**Documentazione per la tesi**

**Abstract**

All’interno di questa tesi verrà discusso il lavoro svolto che ha portata 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 dateset 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.

**Introduzione**

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.

**Capitolo 1**

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.

All’interno di questi capitolo andremo a discutere alcuni delle soluzioni presenti in rete. Nella prima parte di questo capitolo ci concentreremo su alcune soluzioni, che presentano approcci differenti dal nostro e che sono già distribuite nel mercato. Mentre nella seconda parte ci concentreremo su alcuni paper scientifici che propongono soluzioni con un approccio simile al nostro.

**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, reccomendation 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]

**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 model(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.

**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.

**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 portano 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.

Immagine che contiene testo, interni, piastrellato, tegola

Descrizione generata automaticamente

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.

**Capitolo ?**

All’interno di questi capitolo andremo a discutere alcuni delle soluzioni presenti in rete. Nella prima parte di questo capitolo ci concentreremo su alcune soluzioni, che presentano approcci differenti dal nostro e che sono già distribuite nel mercato. Mentre nella seconda parte ci concentreremo su alcuni paper scientifici che propongono soluzioni con un approccio simile al nostro.

**Stato dell’arte**

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 classificazio.[25]

Nello studio proposto da [25] è stato proposto un sistema di coaching di toga 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](https://link.springer.com/article/10.1007/s11227-021-04076-w#ref-CR28),[29](https://link.springer.com/article/10.1007/s11227-021-04076-w#ref-CR29),[30](https://link.springer.com/article/10.1007/s11227-021-04076-w#ref-CR30),[31](https://link.springer.com/article/10.1007/s11227-021-04076-w#ref-CR31),[32](https://link.springer.com/article/10.1007/s11227-021-04076-w#ref-CR32)] 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 adotta 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 algortimo 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.

**Capitolo 3**

**Strumenti utilizzati**

In questa capitolo descriveremo gli strumenti e le tecnologie utilizzatw 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.

**Linguaggi e framework utilizzati**

Il progetto è stato sviluppato interamente attraverso il linguaggio Python.

Nell’ultima fase oltre ad esser 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

**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 progetta per la lavorare con le librerire NumPy e SciPy.

**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.

**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

**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)

Nonstante 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.

**Capitolo 2**

**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.

**Dataset**

**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 |

 Immagine che contiene persona, interni, pavimento, ragazza

Descrizione generata automaticamente 

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.

**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**

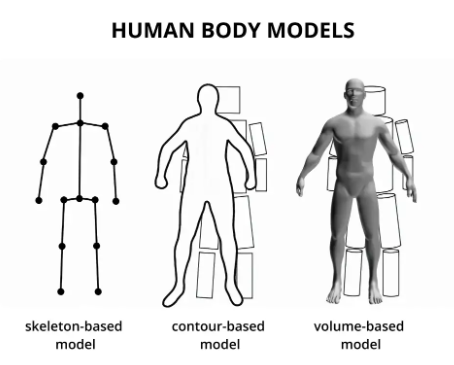
viene utilizzato per la stima della posa 2D e per la stima della posa 3D. Questo modello di corpo umano flessibile e intuitivo include una serie di posizioni articolari, joints e orientamenti degli arti per rappresentare la struttura del corpo umano. Pertanto, i modelli di stima della posa dello scheletro 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.

[11]

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. **Botton 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.

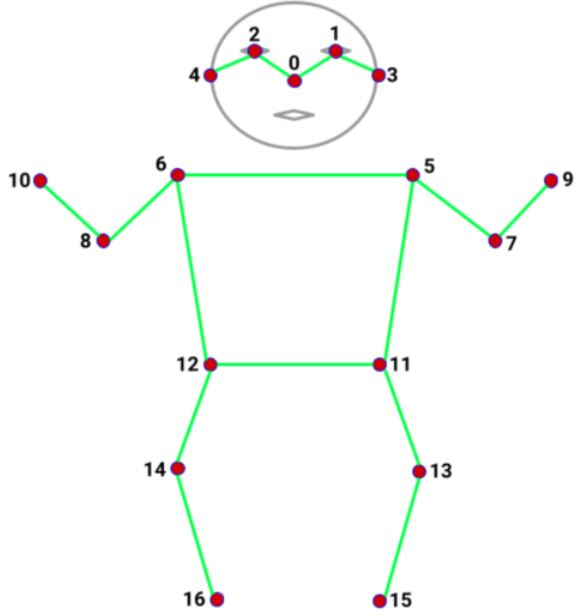
Si è deciso di usare Move-Net oltre che per le sue peculiarità quali la velocità nella rilevazione dei punti, la capacità di individuare i punti per più persone anche perché come vedremo è stato addestrato su un dataset contenente esercizi fisici e posizioni di yoga.

**Move-Net**

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.

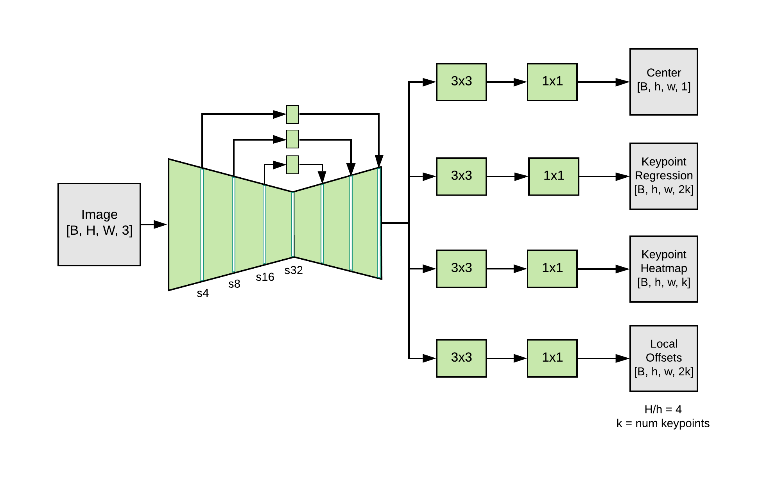
Entrambi i modelli funzionano più velocemente del tempo reale (oltre 30 FPS) sulla maggior parte dei desktop, laptop e telefoni moderni, il che si rivela fondamentale per le applicazioni di fitness, salute e benessere dal vivo.

****

**Architettura 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 princiali:

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

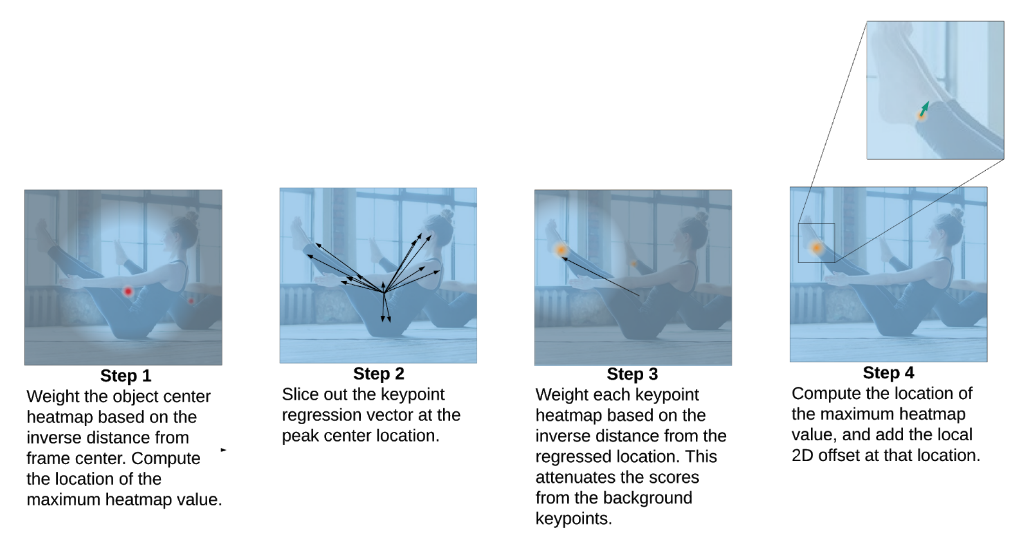


Lo schema di predizione segue largamenete quello di CenterNet, con dei cambiamenti che migliorano l’accuratezza e la velocità.Tutti i modelli sono trainati usando TensorFlow Object Detection API.

Il Feature extractor in MoveNet è MobileNetV2 unito ad una feature pyramid network(FPN), che consente un output della features map semanticamente ricca e ad alta risoluzione (output stride 4).

Ci sono 4 prediction heads attaccate al feature extractor, responsabili delle seguente 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.



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).

**Creazione del 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.

*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.*

*#aggiungere algoritmo fosrse*

**Divisione del dataset**

Affinché potessimo creare dei set di traning 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 defininendo 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 categoricha 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.

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

#mettere la differenza per i dataset tra i modelli ML e i modelli DL

**Costruzione dei modelli & Addestramento**

**METRICHE**

#Decidere se fare anche una spiegazione sugli errori di prima e seconda specie e se spiegare i 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.

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.

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.

Immagine che contiene testo

Descrizione generata automaticamente

* **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.[0,1]

Immagine che contiene testo

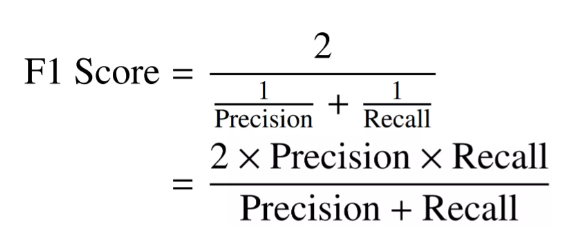
Descrizione generata automaticamente

* **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.

Immagine che contiene tavolo

Descrizione generata automaticamente

* **F-1-score** 
  + L’F1-score è la media armonica della precision e della recall ,utile per avere una visione complessiva delle due metriche

****

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.

**Immagine che contiene tavolo

Descrizione generata automaticamente**

**MACHINE LEARNING SEMPLICE**

**VALIDATION**

Prima di passare alla fase di test sia per l’SVM sia per sia per il random forest è stata svolta la fase di test 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 stat 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-prametri durante la cross-validation, acui passiamo come estimator(come modello)non il semplice modello, ma una pipeline in modo tale che se in futuro dovessiomo 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 nel paragrafo precedente.

**MODELLI**

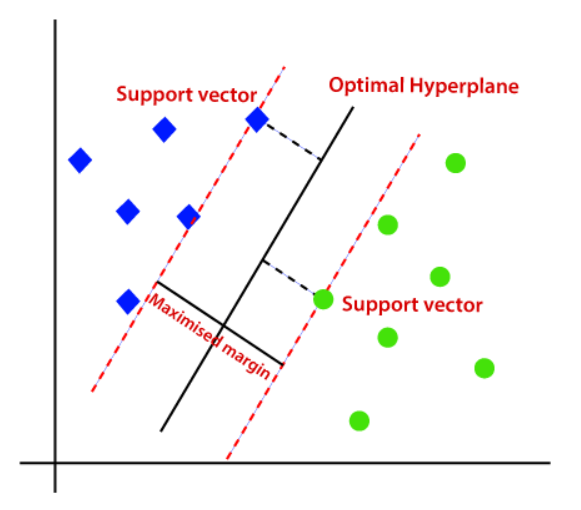
Passando ora ai modelli di ML ,abbiamo utilizzato il modello SVM e il modello Random Forest

**SVM**

Il modello svm è un modello di apprendimento supervisionato che nativamente suporta 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 so che vettori) appartenti alle rispettive classi, dove un iper-piano è un sottospazio lineare di dimensioni n-1 rispetto allo spazio che lo contiene.

Poiché esitono 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

**[15]**

Durante la fase di validation sono stati provati, come detto precedentemete, 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 il trade off tra tra un margine ampio e un basso numero di punti dati classificati erroneamente.(gestisce l’overfitting).Un avalore più grande ci C significa un maggior numero di punti del set di training correttamenti classificati(bisogna stare attenti all’overfitting)

L’iperparametro gamma definisce fino a che punto arriva l’influenza di un singolo samples del set di training .Se ha un valore basso soognifica che 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 è una funzione di base radiale gaussiano; rispetto agli altri kernel, lineari e polinomiali offre una maggiore flessibilità.

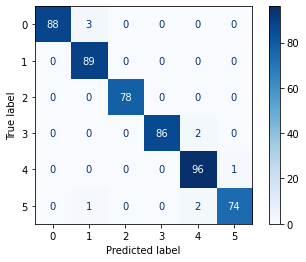
In seguito alla cross validation siamo giunti che il modello migliore è ottimizzatosecondo i seguenti values, assegnati ai rispettivi parametri:

* **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:



**RANDOM FOREST**

Il modello Random Forest è un modello di apprendimento supervisionato nel quale, come suggerisce un nome sono usati diversi alberi di di classificazioni. Perntanto la previsione finale sarà data dalla nedua tra i risultati ottenuti dai vari alberi ,oppurà sarà usato il meccanismo del voto.

E’ 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 effetivamente 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, unaserie 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’iper-parametro indica il numero di alberi di decisione che verranno usati nella foresta.

L’iperparametro max-depth indica invece la profondità massima dell’albero.

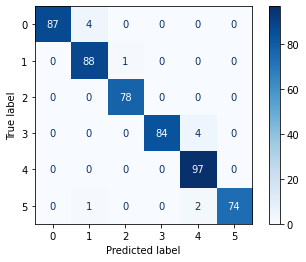
In seguito alla cross validation siamo giunti che il modello migliore è ottimizzatosecondo i seguenti values, assegnati ai rispettivi parametri:

* **n\_estimators**: 100
* **max\_depth:** 10

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

* **accuracy:** 0.9769230769230769
* **precision:** 0.9792209840178696
* **recall:** 0.9767320694286985
* **f1:** 0.9775113777577754

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



**MODELLI DL**

Passando ora invece ai modelli di deep learning si è deciso di focalizzarsi sui modelli di reti neurali ricorreni.In particolare si è deciso di esplorare i seguenti modelli neurali ricorrenti RNN LSTM e GRU. Per ognunsa presenteremo una spiegazione di natura teorica seguita dalla soluzione adottata per ogni re neurale.

**RNN**

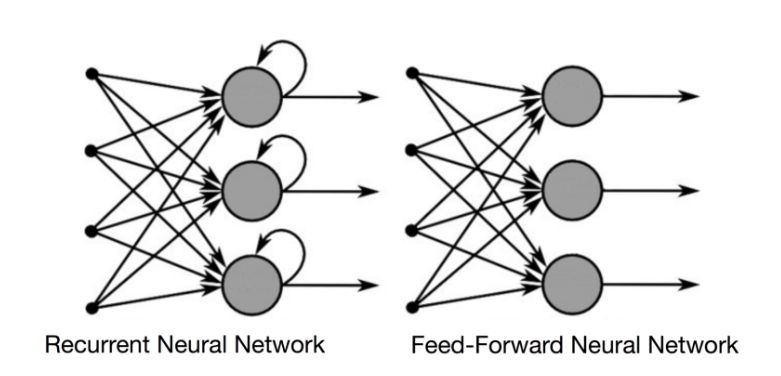
Le RNN sono un potente e robusto tipo di reti neurali che negli ultimi anni stanno ottenendo molto successo poiché sono gli unici modelli con una memoria interna.

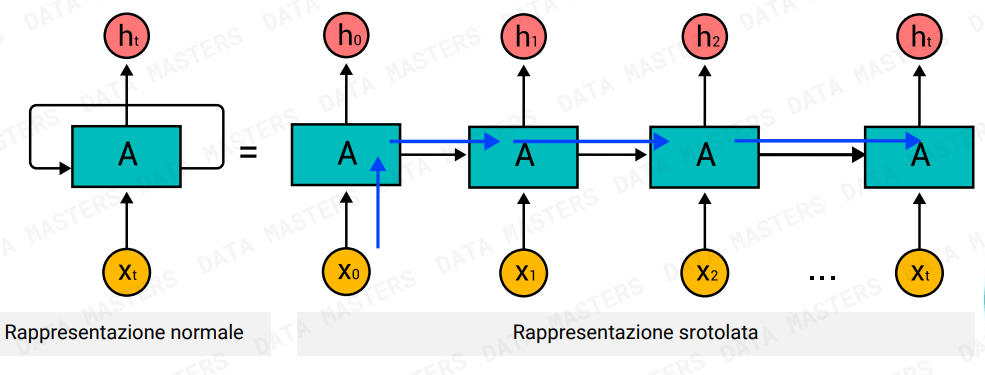
Come gli altri mofelli di deep learning anche le reti neurali ricorrenti sono relativamente vecchie.Difatti vennero inizialmente create nel 1980 ma solo nel recente periodo abbiamo vissto il loro reale potenziale grazie alla mole di dati a nostra disposizione e grazie all’umento della potenza computazionale.

Questi ti 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 finanaziari e olto altro. Non a caso ono molto usati nel Natural language processing NLP.

Oltre ad essere usati per task di NLP, questi algoritmi sono utili anche per i task si riconoscimento dell’attività o di classificazione della posa.Difatti nel caso dello yoga il contsto o l’infromazione della posa intermedia o o iniziale è importante per predirre laposa 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.

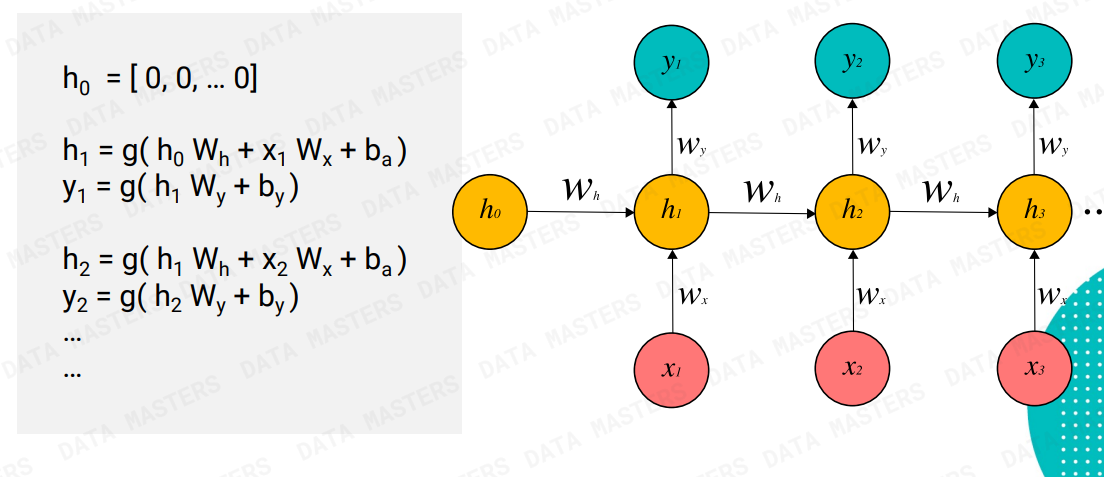
Per poter mantenre la memoria interna le rnn a differenza di quelle fee forward hanno dei loop al loro interno che permettono alle singole informazioni dedotte da un input di persistere nella rete.

****

****

In particolare avremo quindi che l’output all’i-esimo input non dipende solo da questo ma anche dagli (1-1) input precedenti.

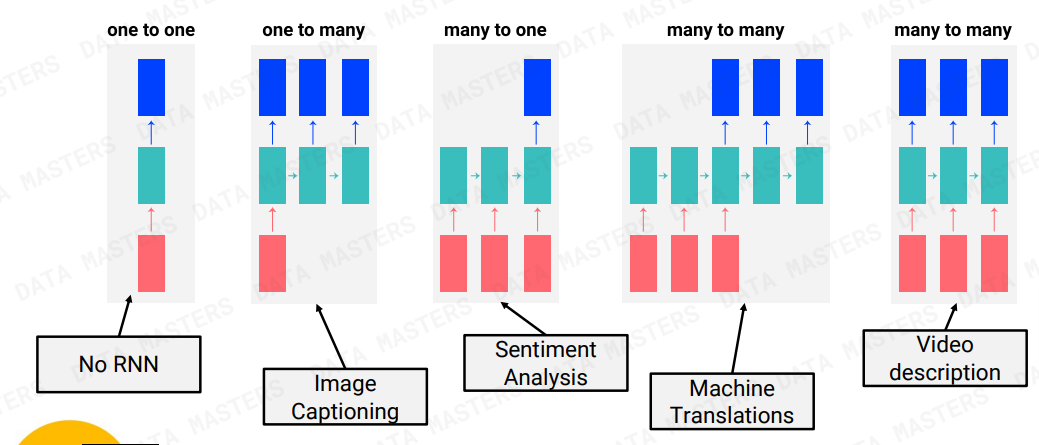
Ossia l’output non dipede solo dal vettore in input e dai relativi pesi associati ma anche dai pesi associati ad un vettore nascoto che tiene traccia dei precedenti input e output della rete



Le reti RNN soffrono prò di due problemi l’esposione del gradiente, che si verifica quando, per qualche motivo, l’algoritmo assegna dei pesi altissimi e la scomparsa del gradiente che si verica quando il gradiente assume un valore nullo o asintotico allo zero impedendo cosi 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 abbiano che la rete è cosi strutturata: Immagine che contiene testo

Descrizione generata automaticamente

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.

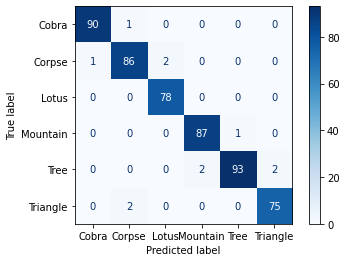
Mentre come ottimizzatore è stato usato Adam.

La rete è stata addestrata per 100 epoche.

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

* **accuracy**: 0.9788461538461538
* **precision**: 0.9785364816472873
* **recall**: 0.9794547246837876
* **f1:** 0.9789239511677909

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



**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.

Gli 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 “gated”(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 (gate dimenticato) o lasciare che incidano sull'output nella fase temporale corrente (gate di output).

In particolare avremo che nel forget Gate

#INSERIRE SLIDE E FOTO

Nell’input gate

#INSERIRE SLIDE E FOTO

Nell’otput gate

#INSERIRE SLIDE E FOTO

Passando ora invece alla nostra architettura abbiano che la rete è cosi strutturata:

Immagine che contiene testo

Descrizione generata automaticamente

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.

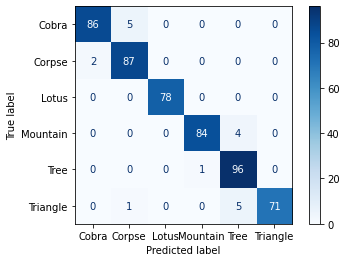
Mentre come ottimizzatore è stato usato Adam.

La rete è stata addestrata per 100 epoche.

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

* **accuracy**: 0.9653846153846154
* **precision**: 0.9692129344406384
* **recall**: 0.9648161888692411
* **f1**: 0.9663317642754872

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



**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 rispoetto alle reti LSTM avendo meno parmetri ma risulta anche essere meno potente per il medesimo motivo.

Passando ora invece alla nostra architettura abbiano che la rete è cosi strutturata:

Immagine che contiene testo, monitor, telefono, cellulare

Descrizione generata automaticamente

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.

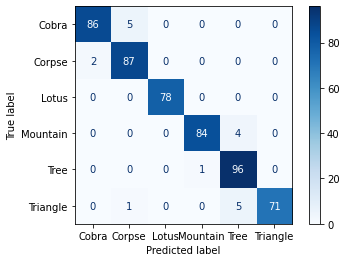
Mentre come ottimizzatore è stato usato Adam.

La rete è stata addestrata per 100 epoche.

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

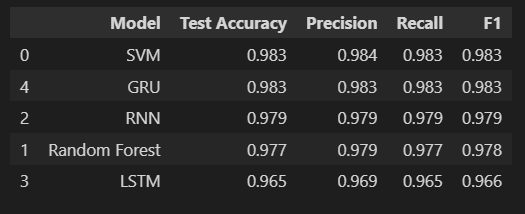
* **accuracy**: 0.9826923076923076
* **precision**: 0.9826634118066703
* **recall**: 0.9833161596887995
* **f1**: 0.9829539964492509

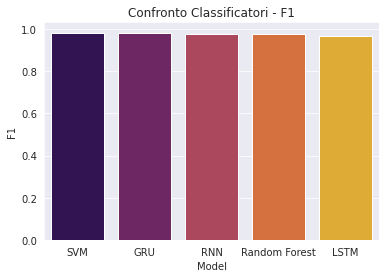
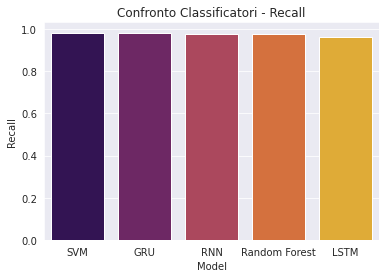
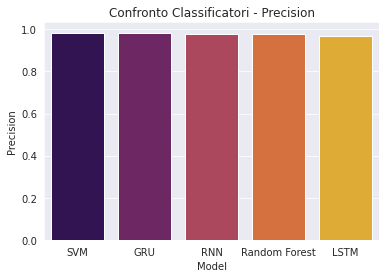
Di seguito rappresentiamo la matrice di confusione che ci permette di avere una panoramica completa dei risultati ottenuti:



**Confronto Riassiuntivo**

Dopo aver addestrato i vari modelli li abbiamo messi a confronto con un 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 discrminanate che i modelli di machine lerning hanno impiegato una quantità di risorse(tempo) minina rispetto ai modelli deep.





**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'ovevrfitting.

Per valutare quindi in maniera più corretta la situazione proviamo ad 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 dovò prima ottenere gli indici sui quali dividere il set 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 un shuffle sul nostro dataset poichè la struttura sequenziale del nostro dataset potrebbe influenzare i risultati dei nostri modelli deep.

Anche in questo caso però abbiamo ottenuto i medesimi risultati.

**Capitolo 3**

**Correzione**

All’interno di questo capitolo 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.

**DATASET**

**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:

1. [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:

1. [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 varierè da posa a posa.

In seguito a questa fase avremo quindi i seguenti dataset:

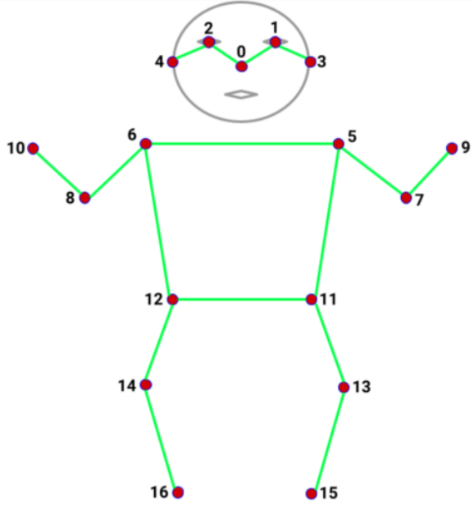
|  |  |  |
| --- | --- | --- |
| **dataset\_Posa** | **# Frames** | **#Key point** |
| Cobra | 20721 | 51 |
| Corpse | 20492 | 51 |
| Lotus | 17884 | 51 |
| Mountain | 20061 | 51 |
| Tree | 22204 | 51 |
| Triangle | 17517 | 51 |

Per poter semplificare l’accesso ai valori x e y per ogni keypoint effettuiamo la reshape da

**(#frames, 51) 🡪 (#frames, 17, 3)**

**SISTEMA ADOTTATO**

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. #to do inserire angoli disegnati

****

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],

}

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

****

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:

Immagine che contiene tavolo

Descrizione generata automaticamente

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

**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 la funzione prima descritta per calcolare gli angoli mentre nel caso in cui K sia stato definito con un valore >maggiore di 1 verrà calcolati il centroide dei vettori neighbors e sulla base di questo verranno calcolati gli angoli.

**SITEMA 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 coretti, dove per corretti si intende superiori o inferiori rispetto a un valore di treshold 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 esguita in maniera corretta verrà riferiferito all’utente che sta svolgendo la posizione in maniera corretta.

Per ogni posa sono stati definiti dei feedback differenti riassunti nei seguenti flow chart:

**Capitolo 3**

**Web-App**

In quetso capitolo verrà descritto il sistema sul quale si è deciso di implementari il sistema prima descritto fornendo una panoramica generale della struttura del progetto e spoegando le varie scelte progettuali.

Il sistema ralizzato è 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 SLQalchemy mentre come dialetto di sql è stato usato sqlite. Strumenti già descritti nei primi capitoli di questo lavoro di tesi.

Per dare un idea generale e esplicitare i concetti fondamentali della nostra applicazione abbiamo realizzato il seguente modello di dominio.#inserire modello di dominio

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

* La parte Model si compone dal’insieme di classi che rappresentanno il sdominio 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 buisness tra cui i file che gestiscono le routes.

In questo tipo di sistema avremo che le classi facenti parte della parte Model possono comunicare tra loro e con le classu facenti parte della parte Control.

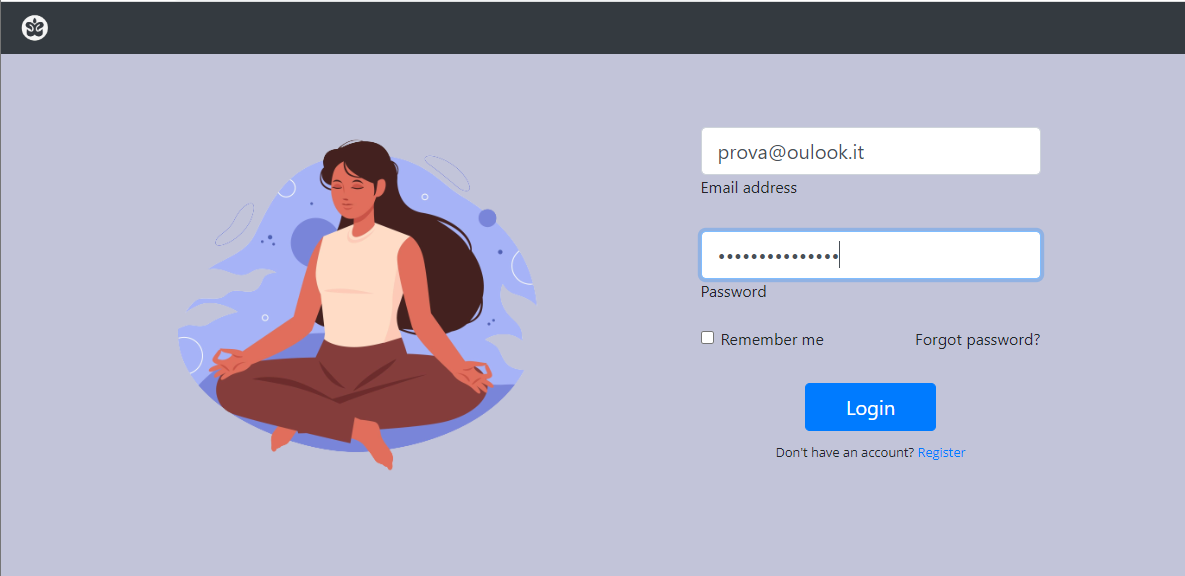
Le classi/file facenti parte della parte view possono comunicare tra loro e con le classi Control.

Le classi/ file facenti parte della parte Control possono comunicare comunicare con tutte le classi/file ad eccetto che con gli attori.

Passando ora invece alle funzionalità della nostra piattaforma queste verranno descritte verranno descritte via via che le pagine/scene realizaate verranno presentate.

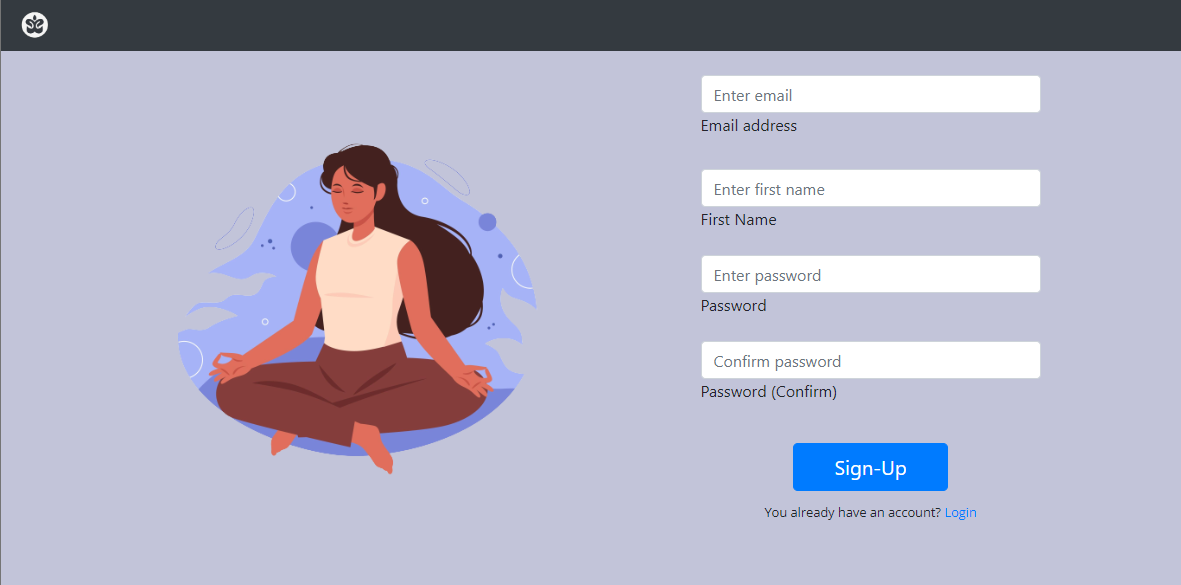
PAGINA LOGIN

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



PAGINA SIGN-IN

Nel caso invece in cui non sia staa già effettuata la regitrazione bisogna effettuala tramite l’apposita sezione



HOMEPAGE

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 nome il logout e quindi torneremo alla pagina dedica al login
* Sessione Privata
  + Questo pulsante ci permetterà di essere trasportati alla pagina dedica alla pagina dedica alla gestione della propria sessione di yoga



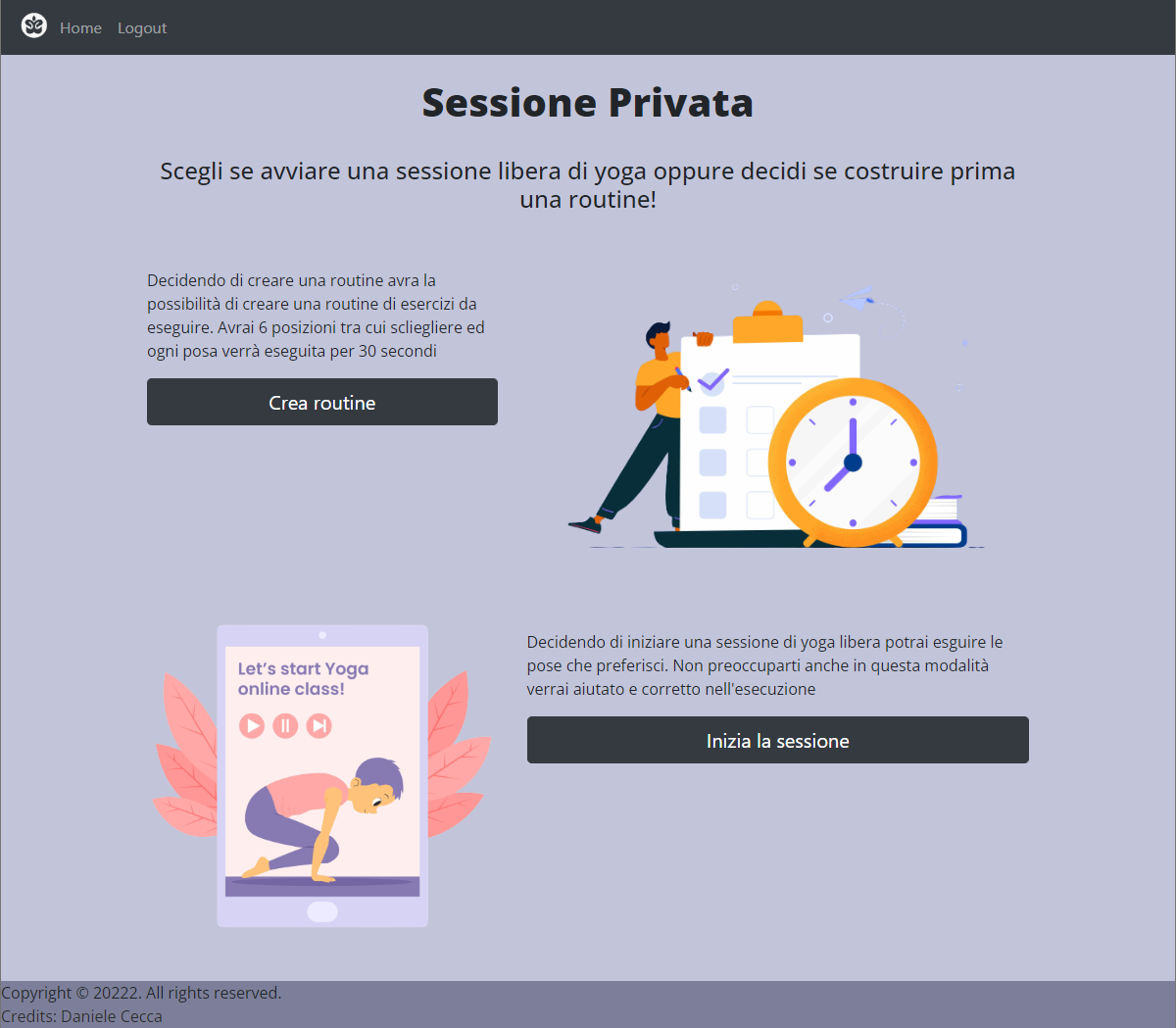
PAGINA SESSIONE PRIVATA

Una volta piggiato il pulsate sessione priva verremo trasportati nella pagina dedicata alla gestione della propria sessione di yoga 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 nome il logout e quindi pigiandolo torneremo alla pagina dedica al login
* Aggiungi esercizio
  + Questo pulsante ci permetterà di aggiungere un esercizio alla nostra routine
* (X) Eliminare esercizion
  + 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

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



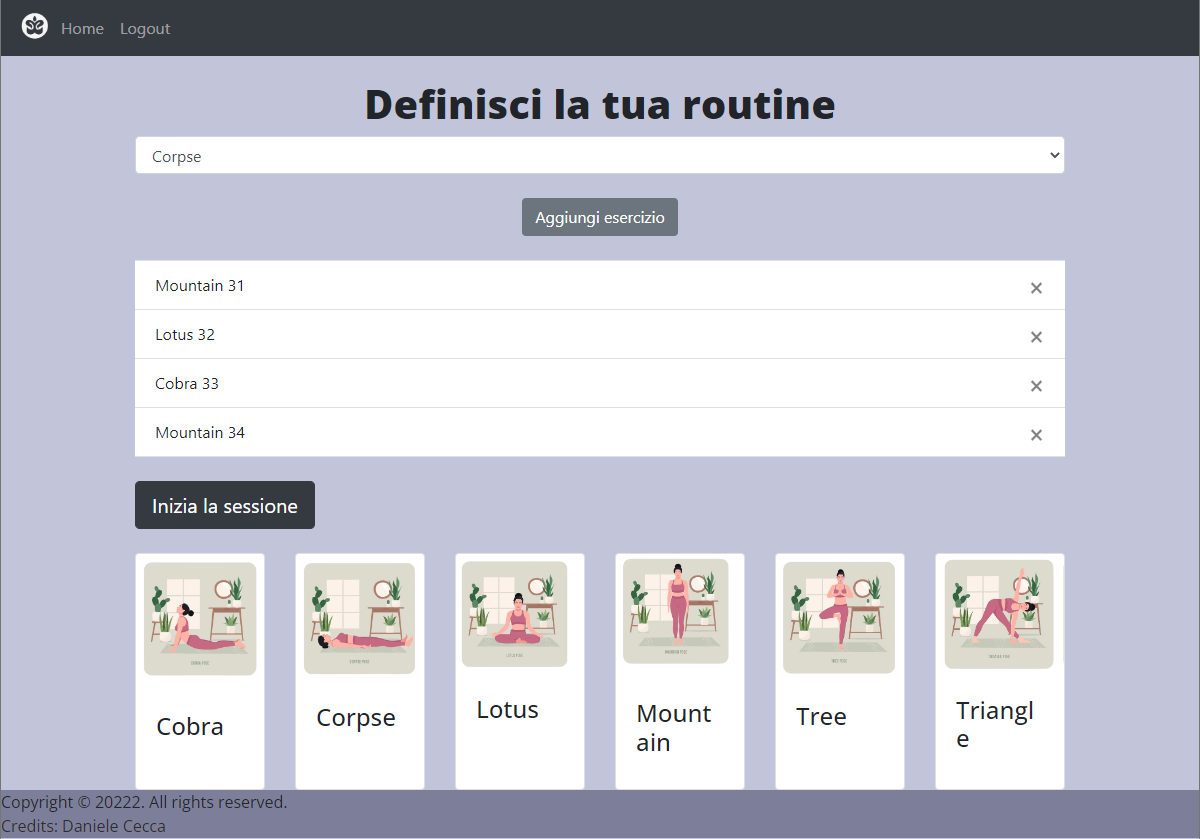
PAGINA CREA ROUTINE

Una volta pigiato il pulsate 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 nome il logout e quindi pigiandolo torneremo alla pagina dedica al login
* Crea routine
  + Questo pulsante ci permetterà di essere trasportati alla pagina dedica 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

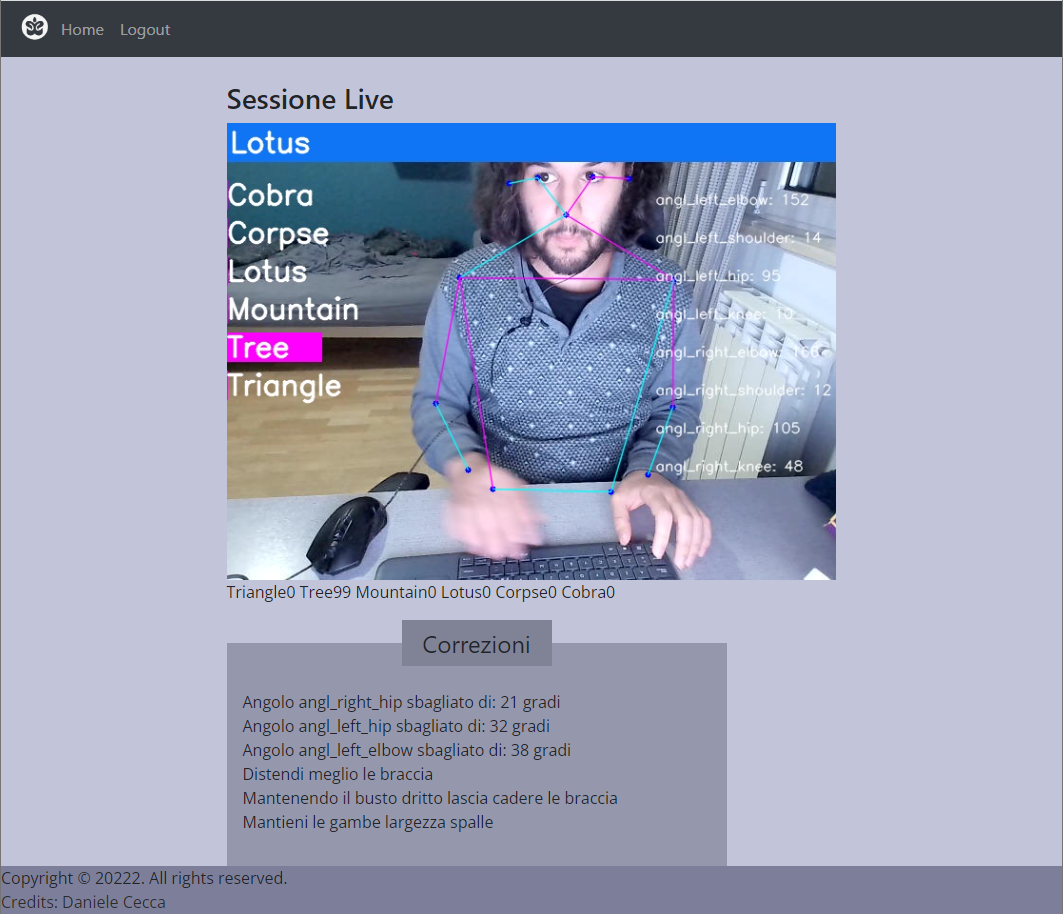
Per facilitare la comprensione della posa che si sta inserendo all’interno della routine per ogni posa è stata inserita una illustrazione.



PAGINA SESSIONE LIVE (LIBERA)

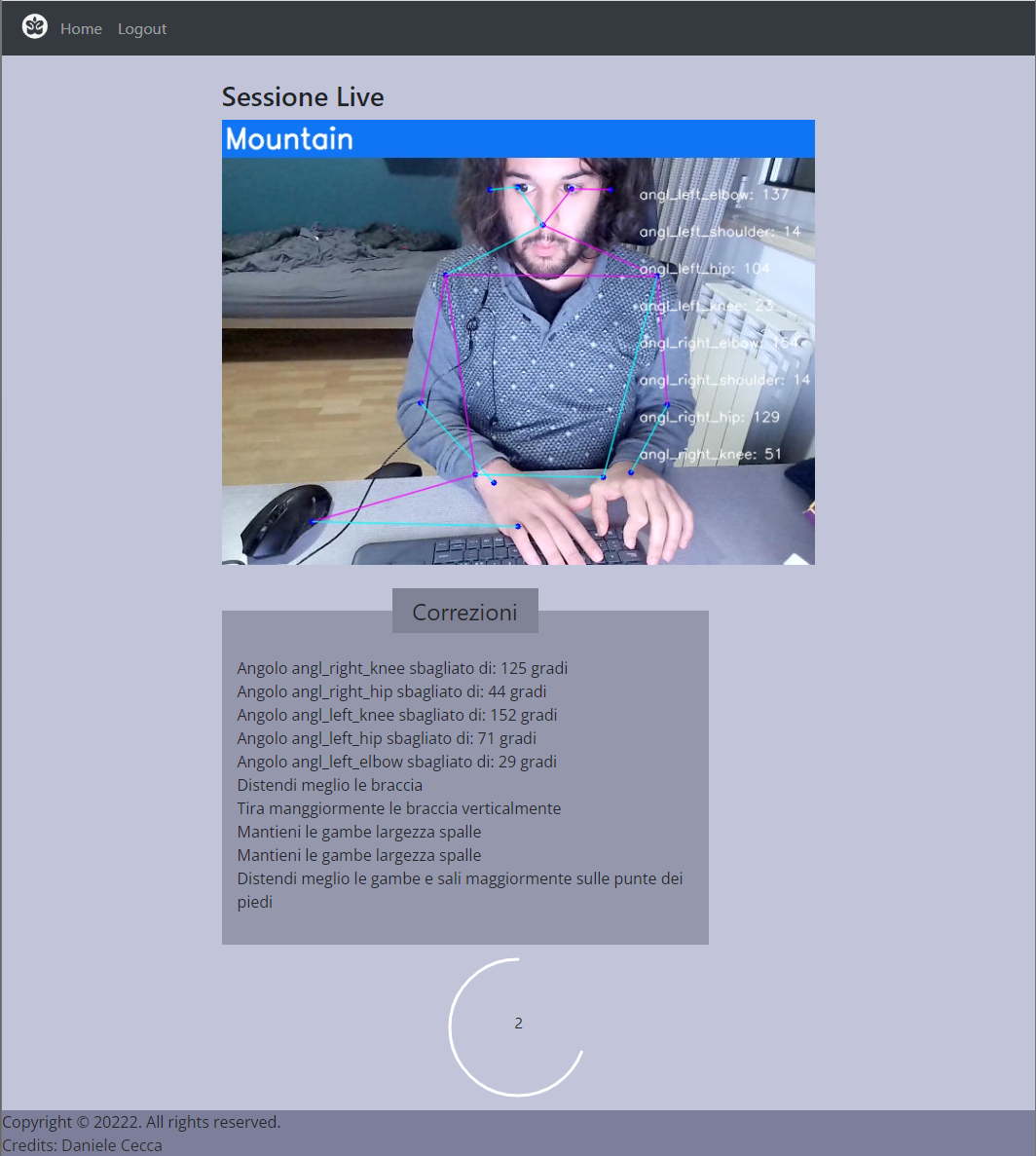
Dopo aver pigiato il pulsante nella pagina sessione privata verremo trasportati nella pagina in cui avverrà effettivamente la sessione di yoga e quindi dove verrà erogato il servizio descritto nella pagina inziale. In particolare in questa pagina avverrà sia la classificazione che la correzione della posa descritti rispettivamente nelle fasi 1 e 2.

All’interno di questa pagina verrà poi mostrata una finestra in cui ci sarà il nostro video streaming mentre eseguiamo le pose. Per ogni posa che stiamo eseguendo averrà 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.



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 inziale. In particolare in questa pagina avverrà solamente la correzione della posa che si sta eseguendo. In questa modalità nonsaremo liberi di svolgere qualsiasi posa ma dovremo svolgere la posa presente nella 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



BIBLIOGRAFIA

1. <https://www-formal.stanford.edu/jmc/whatisai.pdf>
2. <https://www.ibm.com/topics/artificial-intelligence>
3. <https://link.springer.com/chapter/10.1007/978-3-030-69978-9_4>
4. <http://infolab.stanford.edu/pub/voy/museum/samuel.html>
5. <https://developers.google.com/machine-learning/intro-to-ml/what-is-ml>
6. <https://www.ibm.com/topics/deep-learning>
7. <https://www.qut.edu.au/study/business/insights/3.2-billion-images-and-720000-hours-of-video-are-shared-online-daily.-can-you-sort-real-from-fake>
8. <https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/>
9. <https://archive.org/details/YogaVidCollected>
10. <https://link.springer.com/article/10.1007/s00521-019-04232-7#Sec2>
11. <https://arxiv.org/pdf/2006.01423.pdf>
12. <https://viso.ai/deep-learning/pose-estimation-ultimate-overview/>
13. <https://maureentkt.medium.com/selecting-your-2d-real-time-pose-estimation-models-7d0777bf935f>
14. <https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html>
15. <https://medium.com/@viveksalunkhe80/support-vector-machine-svm-88f360ff5f38>
16. <https://www.smartmat.com/about/>
17. <https://www.wearablex.com/pages/how-it-works>
18. <https://yoganotch.com/>
19. <https://www.mirror.co/>