



**UNIVERSITÀ ALDO MORO DI BARI**

DIPARTIMENTO DI INFORMATICA  
Corso di Laurea in Informatica

---

Tesi di Laurea in Sistemi ad Agenti

**WEB-APP PER IL RICONOSCI-  
MENTO E MONITORAGGIO DI PO-  
SIZIONI YOGA MEDIANTE CV**

Relatore  
**Prof.ssa Nadja Berardina De Caroliso**

Laureando  
**Daniele Cecca**

---

Anno Accademico 2021 - 2022



*Dedication,  
continuing dedication.*



# Abstract

All'interno di questa tesi verrà discusso il lavoro svolto che ha portato alla creazione di una web-app che permette la classificazione e correzione di pose di yoga spiegando, le scelte prese durante la sua realizzazione.

È stato utilizzato un dataset pubblico di 6 posizioni di Yoga(Cobra ,Corpse, Mountain, Tree, Triangle, Lotus) creato registrando, con una normale webcam RGB, 15 individui(10 maschi e 5 femmine). Per la classificazione sono state esplorate diverse soluzioni sia Deep che non (SVM, Random Forest, RNN, CNN, GRU) e tutte hanno portato ad ottimi risultati, difatti abbiamo per tutte le soluzioni un accuracy superiore al 90%. Per l'addestramento di questi modelli non sono stati utilizzati direttamente i video ma il dataset composto dai keypoints derivanti dall'uso di MoveNet come modello di Human Pose Estimation. Per la correzione invece si è deciso di calcolare per ogni frame gli 8 angoli più importanti e confrontarli con un angolo medio individuato tramite il KNN. I due sistemi, di classificazione e correzione, sono stati poi inserite all'interno di un'applicazione web creata seguendo il pattern MVC e che utilizza come framework back-end Flask.



# Indice

<b>Elenco delle figure</b>	V
<b>1 Introduzione</b>	1
1.1 Motivazioni e obiettivi . . . . .	1
1.2 Struttura della tesi . . . . .	2
<b>2 Stato dell'arte</b>	3
2.1 Artificial Intelligence . . . . .	3
2.2 Machine Learning . . . . .	5
2.3 Deep Learning . . . . .	6
2.4 Computer Vision . . . . .	7
2.5 Lavoro correlato . . . . .	8
2.5.1 Subsection Title . . . . .	10
<b>3 Tecnologie utilizzate</b>	11
3.1 Linguaggi e framework . . . . .	11
3.1.1 Sci-Kit-Learn . . . . .	12
3.1.2 Keras . . . . .	12
3.1.3 Open-CV . . . . .	12
3.1.4 Flask . . . . .	13
<b>4 Sistema Proposto</b>	15
4.1 Fase 1: Classificazione . . . . .	15
4.1.1 Dataset . . . . .	15
4.1.1.1 Organizzazione e Analisi esplorativa dei dati . . . .	15
4.1.1.2 Human Pose Estimation . . . . .	16
4.1.1.3 MoveNet . . . . .	18
4.1.1.4 Creazione dataset . . . . .	20
4.1.1.5 Divisione del dataset . . . . .	22
4.1.2 Costruzione dei modelli & Addestramento . . . . .	24
4.1.2.1 Metriche . . . . .	24
4.1.3 Modelli Machine Learning . . . . .	26

4.1.3.1	Validation . . . . .	26
4.1.3.2	SVM . . . . .	26
4.1.3.3	Random Forest . . . . .	29
4.1.4	Modelli Deep Learning . . . . .	30
4.1.4.1	Reti Neurali Ricorrenti-RNN . . . . .	30
4.1.4.2	Long Short Term Memory-LSTM . . . . .	33
4.1.4.3	Gated Recurrent Unit-GRU . . . . .	35
4.1.5	Confronto riassuntivo . . . . .	37
4.1.6	Osservazioni . . . . .	37
4.2	Fase 2: Correzione . . . . .	39
4.2.1	Dataset . . . . .	39
4.2.1.1	Creazione del dataset . . . . .	39
4.2.2	Calcolo degli angoli . . . . .	40
4.2.3	KNN-Angoli corretti . . . . .	41
4.2.4	Sistema di feedback . . . . .	41
4.3	Fase 3: Web-App . . . . .	45
4.3.1	Descrizione del progetto . . . . .	45
4.3.2	System design . . . . .	46
4.3.2.1	Model-View-Controller . . . . .	46
4.3.3	Funzionalità . . . . .	46
4.3.3.1	Pagina Login . . . . .	46
4.3.3.2	Pagina Sign-In . . . . .	47
4.3.3.3	Pagina Home . . . . .	48
4.3.3.4	Pagina Sessione Privata . . . . .	48
4.3.3.5	Pagina Crea Routine . . . . .	50
4.3.3.6	Pagina Sessione Live-Libera . . . . .	52
4.3.3.7	Pagina Sessione Live-Routine . . . . .	52
<b>5</b>	<b>Conclusioni e sviluppi futuri</b>	<b>55</b>
5.1	Conclusioni . . . . .	55
5.1.1	Sviluppi futuri . . . . .	55
	<b>Bibliografia</b>	<b>59</b>



# Elenco delle figure

2.1	Rete neurale feed forward . . . . .	7
2.2	Rappresentazione scala di grigi . . . . .	8
3.1	Logo Sci-Kit-Learn . . . . .	12
3.2	Logo keras . . . . .	12
3.3	Logo openCV . . . . .	13
3.4	Logo Flask . . . . .	13
4.1	Frames di pose dal dataset . . . . .	16
4.2	Human body models . . . . .	17
4.3	Scheletro individuato da MoveNet . . . . .	18
4.4	Architettura MoveNet . . . . .	19
4.5	Pipeline di lavoro di MoveNet . . . . .	20
4.6	Plot del bilanciamento del dataset . . . . .	24
4.7	Esempio matrice di confusione per valori binari . . . . .	26
4.8	Support Vector Machines . . . . .	27
4.9	Matrice di confusione per SVM . . . . .	28
4.10	Matrice di confusione per Random Forest . . . . .	30
4.11	Rete neurale srotolata . . . . .	31
4.12	Rete neurale ricorrente . . . . .	31
4.13	Matrice di confusione per RNN . . . . .	33
4.14	Cella LSTM . . . . .	34
4.15	Matrice di confusione per LSTM . . . . .	35
4.16	Matrice di confusione per GRU . . . . .	37
4.17	Confronto Precision . . . . .	38
4.18	Confronto Recall . . . . .	38
4.19	Confronto F-1 . . . . .	38
4.20	Dizionario 8 angoli . . . . .	40
4.21	Pagina della web-app dedicata al login . . . . .	47
4.22	Pagina della web-app dedicata al sign-in . . . . .	48
4.23	Pagina della web-app dedicata alla home . . . . .	49
4.24	Pagina della web-app dedicata alla sessione privata . . . . .	50

4.25	Pagina della web-app dedicata alla creazione della routine . . . . .	51
4.26	Pagina della web-app dedicata alla sessione privata libera . . . . .	52
4.27	Pagina della web-app dedicata alla sessione privata routine . . . . .	53
5.1	Ringraziamenti . . . . .	57

# Capitolo 1

## Inroduzione

### 1.1 Motivazioni e obbiettivi

Lo studio dello yoga e delle varie pose, chiamate “asanas” ha una lunga e ricca storia, che abbraccia centinaia di anni. Ad oggi lo yoga è riconosciuto per i suoi innumerevoli benefici, come il miglioramento della salute fisica, la riduzione dello stress e delle ansie e il miglioramento della concentrazione e della consapevolezza di se. Di conseguenza, la pratica dello yoga è diventata sempre più popolare sia tra i più giovani che tra i più anziani, con milioni di persone in tutto il mondo che la incorporano nella loro routine quotidiana. Tuttavia, la corretta esecuzione delle posizioni yoga è fondamentale per evitare lesioni e massimizzare i benefici della pratica. Garantire una corretta esecuzione non è però un compito facile, difatti ci sono molti fattori, come l’anatomia individuale, la flessibilità, la forza, che possono incidere sull’esecuzione della posa. Pertanto, valutare accuratamente la propria forma può risultare complicato sia per i principianti sia per i più esperti. Ed è qui che nasce l’esigenza di un sistema che permetta la classificazione e la correzione dell’esecuzione di pose di yoga. Inoltre, dopo la pandemia COVID-19, l’aumento della domanda per sistemi di yoga di questo tipo e più in generale di sistemi per il fitness, ha reso questo problema ancor più rilevante. Questo tipo di sistema fornisce quindi all’utente una totale autonomia garantendogli una maggior sicurezza ed efficacia nell’esecuzione degli esercizi. Oltre a poter essere utilizzato in maniera autonoma dall’utente, questo tipo di sistemi potrebbe essere usato a sostegno di istruttori, sia che questi svolgano le lezioni all’interno di una palestra, sia che questi svolgano le lezioni online su piattaforme dedicate, che come detto precedentemente, a causa della pandemia sono molto diffuse, fornendogli tutte le informazioni per poter seguire in maniera oculata l’esecuzione della posa da parte di tutti i suoi allievi e non solo una parte di questi. Questa tesi, quindi, avrà come obbiettivo quello di realizzare il sistema sopra descritto, e per farlo verranno utilizzate tecniche di computer vision e di machine learning, con un focus sul deep learning.

## 1.2 Struttura della tesi

# Capitolo 2

## Stato dell'arte

Come anticipato il nostro sistema farà uso di tecniche di computer vision machine learning e deep learning; pertanto, nelle prime sezioni di questo capitolo andremo ad esplorare questi argomenti ad alto livello per dare al lettore una panoramica generale di quello che verrà approfondito nei capitoli successivi. Mentre nelle sezioni successive esploreremo alcune soluzioni proposte da aziende o ricercatori sia che queste siano simili alla soluzione da noi proposta sia che queste differiscano.

### 2.1 Artificial Intelligence

Cosa è l'intelligenza artificiale? Esistono diverse definizioni di intelligenza artificiale che si sono susseguite nel corso del tempo, tra le più accreditate vi è la definizione di John McCarthy, il quale offre la seguente definizione nel paper [1], rilasciato nel 2004:

*" It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."*

Definizione che trova riscontro nelle principali applicazioni dell'Intelligenza Artificiale come speech recognition, customer service, computer vision, recommendation engines, automated stock trading, ossia applicazioni che normalmente richiedono intelligenza umana ma non solo. L'AI può essere divisa in 2 categorie[2]:

- Weak AI: la Weak AI, chiamata anche Narrow AI o Artificial Narrow Intelligence (ANI), è un' AI addestrata e focalizzata per eseguire compiti specifici. L'intelligenza artificiale debole è sicuramente il tipo di AI ad oggi più diffuso. 'Narrow' (Stretto) potrebbe però essere una descrizione più accurata per questo tipo di AI in quanto è tutt'altro che debole; infatti si trova in alcune

delle applicazioni ad oggi più robuste, come Siri di Apple, Alexa di Amazon e nelle auto a guida autonoma.

- Strong AI: la Strong AI è composta dall'Intelligenza Generale Artificiale (AGI) e Super Intelligenza Artificiale (ASI). L'intelligenza artificiale generale (AGI), è una forma teorica di AI in cui una macchina avrebbe un'intelligenza pari a quella umana; avrebbe una coscienza autocosciente che ha la capacità di risolvere problemi, imparare e pianificare il futuro. La super intelligenza artificiale (ASI), nota anche come super intelligenza, supererebbe l'intelligenza e l'abilità del cervello umano. Sebbene l'AI forte sia ancora interamente teorica senza esempi pratici in uso oggi, ciò non significa che i ricercatori di intelligenza artificiale non stiano anche esplorando il suo sviluppo. Nel frattempo, i migliori esempi di ASI potrebbero provenire dalla fantascienza, come HAL, l'assistente informatico sovrumano in "2001: Odissea nello spazio".

Oltre a poter dividere l'AI in queste due categorie, possiamo individuare anche due paradigmi:

- Symbol AI
- Machine Learning

Symbol AI implica lo sviluppo di regole, fatti e modelli espliciti per rappresentare la conoscenza ed eseguire compiti. L'apprendimento automatico, d'altra parte, comporta lo sviluppo di algoritmi in grado di apprendere automaticamente dai dati; in particolare ai fini di questa tesi ci concentreremo sul machine learning.

L'intelligenza artificiale ha quindi il potenziale per trasformare un'ampia gamma di settori, dalla sanità ai trasporti fino ad arrivare alla finanza e al commercio al dettaglio. Ad esempio, nel settore sanitario, gli algoritmi di intelligenza artificiale vengono utilizzati per migliorare l'accuratezza delle diagnosi mediche e per sviluppare piani di trattamento personalizzati mentre nei trasporti, l'intelligenza artificiale viene utilizzata per sviluppare veicoli autonomi e migliorare la gestione del traffico.

Tuttavia, lo sviluppo e la diffusione dell'AI sollevano anche una serie di importanti preoccupazioni etiche e sociali. Ad esempio, il crescente utilizzo dell'IA nel processo decisionale solleva importanti questioni sulla responsabilità e sulla trasparenza. Inoltre, il potenziale dell'intelligenza artificiale di automatizzare i lavori e spostare i lavoratori solleva importanti domande sul futuro del lavoro e sulla disparità di reddito. Oltre a questi problemi ne sorgono molti altri di più varia natura, raccolti all'interno dello studio[3].

## 2.2 Machine Learning

Machine Learning Come anticipato precedentemente una sottocategoria di intelligenza artificiale è il machine learning, il quale in accordo con uno dei suoi pionieri, Arthur Samuel, è definito come :

*“the field of study that gives computers the ability to learn without explicitly being programmed.”*[4]

Difatti il ML si pone come obbiettivo quello di realizzare sistemi che possono apprendere dai dati e fare previsioni o prendere decisioni senza che questi siano stati esplicitamente programmati per eseguire compiti specifici. In altri termini il ML è il processo di training(addestramento) di un pezzo di software, chiamato modello(modello), che permette di fare predizioni utili a partire dai dati. Quindi un modello altro non rappresenta che la relazione matematica tra gli elementi dei dati che un sistema di ML usa per fare predizioni. Gli algoritmi di machine learning possono essere divisi in 3 categorie a seconda della modalità con cui avviene la fase di apprendimento[5]:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Nei modelli di apprendimento supervisionato le previsioni vengono fatte dopo aver visto molti dati con le risposte corrette e quindi aver scoperto le connessioni tra gli elementi nei dati che producono le risposte corrette. Da un punto di vista più tecnico avremo che il nostro modello verrà addestrato su una serie di dati dove oltre alle features di input sono definite le features target. Due dei task più comuni che vengono risolti con sistemi basati sull'apprendimento supervisionato sono i task di classificazione e regressione. Un modello di regressione predice un valore numerico mentre i modelli di classificazione predicono la probabilità(la likelihood) con la quale il sample preso sotto esame appartenga ad una data categoria. Inoltre i modelli di classificazione possono effettuare una classificazione binaria o una classificazione multi-classe.

Nei modelli di apprendimento non supervisionati le previsioni vengono fatte ricevendo dati che non contengono risposte corrette. L'obiettivo di un modello di apprendimento non supervisionato è identificare pattern significativi tra i dati. In altre parole, il modello non ha suggerimenti su come classificare ogni dato, ma deve invece inferire le proprie regole. Da un punto di vista più tecnico avremo che il nostro modello verrà addestrato su una serie di dati dove sono definite solo le features di input e non le features target. Una delle tecniche più usate nell'apprendimento

non supervisionato è il clustering, dove a partire dai dati vengono creati dei cluster che andranno a dividere i nostri samples.

Nei modelli di apprendimento per rinforzo le previsioni vengono effettuate ottenendo ricompense o penalità in base alle azioni eseguite all'interno di un ambiente. Un sistema di apprendimento per rinforzo genera una policy (politica) che definisce la migliore strategia per ottenere il maggior numero di ricompense. L'apprendimento per rinforzo viene utilizzato per addestrare i robot a svolgere compiti, come camminare per una stanza, e programmi software come AlphaGo per giocare al gioco del Go.

Nonostante le potenzialità del ML anche questi tipi di sistemi hanno delle importanti limitazioni e problematiche da affrontare. Una di queste è l'Overfitting, che si verifica quando l'algoritmo apprende troppo bene i dati di addestramento, al punto da non essere più in grado di generalizzare su nuovi dati, non ancora osservati. L'overfitting può però essere mitigato attraverso tecniche come la regolarizzazione, la cross validation e l'arresto anticipato.

Un'altra problematica da affrontare è il problema del bias nell'apprendimento. Gli algoritmi di machine learning possono essere "distorti" o "influenzati" se i dati di addestramento contengono degli errori sistematici o se l'algoritmo è stato progettato con determinati bias. Questo può comportare ad ottenere risultati non corretti o discriminatori, pertanto bisogna stare molto attenti a questi problemi quando si decide di mettere in produzione un algoritmo di ML.

## 2.3 Deep Learning

Un sottocampo del ML è il Deep learning, il quale si concentra sullo sviluppo di algoritmi e modelli ispirati alla struttura e al funzionamento del cervello. A differenza degli algoritmi di machine learning gli algoritmi di deep learning hanno però bisogno di una mole molto più elevata.

Gli algoritmi di deep learning sono costituiti da più layer di neuroni artificiali, che sono collegati da connessioni ponderate. L'input al primo livello sono i dati grezzi, come un'immagine o una frase. Dopo di che l'output di ogni livello viene passato come input al livello successivo, finché il livello finale non produce la previsione o decisione finale. Durante la fase di training l'algoritmo regola i pesi delle connessioni per ridurre al minimo l'errore tra gli output previsti e quelli effettivi.

Uno dei principali vantaggi nell'uso di algoritmi di DL rispetto a algoritmi di ML è che il deep learning elimina parte del processo di pre-processing dei dati tipicamente coinvolto nel machine learning. Infatti, questi algoritmi, possono importare ed elaborare dati non strutturati (raw-data), come testo e immagini, e automatizzare l'estrazione delle features, eliminando la dipendenza da degli esperti. Ad esempio, supponiamo di avere una serie di foto di diversi animali domestici e di volerli classificare in base a 'gatto', 'cane', 'criceto', eccetera. Gli algoritmi di deep



learning possono determinare quali caratteristiche (ad esempio le orecchie) sono più importanti per distinguere un animale da un altro. Nel machine learning “semplice”, questa gerarchia di funzionalità viene stabilita manualmente da un esperto umano.[6]

Tuttavia, il deep learning oltre a presentare gli stessi problemi del machine learning presenta anche il problema dell’interpretabilità, che si riferisce alla difficoltà di capire perché un modello di deep learning abbia fatto una particolare previsione o decisione. Questa può essere una sfida importante per applicazioni come la diagnosi medica, dove è importante comprendere le basi per una diagnosi.

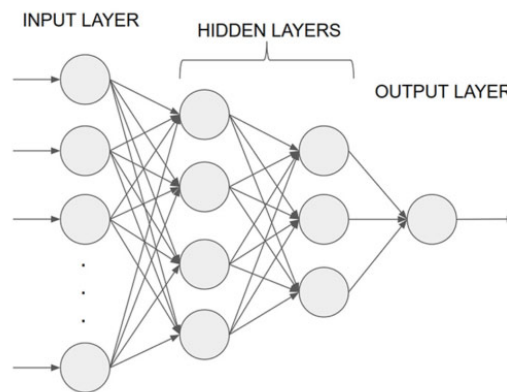


Figura 2.1: Rete neurale feed forward

## 2.4 Computer Vision

La Computer vision come anticipato è una branchia dell’intelligenza artificiale che si occupa dell’interpretazione e dell’analisi delle informazioni visive. Implica lo sviluppo di algoritmi, modelli e tecniche per consentire ai computer di comprendere, interpretare e manipolare le informazioni visive dal mondo, come immagini e video. Il campo della computer vision è motivato dal desiderio di replicare la percezione visiva umana e di fornire ai computer la capacità di eseguire attività che normalmente richiederebbero l’intelligenza visiva umana, come riconoscere oggetti, rilevare schemi e prendere decisioni basate su dati visivi. Inoltre grazie ai progressi dell’intelligenza artificiale e alle innovazioni nel deep learning e nelle reti neurali, il campo ha potuto fare grandi passi avanti negli ultimi anni ed è stato in grado di superare gli umani in alcuni compiti relativi al rilevamento e all’etichettatura di oggetti.

Uno dei fattori principali ad oggi che ha portato alla crescita della computer vision è la quantità di dati che generiamo e che poi usiamo nei nostri modelli, si

stima infatti che oggi vengano condivise più di 3 miliardi di immagini e 720000 ore di video ogni giorno[7].

Immagini che vengono altro non sono delle matrici bidimensionale di pixel, ognuno con la propria codifica a seconda dello spazio di colore usato. Ad esempio, nel seguente esempio[8] è stata usata la scala di grigi, pertanto ogni pixel sarà rappresentato da un numero a 8-bit il cui range va da  $[0,255]$ , dove 0 corrisponde a nero e 255 a bianco.

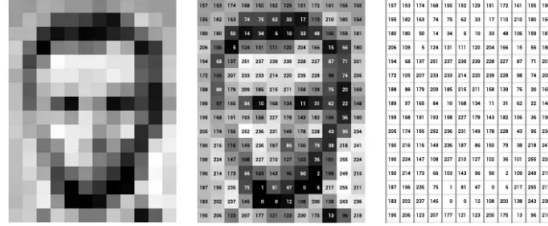


Figura 2.2: Rappresentazione scala di grigi

Oltre alla scala di grigi potremo avere come modelli colore anche quello RGB e HSV. Sia nel primo caso che nel secondo avremo che ogni pixel sarà rappresentato da 3 byte, ma mentre nel primo caso ogni byte codifica rispettivamente il colore, red, green e blue nel secondo caso 1 byte viene usato per la tonalità, 1 byte per la saturazione e 1 byte per la luminosità.

## 2.5 Lavoro correlato

Ad oggi esistono diverse aziende che propongono diverse soluzioni per il controllo e la correzione di esercizi yoga da casa.

SmartMat[16] come suggerisce il nome propone un tappetino da yoga intelligente che riesce, tramite dei sensori posti nel tappetino, a ricavare la pressione che si sta applicando in determinati punti e, tramite i dati ricavati da questi sensori riesce a determinare se la posa indicata sia svolta in maniera corretta e nel caso non lo sia viene fornito un feedback nell'apposita applicazione.

L'azienda Wearable X[17] propone invece dei pantaloni da yoga intelligenti che inviano una serie di dati ad una applicazione mobile tramite la quale è possibile controllare che si stia svolgendo in maniera corretta o meno la posa di yoga.

YogaNotch[18] propone dei dispositivi indossabili, dotati di casse interne che oltre a permettere di determinare se la posa venga svolta in maniera corretta permettono di avere anche un feedback sonoro.

Un altro dispositivo molto popolare è lo smart mirror proposto da MIRROR[19], il quale rispetto agli altri strumenti descritti finora permette di svolgere molte più attività oltre lo yoga come kickboxing, danza, esercizi posturali ecc. e mette a disposizione dell'utente un sistema molto più ampio e completo con lezioni in streaming ed altre funzionalità utili per gestire il proprio workout. Le informazioni circa gli esercizi che si stanno svolgendo sono raccolte da una telecamera e da alcuni sensori posti al di sotto dello schermo. Unico difetto di questo prodotto è il costo dello smart mirror e del servizio proposto in abbonamento senza il quale il sistema non funzionerebbe. Costo che quindi non gli permette di essere un prodotto di massa.

Anche nella letteratura scientifica molti sono gli esempi di sistemi di questo tipo, alcuni dei quali adottano delle soluzioni simili alla soluzione da noi adottata. Nella maggior parte dei lavori proposti sono usate reti neurali ricorrenti LST[24] e reti neurali convolutive CNN per la classificazione.[25]

Nello studio proposto da [25] è stato proposto un sistema di coaching di yoga basato sul transfer learning. Sono state collezionati video per 14 pose ognuna dei quali registrato 10 volte da 8 volontari(6 donne e 2 uomini). Per la classificazione è stata adottata una CNN, mentre come modelli pre-addestrati sono stati confrontati i seguenti modelli VGG16, VGG19, MobileNet, MobileNetV2, InceptionV3, and DenseNet201 [28,29,30,31,32] addestrati su ImageNet, riuscendo a raggiungere un'accuracy del 98.43%. La correzione invece avviene considerando gli angoli e la loro differenza rispetto a degli angoli di riferimento per quella determinata posa. Il feedback però rilasciato dal sistema non rilascia dei feedback personalizzati ma ci dice solamente se la posa è stata eseguita in maniera corretta o meno

Nell studio[26] è stato utilizzato come modello di Human Pose Estimation Mediapipe e i vettori di keypoints sono stati raccolti in formato JSON. Come dataset è stato utilizzato il nostro stesso dataset mentre come modello di classificazione è stato adottato un approccio ibrido CNN+LSTM riuscendo ad ottenere un accuracy del 99.53%. Per determinare la correttezza di una posa invece è stata adottata la similarità del coseno.

Anche nello studio[27] è stato proposto come modello di classificazione il modello CNN+LSTM ottenendo un'accuracy sul set di test pari al 99.04% ed anche in questo caso è stato utilizzato lo stesso dataset. In questo caso però è stato usato come modello di Human Pose Estimation Open Pose. In questo studio però non viene proposto alcun sistema di correzione.

Lo studio [28] a differenza degli studi precedenti sebbene utilizzi lo stesso dataset utilizza come metodo di estrazione dei punti Keras multiperson pose che permette

di estrarre 12 keypoints. Mentre come algoritmo di classificazione utilizza il modello MLP ottenendo un'accuracy del 99.58%, un risultato equiparabile ai risultati visti fino ad ora con un modello però che risulta essere molto più leggero. In questo caso però il dataset usato per la classificazione non sarà composto direttamente dai keypoints ma dai 12 angoli calcolati. Per il sistema di correzione invece per ogni posa sono calcolati i 12 angoli medi e nel momento della classificazione gli angoli formati in real-time sono confrontati con gli angoli medi.

### **2.5.1 Subsection Title**

repellat [1].

# Capitolo 3

## Tecnologie utilizzate

In questo capitolo descriveremo gli strumenti e le tecnologie utilizzati durante le varie fasi per la realizzazione del progetto. Nella prima parte del capitolo verranno elencati tutti gli strumenti utilizzati mentre nella seconda parte verranno approfonditi solo gli strumenti più rilevanti.

### 3.1 Linguaggi e framework

Il progetto è stato sviluppato interamente attraverso il linguaggio Python. Nell'ultima fase oltre ad essere stato utilizzato Python è stato usato anche il linguaggio Javascript, SQLAlchemy e SQLite per gestire il database, il linguaggio di mark-up Html e i fogli di stile CSS.

Come IDE invece è stato utilizzato Visual Studio Code, in particolare nelle prime due fasi abbiamo utilizzato come file di lavoro i file notebooks. Per quanto riguarda i framework e le librerie utilizzate invece abbiamo utilizzato:

- Per il sistema di classificazione e correzione:
  - sklearn: per i modelli di apprendimento supervisionato
  - keras : per i modelli di apprendimento deep
  - open-cv: per lavorare con i file video
  - numpy: per lavorare con i vettori
  - pandas: per confrontare i risultati
  - matplotlib, seaborn: per la realizzazione e visualizzazione di grafici
  - imblearn: per l'automatizzazione di processi tramite pipeline
- Per la realizzazione della web-app:

- Flask : per gestire il back-end
- Bootstrap: per la gestione di alcuni elementi (button,navbar,..) nelle pagine web

### 3.1.1 Sci-Kit-Learn

Scikit-learn è una libreria chiave per il linguaggio di programmazione Python, che viene tipicamente utilizzata nei progetti di machine learning . Scikit-learn fornisce una serie di strumenti efficienti per l' apprendimento automatico, inclusi algoritmi matematici, statistici e generici che costituiscono la base per molte tecnologie di apprendimento automatico. Tra i vari strumenti proposti abbiamo algoritmi di classificazione, regressione e clustering. Inoltre, questa libreria, scritta in gran parte in Python, è progettata per lavorare con le librerie NumPy e SciPy.



Figura 3.1: Logo Sci-Kit-Learn

### 3.1.2 Keras

Keras è un'API di deep learning ad alto livello sviluppata da Google per l'implementazione di reti neurali. È scritta in Python e viene utilizzata per semplificare l'implementazione delle reti neurali. Keras è relativamente facile da imparare e utilizzare perché fornisce un' interfaccia Python con un alto livello di astrazione pur avendo la possibilità di utilizzare in base allo scopo diversi back-end. Questo rende Keras più lento di altri framework di deep learning, ma estremamente adatto ai neofiti.



Figura 3.2: Logo keras

### 3.1.3 Open-CV

OpenCV è una libreria open source per la visione artificiale, l'apprendimento automatico e l'elaborazione delle immagini che ora svolge un ruolo importante nelle

operazioni in tempo reale che sono molto importanti nei sistemi odierni. Usandolo, è possibile elaborare immagini e video per identificare oggetti, volti o persino la calligrafia di una persona. Per sviluppare OpenCV è stato utilizzato il C++ ma è possibile interfacciarsi anche con il C, Python e Java

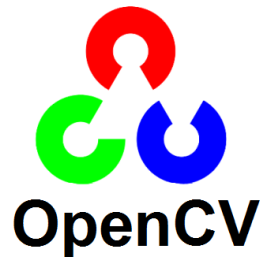


Figura 3.3: Logo openCV

#### 3.1.4 Flask

Flask è un framework web scritto in Python. Ciò significa che Flask fornisce strumenti, librerie e tecnologie che consentono di creare un'applicazione Web. Flask fa parte delle categorie del micro-framework. I micro-framework sono framework con dipendenze minime o nulle da librerie esterne. Nel caso di Flask le dipendenze sono:

- Werkzeug
- jinja2 (motore di template)

Nonostante Flask sia un micro-framework supporta estensioni che possono aggiungere funzionalità a un'applicazione come se fossero implementate dallo stesso Flask. Ci sono per esempio estensioni per la validazione dei formulari, la gestione del caricamento dei file, varie tecnologie di autenticazione e altro.



Figura 3.4: Logo Flask





# Capitolo 4

## Sistema Proposto

### 4.1 Fase 1: Classificazione

All'interno di questo capitolo verrà descritta l'intera pipeline di lavoro che ha portato al risultato ottenuto in questa prima fase, a partire dai dati a nostra disposizione fino ad arrivare ai modelli usati per la classificazione. In particolare, ogni scelta presa verrà spiegata in maniera approfondita con apposite regressioni di natura teorica ove necessario.

#### 4.1.1 Dataset

##### 4.1.1.1 Organizzazione e Analisi esplorativa dei dati

Per la realizzazione di questo sistema abbiamo utilizzato il seguente dataset[9], che fa parte di una collezione Open Source realizzata nello studio proposto [10]. Questo dataset si compone di 88 video, rappresentanti 15 diversi individui(5 femmine e 10 maschi), di diversa età, che svolgono 6 diverse pose di yoga:

- Cobra (Bhujangasana)
- Tree (Vrikshasana)
- Mountain (Tadasana)
- Lotus (Padmasana)
- Triangle (Trikonasana)
- Corpse (Shavasana)

Ogni video ha una durata che si aggira tra i 30 e i 60 secondi per un totale di 1 ora 6 minuti e 5 secondi. Tutti i video sono stati registrati a 30 fps con una qualità pari a 1366x768. Inoltre, tutti i video sono stati registrati all'interno di una stanza

ad una distanza di 4 metri dalla camera.

No.	Yoga Pose	#Persone	#Video
1	Cobra	15	16
2	Lotus	14	14
3	Corpse	15	15
4	Mountain	15	15
5	Triangle	13	13
6	Tree	15	15
Numero totale di video			88



Figura 4.1: Frames di pose dal dataset

Prima di passare effettivamente al pre-processing dei nostri dati abbiamo ripartito i video in maniera manuale in 6 diverse cartelle rappresentanti le 6 diverse pose. Inoltre, per garantire una maggior reliability abbiamo preso alcuni accorgimenti sulla struttura e l'organizzazione del progetto e abbiamo realizzato una serie di script che permettono di lavorare più facilmente con i dati a nostra disposizione.

#### 4.1.1.2 Human Pose Estimation

Come appena detto nella sezione precedente il nostro dataset si compone di una serie di video in cui vengono eseguite diverse pose ma, per poter effettuare una classificazione dei video, dobbiamo affrontare il problema della Human Pose Estimation.

La Human Pose Estimation (HPE) è un task della computer vision che si concentra sull'identificazione della posizione di un corpo umano all'interno di una scena

specifica. Più precisamente è un modo per catturare un insieme di coordinate per ogni articolazione (braccio, testa, busto, ecc.) , che è nota come key-point e che vengono usate per descrivere una posa di una persona. La connessione tra due punti è nota come coppia.

La connessione formata tra i punti deve però essere significativa, il che significa che non tutti i punti possono formare una coppia. Fin dall'inizio, l'obiettivo di HPE è quello di formare una rappresentazione 2D o 3D di un modello del corpo umano.

Questi modelli quindi sono sostanzialmente delle mappature dei joints del corpo umano, sia che questo stia fermo, sia che questo sia in movimento. Esistono tre tipi di modelli per la modellazione del corpo umano[12]:

- Kinematic model o Skeleton-based model viene utilizzato per la stima della posa 2D e per la stima della posa 3D. Questo modello di corpo umano flessibile e intuitivo include una serie di posizioni articolari, joints e orientamenti degli arti per rappresentare la struttura del corpo umano. Pertanto, i modelli di stima della posa dello scheletro vengono utilizzati per catturare le relazioni tra le diverse parti del corpo. Tuttavia, i modelli cinematici sono limitati nella rappresentazione delle informazioni sulla texture o sulla forma .
- Planar model o Countourn-based model viene utilizzato per la stima della posa 2D. I modelli planari sono usati per rappresentare l'aspetto e la forma di un corpo umano. Di solito, le parti del corpo sono rappresentate da più rettangoli che si avvicinano ai contorni del corpo umano.
- Volumetric model o Volume-based model viene utilizzato per la stima della posa 3D. Esistono diversi modelli di corpo umano 3D popolari utilizzati per la stima della posa umana 3D basata sull'apprendimento profondo per il recupero della mesh umana 3D.

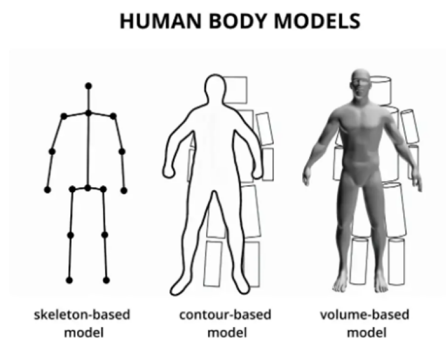


Figura 4.2: Human body models

Concentrandoci sugli skeleton based-models, essendo questi di nostro interesse, abbiamo che i modelli possono essere suddivisi in due categorie a seconda dell'approccio adottato per determinare lo scheletro della persona:

1. Top down: vengono localizzati prima le persone all'interno della scena e poi sono individuate le parti del corpo
2. Bottom Up: vengono prima localizzate le parti del corpo e a partire da quelle si individua l'intero corpo

Dopo una serie di ricerche e sfruttando anche lo studio [13] in cui vengono confrontati i punti di forza di vari modelli si è deciso di utilizzare il modello Move-Net.

Si è deciso di usare Move-Net oltre che per le sue peculiarità quali la velocità nella rilevazione dei punti, la capacità di individuare i punti per più persone anche perché come vedremo è stato addestrato su un dataset contenente esercizi fisici e posizioni di yoga.

#### 4.1.1.3 MoveNet

MoveNet è un modello ultraveloce e preciso che rileva 17 punti chiave di un corpo. Il modello è offerto su TF Hub con due varianti, note come Lightning e Thunder. Lightning è destinato ad applicazioni critiche per la latenza, mentre Thunder è destinato ad applicazioni che richiedono un'elevata precisione. Mentre il primo riceve in input immagini o frames di dimensioni 192x192 il secondo riceve in input immagini 256x256. Entrambi i modelli funzionano più velocemente del tempo reale (oltre 30 FPS) sulla maggior parte dei desktop, laptop e telefoni moderni, il che si rivela fondamentale per le applicazioni di fitness, salute e benessere dal vivo.

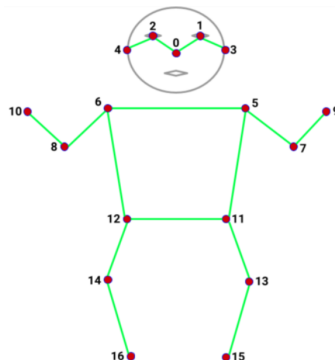


Figura 4.3: Scheletro individuato da MoveNet

Movenet è un modello bottom-up, che fa uso di heatmaps (mappe di calore) per localizzare i keypoints sul corpo di una persona. L'architettura consiste di 2 componenti principali:

1. Il Feature extractor
2. Un insieme di prediction heads

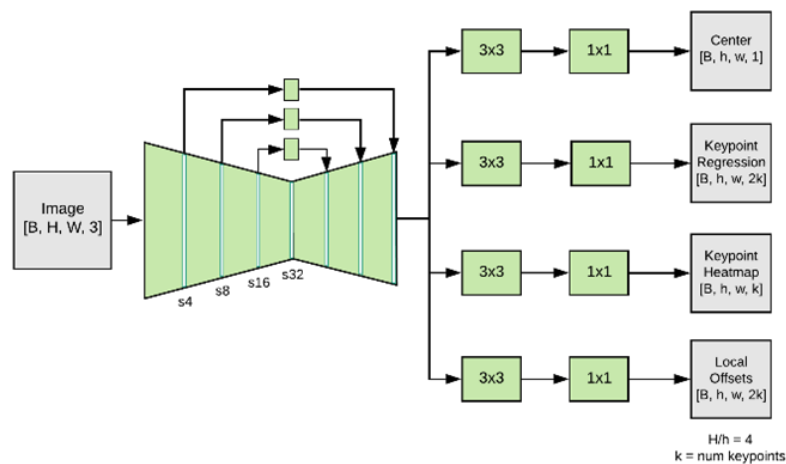


Figura 4.4: Architettura MoveNet

Lo schema di predizione segue largamente quello di CenterNet, con dei cambiamenti che migliorano l'accuratezza e la velocità. Tutti i modelli sono trainati usando TensorFlow Object Detection API. Il Feature extractor in MoveNet è MobileNetV2 unito ad una feature pyramid network (FPN), che consente un output della feature map semanticamente ricca e ad alta risoluzione (output stride 4).

Ci sono 4 prediction heads attaccate al feature extractor, responsabili delle seguenti predizioni:

- Person center heatmap: predice il centro geometrico (baricentro) di una persona
- Keypoint regression field: predice l'insieme dei keypoints a partire dal baricentro di quella persona ad un istante  $t$
- Heat map of key points: Utilizzando il regression field, il sistema prevede la posizione di tutti i keypoints di una persona, prendendo in considerazione solo la persona in primo piano.

- Two-dimensional shift field: predice l'off-set in pixel di ciascuna heatmap di ciascun keypoint per perfezionare il risultato finale.

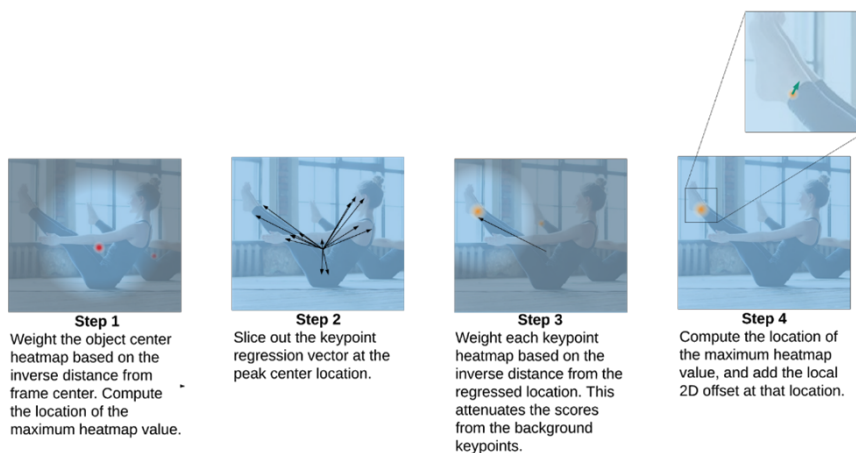


Figura 4.5: Pipeline di lavoro di MoveNet

MoveNet è stato addestrato su due set di dati: COCO e un set di dati interno di Google chiamato Active. Sebbene COCO sia il set di dati di riferimento standard per il rilevamento, a causa della sua diversità di scene e dimensioni, non è adatto per applicazioni di fitness e danza, che presentano pose impegnative e sfocature di movimento significative. Active è stato prodotto etichettando i punti chiave (adottando i 17 punti chiave del corpo standard di COCO) su video di yoga, fitness e danza da YouTube. Non vengono selezionati più di tre fotogrammi da ciascun video per la formazione, per promuovere la diversità di scene e individui.

Le valutazioni sul set di dati di convalida attiva mostrano un significativo aumento delle prestazioni rispetto ad architetture identiche addestrate utilizzando solo COCO. Ciò non sorprende poiché COCO esibisce raramente individui con pose estreme (ad esempio yoga, flessioni, headstands e altro).

#### 4.1.1.4 Creazione dataset

Come indicato all'inizio di questo capitolo per effettuare la classificazione come dataset non utilizzeremo direttamente i video o i frames ma andremo a costruire un nuovo dataset a partire dai landmarks individuati e restituiti da Move-Net.

Pertanto, definiamo una funzione che effettui l'estrazione dei landmarks a partire dall'output del modello appena descritto. Avremo quindi che per ogni frame verrà restituito un tensore che verrà trasformato dapprima in un array bidimensionale che avrà 17 righe corrispondenti ai 17 joints e 3 colonne corrispondenti a x, y e confidence e poi verrà reso flatten ossia ad una dimensione. Tale vettore sarà ritornato dalla funzione. Quindi saremo passati da una shape:

1. (1, 1, 17, 3)
2. (17, 3)
3. (51,)

Si noti che il vettore è stato reso flatten poiché le LSTM(un modello di deep learning per la classificazione) lavorano con questo formato di input.

Per garantire l'organicità del progetto e mantenere l'organizzazione congrua all'organizzazione fino ad ora descritta, definiamo una funzione che per ogni video crei una rispettiva cartella, nominata con nome "video + \_landmarks", all'interno della quale verranno salvati i keypoints raccolti per ogni frame.

Dopo aver creato le cartelle definiamo la funzione che andrà effettivamente a creare il nostro dataset, infatti questa funzione per ogni frame di ogni video andrà ad estrarre, tramite la funzione prima descritta, i keypoints che verranno salvati come numpy array all'interno della rispettiva cartelle.

```
1 def extract_all_keypoints(directory_path, actions):
2     for file in actions:
3
4         file_path = os.path.join(directory_path, file)
5
6         video_list = []
7         for f in os.listdir(file_path):
8             name, ext = os.path.splitext(f)
9             if ext == '.mp4':
10                 video_list.append(f)
11
12         for video in video_list:
13
14             video_path = os.path.join(file_path, video)
15             cap = cv2.VideoCapture(video_path)
16
17             #Ottengo il numero di frames totali
18             video_frames_count= int(cap.get(cv2.
CAP_PROP_FRAME_COUNT))
19
20             video_height, video_width = [768, 1366]# sostituire con
funzione
21             crop_region = init_crop_region(video_height,
video_width)
22
23             i=1
24
25             for frame_counter in range(video_frames_count):
26                 try:
27                     #leggo il frame
28                     ret, frame = cap.read()
```

```
29
30         #se non viene letto esco
31         if not ret:
32             break
33
34         image = frame.copy()
35         cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
36
37         keypoints_with_scores = run_inference(
38             movenet, image, crop_region,
39             crop_size=[256, 256])# sostituire con variabile
40
41         crop_region = determine_crop_region(
42             keypoints_with_scores, video_height,
43             video_width)
44
45         keypoints = extract_keypoints(
46             keypoints_with_scores)
47
48         npy_path = os.path.join(file_path, video + "
49         _landmarks", str(i))
50         np.save(npy_path, keypoints)
51         i+=1
52         print(i)
53         print(npy_path)
54     except:
55         pass
56
57     cap.release()
58     cv2.destroyAllWindows()
```

*N.B da questo momento in poi quando parliamo di video o frames facciamo sempre riferimento ad array di 51 points o a gruppi di questi nel caso di video.*

#### 4.1.1.5 Divisione del dataset

Affinché potessimo creare dei set di training e test i samples del dataset devono essere etichettati, ossia ogni sample deve essere associato alla categoria di appartenenza.

Pertanto definiamo un dizionario che associa ad ogni posa una label, che andrà ad indicare la classe di appartenenza, quindi in questo momento stiamo definendo il dominio della nostra feature target.

'Cobra': 0, 'Corpse': 1, 'Lotus': 2, 'Mountain': 3, 'Tree': 4, 'Triangle': 5

Si noti come creando questo dizionario si sia passati da una rappresentazione categorica ad una numerica senza sfruttare il label encoding fornito dalla libreria Sci-Kit Learn.



Nel momento dell'assegnazione delle label abbiamo deciso di non assegnare la label al singolo video composto da centinaia se non migliaia di frame ma si è deciso di dividere i video in subset di 45 frames, quindi in video da 1,5 secondi, ed assegnare ad ognuno di questi una label. In questo modo quindi avremo un dataset composto non da soli 88 video ma da 2599 video, ognuno dei quali composto da 45 frames. Questa fase è stata realizzata tramite il seguente script:

```
1 sequences=[]
2 group_sequences=[]
3 group=[]
4 labels=[]
5 j=0
6 for file in actions:
7     file_path = os.path.join(DATA_PATH, file)
8
9     key_dir_list = []
10    for f in os.listdir(file_path):
11        name, ext = os.path.splitext(f)
12        if not ext == '.mp4':
13            key_dir_list.append(f)
14
15
16    i=0
17    for keypoint_dir in key_dir_list:
18        window = []
19        for frame_num in range(1, info_videos[j][1][i][1]):
20            group=[x for x in range(len(key_dir_list))]
21            frame_path = np.load(os.path.join(file_path,
22            keypoint_dir, "{}.npy".format(frame_num)))
23            window.append(frame_path)
24            if frame_num%45==0:
25                sequences.append(window)
26                group_sequences.append(group[i])
27                labels.append(label_map[file])
28                window=[]
29
30        i+=1
31    j+=1
```

Arrivati a questo punto effettuiamo la divisione del nostro set di dati in set di training e set di test, andando a comporre rispettivamente il set di training con l'80% dei dati, mentre il set di test con il 20% dei dati. Essendo comunque il nostro dataset un dataset bilanciato, avendo circa gli stessi samples per ogni categoria, si è deciso di non splittare in modo casuale ma di utilizzare il parametro stratify nella funzione di splitting, in modo tale da mantenere le proporzioni del vettore assegnato; in questo caso il vettore è il vettore colonna rappresentante le features

target.

Come risultato avremo il seguente numero di video:

- 520 video per il test
- 2079 video per il train

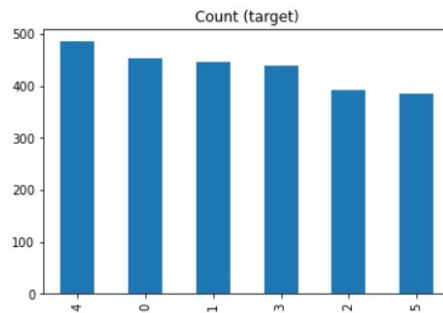


Figura 4.6: Plot del bilanciamento del dataset

## 4.1.2 Costruzione dei modelli & Addestramento

### 4.1.2.1 Metriche

Nel momento in cui un modello viene addestrato per la valutare la bontà delle sue predizioni vengono considerati i seguenti quattro casi:

- True positive (TP): Se la classe prevista è SI ed è uguale alla classe effettiva, si tratta di un caso di true positive (vero positivo). Il modello ha risposto correttamente SI.
- True negative (TN): Se la classe prevista è NO ed è uguale alla classe effettiva, si tratta di un caso di true negative (vero negativo). Il modello ha risposto correttamente NO.
- False positive (FP): Se la classe prevista è SI ma è diversa dalla classe effettiva, si tratta di un caso di false positive (falso positivo). Il modello ha sbagliato a rispondere SI. Si tratta di un errore di prima specie.
- False negative (FN): Se la classe prevista è NO ma è diversa dalla classe effettiva, si tratta di un caso di false negative (falso negativo). Il modello ha sbagliato a rispondere NO. Si tratta di un errore di seconda specie.

Si noti come commettere un errore di seconda specie sia più grave che compiere un errore di prima.

Per valutare i nostri modelli si è deciso di adottare le seguenti metriche:

- Accuracy

- L'accuratezza è generalmente una metrica che descrive il comportamento del modello, considerando tutte le classi. È utile quando tutte le classi hanno la stessa importanza. È calcolata come il rapporto tra il numero di predizioni corrette e il numero totale di predizioni.

$$Accuracy = \frac{True}{True_{positive} + True_{negative} + True_{negative}}$$

- Precision

- La precision descrive l'affidabilità del modello nel classificare i campioni come positivi. È calcolata come il rapporto tra il numero di samples positivi correttamente classificati e il numero di esempi totali classificati come positivi.

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

- Recall

- La recall misura l'abilità del modello di individuare samples positivi. È calcolata come il rapporto tra il numero che campioni Positivi correttamente classificati come positivi e il numero totale di campioni positivi.

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

- F-1-score

- L'F1-score è la media armonica della precision e della recall ,utile per avere una visione complessiva delle due metriche

$$F1Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Oltre a queste metriche si è deciso di valutare la bontà delle nostre predizioni anche mediante una matrice di confusione che risulta essere più facile da leggere e interpretare anche per i profili meno tecnici. Infatti, la matrice di confusione ci permette di visualizzare quanti samples sono stati classificati correttamente e quanti no e che label è stata assegnata ai samples classificati in maniera non corretta.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figura 4.7: Esempio matrice di confusione per valori binari

### 4.1.3 Modelli Machine Learning

Passando ora ai modelli di ML ,abbiamo utilizzato il modello SVM e il modello Random Forest.

#### 4.1.3.1 Validation

Prima di passare alla fase di test sia per l'SVM sia per il random forest abbiamo svolto la fase di validation mediante la stratify k-fold cross validation che, ci ha permesso di poter ottenere il miglior modello.

Infatti durante la fase di cross-validation i dati del set di training sono stati splittati in k fold, dove ogni fold è stata usata come set di validation mentre gli altri k-1 come set di training, il tutto ripetuto per k volte.

In questo modo è possibile ottimizzare i parametri per ogni modello , e come vedremo grazie all'utilizzo di particolari funzioni siamo stati in grado di ottimizzare i modelli di classificazione anche sulla base di alcuni iper-parametri, che saranno differenti nei due modelli.

Nel nostro caso abbiamo scelto k=3 per motivi legati ai limiti di potenza computazionale imposti dai nostri calcolatori.

Pe individuare il modello migliore abbiamo utilizzati la funzione GridSearchCV, che come anticipato ci permette di provare diversi iper-parametri durante la cross-validation, a cui passiamo come estimator(come modello) non il semplice modello, ma una pipeline in modo tale che se in futuro dovessimo effettuare operazioni di oversampling o under-samplig o normalizzazzione, queste risulterebbero più facili da aggiungere. Nella ricerca del modello migliore è stata usata come metrica di valutazione l'accuracy, già descritta nella sezione precedente.

#### 4.1.3.2 SVM

Il modello SVM è un modello di apprendimento supervisionato che nativamente supporta la classificazione binaria, ma come accade in molti casi il problema richiede una classificazione su più classi. Pertanto, viene adottata la strategia one vs all o one vs one. Nel nostro caso verrà adottata la strategia one vs all.

L'idea alla base dell'SVM è quella di trovare un iperpiano che divida i punti (che altro non sono che vettori) appartenenti alle rispettive classi, dove un iperpiano è un sottospazio lineare di dimensioni  $n-1$  rispetto allo spazio che lo contiene. Poiché esistono un numero potenzialmente infinito di iperpiani, SVM trova l'iperpiano che massimizzi il margine, dove con margine si intende la distanza minima dalla retta ai punti delle due classi, chiamati vettori di supporto.

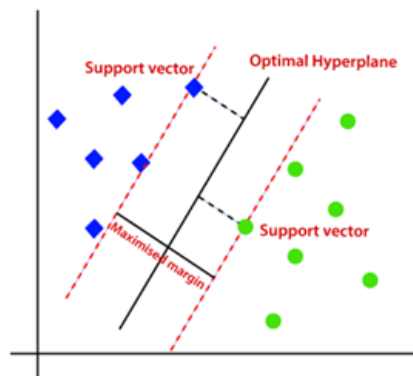


Figura 4.8: Support Vector Machines

Durante la fase di validation sono stati provati, come detto precedentemente, una serie di iper-parametri; Gli iper-parametri che sono stati scelti per ottimizzare il modello con i rispettivi values da provare, sono i seguenti:

- C: [0.1, 1, 10, 100, 1000]
- gamma: [1, 0.1, 0.01, 0.001, 0.0001]
- kernel: [rbf]

L'iperparametro C è un parametro di regolarizzazione che controlla il trade off tra un margine ampio e un basso numero di punti dati, classificati erroneamente (gestisce l'overfitting). Un valore più grande di C significa un maggior numero di punti del set di training classificati correttamente (bisogna stare attenti all'overfitting).

L'iperparametro gamma definisce fino a che punto arriva l'influenza di un singolo sample del set di training. Se ha un valore basso significa che i samples avranno una portata ampia mentre se ha un valore alto avranno una portata bassa e pertanto saranno più discriminanti i valori più vicini all'iperpiano.

Rbf, radial basis function, invece è il kernel predefinito e più popolare che altro non è che una funzione di base radiale gaussiana; rispetto agli altri kernel, lineari e polinomiali, offre una maggiore flessibilità.

In seguito alla cross validation siamo giunti alla conclusione che il modello migliore è ottimizzato secondo i seguenti values, assegnati ai rispettivi parametri:

- C: 10
- gamma: 0.01
- kernel: rbf

I risultati della nostra fase di test sono descritti dal risultato delle seguenti metriche:

- accuracy: 0.9826923076923076
- precision: 0.9839426523297491
- recall: 0.98250589616569
- f1: 0.9830119080797531

Di seguito rappresentiamo la matrice di confusione che ci permette di avere una panoramica completa dei risultati ottenuti:

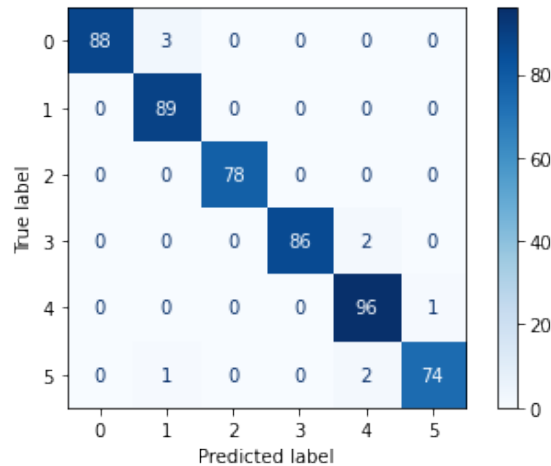


Figura 4.9: Matrice di confusione per SVM

#### 4.1.3.3 Random Forest

Il modello Random Forest è un modello di apprendimento supervisionato nel quale, come suggerisce il nome sono usati diversi alberi di classificazioni. Pertanto, la previsione finale sarà data media tra i risultati ottenuti dai vari alberi ,oppure sarà usato il meccanismo del voto. È stato scelto questo modello, essendo un modello migliore di un semplice albero di decisione, poiché un albero di decisione non è un ottimo predictor a causa della sua imprecisione nel classificare nuovi esempi (non di training, difatti si potrebbe verificare più facilmente l'overfitting). Ciò che rende effettivamente migliore questo modello è la varietà della selezione delle features di input(set di input), le quali vengono selezionate in maniera randomica per ogni albero, portando a predizioni diverse fra loro. Durante la fase di validation sono stati provati, come detto precedentemente, una serie di iper-parametri; Gli iper-parametri che sono stati scelti per ottimizzare il modello , con i rispettivi values da provare, sono i seguenti:

- `n_estimators`: [50, 100, 200]
- `max_depth`: [4, 6, 10, 12]

L'iperparametro `n_estimators` indica il numero di alberi di decisione che verranno usati nella foresta. L'iperparametro `max_depth` indica invece la profondità massima dell'albero. In seguito alla cross validation siamo giunti alla conclusione che il modello migliore è ottimizzato secondo i seguenti values, assegnati ai rispettivi parametri:

- `n_estimators`: 100
- `max_depth`: 12

I risultati della nostra fase di test sono descritti dal risultato delle seguenti metriche:

- `accuracy`: 0.9788461538461538
- `precision`: 0.9809351806829779
- `recall`: 0.9782930084896376
- `f1`: 0.9792064633651226

Di seguito rappresentiamo la matrice di confusione che ci permette di avere una panoramica completa dei risultati ottenuti:

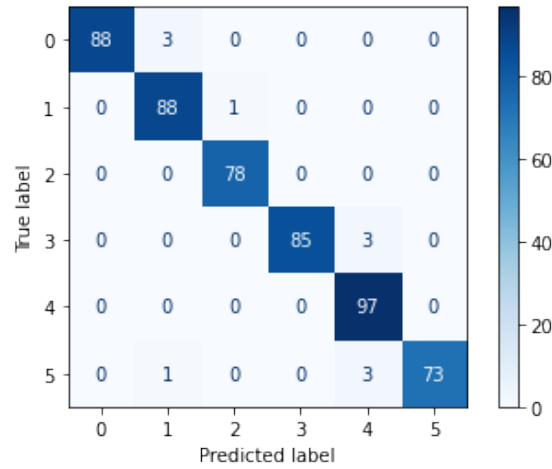


Figura 4.10: Matrice di confusione per Random Forest

#### 4.1.4 Modelli Deep Learning

Passando ora invece ai modelli di deep learning si è deciso di focalizzarsi sui modelli di reti neurali ricorrenti. In particolare si è deciso di esplorare i seguenti modelli neurali ricorrenti RNN, LSTM e GRU.

Per ognuno presenteremo una spiegazione di natura teorica seguita dalla soluzione adottata per ogni rete neurale.

Per quanto riguarda il dataset la feature target verrà codificata secondo la codifica One-Hot-Encoder, lavorando le reti neurali con questo tipo di codifica. Quindi per ogni label verrà creata una "dummy" variable o variabile indicatrice.

[0	0	0	0	0	1]	Cobra
[0	0	0	0	1	0]	Corpse
[0	0	0	1	0	0]	Lotus
[0	0	1	0	0	1]	Mountain
[0	1	0	0	0	0]	Tree
[1	0	0	0	0	0]	Triangle

##### 4.1.4.1 Reti Neurali Ricorrenti-RNN

Le RNN sono un potente e robusto tipo di reti neurali che negli ultimi anni stanno ottenendo molto successo poiché sono gli unici modelli con una memoria interna. Come gli altri modelli di deep learning anche le reti neurali ricorrenti sono relativamente vecchie. Difatti vennero inizialmente create nel 1980 ma solo nel recente periodo abbiamo visto il loro reale potenziale grazie alla mole di dati a nostra disposizione e grazie all'aumento della potenza computazionale. Questi tipi di algoritmi,



per via della loro capacità di ricordare informazione dagli input ricevuti, vengo adottati per risolvere problemi in cui sono richiesti dati sequenziali come audio, frasi, video, dati finanziari e molto altro. Non a caso sono molto usati nel natural language processing NLP. Oltre ad essere usati per task di NLP, questi algoritmi sono utili anche per i task di riconoscimento dell'attività o di classificazione della posa. Infatti nel caso dello yoga il contesto o l'informazione della posa intermedia o iniziale è importante per predire la posa finale. Questo perché una posa si compone di più azioni eseguite nel tempo, pertanto per determinare la posa che si sta eseguendo è utile comprendere l'insieme delle azioni e non la singola azione. Per poter mantenere la memoria interna le RNN a differenza di quelle feed-forward hanno dei loop al loro interno che permettono alle singole informazioni dedotte da un input di persistere nella rete.

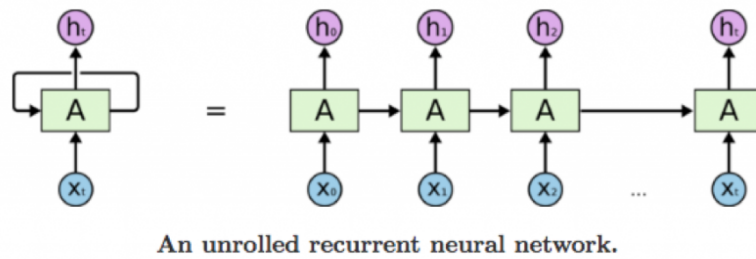


Figura 4.11: Rete neurale srotolata

In particolare, avremo che l'output all' $i$ -esimo input non dipende solo da questo ma anche dagli input precedenti. Ossia l'output non dipende solo dal vettore in input e dai relativi pesi associati ma anche dai pesi associati ad un vettore nascosto che tiene traccia dei precedenti input e output della rete

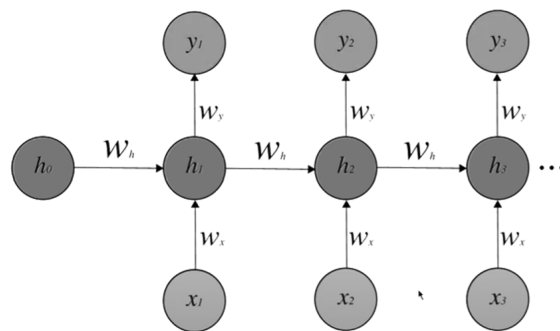


Figura 4.12: Rete neurale ricorrente

Le reti RNN soffrono però di due problemi l'esplosione del gradiente, che si verifica quando, per qualche motivo, l'algoritmo assegna dei pesi altissimi e la

scomparsa del gradiente che si verifica quando il gradiente assume un valore nullo o asintotico allo zero impedendo così l'aggiornamento dei pesi. In particolare, questo ultimo problema non permette alle RNN di preservare dipendenze a lungo termine.

Esisto 4 architetture di RNN utilizzate a seconda del compito da svolgere:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

Passando ora invece alla nostra architettura abbiamo che la rete è così strutturata:

```

1 Model: "sequential"
2 -----
3 Layer (type)                Output Shape                Param #
4 =====
5 simple_rnn (SimpleRNN)      (None, 45, 64)              7424
6
7 simple_rnn_1 (SimpleRNN)    (None, 45, 128)             24704
8
9 simple_rnn_2 (SimpleRNN)    (None, 64)                   12352
10
11 dense (Dense)               (None, 64)                   4160
12
13 dense_1 (Dense)             (None, 32)                   2080
14
15 dense_2 (Dense)             (None, 6)                    198
16
17 =====
18 Total params: 50,918
19 Trainable params: 50,918
20 Non-trainable params: 0
21 -----

```

In tutti i layer fatta eccezione per l'ultimo abbiamo utilizzato come funzione di attivazione la funzione ReLu mentre nell'ultimo layer abbiamo come funzione di attivazione la funzione Softmax che ci permette di attribuire ad ogni label una likelihood(probabilità). Come funzione di loss abbiamo utilizzato la categorical cross – entropy anche chiamata softmax-loss essendo questo tipo di loss usata per la classificazione multi-classe.

Mentre come ottimizzatore è stato usato Adam.

La rete è stata addestrata per 50 epoche.

I risultati della nostra fase di test sono descritti dal risultato delle seguenti metriche:

- accuracy: 0.9596153846153846
- precision: 0.96342504883462
- recall: 0.9566489222388098
- f1: 0.9586039392372435

Di seguito rappresentiamo la matrice di confusione che ci permette di avere una panoramica completa dei risultati ottenuti:

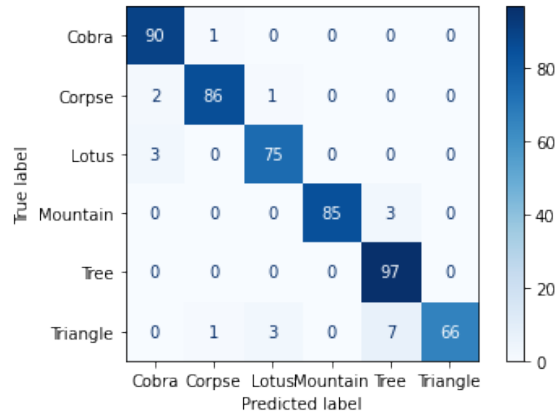


Figura 4.13: Matrice di confusione per RNN

#### 4.1.4.2 Long Short Term Memory-LSTM

Il problema della non dipendenza a lungo termine viene superata con un altro tipo di reti neurali ricorrenti le Long short Term memory LSTM, ossia un'estensione delle reti neurali appena descritte. Le LSTM consentono agli RNN di ricordare gli input per un lungo periodo di tempo. Questo perché gli LSTM contengono informazioni in una memoria, proprio come la memoria di un computer. L'LSTM può leggere, scrivere e cancellare informazioni dalla sua memoria. Questa memoria può essere vista come una cella dotata di "gate"(porte), ossia la cella decide se archiviare o eliminare informazioni (ovvero, se apre le porte o meno), in base all'importanza che assegna all'informazione. L'attribuzione dell'importanza avviene tramite pesi, anch'essi appresi dall'algoritmo. Ciò significa semplicemente che apprende nel tempo quali informazioni sono importanti e quali no. In una cella di memoria ci sono tre porte:

1. Input gate

2. Forget gate

3. Output gate

Queste porte determinano se consentire o meno l'ingresso di nuovi input (gate di input), eliminare le informazioni perché non sono importanti (forget gate) o lasciare che incidano sull'output nella fase temporale corrente (gate di output).

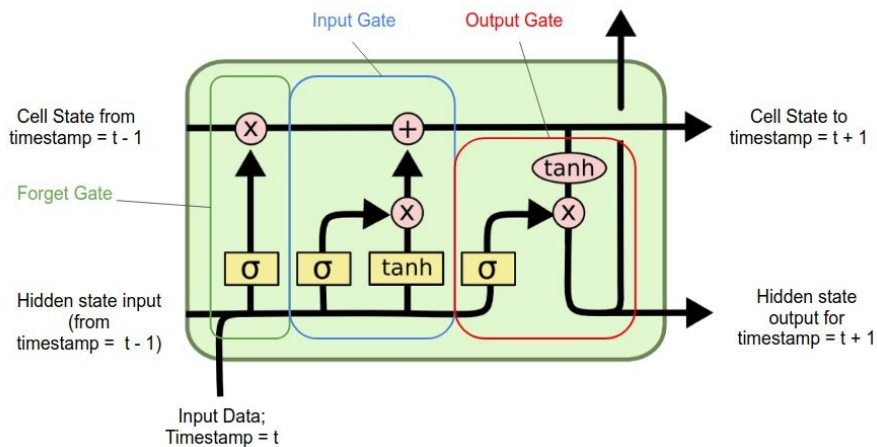


Figura 4.14: Cella LSTM

Passando ora invece alla nostra architettura abbiamo che la rete è così strutturata:

```

1 Model: "sequential_1"
2 -----
3 Layer (type)                Output Shape                Param #
4 =====
5 lstm (LSTM)                  (None, 45, 64)              29696
6
7 lstm_1 (LSTM)                (None, 45, 128)             98816
8
9 lstm_2 (LSTM)                (None, 64)                   49408
10
11 dense_3 (Dense)              (None, 64)                   4160
12
13 dense_4 (Dense)              (None, 32)                   2080
14
15 dense_5 (Dense)              (None, 6)                    198
16
17 =====
18 Total params: 184,358
19 Trainable params: 184,358

```

20 `Non-trainable params: 0`

21 -----

In tutti i layer fatta eccezione per l'ultimo abbiamo utilizzato come funzione di attivazione la funzione ReLu mentre nell'ultimo layer abbiamo come funzione di attivazione la funzione Softmax che ci permette di attribuire ad ogni label una likelihood(probabilità). Come funzione di loss abbiamo utilizzato la categorical cross – entropy anche chiamata softmax-loss. Mentre come ottimizzatore è stato usato Adam. La rete è stata addestrata per 50 epoche.

I risultati della nostra fase di test sono descritti dal risultato delle seguenti metriche:

- accuracy: 0.9538461538461539
- precision: 0.9567968673965136
- recall: 0.9528346466910698
- f1: 0.9536375894472574

Di seguito rappresentiamo la matrice di confusione che ci permette di avere una panoramica completa dei risultati ottenuti:

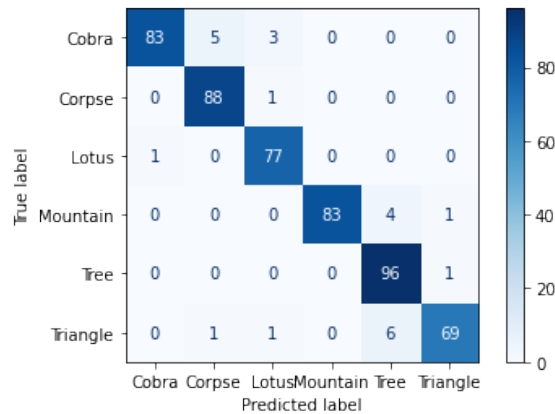


Figura 4.15: Matrice di confusione per LSTM

#### 4.1.4.3 Gated Recurrent Unit-GRU

Nel nostro lavoro abbiamo preso anche in considerazione una variante dell'LSTM ossia la rete GRU Gated Recurrent Unit. All'interno di questa rete come suggerisce il nome il forget gate e l'input gate sono combinate in unico Gate. Quindi Combina lo stato e l'output della cella Questa rete risulta essere più veloce rispetto alle

reti LSTM avendo meno parametri ma risulta anche essere meno potente per il medesimo motivo.

Passando ora invece alla nostra architettura abbiamo che la rete è così strutturata:

```

1 Model: "sequential_2"
2 -----
3 Layer (type)                Output Shape                Param #
4 -----
5 gru (GRU)                   (None, 45, 64)             22464
6
7 gru_1 (GRU)                 (None, 45, 128)            74496
8
9 gru_2 (GRU)                 (None, 64)                 37248
10
11 dense_6 (Dense)            (None, 64)                 4160
12
13 dense_7 (Dense)            (None, 32)                 2080
14
15 dense_8 (Dense)            (None, 6)                 198
16
17 =====
18 Total params: 140,646
19 Trainable params: 140,646
20 Non-trainable params: 0
21 -----

```

In tutti i layer fatta eccezione per l'ultimo abbiamo utilizzato come funzione di attivazione la funzione ReLu mentre nell'ultimo layer abbiamo come funzione di attivazione la funzione softmax che ci permette di attribuire ad ogni label una likelihood(probabilità). Come funzione di loss abbiamo utilizzato la categorical cross – entropy anche chiamata softmax-loss. Mentre come ottimizzatore è stato usato Adam. La rete è stata addestrata per 50 epoche.

I risultati della nostra fase di test sono descritti dal risultato delle seguenti metriche:

- accuracy: 0.9711538461538461
- precision: 0.9719481862223797
- recall: 0.9720634519154733
- f1: 0.9716922627502504

Di seguito rappresentiamo la matrice di confusione che ci permette di avere una panoramica completa dei risultati ottenuti:

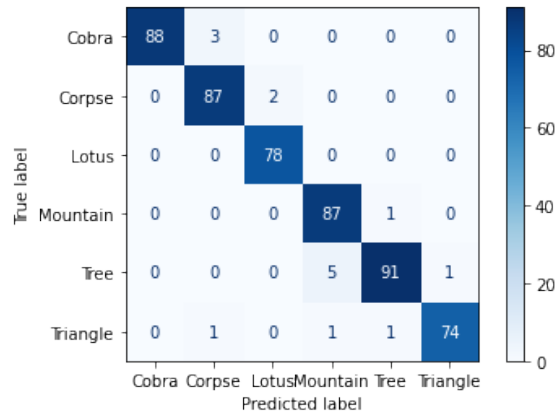


Figura 4.16: Matrice di confusione per GRU

### 4.1.5 Confronto riassuntivo

Dopo aver addestrato i vari modelli li abbiamo messi a confronto con una tabella riassuntiva e con una serie di grafici che confronteranno le singole metriche per ogni modello. Come si evince dai grafici il modello migliore è risultato il modello SVM, sebbene nel complesso si siano comportati tutti ottimamente, se non allo stesso modo, con l'unica discriminante che i modelli di machine learning hanno impiegato una quantità di risorse(tempo) minima rispetto ai modelli deep.

	Model	Test Accuracy	Precision	Recall	F1
0	SVM	0.983	0.984	0.983	0.983
1	Random Forest	0.979	0.981	0.978	0.979
4	GRU	0.971	0.972	0.972	0.972
2	RNN	0.960	0.963	0.957	0.959
3	LSTM	0.954	0.957	0.953	0.954

### 4.1.6 Osservazioni

Come abbiamo notato tutti i modelli, a partire da quelli più semplici, fino ad arrivare a quelli deep hanno prodotto dei risultati fin troppo ottimi; pertanto, potremmo essere andati incontro al fenomeno dell'over-fitting.

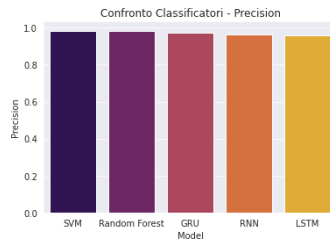


Figura 4.17: Confronto Precision

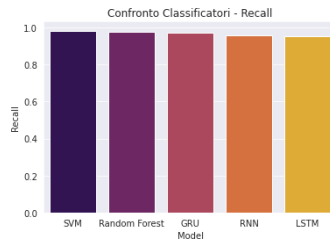


Figura 4.18: Confronto Recall

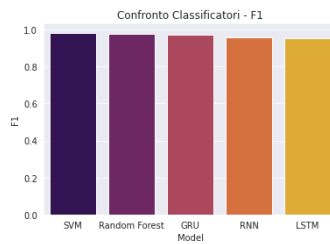


Figura 4.19: Confronto F-1

Per valutare quindi in maniera più corretta la situazione proviamo a riaddestrate e trainare i modelli utilizzando questa volta la tecnica del Leave One Person Out. La tecnica del leave one person out si basa sull'idea di lasciare dal set di training una o più persone fuori e effettuare il test sui samples di quella persona. Pertanto, per fare questo dovremo prima ottenere gli indici sui quali dividere il set di training, in modo tale da lasciare fuori una sola persona. Una volta ottenuti gli indici riaddestriamo i modelli.

Come si può notare dalla tabella riassuntiva i risultati ottenuti sono equivalenti a quelli ottenuti precedentemente quindi, possiamo affermare che prima i nostri modelli non soffrivano di over-fitting e lo si può appurare anche dal fatto che i modelli funzionano bene anche in real-time.

Come ultima prova proviamo ad addestrate una rete neurale ricorrente(abbiamo scelto la GRU essendosi comportata meglio rispetto alle altre reti) effettuando uno shuffle sul nostro dataset poiché, la struttura sequenziale del nostro dataset



	Model	Test Accuracy	Precision	Recall	F1
0	SVM	0.982	0.983	0.985	0.984
1	Random Forest	0.976	0.978	0.973	0.974
4	GRU	0.945	0.937	0.932	0.933
3	LSTM	0.945	0.937	0.932	0.934

potrebbe influenzare i risultati dei nostri modelli deep. Anche in questo caso però abbiamo ottenuto i medesimi risultati, se non di un punto percentuale migliore.

- accuracy: 0.9788461538461538
- precision: 0.9793012224046707
- recall: 0.9788738665867126
- f1: 0.9788738665867126

## 4.2 Fase 2: Correzione

All'interno di questa sezione verrà descritta l'intera pipeline di lavoro che ha portato al risultato ottenuto nella seconda fase, a partire dalla creazione di nuovi datasets fino ad arrivare alla soluzione adottata per la correzione. In particolare, ogni scelta presa verrà spiegata in maniera approfondita con apposite regressioni di natura teorica ove necessario.

### 4.2.1 Dataset

#### 4.2.1.1 Creazione del dataset

Per la correzione della posa si è deciso di non utilizzare il dataset creato per la fase precedente ma di creare un dataset apposito per ogni posa in modo tale che fosse più facile lavorare con dataset di dimensioni ridotte e con una struttura più facile da navigare. Inoltre, lavorare con un dataset di dimensioni ridotte comporta sicuramente anche un minor costo computazionale. Più in dettaglio siamo passati da un dataset avente la seguente struttura:

[Vettore contenente tutti i video composti da 45 frame[Vettore contenente 45 vettori di keypoint[vettore contenente 51 punti 17 punti 3(x,y,confidence)]]]

Quindi con una shape di questo tipo (2599, 45, 51).

Ad un dataset avente la seguente composizione per ogni posa:

[Vettore contenete tutti i frame[vettore contenente 51 punti 17 punti 3(x,y,confidence)]]  
Quindi con una shape di questo tipo (17517, 51) N.B. (17517, 51) è la shape per

il dataset della posa triangle, il primo valore indica il numero di frames pertanto varierà da posa a posa.

In seguito a questa fase avremo quindi i seguenti dataset:

dataset_Posa	# Frames	#Key point
Cobra	20721	51
Corpse	20492	51
Lotus	17884	51
Mountain	20061	51
Tree	22204	51
Triangle	17517	51

Per poter semplificare l'accesso ai valori x e y per ogni keypoint effettuiamo la reshape da (#frames, 51)  $\rightarrow$  (#frames, 17, 3)

### 4.2.2 Calcolo degli angoli

Il sistema di correzione proposto si basa sulla correttezza degli angoli. Pertanto, definiamo un dizionario in cui andiamo ad individuare quelli che sono gli angoli più importanti durante l'esecuzione di tutti gli esercizi. Gli angoli saranno individuati da una terna di valori dove ogni valore fa riferimento al joint rispettivo; in totale sono stati individuati 8 angoli.

```

ANGL_DICT = {
    'angl_left_elbow': [10,8,6],
    'angl_left_shoulder': [8,6,12],
    'angl_left_hip': [6,12,14],
    'angl_left_knee': [12,14,16],
    'angl_right_elbow': [9,7,5],
    'angl_right_shoulder': [7,5,11],
    'angl_right_hip': [5,11,13],
    'angl_right_knee': [11,13,15],
}

```

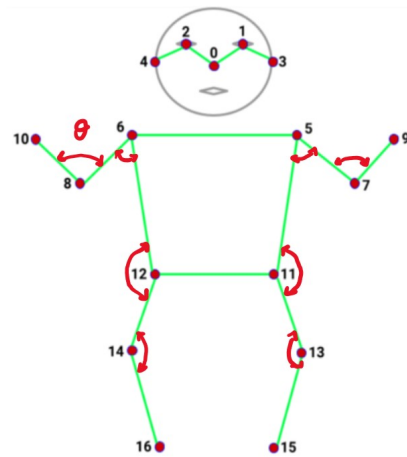


Figura 4.20: Dizionario 8 angoli

Per calcolare gli angoli definiamo una funzione in cui sfruttiamo la seguente funzione trigonometrica

$$radians = \text{arctan2}(a1 - o1) - \text{arctan2}(a2 - o2)$$

Dove  $a1$  è la  $x$  (l'ascissa del primo punto),  $o1$  è la  $y$  (l'ordinata del primo punto),  $a2$  è la  $x$  del secondo punto e  $o2$  è la  $y$  del secondo punto. Successivamente si è ricavato il valore dell'angolo in gradi utilizzando la seguente formula:

Dopo di che definiamo un funzione che calcoli per ogni frame, che sarà rappresentato da un vettore(17,3), tutti gli angoli e li inserisca all'interno di una lista.

$$angle = \left\lfloor \frac{radians * 180.00}{\pi} \right\rfloor$$

### 4.2.3 KNN-Angoli corretti

Per poter individuare l'angolo corretto per ogni posa, che poi fungerà da punto di riferimento per la correzione, abbiamo deciso di utilizzare il K Nearest Neighbors KNN.

In generale il KNN dato un nuovo sample individua sulla base di una metrica di similarità, come può essere la similarità del coseno, i K samples più simili al campione sotto-esame e sulla base dei valori della features-target di questi neighbors effettua la sua predizione.

Nel nostro caso abbiamo utilizzato il KNN solamente come ricerca e abbiamo utilizzato come misura di similarità la misura minkowski (la misura di default). Dati quindi i risultati del KNN nel caso K sia stato definito uguale ad 1 abbiamo utilizzato di conseguenza la funzione prima descritta per calcolare gli angoli mentre, nel caso in cui K sia stato definito con un valore maggiore di 1, verrà calcolato il centroide dei vettori neighbors e sulla base di questo verranno calcolati gli angoli.

### 4.2.4 Sistema di feedback

In seguito, abbiamo definito una funzione che andasse a confrontare gli angoli individuati in seguito al KNN, con gli angoli che si stanno effettivamente eseguendo e, nel caso in cui gli angoli non siano corretti, dove per corretti si intende superiori o inferiori rispetto a un valore di threshold definito nei parametri della funzione, verrà fornito un feedback all'utente che suggerirà una o più correzioni sulla base di quanti angoli non siano stati rispettati. In caso la posizione venga eseguita in maniera corretta verrà riferito all'utente che sta svolgendo la posizione in maniera corretta.

Per ogni posa sono stati definiti dei feedback differenti descritti dalle seguenti funzioni:

```
1 def correct_angles_cobra(threshold, angles_vid, angles_u):
```

```
2     correct_angles=0
3     for i in range(len(angles_vid)):
4         diff=abs(angles_vid[i][1] - angles_u[i][1])
5         if diff > threshold:
6             if i == 0 or i == 4:
7                 print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
8                 print("Distendi le braccia")
9                 correct_angles+=1
10            elif i == 2 or i == 5:
11                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
12                print("Mantieni le mani larghezza spalle")
13                correct_angles+=1
14            elif i == 3 or i == 6:
15                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
16                print("Mantieni le gambe larghezza spalle")
17                correct_angles+=1
18            elif i == 4 or i == 7:
19                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
20                print("Distendi le gambe")
21            if correct_angles==0 :
22                print("posizione corretta")

1 def correct_angles_corpse(threshold, angles_vid, angles_u):
2     correct_angles=0
3     for i in range(len(angles_vid)):
4         diff=abs(angles_vid[i][1] - angles_u[i][1])
5         if diff > threshold:
6             if i == 0 or i == 4:
7                 print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
8                 print("Distendi meglio le braccia")
9                 correct_angles+=1
10            elif i == 2 or i == 5:
11                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
12                print("Disponi le braccia lungo i fianchi")
13                correct_angles+=1
14            elif i == 3 or i == 6:
15                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
16                print("Mantieni le gambe larghezza spalle")
17                correct_angles+=1
18            elif i == 4 or i == 7:
```

```
19         print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
20         print("Distendi meglio le gambe")
21         correct_angles+=1
22     if correct_angles==0 :
23         print("posizione corretta")

1 def correct_angles_lotus(threshold, angles_vid, angles_u):
2     correct_angles=0
3     for i in range(len(angles_vid)):
4         diff=abs(angles_vid[i][1] - angles_u[i][1])
5         if diff > threshold:
6             if i == 0 or i == 4:
7                 print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
8                 print("Distendi meglio le braccia")
9                 correct_angles+=1
10            elif i == 2 or i == 5:
11                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
12                print("Mantenendo il busto dritto lascia cadere le
braccia")
13                correct_angles+=1
14            elif i == 3 or i == 6:
15                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
16                print("Mantieni le gambe larghezza spalle")
17                correct_angles+=1
18            elif i == 4 or i == 7:
19                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
20                print("Incrocia meglio le gambe")
21                correct_angles+=1
22        if correct_angles==0 :
23            print("posizione corretta")

1 def correct_angles_mountain(threshold, angles_vid, angles_u):
2     correct_angles=0
3     for i in range(len(angles_vid)):
4         diff=abs(angles_vid[i][1] - angles_u[i][1])
5         if diff > threshold:
6             if i == 0 or i == 4:
7                 print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
8                 print("Distendi meglio le braccia")
9                 correct_angles+=1
```

```

10         elif i == 2 or i == 5:
11             print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
12             print("Tira maggiormente le braccia verticalmente"
)
13             correct_angles+=1
14         elif i == 3 or i == 6:
15             print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
16             print("Mantieni le gambe larghezza spalle")
17             correct_angles+=1
18         elif i == 4 or i == 7:
19             print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
20             print("Distendi meglio le gambe e sali maggiormente
sulle punte dei piedi")
21             correct_angles+=1
22         if correct_angles==0 :
23             print("posizione corretta")

1 def correct_angles_tree(threshold, angles_vid, angles_u):
2     correct_angles=0
3     for i in range(len(angles_vid)):
4         diff=abs(angles_vid[i][1] - angles_u[i][1])
5         if diff > threshold:
6             if i == 0 or i == 4:
7                 print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
8                 print("Unendo le mani cerca di formare un angolo
retto con il gomito")
9                 correct_angles+=1
10            elif i == 2 or i == 5:
11                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
12                print("Non tenere i gomiti esternamente ma lasciali
vicino al corpo")
13                correct_angles+=1
14            elif i == 3 or i == 6:
15                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
16                print("Mantieni le gambe larghezza spalle piegandone
una")
17                correct_angles+=1
18            elif i == 4 or i == 7:
19                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
20                print("Distendi meglio una gamba e cerca di portare
il piede in prossimit del ginocchio")

```

```
21         correct_angles+=1
22     if correct_angles==0 :
23         print("posizione corretta")

1 def correct_angles_triangle(threshold, angles_vid, angles_u):
2     correct_angles=0
3     for i in range(len(angles_vid)):
4         diff=abs(angles_vid[i][1] - angles_u[i][1])
5         if diff > threshold:
6             if i == 0 or i == 4:
7                 print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
8                 print("Distedi meglio le braccia ")
9                 correct_angles+=1
10            elif i == 2 or i == 5:
11                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
12                print("Disponi le braccia orizzontalmente e cerca
di piegarti maggiormente verso la punta del piede")
13                correct_angles+=1
14            elif i == 3 or i == 6:
15                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
16                print("Metti le gambe un po' pi larghe rispetto
alle spalle")
17                correct_angles+=1
18            elif i == 4 or i == 7:
19                print("Angolo {} sbagliato di: {} gradi".format(
angles_vid[i][0], diff))
20                print("Mantieni le gambe distese")
21                correct_angles+=1
22        if correct_angles==0 :
23            print("posizione corretta")
```

## 4.3 Fase 3: Web-App

In questa sezione verrà descritta la piattaforma sulla quale si è deciso di implementare il sistema prima descritto fornendo una panoramica generale della struttura del progetto e spiegando le varie scelte progettuali.

### 4.3.1 Descrizione del progetto

Il sistema realizzato è una web app realizzata principalmente con Flask per gestire la parte di back-end, Html e CSS per gestire la parte di front-end mentre per gestire

i dati è stato utilizzato SQLAlchemy e come dialetto di SQL è stato usato SQLite. Strumenti già descritti nei primi capitoli di questo lavoro di tesi.

### **4.3.2 System design**

Per avere una visione più completa del progetto vedere la repository "REPOSITORY"

#### **4.3.2.1 Model-View-Controller**

Da un punto di vista strutturale e progettuale si è deciso di adottare come stile architetturale ossia come system design il pattern Model-View-Controller, separando la logica di business(model) da quella di rappresentazione dei dati(view).

- La parte Model si compone dall'insieme di classi che rappresentano il dominio dell'applicazione.
- La parte view si compone dai file html e css che si occupano dell'interazione con l'attore.
- La parte Controller si compone invece dei file che gestiscono la logica di business tra cui i file che gestiscono le routes. Dove le routes sono gli url associati a specifiche funzioni.

In questo tipo di sistema avremo che le classi facenti parte della parte Model possono comunicare tra loro e con le classi facenti parte della parte Control. Le classi-file facenti parte della parte view possono comunicare tra loro e con le classi Control. Le classi-file facenti parte della parte Control possono comunicare con tutte le classi/file ad eccezione che con gli attori.

### **4.3.3 Funzionalità**

Passando ora invece alle funzionalità della nostra piattaforma queste verranno descritte verranno descritte via via che le pagine di cui si compone la web-app verranno presentate.

#### **4.3.3.1 Pagina Login**

Per utilizzare effettivamente le funzionalità del nostro sistema bisogna dapprima effettuare il login inserendo e-mail e password nel caso sia stata già effettuata la registrazione.



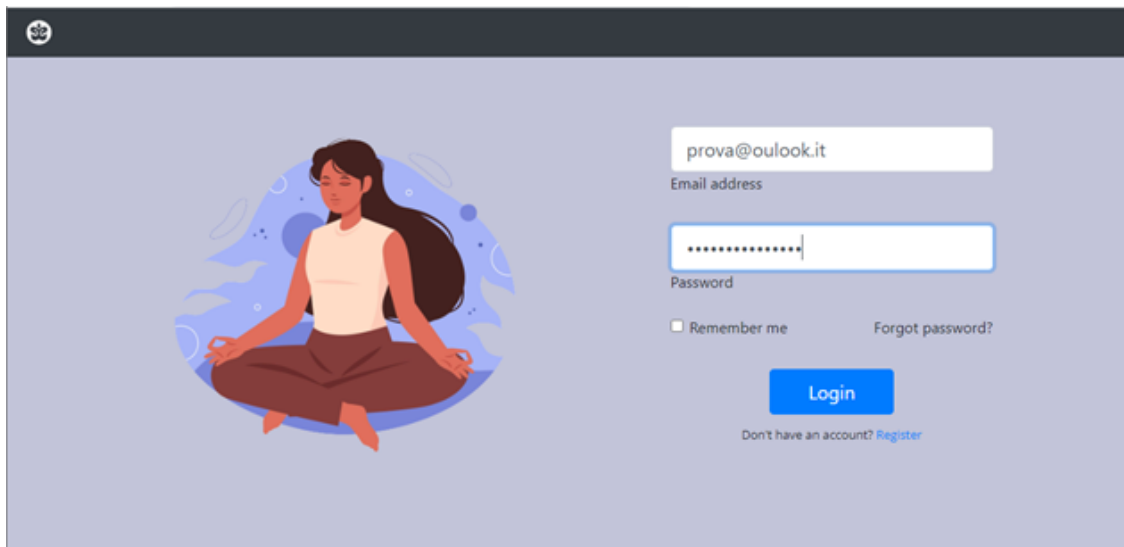


Figura 4.21: Pagina della web-app dedicata al login

#### 4.3.3.2 Pagina Sign-In

Nel caso invece in cui non sia stata già effettuata la registrazione bisogna effettuarla tramite l'apposita sezione

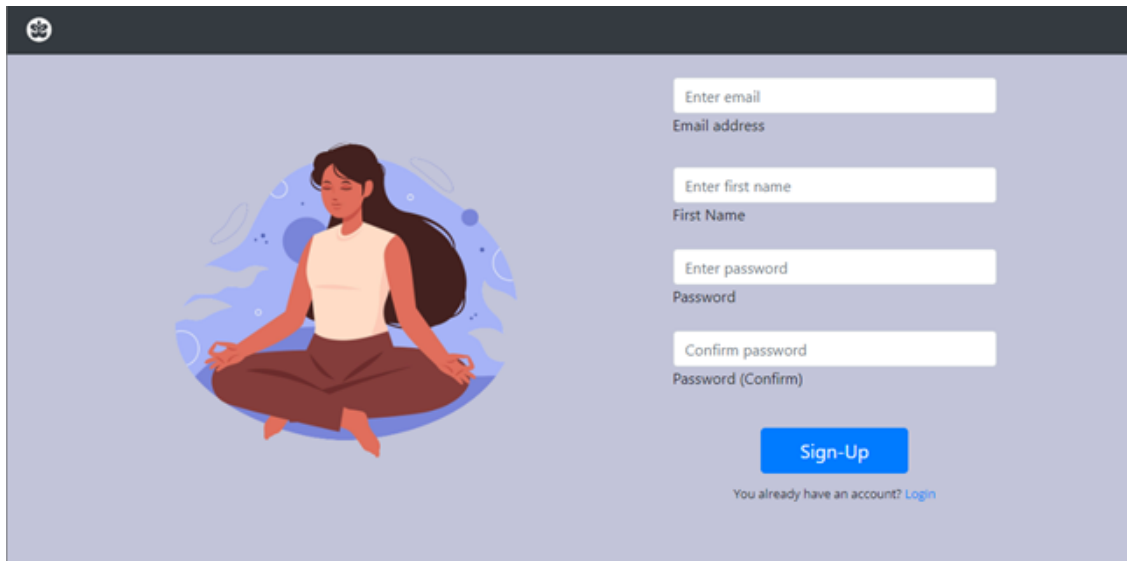


Figura 4.22: Pagina della web-app dedicata al sign-in

#### 4.3.3.3 Pagina Home

Una volta effettuato il login ci verrà presentata la home page dove ci sarà una breve descrizione del servizio che la piattaforma offre e i seguenti pulsanti:

- Home
  - Questo pulsante presente all'interno della nav bar ci permetterà di tornare alla home page
- Logout
  - Questo pulsante presente all'interno della nav bar ci permetterà di effettuare come suggerisce il logout e quindi torneremo alla pagina dedicata al login
- Sessione Privata
  - Questo pulsante ci permetterà di essere trasportati alla pagina dedicata alla gestione della propria sessione di yoga

#### 4.3.3.4 Pagina Sessione Privata

Una volta pigiato il pulsante sessione privata verremo trasportati nella pagina dedicata alla gestione della propria sessione di yoga dove avremo i seguenti pulsanti:



Figura 4.23: Pagina della web-app dedicata alla home

- Home
  - Questo pulsante presente all'interno della nav bar ci permetterà di tornare alla home page
- Logout
  - Questo pulsante presente all'interno della nav bar ci permetterà di effettuare come suggerisce il logout e quindi torneremo alla pagina dedicata al login
- Crea routine
  - Questo pulsante ci permetterà di essere trasportati alla pagina dedicata alla creazione di una routine di posizioni di yoga.
- Inizia la sessione
  - Questo pulsante ci permetterà di iniziare direttamente la sessione di yoga senza definire una routine

Inoltre, i pulsanti crea routine e inizia la sessione saranno accompagnati da una descrizione e da un'illustrazione che ne spiega il funzionamento

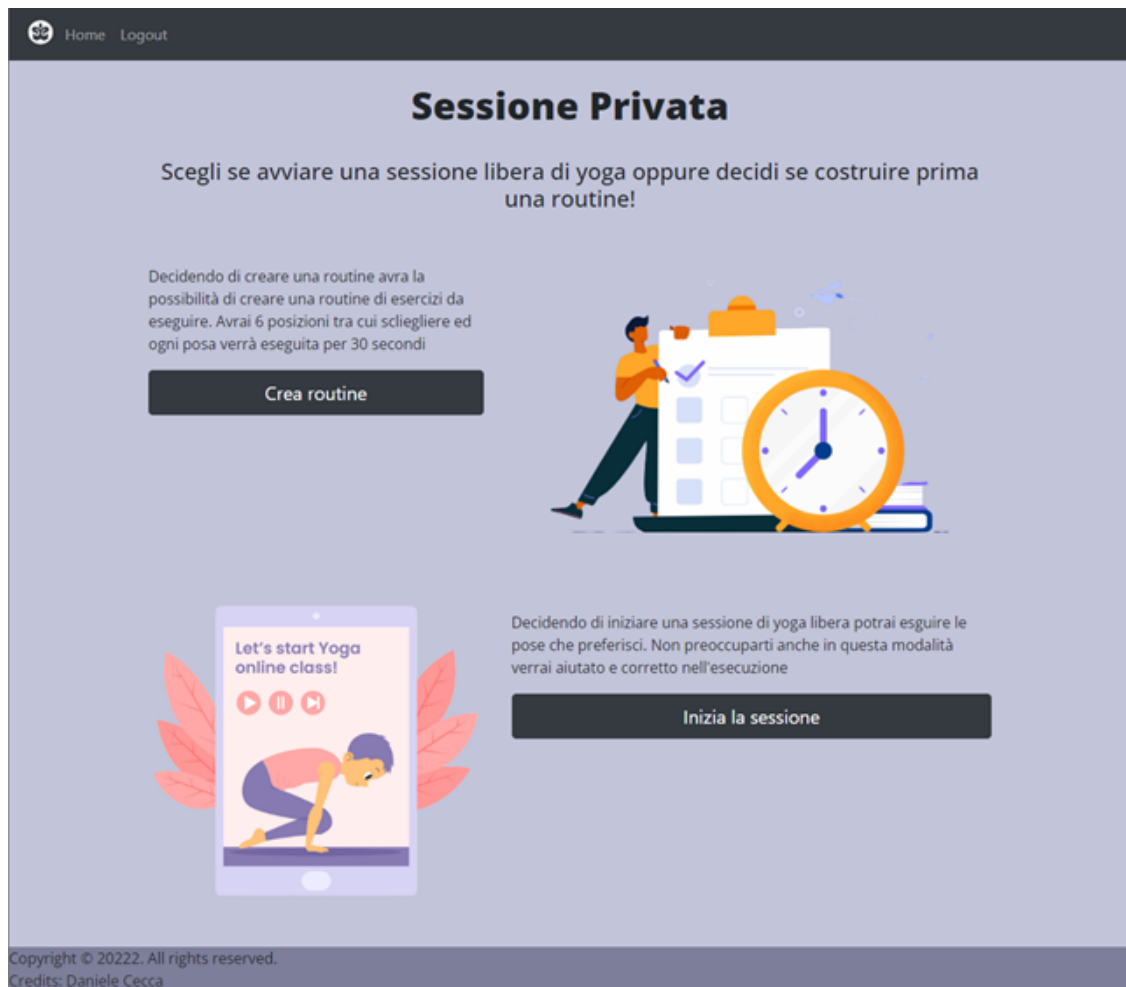


Figura 4.24: Pagina della web-app dedicata alla sessione privata

#### 4.3.3.5 Pagina Crea Routine

Una volta pigiato il pulsante crea routine verremo trasportati nella pagina dedicata alla creazione della routine dove avremo i seguenti pulsanti:

- Home
  - Questo pulsante presente all'interno della nav bar ci permetterà di tornare alla home page
- Logout
  - Questo pulsante presente all'interno della nav bar ci permetterà di effettuare come suggerisce il logout e quindi torneremo alla pagina dedicata al login

- Aggiungi esercizio
  - Questo pulsante ci permetterà di aggiungere un esercizio alla nostra routine
- (X) Eliminare esercizio
  - Questo pulsante ci permetterà di eliminare un esercizio dalla nostra routine
- Inizia la sessione
  - Questo pulsante ci permetterà di iniziare la sessione di yoga dopo aver definito una routine

Per facilitare la comprensione della posa che si sta inserendo all'interno della routine per ogni posa è stata inserita una illustrazione.

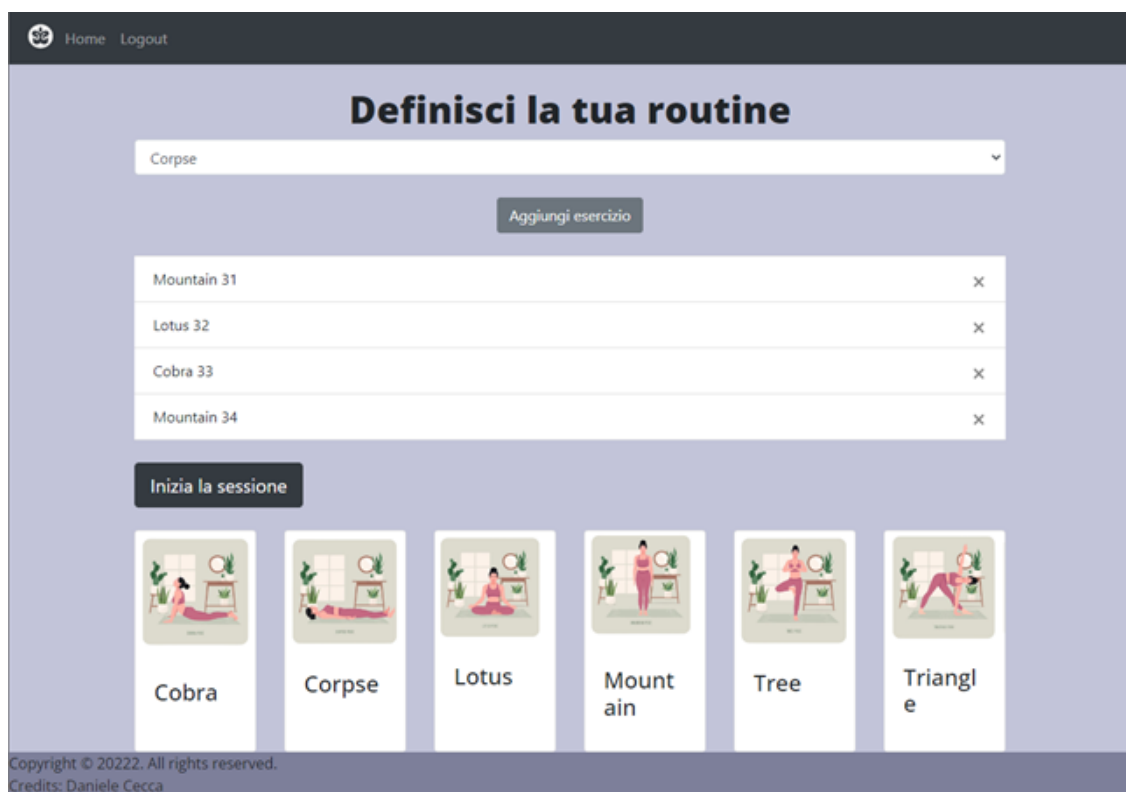


Figura 4.25: Pagina della web-app dedicata alla creazione della routine

#### 4.3.3.6 Pagina Sessione Live-Libera

Dopo aver pigiato il inizia sessione nella pagina sessione privata verremo trasportati nella pagina in cui avverrà effettivamente la sessione di yoga e quindi dove verrà erogato il servizio descritto nella pagina iniziale. In particolare, in questa pagina avverrà sia la classificazione che la correzione della posa descritti rispettivamente nelle fasi 1 e 2. All'interno di questa pagina verrà mostrata una finestra in cui ci sarà il nostro video streaming mentre eseguiamo le pose. Per ogni posa che stiamo eseguendo avverrà una classificazione che ci dirà in tempo reale che posizione stiamo svolgendo e in base alla posizione che stiamo eseguendo ci verranno proposte una serie di correzioni nel caso la posa non sia eseguita in maniera corretta.

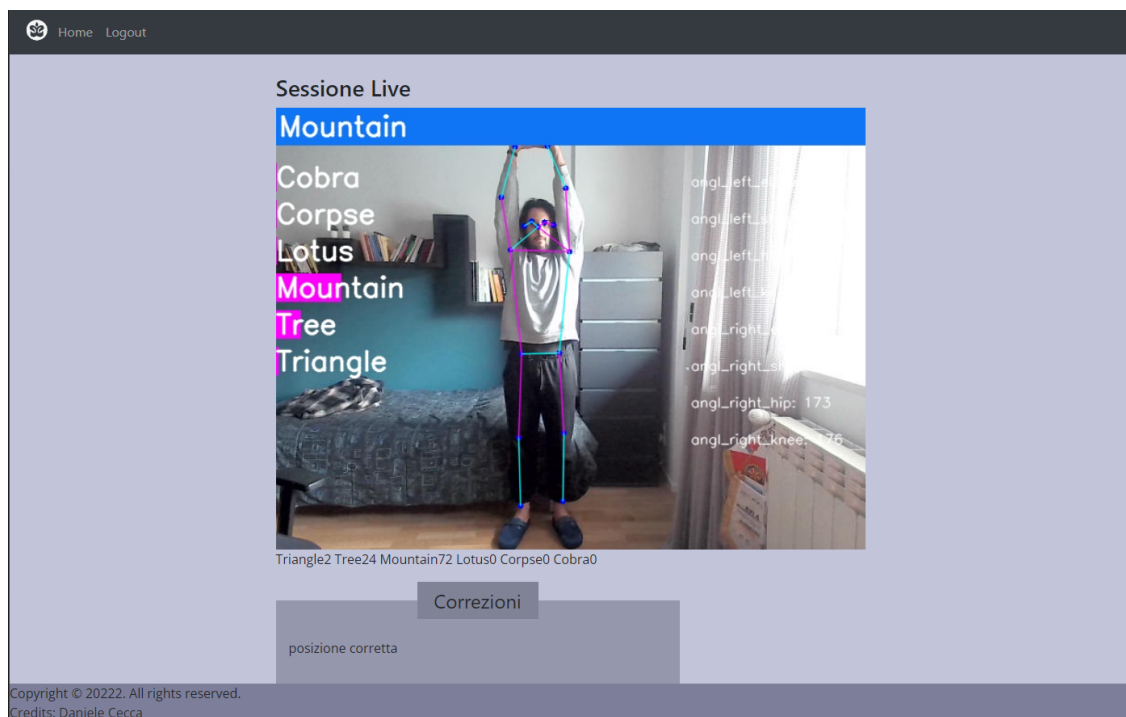


Figura 4.26: Pagina della web-app dedicata alla sessione privata libera

#### 4.3.3.7 Pagina Sessione Live-Routine

Dopo aver pigiato il pulsante inizia sessione nella pagina crea routine verremo trasportati nella pagina in cui avverrà effettivamente la sessione di yoga e quindi dove verrà erogato il servizio descritto nella pagina iniziale. In particolare, in questa pagina a differenza della pagina precedente, avverrà solamente la correzione della posa che si sta eseguendo. In questa modalità non saremo liberi di svolgere qualsiasi posa ma dovremo svolgere la posa presente nella nostra routine. Inoltre, una posa

si susseguirà all'altra dopo 30 sec, rappresentati da un timer mostrato a schermo . Al termine della routine verremo riportati nella home.Come per la pagina prima descritta anche in questa verranno mostrati gli errori commessi.

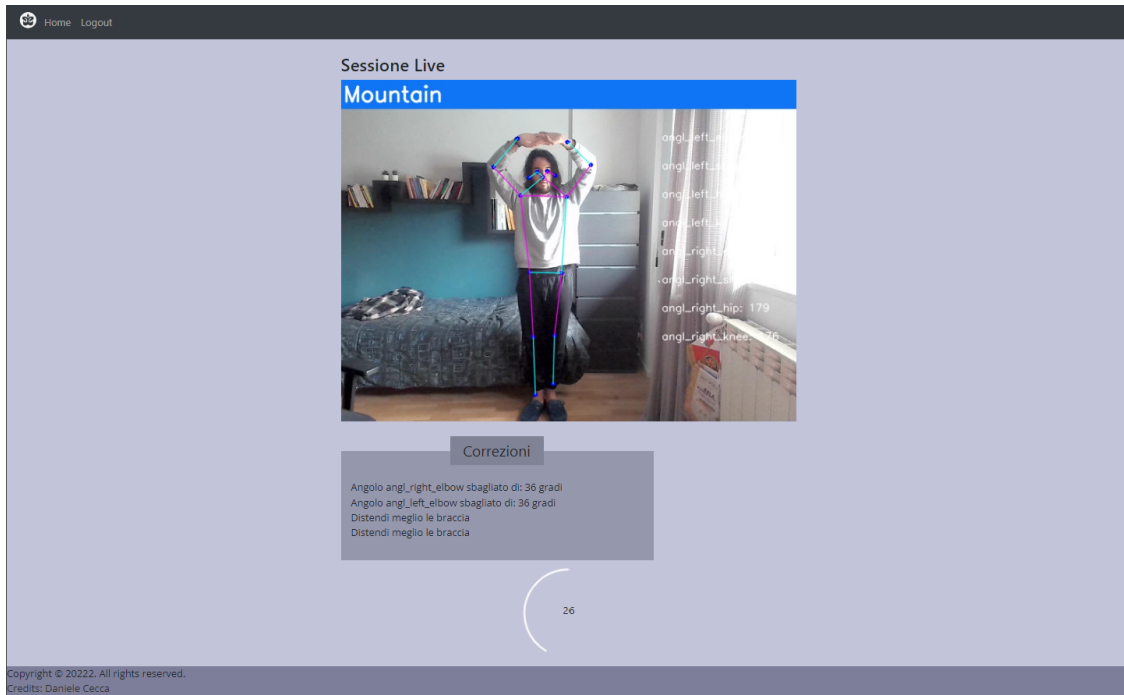


Figura 4.27: Pagina della web-app dedicata alla sessione privata routine





# Capitolo 5

## Conclusioni e sviluppi futuri

### 5.1 Conclusioni

Il progetto nel suo complesso ha avuto un esito positivo sia per quanto riguarda la realizzazione del sistema di classificazione e correzione sia per la realizzazione della piattaforma web.

I risultati ottenuti dai modelli proposti per la classificazione risultano equiparabili a quelli dello stato dell'arte ed in particolare abbiamo ottenuto un risultato migliore, rispetto ai lavori analizzati, per il modello SVM e Random Forest. Inoltre abbiamo ottenuto questi risultati nonostante sia stato usato come modello per la Human Pose Estimation il modello MoveNet; un modello più recente rispetto a quelli già elencati (OpenPose, MediaPipe,..) e che individua meno keypoints, si veda infatti Mediapipe che individua 33 punti.

Per la prima volta per la correzione è stato usato un algoritmo KNN per individuare gli angoli corretti invece di calcolarli a priori o di stabilire semplicemente un threshold senza confrontarli prima con degli angoli corretti. Inoltre, a differenza di alcuni lavori, nel nostro sistema di classificazione vengono proposti feedback differenti a seconda dell'errore commesso.

#### 5.1.1 Sviluppi futuri

La piattaforma web, da me creata, è da considerarsi solo come una versione preliminare e di conseguenza potrebbe essere ulteriormente ampliata e raffinata, aggiungendo diverse funzionalità. Eventuali sviluppi futuri potrebbero orientarsi sui seguenti punti:

- Aggiungere una modalità istruttore che permetta all'istruttore di aggiungere delle nuove posizioni
- Aggiungere la possibilità di svolgere sessioni di yoga di gruppo svolte fisicamente

- Aggiungere la possibilità di svolgere sessioni di yoga di gruppo online creando delle stanze virtuali
- Migliorare l'interfaccia grafica rendendola maggiormente responsive
- Dare la possibilità di creare più routine per un solo utente
- Aggiungere più pose
- Migliorare i feedback per la correzione
- Rendere i feedback sonori
- Risolvere alcuni bug

# Ringraziamenti



Figura 5.1: Ringraziamenti



# Bibliografia

- [1] Francesco Ricci, Lior Rokach, and Bracha Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.