

- Livello utente B2
- Disponibilità **3** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 55+
  Optimum    : unknown
Optimization : 0 -3 -16
Calls        : 1
Time         : 4594.042s (Solving: 4485.01s 1st Model: 1.70s Unsat: 0.00s)
CPU Time     : 4412.828s
```

Fig. 47 - Ricerca se si trovano soluzioni migliori per scenario B2 con 3 giorni

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 55+
  Optimum    : unknown
Optimization : 0 -3 -16
Calls        : 1
Time         : 1212.584s (Solving: 1156.05s 1st Model: 1.26s Unsat: 0.00s)
CPU Time     : 1207.531s
```

Fig. 48 - Risultato per scenario B2 con 3 giorni

Questo scenario, simile a quello precedente, ma con 3 giorni di allenamento da 30 minuti.

Il primo test ha individuato 55 modelli sub-ottimi ed è stato interrotto a 4594 secondi.

Trova la soluzione sub-ottima a 1212 secondi (20 minuti e 12 secondi).

Scenario B2 - 3

- Livello utente B2
- Disponibilità **4** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

```
SATISFIABLE

TIME LIMIT   : 1
Models       : 49+
  Optimum    : unknown
Optimization : 27 -4 -17
Calls        : 1
Time         : 3000.017s (Solving: 2867.30s 1st Model: 43.23s Unsat: 0.00s)
CPU Time     : 2976.094s
```

Fig. 49 - Ricerca se si trovano soluzioni migliori per scenario B2 con 4 giorni

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 49+
  Optimum    : unknown
Optimization : 27 -4 -17
Calls        : 1
Time         : 1082.229s (Solving: 939.04s 1st Model: 33.26s Unsat: 0.00s)
CPU Time     : 1076.344s
```

Fig. 50 - Risultato per scenario B2 con 4 giorni

Scenario simile a quello precedente, ma con 4 giorni di allenamento da 30 minuti.

Il primo test ha individuato 49 modelli sub-ottimi ed è stato interrotto a 3000 secondi.

Trova la soluzione sub-ottima a 1082 secondi (18 minuti e 2 secondi).

Scenario B3 - 1

- Livello utente **B3**
- Disponibilità **2** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 138+
  Optimum    : unknown
Optimization : 0 -5 -12
Calls        : 1
Time         : 4829.932s (Solving: 4750.35s 1st Model: 2.94s Unsat: 0.00s)
CPU Time     : 4777.406s
```

Fig. 51 - Ricerca se si trovano soluzioni migliori per scenario B3 con 2 giorni

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 138+
  Optimum    : unknown
Optimization : 0 -5 -12
Calls        : 1
Time         : 3356.570s (Solving: 3293.83s 1st Model: 2.26s Unsat: 0.00s)
CPU Time     : 3332.406s
```

Fig. 52 - Risultato per scenario B3 con 2 giorni

In questo scenario, passiamo al livello B3, quindi troveremo esercizi sia di livello B3 che del livello precedente B2.

Il primo test ha individuato 138 modelli sub-ottimi ed è stato interrotto a 4829 secondi.

Trova la soluzione sub-ottima a 3356 secondi (55 minuti e 56 secondi).

Scenario B3 - 2

- Livello utente B3
- Disponibilità **3** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 51+
  Optimum    : unknown
Optimization : 0 -5 -13
Calls        : 1
Time         : 6459.265s (Solving: 6359.37s 1st Model: 2.05s Unsat: 0.00s)
CPU Time     : 6449.141s
```

Fig. 53 - Ricerca se si trovano soluzioni migliori per scenario B3 con 3 giorni

```
SATISFIABLE

INTERRUPTED  : 1
Models       : 51+
  Optimum    : unknown
Optimization : 0 -5 -13
Calls        : 1
Time         : 5504.625s (Solving: 5406.06s 1st Model: 2.04s Unsat: 0.00s)
CPU Time     : 5441.625s
```

Fig. 54 - Risultato per scenario B3 con 3 giorni

Il primo test ha individuato 51 modelli sub-ottimi ed è stato interrotto a 6459 secondi.

Trova la soluzione in 5504 secondi (31 minuti e 44 secondi).

Scenario B3 - 3

- Livello utente B3
- Disponibilità **4** giorni da 30 minuti
- Accessibile a tutti gli strumenti disponibili
- Preferenza gruppo muscolare parte superiore

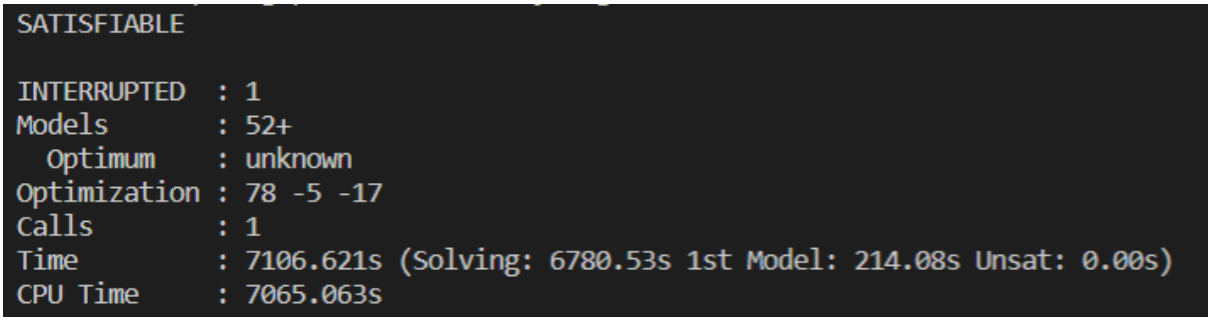


Fig. 55 - Risultato per scenario B3 con 4 giorni

In questo ultimo scenario è stato effettuato un unico test per semplicità. Si nota che la soluzione trovata è sub-ottimale con 78 secondi di margine ed è accettabile in quanto significa sforare di 78 secondi distribuito su 4 giorni di allenamento.

Grafico riassuntivo performance

Questa tabella riassume i tempi impiegati dal solver per trovare una soluzione che si avvicina di più alle richieste date in input.

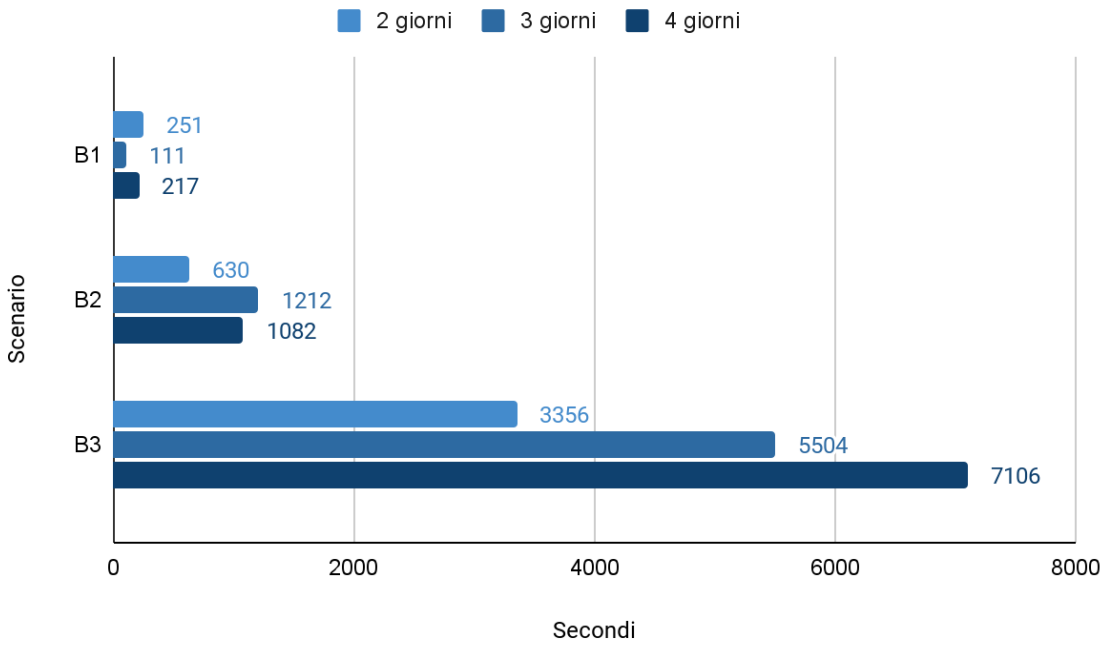


Fig. 56 - Grafico riassuntivo per ogni scenario

Si può subito notare che all'aumentare del livello, aumentano anche i tempi di calcolo. Questo perché aumenta di molto lo spazio di ricerca prodotto da nuove combinazioni di esercizi in base ai loro fattori di utilità, durate di esecuzione e riposo, strumento, per ogni giorno di allenamento disponibile.

Nello scenario B1 con 2 giorni di allenamento ha esaurito tutte le possibili combinazioni nello spazio di ricerca impiegando 251 secondi. Le due successive, con 3 e 4 giorni, sono state trovate soluzioni sub-ottime rispettando i requisiti, in quanto il solver non aveva ancora esaurito lo spazio di ricerca ma i risultati erano già soddisfacenti.

Nello scenario B2 troviamo 630 secondi per 2 giorni, 1212 secondi per 3 giorni e 1082 secondi per 4 giorni. Si nota che per 4 giorni è risultato più veloce perché è stato interrotto prima producendo con valori 27 -4 -17, quindi trovando un sub-ottimo che comunque soddisfa i requisiti sforando di 27 secondi suddiviso su 4 giorni. Si è comunque effettuato 3000 secondi di test per valutare se la soluzione potesse migliorare e come presentato dallo screenshot, il risultato è invariato.

Nello scenario B3 presenta risultati con lo stesso andamento, quindi all'aumentare del livello, aumentano la lista degli esercizi esistenti e la complessità nello spazio di ricerca, conseguentemente un aumento del tempo richiesto per individuare una soluzione.

Conclusione

Il progetto è nato con l'idea di insegnare e correggere gli esercizi del *calisthenics*, ma data la scarsità di dataset, si è semplificato ed effettuato l'analisi su un unico semplice movimento delle alzate laterali delle braccia.

Si prospetta di estendere il progetto per movimenti più complessi, ma basilari per la disciplina, come i push-ups, pull-ups, dips, squats ed eventualmente evolvere in skills come muscle up, handstand, front lever, planche…

Il nostro progetto è molto versatile e di conseguenza applicabile in diversi campi, per esempio modificando i modelli di machine learning e gli esercizi da svolgere, si può utilizzare come supporto alla riabilitazione per la persona [29].

Future works

Attualmente nell'implementazione della parte di planning, sono presenti fattori utilità generici suddivisi in 3 sezioni del corpo, parte superiore, core e parte inferiore. In futuro si potrebbero implementare fattori di utilità più specifici arto per arto o muscolo per muscolo, utile dal punto di vista riabilitativo.

Un'altra idea per migliorare il progetto è di integrare con dispositivi esterni wearable per valutare il rischio di infortuni [25] e ottenere misurazioni dell'intensità degli allenamenti che influenzeranno al meglio gli allenamenti della settimana successiva. Attualmente non avendo a disposizione questi dati, si controlla solo la forma dell'esecuzione ma per esempio non si possono sapere i battiti cardiaci che è un'informazione utile per stabilire lo "stato" della persona (se sta affaticando o no). Un altro valore che si potrebbe introdurre, ottenuto dal dispositivo wearable, è la percentuale di ossigeno nel sangue.

Si potrebbe realizzare un'interfaccia grafica in cui un esperto del settore può aggiungere personalmente esercizi in più alla lista degli esercizi proposti.

Implementazione di esercizi "ibridi" che contengono ripetizioni con isometrie.
Un esempio:

- salita pull-up (concentrica)
- mantenimento posizione per tot secondi (isometria)
- discesa pull-up (eccentrica)

Nei lavori futuri saranno presenti le analisi e produzione di dataset su più esercizi.

Un'idea molto interessante, questo progetto potrebbe essere il punto di partenza per la nascita di parchi fitness "tecnologici" pubblici ed incentivare i cittadini ad una vita più fit.

Ringraziamenti

Luca.

Vorrei dedicare questo spazio ai ringraziamenti di chi mi ha accompagnato e supportato in questo percorso universitario:
Ringrazio il mio collega Gabriele, compagno di mille progetti con il quale ho passato moltissime ore ad imparare, sviluppare e risolvere problemi.
Ringrazio i relatori prof. Gualtiero Volpe e prof. Marco Maratea per la pazienza e il tempo che ci hanno dedicato durante la realizzazione della tesi.
Ringrazio la mia famiglia per avermi supportato economicamente e in innumerevoli momenti psicologicamente complicati, dove mi hanno dato la forza di andare avanti e raggiungere questo bellissimo obiettivo.
Infine, ringrazio anche voi lettori di questo elaborato per aver dedicato il vostro tempo alla lettura di questa tesi.

Gabriele.

Questo mio percorso universitario è stato lungo e pieno di ostacoli, ma è il luogo dove ho imparato tante cose, sia dai successi che dagli sbagli, e dove ho conosciuto tante persone con cui ho condiviso momenti difficili da dimenticare e che vorrei ringraziare per il tempo dedicatomi.
Ringrazio il mio collega Luca che mi ha affiancato in molti progetti della magistrale, è in gamba e possiede ottime doti di problem solving che ci ha portato a superare i vari momenti difficili. Ringrazio il prof. Gualtiero Volpe e il prof. Marco Maratea per la disponibilità e pazienza dedicatoci per la realizzazione di questa tesi. Ringrazio i miei colleghi di università, Gerardo, Chenghao, Daniele, Ash, Alessio, Nicholas, Gianpiero e i miei amici di allenamento, Omi, Yulin, Olimpia, Fencky, Alessio, Michael per aver condiviso i momenti di gioia, sacrificio e successo insieme. Inoltre ringrazio Sofia per il cibo, Greta per i vari consigli e Laura per il supporto.

Infine, ringrazio profondamente la mia famiglia per avermi sempre supportato sia economicamente che psicologicamente, è grazie al loro sostegno e al loro incoraggiamento se oggi sono riuscito a raggiungere questo traguardo.

Poter unire le mie tre passioni, la tecnologia, la fotografia/videomaking, il calisthenics in questa tesi mi ha reso molto soddisfatto nel poter lavorare su qualcosa che veramente mi appassiona e sprono a voi lettori a fare ciò perché quando si fa qualcosa che piace e appassiona, non si smette mai di imparare e crescere.
Spero vivamente che questa tesi sia un punto di partenza per futuri progetti simili.

Detto ciò, si conclude la mia carriera universitaria e l'inizio di un nuovo capitolo della mia vita.

References

In quest'ultimo capitolo sono riportate le fonti che contengono alcune informazioni riportate nella tesi.

1. How to avoid exercise injuries: MedlinePlus Medical Encyclopedia
   https://medlineplus.gov/ency/patientinstructions/000859.htm

2. Can exercise extend your life? - Harvard Health
   https://www.health.harvard.edu/blog/can-exercise-extend-your-life-2019031316207

3. Exercise is key to living longer
   https://www.allinahealth.org/healthysetgo/move/exercise-is-key-to-living-longer

4. R. Niewiadomski, K. Kolykhalova, S. Piana, P. Alborno, G. Volpe, and A. Camurri, "Analysis of Movement Quality in Full-Body Physical Activities", ACM Transactions on Interactive Intelligent Systems, vol. 9, no. 1, article 1, 20 pages, 2019.
   https://dl.acm.org/doi/pdf/10.1145/3132369

5. Best Fitness and Exercise Apps
   https://www.healthline.com/health/fitness/top-iphone-android-apps#our-picks

6. Le 7 migliori App per il fitness del 2021
   https://www.gazzetta.it/fitness/02-03-2021/7-migliori-app-il-fitness-2021-400110574 5920.shtml

7. Nymoen, K. (2013). Methods and Technologies for Analysing Links Between Musical Sound and Body Motion. chapter 3.
   https://www.uio.no/studier/emner/hf/imv/MUS2006/v15/litteratur/knkap3-4.pdf

8. OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation
   https://github.com/CMU-Perceptual-Computing-Lab/openpose

9. Pose Detection
   https://github.com/tensorflow/tfjs-models/tree/master/pose-detection#example-code-and-demos

10. DensePose
    http://densepose.org/

11. GitHub - MVIG-SJTU/AlphaPose: Real-Time and Accurate Full-Body Multi-Person Pose Estimation&Tracking System
    https://github.com/MVIG-SJTU/AlphaPose

12. Deep High-Resolution Representation Learning for Human Pose Estimation
https://jingdongwang2017.github.io/Projects/HRNet/PoseEstimation.html

13. AI Pose Estimation with Python and MediaPipe | Plus AI Gym Tracker Project by Nicholas Renotte
https://youtu.be/06TE_U21FK4

14. T-pose detector in JavaScript for the Web
https://t-pose-detector.netlify.app/

15. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., and Schaub, T. (2019). Asp-core-2 input language format. Theory and Practice of Logic Programming.

16. Dodaro, C., Leone, N., Nardi, B., Ricca, F. (2015). Allotment Problem in Travel Industry: A Solution Based on ASP. In: ten Cate, B., Mileo, A. (eds) Web Reasoning and Rule Systems. RR 2015. Lecture Notes in Computer Science(), vol 9209. Springer, Cham.

17. Amendola, G., Dodaro, C., Leone, N., Ricca, F. (2016). On the Application of Answer Set Programming to the Conference Paper Assignment Problem. In: Adorni, G., Cagnoni, S., Gori, M., Maratea, M. (eds) AI*IA 2016 Advances in Artificial Intelligence. AI*IA 2016. Lecture Notes in Computer Science(), vol 10037. Springer, Cham.

18. Potassco/Clingo
https://potassco.org/

19. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Comput. 9(3/4), 365–386 (1991)

20. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)

21. I. Niemelä, Logic programs with stable model semantics as a constraint programming paradigm, Annals of Mathematics and Artificial Intelligence 25 (3–4) (1999) 241–273.

22. M. Gelfond, N. Leone, Logic programming and knowledge representation — The A-Prolog perspective, Artificial Intelligence 138 (1–2) (2002) 3–38.

23. M. Gelfond, Answer sets, in: V. Lifschitz, F. van Harmelen, B. Porter (Eds.), Handbook of Knowledge Representation, Elsevier, 2008, pp. 285–316, Chapter 7.

24. M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, S. Thiele, A user's guide to gringo, clasp, clingo, and iclingo, available at: http://potassco.sourceforge.net.

25. Ariel V. Dowling, Julien Favre, and Thomas P. Andriacchi. 2011. A wearable system to assess risk for anterior cruciate ligament injury during jump landing: Measurements of temporal events, jump height, and sagittal plane kinematics.Journal of Biomechanical Engineering 133, 7 (2011), 071008–071008–7.

26. M. Pino Daz-Pereira, Ivn Gmez-Conde, Merly Escalona, and David N. Olivieri. 2014. Automatic recognition and scoring of Olympic rhythmic gymnastic movements. Human Movement Science 34 (2014), 63–80. DOI:https://doi.org/10.1016/j.humov.2014.01.001

27. Ksenia Kolykhalova, Antonio Camurri, Gualtiero Volpe, Marcello Sanguineti, Enrico Puppo, and Radoslaw Niewiadomski. 2015. A multimodal dataset for the analysis of movement qualities in karate martial art. In 7th International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN'15). IEEE, 74–78. https://eudl.eu/pdf/10.4108/icst.intetain.2015.260039

28. D. S. Alexiadis, P. Kelly, P. Daras, N. E. OConnor, T. Boubekeur, and M. Moussa, "Evaluating a dancers performance using kinectbased skeleton tracking," in Proceedings of the 19th ACM International Conference on Multimedia, ser. MM 11, 2011, pp. 659–662 https://core.ac.uk/download/pdf/11310464.pdf

29. R. Houmanfar, M. Karg and D. Kulić, "Movement Analysis of Rehabilitation Exercises: Distance Metrics for Measuring Patient Progress," in IEEE Systems Journal, vol. 10, no. 3, pp. 1014-1025, Sept. 2016, DOI: 10.1109/JSYST.2014.2327792.