Università Politecnica delle Marche
Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Progettazione e implementazione di un'applicazione Android basata su Google ML Kit per il riconoscimento delle pose

Design and development of an Android application based on Google ML Kit for pose detection

Relatore

Prof. Domenico Ursino

Candidato

Giordano Angelini

Anno Accademico 2022-2023

All'Ing. Marco Marchetti

Sommario

Google ML Kit è un insieme di librerie, messe a disposizione da Google, per realizzare applicativi che soddisfino task di Computer Vision e Natural Language Processing. In particolare, questo documento descrive come si possano integrare le funzionalità di ML Kit all'interno di un'applicazione mobile Android che permetta il riconoscimento in tempo reale della posa corporea di un soggetto umano, inquadrato dalla fotocamera di un dispositivo mobile. In esso sono anche incluse: *(i)* un'analisi dello stato dell'arte delle principali soluzioni per la riabilitazione fisica a distanza, mediante realtà aumentata; *(ii)* la descrizione del processo di tracciamento del corpo umano, sfruttando le librerie di MediaPipe; *(iii)* l'implementazione di un algoritmo k-NN per la classificazione e il riconoscimento delle varie pose corporee. Infine, viene presentata l'applicazione completa e viene condotta una breve analisi sui principali orizzonti di sviluppo futuro del progetto.

Keyword: ML Kit, Google, Computer Vision, pose detection, body tracking, realtà aumentata, sviluppo mobile

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La tesi in esame conclude un percorso di laurea magistrale in Ingegneria Informatica e dell'Automazione condotto presso l'Università Politecnica delle Marche, includendo vari argomenti che sono stati oggetto di studio di diversi corsi durante il biennio.

Tale lavoro rappresenta il sunto di un progetto di tirocinio in ambito aziendale svoltosi da ottobre a dicembre 2023. L'idea alla base del prodotto nasce dall'azienda ospitante il tirocinio, *Airbag Studio srl*, desiderosa di studiare la realizzabilità di un'applicazione di riconoscimento della posa corporea in un contesto di riabilitazione motoria. In particolare, l'azienda ha tanti clienti che si occupano di telemedicina, e una tecnologia del genere rappresenta un buon orizzonte di sviluppo nel mercato odierno. In un mondo che è sempre più aperto a soluzioni innovative in ogni ambito di lavoro, l'applicazione realizzata pone le proprie basi in due rami dell'Informatica che, ultimamente, hanno riscosso grande successo nell'immaginario collettivo. Tali rami sono:

- *La realtà aumentata.* La pandemia di *COVID-19* ha dimostrato quanto ci sia bisogno di poter disporre di tecnologie che abbattano le limitazioni poste dalla distanza geografica, la quale, spesso, impedisce di svolgere attività in diversi contesti. Tra questi compare, per l'appunto, anche la riabilitazione fisica, ovvero quell'insieme di trattamenti per permettere il recupero delle capacità fisiche e dell'attività motoria, spesso necessaria dopo un infortunio. La realtà aumentata, oggetto di una sostanziosa parte del documento in esame, rappresenta un nuovo orizzonte tecnologico che aiuterà, col tempo, ad abbattere le barriere appena descritte.

- *L'Intelligenza Artificiale.* Il 2023 è stato l'anno in cui il mondo ha toccato con mano le vere potenzialità dell'Intelligenza Artificiale, la quale prevede anche alcune funzionalità integrate nell'app oggetto della presente tesi. In particolare, l'applicazione costruita fa uso di algoritmi di Machine Learning che permettono, una volta addestrati opportunamente, di distinguere una posa corporea dall'altra.

Quanto detto implica che il lavoro svolto ha grandi potenzialità e margini di sviluppo futuro. Esso si è attenuto a diverse fasi, descritte di seguito:

1. *Raccolta dei requisiti.* Tramite due differenti riunioni telematiche, assieme all'Ing. Omar Cafini, tutor aziendale del tirocinio, e al Prof. Domenico Ursino, relatore della presente tesi, si è deciso di affrontare questo tipo di task e si sono definiti i requisiti dell'applicazione da realizzare.

2. *Valutazione dello stato dell'arte.* Prima di definire, in un'ultima riunione, la tecnologia da utilizzare, è stata verificata la presenza di soluzioni simili a quella implementata nel corso del progetto. Ciò ha incluso la ricerca di prodotti finiti esistenti sul mercato, che espletassero le funzionalità elencate durante la raccolta dei requisiti. Le fonti, durante questa seconda fase, sono stati i principali store di applicazioni mobile e i repository GitHub rinvenuti in rete.

3. *Scelta della tecnologia.* Dopo un'analisi dei framework e delle librerie adatti ad adempiere al task iniziale, è stato scelto, con le dovute motivazioni, il pacchetto ML Kit di Google. Questo si è rivelato ottimo per implementare le principali funzioni di Computer Vision, utilizzate nel tracciamento delle pose.

4. *Realizzazione dell'applicazione.* Durante questa fase è stata realizzata una prima versione dell'applicazione, in Android/Java, al fine di testare la libreria prima di apporre i dovuti miglioramenti.

5. *Creazione del dataset.* È stato realizzato un dataset apposito, a partire da quello "dimostrativo", fornito da Google stessa, per generare una libreria di pose corporee note, utile ad addestrare un algoritmo k-NN di Machine Learning, utilizzato durante la classificazione in tempo reale.

6. *Risultati finali.* La fase finale è stata caratterizzata da un utilizzo intensivo dell'applicazione, al fine di evidenziare mancanze o punti di forza della stessa, in modo da poter elencare una serie di migliorie che potranno essere apportate in futuro.

Il presente documento è composto da cinque capitoli, strutturati come di seguito specificato.

- Il Capitolo 1 descrive le prime tre fasi, appena citate, dalla raccolta dei requisiti alla scelta della tecnologia da utilizzare.

- Nel Capitolo 2 compare una descrizione molto dettagliata di ML Kit e, in particolare, del modulo di pose detection.

- Il Capitolo 3 riassume le principali tecniche di Data Mining, approfondendo la classificazione tramite algoritmi "Nearest-Neighbor"; esso si conclude descrivendo come è stato generato un training set per la classificazione, a partire da una serie di immagini multimediali.

- È compito del Capitolo 4 descrivere l'applicazione Android realizzata, includendo i diagrammi di sequenza dei task principali, l'implementazione dell'algoritmo k-NN e la resa visiva del prodotto finito.

- Nel Capitolo 5 vengono presentati i risultati della fase finale del lavoro, ponendo l'accento sulle mancanze dell'applicazione e, dall'altra parte, sulle opportunità di sviluppo della stessa.

CAPITOLO 1

Stato dell'arte e tecnologie esistenti

Prima di iniziare un qualsiasi progetto è bene guardarsi attorno e osservare al meglio ciò che già esiste nel settore all'interno del quale si andrà a operare. Migliore è l'analisi che viene fatta e migliore sarà l'approccio iniziale al lavoro; questo si traduce in un efficiente impiego delle risorse le quali andranno concentrate ove si evidenzieranno le principali mancanze dei prodotti esistenti presi in considerazione. Questo capitolo riassume tale analisi, descrivendo lo stato dell'arte delle soluzioni software inerenti all'ambito del progetto ed elencando le tecnologie più diffuse sul mercato quando si parla di tracciamento del corpo umano.

1.1 Definizione del problema

Un'analisi corretta e funzionale non può certo essere effettuata se non si ha chiara in mente l'entità del problema da risolvere. Il progetto in questione ha come obiettivo principale l'implementazione di un meccanismo di *pose detection* (letteralmente "riconoscimento della posa corporea") che possa essere, successivamente, integrato in un'applicazione mobile[1]. Il meccanismo introdotto deve soddisfare una serie di requisiti specifici; questi ultimi sono riportati di seguito.

1. *Riconoscimento tramite fotocamera.* Il sistema deve essere in grado di operare il riconoscimento della posa a partire da un input multimediale fornito a esso dalla telecamera del dispositivo mobile in utilizzo. Ciò evita di dover utilizzare sensori di movimento spesso costosi e monouso, dunque fin troppo orientati a un solo task e necessari soltanto in ambiti che richiedono una precisione troppo alta (basti pensare ai sensori per il motion tracking nei contesti di produzioni cinematografiche).

2. *Riconoscimento in real time.* Il sistema deve eseguire il riconoscimento delle pose in tempo reale, ovvero non appena la fotocamera del dispositivo acquisisce l'immagine e non in un secondo momento (ad esempio, processando un video precedentemente registrato).

[1] Un'applicazione mobile è un software progettato per essere eseguito su dispositivi mobili, come smartphone e tablet. Queste applicazioni offrono funzionalità specifiche e sono scaricabili e installabili dai negozi di app ufficiali associati ai sistemi operativi mobile, come l'App Store per iOS e Google Play per Android.

3. *Feedback a schermo.* Tale requisito si lega al precedente. Infatti, il riconoscimento della posa corporea in tempo reale permette al sistema di produrre costantemente un output che mostri all'utente i risultati dei task di identificazione. Il requisito in questione presuppone che tale output venga visualizzato nello schermo con un'interfaccia grafica utile a comprendere meglio quale sia l'esito del riconoscimento.

4. *Possibilità di personalizzazione.* Si ha bisogno di implementare un sistema che supporti il riconoscimento di pose personalizzate in modo che si possa costruire una "libreria" di posizioni note in base ai requisiti dell'applicativo finale. Ciò permette agli utenti di utilizzare l'applicazione in vari ambiti differenti; il Capitolo 5 presenta una serie di possibilità di utilizzo del sistema di riconoscimento nel mercato delle applicazioni mobile.

1.1.1 Augmented Reality (AR) & Virtual Reality (VR)

Il requisito numero 3 introduce al tema della realtà aumentata che, da diversi anni a questa parte, occupa una posizione di grande rilevanza nello spettro di prodotti realizzati per il settore manifatturiero, dell'insegnamento o dell'intrattenimento. La realtà aumentata (AR) e la realtà virtuale (VR), più conosciuta anche nella terminologia comune, sono due concetti correlati, ma distinti (Figura 1.1) che offrono esperienze utente diverse [1].

La realtà aumentata combina elementi del mondo reale con elementi generati al computer, sovrapponendo informazioni digitali a quelle della realtà. Gli utenti possono interagire con gli oggetti del mondo reale insieme agli elementi virtuali sovrapposti. Ad esempio, attraverso uno smartphone od occhiali AR, è possibile vedere informazioni aggiuntive su un oggetto reale o interagire con elementi virtuali in uno spazio fisico; ancora, come nel caso dell'applicazione in questione, gli utenti possono ottenere feedback visivi in base al movimento del proprio corpo. Applicazioni di realtà aumentata possono includere filtri fotografici, navigazione stradale, istruzioni in tempo reale riguardo task aziendali e giochi che coinvolgono il mondo reale (qualche anno fa il gioco Pokémon Go[2] ha registrato un enorme successo nelle fasce d'età più giovani).

La realtà virtuale crea un ambiente artificiale che sostituisce il mondo reale, immergendo chi la utilizza in un'esperienza completamente digitale. Gli utenti in VR sono immersi in un ambiente simulato e possono interagire con esso attraverso dispositivi di input, come controller o sensori di movimento. La sensazione di presenza è accentuata, facendo sentire gli utenti come se fossero fisicamente presenti in una simulazione. Tra le applicazioni di realtà virtuale compaiono simulazioni immersive (come quelle di volo), esperienze di gioco 3D, addestramento virtuale (simulazione di ambienti ostili), tour e meeting virtuali.

1.2 Soluzioni presenti nel mercato

Dopo aver brevemente descritto il task da portare a termine, è possibile iniziare a sondare ciò che è già disponibile agli utenti nel campo del riconoscimento delle pose tramite smartphone. Tra i numerosi prodotti presenti nel mercato delle applicazioni mobile, sono stati presi in considerazione quelli che collimavano maggiormente con i requisiti espressi precedentemente. Di seguito è riportato l'elenco delle principali soluzioni a oggi disponibili.

[2]Pokémon GO è un gioco mobile basato sulla realtà aumentata (AR) sviluppato da Niantic in collaborazione con Nintendo e The Pokémon Company. Il gioco è stato rilasciato nel luglio 2016 ed è diventato rapidamente un fenomeno globale.

Figura 1.1: Realtà aumentata e realtà virtuale

- *Exer Health.* Sviluppata da Exer Labs Inc., è un'applicazione progettata per la riabilitazione medica a distanza. I professionisti che decidono di acquistarla hanno a disposizione una libreria predisposta di esercizi di riabilitazione che vengono assegnati ai pazienti; quest'ultimi possono utilizzare lo smartphone per inquadrarsi mentre svolgono i vari esercizi (Figura 1.2) e ricevere feedback visivi sul corretto svolgimento della sessione. L'applicazione informa chi la utilizza su come migliorare l'esecuzione dei movimenti in modo comprensibile per l'utente finale. È disponibile solamente per iOS.

- *Kaia Health.* Realizzata dall'omonima compagnia è un prodotto disponibile per iOS e Android. Si tratta di una piattaforma che rende possibile svolgere esercizi riabilitativi o di allenamento generico. L'utenza è più autonoma rispetto al caso precedente (Exer Health) in quanto sono disponibili dei piani di allenamento, previa sottoscrizione di un abbonamento mensile, da seguire per il raggiungimento dei propri obiettivi fisici. Anche in questo caso l'utente riceve dei feedback accurati sull'andamento dell'esercizio e viene accompagnato per tutta la durata della sessione (Figura 1.3). La pecca più grande di Kaia Health è la disponibilità limitata, nella sua versione completa, solo alla zona geografica degli Stati Uniti, in quanto parte di diversi pacchetti di *health insurance*[3].

- *Mirror AR.* L'ultima app è la più versatile delle tre in quanto è disponibile sia per iOS che per Android ed è stata rilasciata anche in Italia. Conserva le funzioni degli esempi precedenti ma continua a necessitare l'intervento di un professionista della riabilitazione, che prescrive i vari esercizi ai pazienti (Figura 1.4).

1.2.1 Principali mancanze evidenziate nei prodotti esistenti

Dalla descrizione precedente e dalle mancanze delle applicazioni osservate si intendono facilmente quali siano i requisiti aggiuntivi a quelli previamente elencati che il prodotto da realizzare deve necessariamente e congiuntamente avere. Essi sono di seguito specificati.

[3] L'health insurance, o assicurazione sanitaria, è un tipo di copertura assicurativa che fornisce una protezione finanziaria contro i costi associati alle spese mediche. Gli assicurati, ovvero coloro che detengono la polizza di assicurazione sanitaria, pagano premi periodici alla compagnia assicurativa in cambio della copertura di determinati costi legati alla salute.

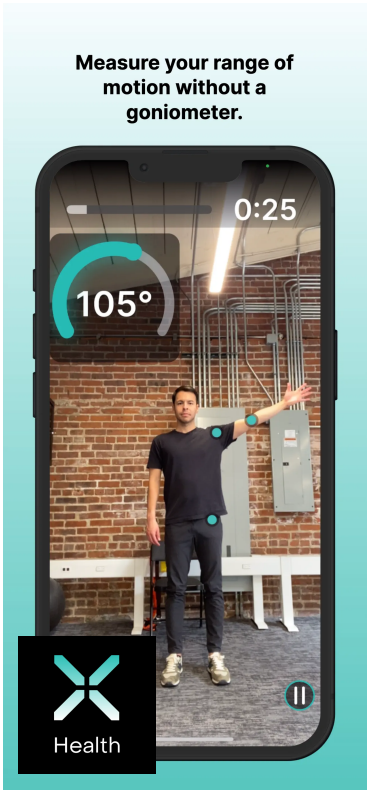

Figura 1.2: Exer Health

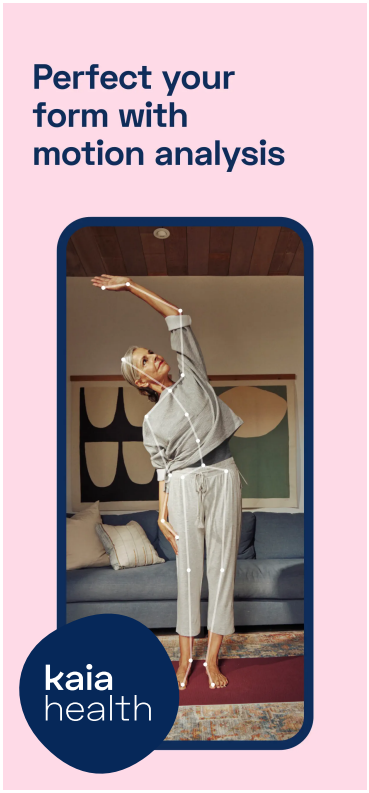

Figura 1.3: Kaia Health

Figura 1.4: Mirror AR

- *Prodotto autonomo e gratuito.* La totalità delle soluzioni presenti sul mercato richiede il pagamento di una quota mensile o il consulto di un medico privato che gestisca gli esercizi e le sessioni di allenamento. Nell'ottica di una tecnologia aperta a più persone possibili si vuole realizzare un prodotto gratuito, che possa essere utilizzato e personalizzato autonomamente per soddisfare tutte le casistiche di utilizzo reale.
- *Prodotto cross-platform.* Nell'eterna sfida tra utilizzatori Apple e Android è bene soddisfare tutte le esigenze e fornire una soluzione adatta a entrambi i mondi, senza precludersi una fetta di utenza che, in entrambi i casi, è importante.
- *Prodotto open source.* Per l'utenza più "avanzata" è bene fornire il codice sorgente del risultato finale in modo che altri sviluppatori possano applicare migliorie, manutenere il codice o utilizzare il prodotto come base per applicazioni e software più complessi.

1.3 Tecnologie e framework per il tracciamento del corpo umano

Delineati i requisiti minimi dell'applicazione, si analizzano le tecnologie esistenti nel campo di sviluppo, le quali saranno utilizzate nell'intera fase di costruzione del progetto finale; per tale motivazione si deve prestare la massima attenzione a scegliere con cura la soluzione migliore e, talvolta, operare compromessi tra una tecnologia e l'altra.

1.3.1 Unity MARS

Se si parla di realtà aumentata e di applicazione cross-platform non si può non tenere in considerazione Unity. Questo è un motore di sviluppo di videogiochi e un ambiente per

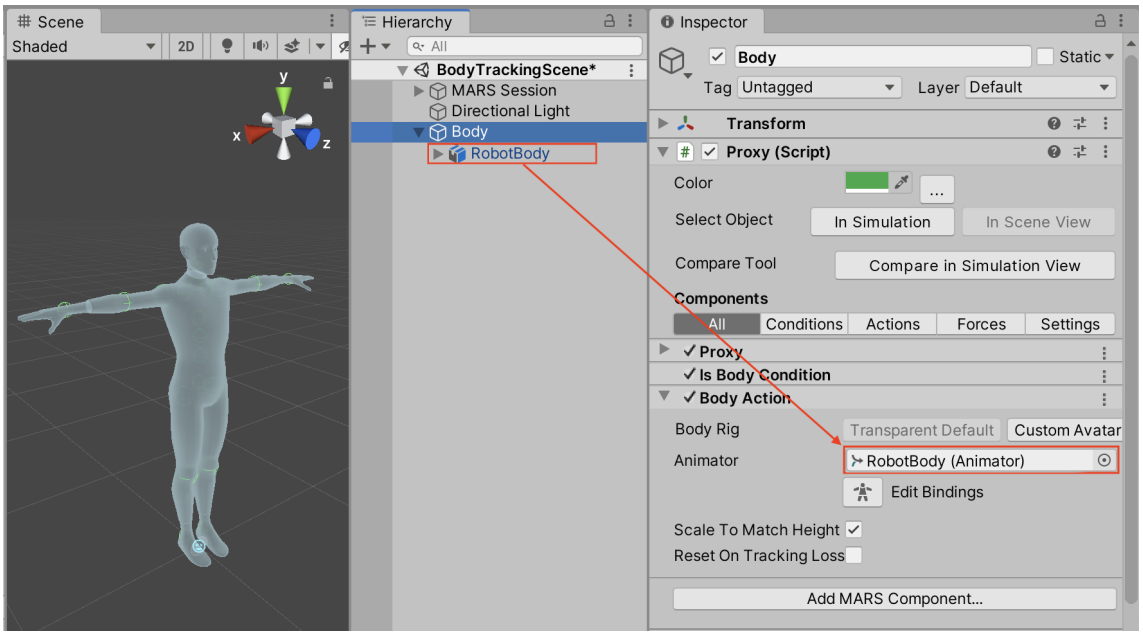

Figura 1.5: Interfaccia di Unity Mars

il development multi-piattaforma. Creato da Unity Technologies, è ampiamente utilizzato per creare giochi per PC, console e dispositivi mobili. Supporta numerosi linguaggi di programmazione, tra cui C# e JavaScript, e offre un'ampia gamma di asset e plugin dalla sua community; Unity Mars è, infatti, uno di questi plugin.

Nella Versione 1.3.1, Mars implementa in modo incredibilmente semplice il tracciamento del corpo umano permettendo di definire un "Body Proxy" (Figura 1.5), ovvero un avatar umanoide che si aggancia al corpo umano rilevato dalla fotocamera del dispositivo in utilizzo. Tramite le funzionalità di Unity e di Mars stesso, si può, poi, procedere a operare un pose matching (riconoscimento della posa corporea mediante il confronto con pose predefinite). Mars ha due principali difetti:

- è un plugin a pagamento;
- è povero di una personalizzazione avanzata (l'interfaccia grafica di Unity è ricca di funzionalità le quali, però, sono predefinite e proprie del plugin);
- supporta la realizzazione di applicazioni esclusivamente per iOS, in quanto sfrutta librerie esclusive di Apple (Sezione 1.3.3).

1.3.2 Unity AR Foundation

Unity AR Foundation è un vero e proprio framework[4] per la costruzione di applicazioni di realtà aumentata, spesso utilizzato per task quali il tracciamento di oggetti, volti, piani (Figura 1.6) o, per l'appunto, pose del corpo umano. È largamente utilizzato sia da sviluppatori iOS che Android e, a differenza di Mars, è gratuito. Tuttavia, le funzionalità per il tracciamento del corpo sono, anche in questo caso, limitate ai dispositivi Apple (Sezione 1.3.3).

[4]Un framework è un'infrastruttura software che fornisce un'architettura di supporto per lo sviluppo di applicazioni. Si tratta di un insieme di librerie, strumenti e convenzioni di progettazione predefinite che consentono agli sviluppatori di costruire applicazioni in modo più efficiente, fornendo una struttura già stabilita.

Figura 1.6: Tracciamento di piani (AR Foundation)

1.3.3 Apple ARKit

Il framework ARKit (kit di realtà aumentata) prodotto da Apple è la "causa" per la quale le precedenti due tecnologie non sono adatte allo sviluppo del progetto da realizzare. Esso contiene tutte le sopracitate librerie necessarie all'esecuzione di task di realtà aumentata nel contesto dei prodotti mobile. È, dunque, maggiormente personalizzabile rispetto alle soluzioni precedenti, ma rimane comprensibilmente pensato per lo sviluppo di applicazioni iOS, in quanto necessita di sensori e accortezze hardware/software che soltanto i dispositivi Apple posseggono.

1.3.4 Google ML Kit

L'ultima tecnologia analizzata, Google ML Kit (Figura 1.7), è un framework prodotto da Google, orientato al Machine Learning[5] e ottimizzato per lo sviluppo mobile e l'utilizzo dell'AR in tempo reale. Permette la realizzazione di applicazioni per Android e iOS, è gratuito e open source; si rivela dunque la soluzione perfetta per il progetto da portare avanti. Una descrizione più approfondita del framework e l'approccio al problema del rilevamento della posa corporea sono questioni affrontate nel Capitolo 2.

Figura 1.7: ML Kit

[5]Il Machine Learning, o apprendimento automatico, è un ramo dell'Intelligenza Artificiale (IA) che si occupa di sviluppare algoritmi e modelli che consentono ai computer di apprendere dai dati e migliorare le proprie prestazioni senza essere esplicitamente programmati. In altre parole, invece di seguire istruzioni dettagliate, i computer possono utilizzare il Machine Learning per riconoscere pattern nei dati e fare previsioni o prendere decisioni basate su tali pattern. Tale argomento verrà approfondito nel Capitolo 3.

CAPITOLO 2

ML Kit

Scelta la tecnologia più adatta, in questo capitolo ne vengono descritte le principali caratteristiche e come tale soluzione affronta il problema del tracciamento del corpo umano e del riconoscimento delle pose. La prima sezione contiene una presentazione di tutte le potenzialità di Google ML Kit mentre, nella seconda, si scende nel dettaglio della questione in esame.

2.1 ML Kit, una visione d'insieme

Come anticipato, Google ML Kit è un framework di Machine Learning fornito da Google che consente agli sviluppatori di integrare facilmente le funzionalità di IA nei loro progetti per dispositivi mobili. ML Kit offre un'ampia gamma di API[1] pronte per l'uso, che coprono diverse applicazioni. ML Kit semplifica l'implementazione di modelli di Machine Learning utilizzando modelli pre-addestrati e offre un'interfaccia utente chiara e semplice da integrare nelle applicazioni Android e iOS. La caratteristica fondamentale del framework è che l'elaborazione avviene sul dispositivo; ciò lo rende veloce e sblocca casi d'uso in tempo reale, come l'analisi dell'input della videocamera. Funziona anche offline e può essere utilizzato per analizzare immagini e testo che devono rimanere sul dispositivo.

ML Kit offre due principali categorie di API: API Vision e API Natural Language; la funzionalità del riconoscimento delle pose appartiene alla prima delle due ma verrà approfondita nella Sezione 2.2. Ognuna delle soluzioni, presentate di seguito, è approfondita nella documentazione [13] del framework.

2.1.1 API Vision

La Computer Vision (CV) è un campo dell'Informatica e dell'Intelligenza Artificiale che si occupa di insegnare ai computer a interpretare le informazioni visive del mondo, similmente a come lo farebbe un essere umano. L'obiettivo principale della CV è quello di fare in modo che le macchine ottengano nella maniera più accurata possibile informazioni

[1] API, acronimo di "Application Programming Interface", è un insieme di regole e protocolli che permettono a due software di comunicare tra loro. Un'API definisce i metodi e le strutture dati che i programmatori possono utilizzare per interagire con un servizio specifico o una libreria di programmazione.

utili dall'*analisi* di immagini e video. Nell'ambito della CV [7] si annoverano principalmente i task riportati di seguito.

- *Analisi di immagini e video.* Ciò include l'estrazione di informazioni significative, come oggetti, persone, luoghi, e la comprensione del contesto visivo.
- *Segmentazione.* Tecnica grazie alla quale si divide un'immagine in regioni più piccole od oggetti. È spesso utilizzata per identificare e isolare specifiche parti di una rappresentazione.
- *Riconoscimento di oggetti.* La CV si occupa di addestrare algoritmi per riconoscere e classificare[2] oggetti all'interno di immagini o video.
- *Tracking.* Il tracking riguarda il monitoraggio del movimento di oggetti o regioni specifiche attraverso sequenze di immagini o video. È utilizzato, ad esempio, per seguire un oggetto in movimento in una scena.

Per raggiungere questi obiettivi, la Computer Vision si basa su algoritmi di apprendimento automatico e reti neurali, che vengono addestrati su grandi quantità di dati visivi per sviluppare la capacità di analizzare e interpretare nuove immagini o video.

La prima categoria di API (denominata, per l'appunto, API Vision) comprende le principali funzionalità di analisi di video e immagini. Le sezioni successive presentano in dettaglio queste soluzioni.

Scansione di codici a barre

Con l'API di scansione dei codici a barre è possibile leggere i dati codificati utilizzando la maggior parte dei formati di codici a barre standard. La scansione viene eseguita sul dispositivo e non richiede una connessione di rete. I codici a barre sono un modo pratico per passare informazioni dal mondo reale alle applicazioni. In particolare, utilizzando formati 2D come i codici QR, è possibile codificare dati strutturati, come contatti o credenziali della rete Wi-Fi.

Rilevamento facciale

Con l'API di rilevamento dei volti di ML Kit, si possono rilevare i volti in un'immagine (Figura 2.1), identificare le principali caratteristiche facciali e definire i contorni dei visi rilevati. È da tener presente che l'API rileva i volti, non riconosce le persone, quindi è possibile ottenere le informazioni necessarie per eseguire attività come creare selfie e ritratti o generare avatar dalla foto di un utente. ML Kit può eseguire il rilevamento dei volti in tempo reale, pertanto è ottimo per videochiamate o giochi che rispondono alle espressioni del giocatore.

Rilevamento di mesh per i volti

Con l'API di rilevamento della mesh dei volti si può generare in tempo reale una mesh[3] ad alta precisione di 468 punti 3D (Figura 2.2) per immagini simili a quelle dei selfie. I volti devono essere al massimo a 2 metri di distanza dalla fotocamera.

[2] In Computer Vision la classificazione si riferisce alla categorizzazione di un'immagine, o di una regione di interesse, in una classe specifica; ad esempio, classificare un animale in base alla sua specie.

[3] Il termine "mesh" si riferisce a una rappresentazione tridimensionale di una superficie o di un oggetto virtuale.

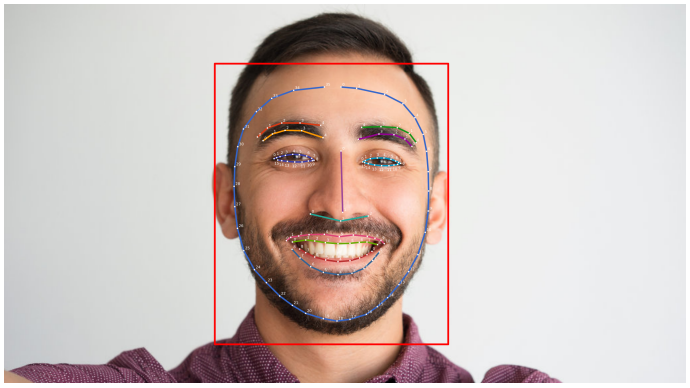

Figura 2.1: Rilevamento facciale (ML Kit)

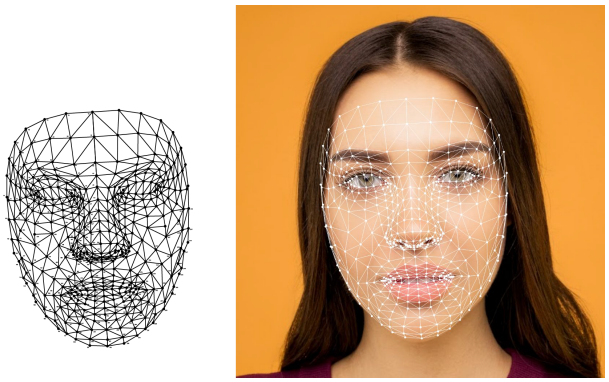

Figura 2.2: Rilevamento di mesh per i volti (ML Kit)

Riconoscimento del testo

L'API "Text Recognition v2" è in grado di riconoscere il testo in qualsiasi set di caratteri, siano essi cinese, devanagari, giapponese, coreano o latino. L'API può essere utilizzata anche per automatizzare le attività di inserimento dati, come l'elaborazione di carte di credito, ricevute e biglietti da visita. Il testo viene suddiviso in blocchi, linee, elementi e simboli (Figura 2.3).

Etichettatura delle immagini

Con le API di etichettatura delle immagini di ML Kit è possibile rilevare ed estrarre informazioni sulle entità in un'immagine in un ampio gruppo di categorie. Il modello di etichettatura delle immagini predefinito può identificare oggetti generici, luoghi, attività, specie animali, prodotti e altro ancora.

Rilevamento e monitoraggio di oggetti

Con l'API di rilevamento e monitoraggio di oggetti on-device di ML Kit, è possibile rilevare e monitorare gli oggetti in un'immagine, o tramite la videocamera dal vivo. In particolare, l'API rileva le entità e individua la loro posizione nell'immagine; essa consente di tenere traccia degli oggetti tra cornici di immagini successive, li classifica in ampie categorie ed è in grado di determinare automaticamente l'oggetto più in evidenza in un'immagine.

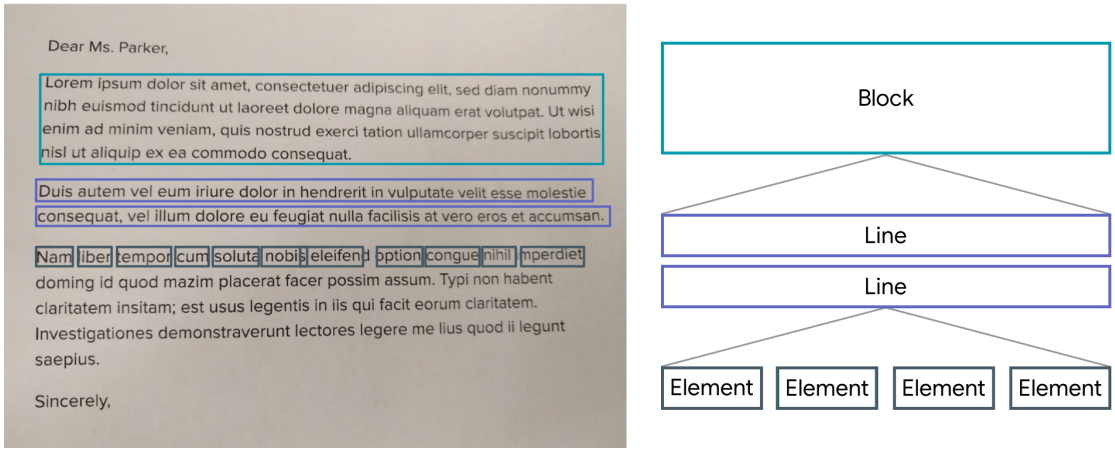

Figura 2.3: Riconoscimento del testo (ML Kit)

Figura 2.4: Riconoscimento dell'inchiostro digitale (ML Kit)

Riconoscimento dell'inchiostro digitale

L'API di riconoscimento dell'inchiostro digitale di ML Kit apre la possibilità all'identificazione di testo scritto a mano, alla classificazione dei gesti su una superficie digitale e degli schizzi (Figura 2.4). Risulta piuttosto utile per l'input sul dispositivo di lingue che possiedono un set di caratteri troppo numeroso per essere confinato in una tastiera, sia essa fisica o digitale, e che necessita l'impiego della scrittura a mano libera.

Segmentazione dei selfie

L'API di segmentazione dei selfie di ML Kit consente agli sviluppatori di separare facilmente lo sfondo dalle persone (Figura 2.5) all'interno di una scena per poi aggiungere effetti ai selfie o inserire i personaggi in ambienti differenti o più interessanti. L'API acquisisce un'immagine di input e produce una maschera[4] di output, funziona con frame statici o video in tempo reale.

Segmentazione dei soggetti

L'API di segmentazione dei soggetti consente di separare più soggetti dallo sfondo in un'immagine, per casi d'uso come la creazione di adesivi o la sostituzione dello sfondo. Per soggetti si intendono le persone, gli animali o gli oggetti più in evidenza nell'immagine. Come nel caso precedente, l'API di segmentazione dei soggetti accetta un'immagine di input e genera una maschera di output per ogni elemento (Figura 2.6).

[4]In Computer Vision, una "maschera" si riferisce a un'immagine binaria, o a tonalità di grigio, utilizzata per identificare e isolare specifiche regioni di interesse all'interno di un'immagine. Queste maschere vengono comunemente utilizzate per segmentare una scena, ovvero dividere l'immagine in diverse parti o regioni in base alle caratteristiche desiderate.

Figura 2.5: Segmentazione dei selfie (ML Kit)

Figura 2.6: Segmentazione dei soggetti (ML Kit)

2.1.2 API Natural Language

Il *Natural Language Processing* (NLP - letteralmente "Elaborazione del Linguaggio Naturale"), è un campo dell'Intelligenza Artificiale che si occupa di consentire alle macchine di comprendere, interpretare e generare il linguaggio umano in modo significativo. L'obiettivo principale dell'NLP è di abilitare la comunicazione tra le persone e i computer attraverso il linguaggio naturale. I concetti chiave dell'NLP [5] sono riassunti di seguito.

- *Analisi del linguaggio e traduzione automatica.* L'NLP si occupa di analizzare il linguaggio umano in ogni sua forma; ciò include la comprensione delle regole grammaticali, la semantica e la struttura del linguaggio stesso. Grazie a tale sistematicità esso permette la traduzione automatica di grandi porzioni di testo.
- *Riconoscimento di entità.* L'NLP può essere utilizzato per identificare e classificare entità all'interno del testo, come persone, luoghi, date, e altro ancora. Questo è particolarmente utile per estrarre informazioni significative da grandi quantità di dati testuali.
- *Generazione di linguaggio.* Questa capacità consente alle macchine di generare testo in modo coerente e significativo. È utilizzata in applicazioni come la creazione di descrizioni automatiche di immagini o la produzione di testo creativo.

Questo campo è in costante evoluzione grazie a progressi nelle tecnologie di apprendimento automatico e all'aumento della disponibilità di dati linguistici. La seconda categoria di API (API Natural Language) contiene le soluzioni offerte da ML Kit nel campo dell'NLP; queste ultime sono elencate nelle prossime sezioni.

Identificazione della lingua

Con l'API di identificazione della lingua si può determinare la lingua di una stringa di testo. Ciò può essere utile quando si lavora con il testo fornito dall'utente, che spesso non include informazioni sulla provenienza. Essa offre un supporto linguistico su vasta scala, infatti, identifica oltre cento lingue diverse e supporta testi in arabo, bulgaro, greco, hindi, giapponese, russo e cinese, sia in scrittura nativa che in caratteri romani.

Traduzione

Grazie all'API di traduzione on-device si è capaci di tradurre in modo dinamico testo tra più di 50 lingue. I modelli di traduzione sono basati sugli stessi modelli utilizzati per la modalità offline del celebre Google Traduttore[5].

Risposta rapida

L'API "Smart Reply" di ML Kit consente di generare automaticamente risposte pertinenti ai messaggi ricevuti. È particolarmente utile su dispositivi con funzionalità di input limitate, come orologi od occhiali smart, sempre più diffusi. Il modello genera suggerimenti di risposta in base al contesto completo di una conversazione, non solo a un singolo messaggio; ciò significa che i suggerimenti sono più utili per gli utenti. Purtroppo soltanto la lingua inglese è attualmente supportata.

Estrazione di entità

La maggior parte delle app offre agli utenti pochissima interazione con il testo oltre alle operazioni di base di taglio, copia e incolla. L'estrazione delle entità migliora l'esperienza utente all'interno dell'applicazione comprendendo il testo in maniera profonda, riuscendo a confinare porzioni dello stesso in entità affini, ad esempio comprendendo quale parte di un input testuale si riferisce a un indirizzo stradale e quale invece a un nome proprio di persona.

2.2 ML Kit e pose detection

Come si è inteso dalla sezione precedente, ML Kit offre una vasta gamma di soluzioni in ambito di Computer Vision e Natural Language Processing; il progetto da realizzare sfrutta un set di API che rientrano nella prima classe (API Vision); tale set è conosciuto come "ML Kit Pose Detection".

2.2.1 Approccio al problema

L'API "ML Kit Pose Detection" è una soluzione versatile e leggera per gli sviluppatori di app che consente di rilevare la posizione del corpo di un soggetto in tempo reale da un video continuo o da un'immagine statica. Una posa descrive la posizione del corpo in un momento temporale con una serie di punti di riferimento scheletrici.

[5]Google Traduttore è un servizio di traduzione automatica fornito da Google. Lanciato nel 2006, questo strumento offre la possibilità di tradurre testi, documenti, pagine web e, persino, discorsi in tempo reale tra molte lingue diverse.

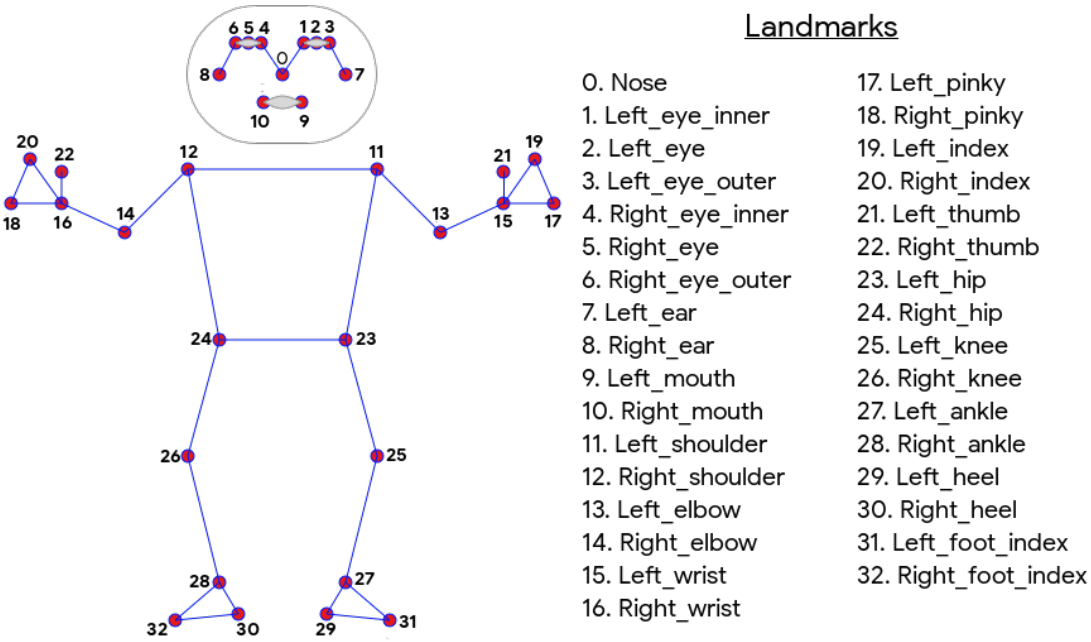

Figura 2.7: Punti di riferimento secondo lo schema BlazePose

MediaPipe e BlazePose

I punti di riferimento corrispondono a diverse parti del corpo, etichettate secondo la topologia[6] *BlazePose* [2]. Si tratta di un nuovo approccio che permette il monitoraggio della posa per dedurre 33 punti scheletrici 2D da un singolo fotogramma. È stato introdotto da poco all'interno di *MediaPipe*, la libreria cardine di ML Kit, sviluppata anch'essa da Google, che rappresenta il core di tutti i processi per l'elaborazione di immagini e video. A differenza degli attuali modelli di posa basati sulla topologia *COCO*[7] standard, BlazePose localizza accuratamente più punti chiave, il che lo rende particolarmente adatto per le applicazioni di fitness. La Figura 2.7 visualizza con precisione i differenti punti scheletrici e come sono collegati tra loro. Localizzando i vari punti dell'immagine in input tramite le coordinate d'immagine[8] è possibile, pertanto, utilizzare la posizione relativa dei punti di riferimento per distinguere una posa dall'altra.

Per la stima della posa, MediaPipe sfrutta la sua collaudata pipeline di Machine Learning con detector/tracker in due fasi (Figura 2.8). Utilizzando un detector, viene individuata, innanzitutto, una regione di interesse (ROI) all'interno del fotogramma, ovvero la porzione dell'immagine nella quale si trova il soggetto da tracciare; successivamente, il tracker prevede tutti i 33 punti chiave della posa all'interno della regione. Nel caso si stesse processando un video il detector viene eseguito solo sul primo fotogramma, per i fotogrammi successivi si ricava la ROI dai punti chiave della posa del fotogramma precedente.

[6] In Informatica, il termine "topologia" è spesso utilizzato per descrivere la struttura fisica o logica di una rete di nodi. La topologia di rete definisce il modo in cui i nodi della rete (in questo caso, i punti del corpo) sono collegati tra loro.

[7] Lo standard attuale per la posa del corpo umano è la topologia *COCO* [2], che consiste di 17 punti di riferimento su busto, braccia, gambe e viso.

[8] Le coordinate d'immagine si riferiscono alla posizione di un punto specifico in un'immagine rispetto al suo sistema di coordinate. La coordinata x indica la posizione orizzontale del punto lungo l'asse delle ascisse, che, solitamente, va da sinistra a destra nell'immagine. La coordinata y indica la posizione verticale del punto lungo l'asse delle ordinate, che, solitamente, va dall'alto verso il basso nell'immagine. Queste coordinate sono spesso espresse in pixel.

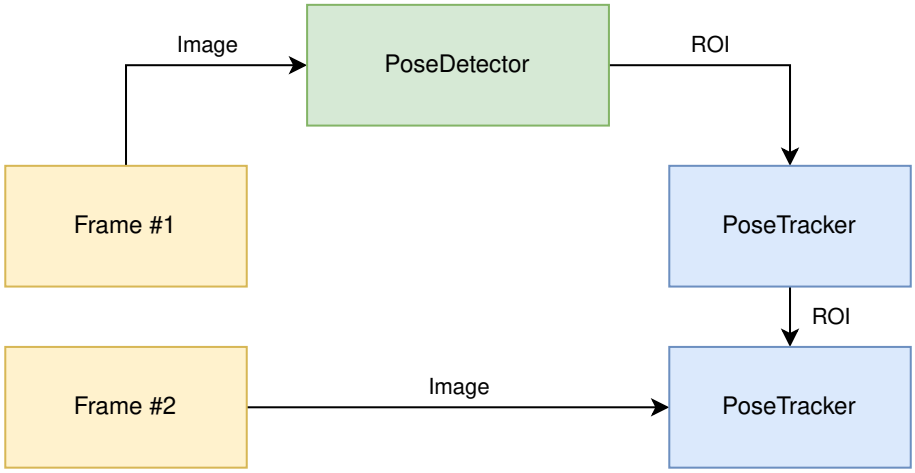

Figura 2.8: Pipeline per il tracking MediaPipe

Figura 2.9: Java e Kotlin

Figura 2.10: Swift e Objective-C

2.2.2 Funzionalità chiave

Supporto per più piattaforme

Come ripetuto più volte, è fondamentale la possibilità di sviluppo dell'applicazione sia per piattaforme Android che per iOS, essendo questi ultimi i principali competitor del mercato odierno. ML Kit supporta entrambe le parti offrendo librerie compatibili con Kotlin e Java[9] o con Swift e Objective-C[10].

Punteggio di confidenza

Per ogni punto di riferimento, la pipeline di MediaPipe calcola una misura che indica la probabilità che il punto si trovi all'interno del frame dell'immagine. Il punteggio ha un intervallo da 0.0 a 1.0, dove 1.0 indica un'elevata affidabilità.

Due SDK ottimizzati

ML Kit mette a disposizione due SDK[11] distinti, uno di base e uno per task che richiedono più precisione. L'SDK di base viene eseguito in tempo reale sui dispositivi, restituisce i risultati, rispettivamente, a una frequenza tra 30 e 45 fps[12] attuando un compromesso con la precisione dei punti di riferimento tracciati, i quali, a ogni modo,

[9]Java e Kotlin (Figura 2.9) sono i principali linguaggi di programmazione per lo sviluppo di applicazioni Android; possono essere utilizzati insieme all'interno di un progetto, beneficiando della loro interoperabilità. Questa capacità consente agli sviluppatori di migrare gradualmente il codice esistente in Java a Kotlin o di integrare nuovi componenti scritti in Kotlin in progetti Java esistenti.

[10]Swift e Objective-C (Figura 2.10) sono uno il linguaggio successore dell'altro, rappresentano la soluzione nativa per lo sviluppo di applicazioni iOS.

[11]SDK, acronimo di "Software Development Kit", è un insieme di strumenti, librerie, documentazione e risorse che agevolano lo sviluppo di software per una specifica piattaforma, sistema operativo o framework.

[12]Frames Per Second, ovvero il numero di fotogrammi al secondo.

Figura 2.11: Cristiano Ronaldo[13]celebra un gol a San Siro

rimangono sostanzialmente molto accurati. L'SDK preciso restituisce, invece, i risultati con una frequenza di fotogrammi più lenta, ma produce valori delle coordinate più precisi.

Coordinata Z

Alle coordinate d'immagine di cui si è discusso precedentemente viene aggiunta anche una coordinata "Z". Anche questa si misura in pixel ma non è un vero valore 3D; essa aiuta, semplicemente, a determinare se parti del corpo del soggetto si trovano davanti o dietro ai fianchi; ciò si rivela utile nel riconoscimento di pose nelle quali il corpo umano risulta piegato in avanti o all'indietro. L'asse z è perpendicolare alla fotocamera e l'origine è posta approssimativamente nel punto centrale tra i fianchi del soggetto. I valori z negativi identificano punti rivolti alla fotocamera; i valori positivi, invece, quelli più lontani; in ogni caso, non c'è un limite superiore o inferiore per tale coordinata.

2.2.3 Risultati d'esempio

Per dare una dimostrazione pratica di quanto detto finora, è stata utilizzata la pipeline di MediaPipe per il tracking di un soggetto particolare. La Figura 2.11 mostra visivamente i risultati del test, mentre la Tabella 2.1 elenca le coordinate dei punti di riferimento del modello BlazePose. Si noti come le coordinate x, y, z sono coerenti con quanto rappresentato nell'immagine, di dimensioni 5568x3712 pixel; ciò certifica la buona accuratezza della soluzione adottata.

[13]Cristiano Ronaldo è considerato da molti il più grande giocatore della storia del calcio. È noto per celebrare i suoi gol con un gesto caratteristico, conosciuto come "Sium" o "Siuuu!".

	Punto di riferimento	x	y	z
0	nose	2956	715	626
1	left_eye_inner	2956	642	391
2	left_eye	2953	640	391
3	left_eye_outer	2949	639	391
4	right_eye_inner	2982	637	631
5	right_eye	2998	633	632
6	right_eye_outer	3015	627	632
7	left_ear	3003	645	-405
8	right_ear	3098	626	665
9	mouth_left	2975	765	368
10	mouth_right	3008	763	679
11	left_shoulder	3015	979	-1113
12	right_shoulder	3532	992	990
13	left_elbow	3026	1517	-1504
14	right_elbow	3738	1550	1013
15	left_wrist	2952	2055	-1474
16	right_wrist	3866	2060	631
17	left_pinky_1	2919	2197	-1666
18	right_pinky_1	3928	2191	632
19	left_index_1	2902	2201	-1631
20	right_index_1	3877	2205	487
21	left_thumb_2	2924	2155	-1442
22	right_thumb_2	3832	2151	553
23	left_hip	3198	2023	-564
24	right_hip	3552	2001	562
25	left_knee	2844	2734	-370
26	right_knee	3839	2725	960
27	left_ankle	2744	3582	99
28	right_ankle	4203	3453	1235
29	left_heel	2816	3711	129
30	right_heel	4268	3596	1244
31	left_foot_index	2443	3650	-151
32	right_foot_index	4103	3561	946

Tabella 2.1: Coordinate *x, y, z* dei punti di riferimento BlazePose

CAPITOLO 3

Classificazione delle pose

In seguito alla raccolta dei requisiti, alla valutazione dello stato dell'arte, all'identificazione e all'analisi della tecnologia più adatta, è, ora, il momento di procedere con l'implementazione software del progetto in questione. Questo capitolo affronta le modalità tramite le quali si può realizzare un sistema di riconoscimento delle pose. In particolare, nella prima sezione, si richiamano le basi indispensabili riguardo alla classificazione nel Machine Learning; nella seconda parte, invece, si spiega come predisporre il dataset[1] utilizzato per il riconoscimento stesso.

3.1 Meccanismo di classificazione

Il progetto che si sta portando avanti, oltre a rientrare nel campo della realtà aumentata, pone le proprie basi nel ramo informatico dell'Intelligenza Artificiale e, in particolare, in quello della Computer Vision e del Machine Learning, come più volte chiarito nel Capitolo 2. La libreria MediaPipe [2] permette di ottenere un set di punti scheletrici a partire da un'immagine rappresentante un soggetto umano; è compito, poi, degli algoritmi di Machine Learning sfruttare tali informazioni per mettere in piedi il meccanismo di riconoscimento vero e proprio.

3.1.1 Richiami di Data Mining

Il Machine Learning, introdotto nel Capitolo 2, può essere considerato come una componente del Data Mining [16], ovvero di quella procedura di esplorazione e analisi dei dati volta a scoprire pattern, relazioni e informazioni significative al loro interno. In particolare, il Data Mining è, a sua volta, uno degli anelli della catena del KD (Knowledge Discovery, letteralmente "scoperta della conoscenza"), ovvero del processo che converte i dati grezzi, come dati sensoristici o analisi statistiche, in informazioni utili e contestualizzate, le quali rappresentano, appunto, conoscenza.

Il Data Mining richiede un livello molto alto di interdisciplinarità [16] (Figura 3.1), in particolar modo necessita di conoscenza riguardo:

[1] Un dataset è un insieme di dati organizzati in una struttura specifica, solitamente rappresentante informazioni relative a un determinato argomento o fenomeno. Un dataset può contenere dati di diverso tipo, come testo, numeri, immagini, o altro, e può essere utilizzato per condurre analisi, studi, o per addestrare algoritmi di Machine Learning.

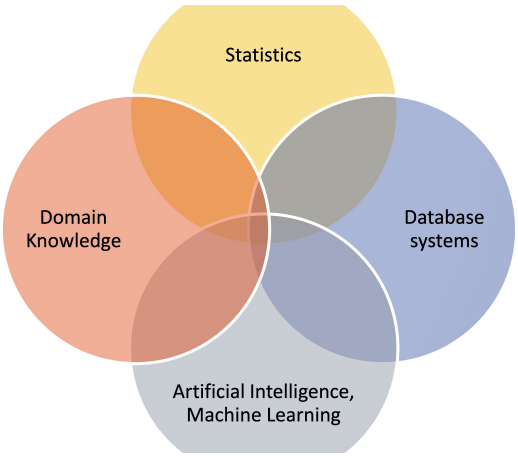

Figura 3.1: Interdisciplinarità del Data Mining

- il dominio d'interesse, per intendere il significato che i dati assumono nel contesto applicativo;
- la Statistica, perchè spesso è necessario effettuare campionamenti[2] dei dati e condurre test d'ipotesi[3];
- i sistemi di database, in quanto la mole dei dati è molto grande e questi vanno conservati in sistemi di memorizzazione efficienti;
- il Machine Learning, il quale comprende tutti gli strumenti specifici per applicare gli algoritmi di IA.

L'attività di Data Mining include, sostanzialmente, due distinte tipologie di task, esposte di seguito.

1. *Task descrittivi.* Fanno parte di questa categoria tutte le attività che descrivono i dati in modo sintetico e comprensibile dall'uomo. I principali esempi di task descrittivi sono:
   - Association Analysis;
   - clustering;
   - Anomaly Detection.
2. *Task predittivi.* Si tratta di quelle attività che utilizzano i dati passati per predire l'andamento di quelli futuri, non ancora misurati. I principali esempi di task descrittivi sono:
   - classificazione (approfondita nella Sezione 3.1.2);
   - regressione.

[2] In Statistica, il campionamento è un processo mediante il quale vengono selezionati e raccolti dati da un sottoinsieme di una popolazione più ampia. In altre parole, invece di raccogliere informazioni da tutti gli elementi di una popolazione, si prende un "campione", cioè un gruppo rappresentativo, e si effettuano le misurazioni o le osservazioni su di esso.

[3] Il test d'ipotesi è una procedura statistica utilizzata per valutare se ci siano prove sufficienti a respingere o meno un'affermazione riguardo a una caratteristica di un campione di dati.

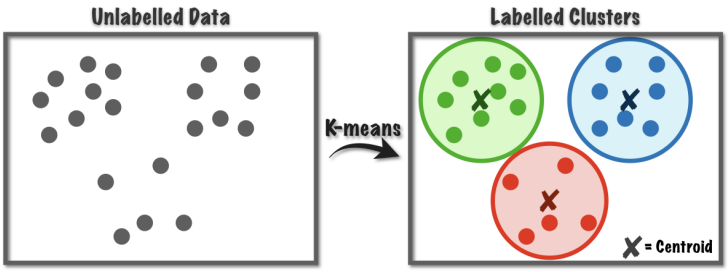

Figura 3.2: Effetti di un algoritmo di Clustering

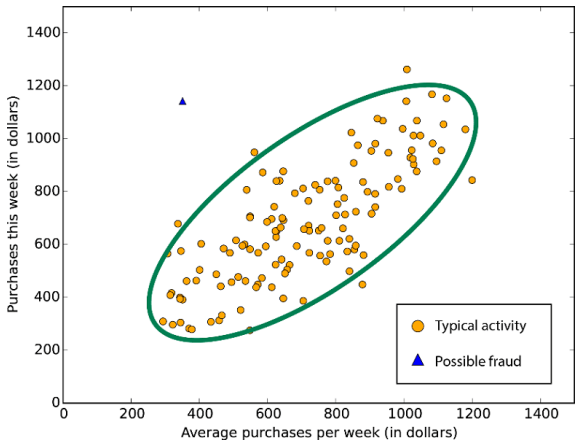

Figura 3.3: Esempio di Fraud Detection

Association Analysis

L'Association Analysis, o scoperta di regole associative, è utilizzata per trovare schemi nascosti nei dati che descrivono caratteristiche fortemente correlate tra loro. Tali regole sono insiemi di condizioni nella forma $R : X \Rightarrow Y$, ovvero regole *if-then*, le quali formalizzano la dipendenza tra l'occorrenza di un dato e l'occorrenza di altri. Un'applicazione dell'Association Analysis è, sicuramente, la Market Basket Analysis [16], utilizzata nei supermercati per cercare prodotti che vengono spesso acquistati assieme (ad esempio, *Birra* $\Rightarrow$ *Patatine*) e che, ragionevolmente, sono posti su scaffali adiacenti, in modo da invogliare i consumatori a spendere di più.

Clustering

Il clustering, o segmentazione, è una tecnica che consiste nel dividere i dati in insiemi omogenei (*cluster*) contenenti elementi simili tra loro e dissimili dagli elementi che compongono gli altri cluster (Figura 3.2). L'obiettivo è quello di massimizzare la similarità interna e minimizzare la similarità esterna tra i cluster.

Anomaly Detection

Il riconoscimento di anomalie consiste nell'identificare elementi le cui caratteristiche sono sostanzialmente differenti da quelle del resto dei dati; tali elementi vengono denominati *outlier* (anomalie). Le applicazioni di questa tecnica risiedono principalmente nel campo del rilevamento delle frodi (*Fraud Detection* - Figura 3.3) e dell'analisi del traffico di reti alla ricerca di intrusi (*Network Intrusion Detection System*).

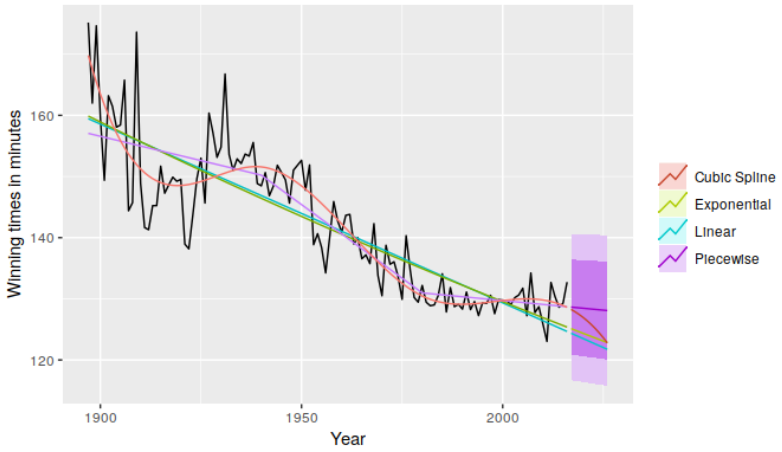

Figura 3.4: Regressione dei tempi record della maratona di Boston

Regressione

La regressione è una tecnica statistica utilizzata per studiare la relazione tra una variabile dipendente (*target*) e una o più variabili indipendenti (*predictor*). L'obiettivo della regressione è modellare questa relazione in modo da poter fare previsioni o inferire il valore numerico della variabile dipendente in base alle variabili indipendenti. La regressione è utilizzata soprattutto nella previsione di serie temporali, le quali compaiono in numerosi ambiti, tra cui la finanza, l'economia, la meteorologia o le scienze ambientali (Figura 3.4).

3.1.2 Classificatori

La classificazione è un processo di apprendimento automatico che consiste nell'assegnare un'etichetta, o una categoria (classe), a un insieme di dati sulla base di modelli indotti[4] da informazioni precedentemente etichettate. In altre parole, l'obiettivo della classificazione è quello di costruire un modello che possa creare generalizzazioni a partire da esempi di dati di cui si conosce la classe di appartenenza per poi fare predizioni su nuovi dati, assegnandoli a una delle categorie predefinite.

Le tecniche di classificazione (o classificatori) sono approcci sistematici per la costruzione di modelli a partire da dataset di input [17]. Generalmente, ciascuno di questi approcci segue una serie di passi comuni, spiegati di seguito.

1. *Generazione di un training set.* La classificazione è un esempio di apprendimento supervisionato, ovvero ha bisogno di partire da un dataset etichettato a priori per poter, poi, indurre un modello di classificazione. Tale dataset prende il nome di *training set*.

2. *Costruzione di un modello di classificazione.* L'assunzione che viene fatta nella classificazione è che i vari elementi abbiano attributi che dipendono dalla classe di appartenenza. Analizzando il training set e come le varie caratteristiche degli elementi etichettati incidono sulla classe assegnata, è possibile, poi, scegliere quale approccio seguire per l'induzione di un modello. Tra le tecniche di classificazione più famose [17] compaiono:

    - alberi di decisione (Decision Tree Classifiers);

[4] L'induzione è un ragionamento logico che consiste nel formulare generalizzazioni o conclusioni basate su osservazioni specifiche o casi particolari.

- classificatori basati su regole (Rule-Based Classifiers);
- classificatori Bayesiani;
- reti neurali (Artificial Neural Network - ANN);
- Support Vector Machines (SVM);
- classificatori "Nearest-Neighbor".

L'approfondimento delle varie soluzioni esula dalle competenze del progetto.

3. *Classificazione e valutazione delle metriche.* Il modello indotto tramite le tecniche sopracitate è utilizzato per predire la classe degli elementi di un secondo dataset, non etichettato a priori, chiamato *test set*. È possibile scegliere un classificatore rispetto a un altro valutandone la bontà, ovvero quanto riesce a classificare correttamente i vari elementi del test set. La metrica più utilizzata per misurare la correttezza dei classificatori è la *accuracy*, calcolata come segue:

$$Accuracy = \frac{\text{Numero di predizioni corrette}}{\text{Numero di predizioni totali}}$$

Gli algoritmi di classificazione, a loro volta, presentano un'ulteriore divisione interna [18]. Essi sono *eager* (diligenti) se attuano la fase di addestramento e di costruzione del modello in modo tempestivo; questo tipo di approccio tende a imparare il massimo possibile dai dati di training, cercando di adattarsi quanto prima alle informazioni fornite. Gli algoritmi di apprendimento diligente possono essere più esigenti in termini di risorse computazionali durante la fase di addestramento, ma, una volta addestrati, sono pronti a fare previsioni o compiere azioni senza dover consultare nuovamente i dati. Esempi di metodi eager sono gli alberi di decisione e i classificatori basati su regole.

Diverso è il discorso per gli algoritmi *lazy* (pigri), i quali, al contrario, adottano un approccio più conservativo; invece di cercare di imparare dai dati il massimo possibile durante il processo di training, posticipano le decisioni di generalizzazione fino a quando non è strettamente necessario, ovvero nella fase di classificazione vera e propria. Un primo esempio di metodo lazy fu il *Rote classifier*, un approccio che memorizzava l'intero training set e classificava un elemento di test identicamente a uno di train soltanto se tutti gli attributi dei due corrispondevano. Un'ovvia problematica di tale tecnica risiede nel fatto che, nella maggior parte dei casi pratici, diversi elementi di test non trovano corrispondenze nel training set e rimangono non classificati. È per tale motivo che sono stati introdotti i classificatori "Nearest-Neighbor", i quali si configurano come un rilassamento del Rote classifier in quanto non cercano all'interno del training set delle corrispondenze esatte, ma, piuttosto, delle similitudini. L'idea di tale approccio è quella di classificare un elemento in base alla classe di appartenenza degli elementi di train a esso più vicini[5], assumendo che dati residenti in zone limitrofe siano caratterizzati da attributi con valori simili.

ML Kit predispone un algoritmo di tipo "Nearest-Neighbor" per la classificazione delle pose, questo è il *k-NearestNeighbor (k-NN)*.

3.1.3 k-NN per la classificazione delle pose

L'approccio di ML Kit alla classificazione delle pose è fortemente ispirato alle tecniche "Nearest-Neighbor", in particolare all'algoritmo k-NN [18]. Quest'ultimo, come le altre

[5]Nel Data Mining, la vicinanza tra due elementi è formalizzata attraverso indici specifici, detti *indici di similarità*, che misurano quanto i dati siano correlati tra loro, ovvero quanto analoghi siano i concetti che descrivono; alcuni esempi sono la distanza Euclidea, la distanza Manhattan, il coefficiente di correlazione o l'indice di Jaccard.

tecniche analoghe, si basa sul principio secondo il quale oggetti simili sono vicini nello spazio delle caratteristiche; in altre parole, se due oggetti hanno caratteristiche simili, è probabile che appartengano a una stessa classe. k-NN segue i passi che sono elencati di seguito e che saranno ulteriormente approfonditi nel Capitolo 4.

1. *Input.* k-NN riceve in input, come gli altri classificatori lazy, un inseme di dati di train e un nuovo elemento del quale si vuole conoscere la classe.

2. *Calcolo delle distanze.* Viene calcolata la distanza (tramite gli indici di similarità, introdotti precedentemente) tra il nuovo elemento e quelli etichettati.

3. *Selezione dei k elementi vicini.* Vengono identificati i k punti di training più vicini al nuovo punto; k è un parametro numerico che viene impostato sulla base dei dati in esame. La scelta del valore di k è critica; un k troppo piccolo può rendere l'algoritmo sensibile al rumore[6] nei dati, mentre un k troppo grande può rendere l'algoritmo meno reattivo alle variazioni degli stessi.

4. *Classificazione.* Si assegna al nuovo elemento la classe che compare più frequentemente tra i k elementi più vicini; si tratta, dunque, di un voto "a maggioranza".

È, dunque, evidente che il requisito fondamentale per avviare il processo di riconoscimento è quello di disporre di un dataset di training con il quale confrontare ogni nuova posa rilevata dalla tecnologia MediaPipe (Sezione 2.2). Tale dataset sarà una vera e propria "libreria" di pose conosciute contenente le informazioni necessarie a distinguere una posa dall'altra. Il processo di costruzione di tale training set è esposto nella sezione successiva.

3.2 Creazione del dataset

Una volta scelto il classificatore, è necessario ragionare su quale sia il modo migliore per memorizzare in un dataset le informazioni sulla posa corporea, le quali, al momento, vengono rappresentate in coordinate d'immagine secondo il modello BlazePose, precedentemente introdotto.

3.2.1 Python e Google Colab

La libreria di ML Kit MediaPipe supporta diversi linguaggi di programmazione; tra questi compare Python, che, a oggi, è una delle soluzioni più utilizzate dagli sviluppatori nel mondo grazie alla sua sintassi chiara, alla facilità di apprendimento e alla versatilità. Il processo di costruzione del training set sfrutta le funzioni di MediaPipe, implementate in Python, per il tracciamento del corpo umano.

Nel repository GitHub [6] associato al progetto è presente un Python Jupiter Notebook[7] che implementa il processo di creazione del training set. Durante lo sviluppo è stato utilizzato anche il servizio Google *Colab*, che fornisce un ambiente di esecuzione di Jupyter Notebook basato su cloud; esso è gratuito e offre risorse computazionali, compreso l'accesso

[6] Il "rumore nei dati" si riferisce a variazioni casuali, irrilevanti o indesiderate che possono essere presenti nei dati raccolti o misurati. Queste variazioni possono essere dovute a diversi fattori, tra cui errori di misurazione, interferenze casuali, eventi imprevisti o altre fonti di incertezza durante il processo di raccolta dei dati.

[7] Un Jupyter Notebook è un ambiente interattivo che consente di creare e condividere documenti contenenti codice, testo descrittivo, immagini e risultati di esecuzione. I notebook Jupyter sono ampiamente utilizzati nelle discipline scientifiche, nell'analisi dei dati, nell'apprendimento automatico e in altri campi in cui è necessario integrare codice e documentazione in modo interattivo.

alle unità di elaborazione grafica (GPU) e alle unità di elaborazione tensoriale[8] (TPU), senza richiedere alcuna configurazione hardware da parte dell'utente.

Il notebook è scritto in maniera piuttosto semplice; è sufficiente, durante l'esecuzione del codice, caricare su Google Colab un archivio ZIP contenente immagini che rappresentano pose corporee che popoleranno il dataset; tali immagini devono essere ulteriormente divise in sottocartelle denominate con l'etichetta che si vuole assegnare a ognuna delle classi. Per ottenere un buon training set, i manutentori ML Kit [2] suggeriscono di fornire al processo almeno 100 immagini per ognuna delle classi di posa.

3.2.2 Dump del dataset

Il processo di costruzione del dataset si basa su una funzione principale, ovvero quella che, a partire da ogni immagine, genera una riga di un file CSV, il quale costituirà il training set effettivo.

```
import cv2
from mediapipe.python.solutions import pose as mp_pose
def extract_landmarks():
    for class_name in pose_folders:
        image_names = sorted([n for n in os.listdir(class_name)
        for image_name in image_names:
            input_frame = cv2.imread(image_name)

            with mp_pose.Pose() as pose_tracker:
                result = pose_tracker.process(image=input_frame)
                pose_landmarks = result.pose_landmarks

            output_frame = input_frame.copy()
            if pose_landmarks is not None:
                mp_drawing.draw_landmarks(
                    image=output_frame,
                    landmark_list=pose_landmarks,
                    connections=mp_pose.POSE_CONNECTIONS)
            cv2.imwrite(image_name, output_frame)

            if pose_landmarks is not None:
                frame_height, frame_width =
                    output_frame.shape[0],
                    output_frame.shape[1]
                pose_landmarks = np.array([[
                    lmk.x * frame_width,
                    lmk.y * frame_height,
                    lmk.z * frame_width]
                    for lmk in pose_landmarks.landmark])
                csv_out_writer.writerow(
                    [image_name] +
                    [class_name] +
                    pose_landmarks.flatten().tolist())

            projection_xz = draw_xz_projection(output_frame, pose_landmarks)
            output_frame = np.concatenate(output_frame, projection_xz)
```

Listato 3.1: Funzione per l'estrazione dei 33 punti scheletrici

[8] L'elaborazione tensoriale si riferisce all'uso di tensori nelle operazioni e nei calcoli matematici; i tensori sono strutture matematiche generalizzate che estendono concetti come scalari, vettori e matrici a dimensioni superiori. L'elaborazione tensoriale è fondamentale in diversi campi, in particolare nell'ambito del Machine Learning.

image	class	nose (x)	nose (y)	nose (z)	...	right_foot_index (z)
001.jpg	squats_down	832	510	-658	...	-17
002.jpg	pushups_down	1699	774	-535	...	528
003.jpg	sium_end	2109	766	1621	...	-1064

Tabella 3.1: Esempio di training set

Il Listato 3.1 riporta, con opportune sintetizzazioni e semplificazioni, la funzione sopracitata. In particolare, per ogni classe di posa e per ogni immagine fornita:

1. si utilizza la libreria Python *OpenCV*, adibita alla Computer Vision, per gestire file binari di tipo multimediale (in questo caso delle immagini);
2. si inizializza un oggetto della classe *Pose* di MediaPipe per estrarre i 33 punti scheletrici del corpo umano;
3. una volta salvata l'immagine ottenuta, simile a quella presentata nella Sezione 2.2, vengono calcolate le esatte coordinate d'immagine (x, y) dei punti scheletrici, per poi salvarle in una riga di un file CSV;
4. viene aggiunta alla riga appena creata anche la coordinata z, stimata tramite il calcolo del punto medio tra i fianchi del soggetto presente in ogni immagine.

Completato il processo di dump[9] del dataset, quest'ultimo si configura come un file CSV, il quale ha tante righe quante sono le immagini fornite in input al processo completo. Tali righe contengono, nell'ordine:

- il nome dell'immagine di input;
- l'etichetta della classe di posa nota a priori, nonché, come anticipato, il nome della sottocartella contenente l'immagine stessa;
- la lista di coordinate x, y, z per ognuno dei 33 punti scheletrici.

Ciò vuol dire che ogni riga del dataset di training contiene 67 attributi; la Tabella 3.1 ne fornisce qualche esempio.

Per lo sviluppo del progetto sono state generate 15 righe di dataset (le quali, dopo i test d'uso del prodotto finale, si sono rivelate sufficienti) a partire da 15 immagini rappresentanti Cristiano Ronaldo, e altri calciatori, nell'intento di eseguire il "Sium", introdotto nel Capitolo 2. Tali righe vanno ad aggiungersi a quelle, già numerose, predisposte dagli sviluppatori di ML Kit all'interno di un dataset dimostrativo; queste ultime comprendono due tipologie di pose, gli squat e i piegamenti sulle braccia.

[9] In campo informatico, un "database dump" è un'esportazione completa o parziale di un database in un formato leggibile o utilizzabile da altri sistemi.

CAPITOLO 4

Applicazione Android

Una volta chiarito il meccanismo grazie al quale si può classificare una posa corporea, giunge il momento di realizzare il prodotto con cui si interfacceranno gli utenti finali, ovvero l'applicazione mobile effettiva. Questo capitolo riassume lo sviluppo di tale applicazione descrivendone gli aspetti e le caratteristiche principali, per poi mostrare opportunamente come si presenta il prodotto finito e installato su un dispositivo mobile.

4.1 Scheletro dell'applicazione

Come più volte ripetuto, ML Kit è predisposto a lavorare sia su piattaforme Android che iOS, tuttavia, le tempistiche entro le quali è stato svolto il progetto in esame hanno obbligato la scelta di una delle due soluzioni. Android ha superato la concorrenza per diverse motivazioni:

- l'IDE[1] principale per lo sviluppo Android è Android Studio, il quale è supportato dalla maggior parte dei sistemi operativi desktop;
- è possibile testare l'applicazione su una gamma più ampia di dispositivi;
- generalmente, Android offre una maggiore flessibilità nell'integrazione con servizi di terze parti.

Android Studio

Android Studio (Figura 4.1) è un IDE basato sul software di JetBrains IntelliJ IDEA. È disponibile al download su Windows, macOS e Linux, diventando l'IDE primario di Google per lo sviluppo nativo di applicazioni Android. Tra le principali funzioni che rendono Android Studio preferibile agli altri IDE vi sono:

- un editor grafico per i file XML, i quali descrivono il layout delle varie schermate dell'app che si sta sviluppando;

[1] L'acronimo "IDE" sta per "Integrated Development Environment" (Ambiente di Sviluppo Integrato). Un IDE è un software che fornisce strumenti per gli sviluppatori, offrendo un ambiente unificato per la scrittura, il debugging e il testing del codice.

Figura 4.1: Android Studio

- un inspector per la directory di progetto che permette un'indicizzazione più ordinata dei file;
- la possibilità di eseguire le applicazioni su emulatori di Android per testarne il funzionamento nelle varie versioni del sistema operativo, con differenti livelli di API Android;
- il supporto a *git* per il *version control*[2].

4.1.1 Activity

In Android, una *Activity* [8] è un componente fondamentale dell'applicazione che rappresenta uno schermo o una finestra dell'interfaccia utente. Ogni schermata visibile dell'app, come la schermata principale, la schermata delle impostazioni o una schermata di dettaglio, è generalmente implementata come una Activity.

Le Activity seguono un ciclo di vita specifico che comprende vari stati, tra cui la creazione, l'avvio, la ripresa, la sospensione e la distruzione; esse sono responsabili della definizione e della gestione dell'interfaccia associata e possono includere layout, pulsanti, campi di testo e altri elementi con cui l'utente può interagire. Gli sviluppatori impostano il layout delle Activity utilizzando file XML che descrivono la disposizione e gli elementi nella schermata.

L'applicazione realizzata, interamente sviluppata in Java e reperibile dal repository GitHub [6] associato al progetto, contiene tre Activity principali:

- *ExerciseChooser*, ovvero l'entry point dell'applicazione, che gestisce le autorizzazioni[3] del sistema operativo e che permette di scegliere quale esercizio svolgere, o, in termini tecnici, con quale posa corporea avviare il processo di riconoscimento;
- *SettingsActivity*, per impostare alcune preferenze utente, approfondite nella sezione successiva;
- *LivePreviewActivity*, la schermata principale dell'applicazione, nonché quella che restituisce i risultati del riconoscimento della posa.

La Figura 4.2 permette di visualizzare i layout delle prime due Activity.

[2]Il version control è un sistema che registra le modifiche apportate ai file nel corso del tempo, consentendo agli sviluppatori di recuperare versioni precedenti del codice e di gestire efficacemente il lavoro collaborativo; il software di controllo di versione più utilizzato dagli sviluppatori è git.

[3]In Android, le autorizzazioni rappresentano il meccanismo mediante il quale le app richiedono l'accesso a determinate risorse o funzionalità del dispositivo. L'applicazione sviluppata richiede, ragionevolmente, l'utilizzo della fotocamera.

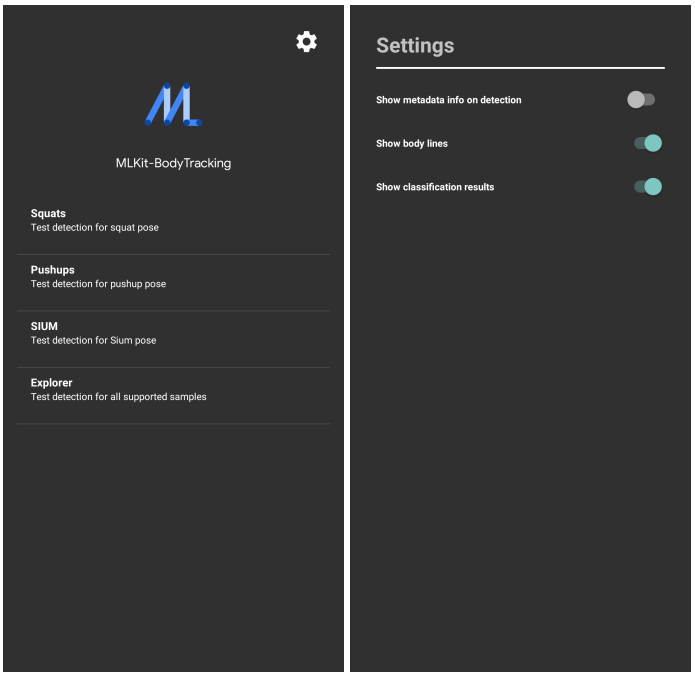

Figura 4.2: Layout di ExerciseChooser e SettingsActivity

LivePreviewActivity

La LivePreviewActivity è la schermata che visualizza graficamente i risultati della classificazione delle pose. Il Listato 4.1 riporta la struttura del file XML, il quale, definisce il layout dell'Activity.

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">

  <com.google.mlkit.vision.posedetection.camera.CameraSourcePreview
      android:id="@+id/preview_view"
      android:layout_width="match_parent"
      android:layout_height="match_parent"/>

  <com.google.mlkit.vision.posedetection.graphic.GraphicOverlay
      android:id="@+id/graphic_overlay"
      android:layout_width="match_parent"
      android:layout_height="match_parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Listato 4.1: Layout XML di LivePreviewActivity

Come si può notare, la struttura consta di tre elementi principali:

- un *ConstraintLayout* [10], ovvero un componente strutturale dell'interfaccia, il quale consente di incapsulare al suo interno altri elementi;
- un *CameraSourcePreview*, che permette la visualizzazione sullo schermo delle immagini catturate dalla fotocamera frontale;

- un *GraphicOverlay*, l'elemento che incorporerà i feedback provenienti dal meccanismo di classificazione.

Impostazioni

Le preferenze che l'utente può esprimere sono, volutamente, piuttosto limitate e permettono di decidere se:

- mostrare i metadati[4] del riconoscimento, come, ad esempio, la dimensione e gli fps dell'immagine catturata dalla fotocamera e inviata al processo di riconoscimento;
- visualizzare uno "scheletro" del corpo, ovvero evidenziare sullo schermo i punti scheletrici di BlazePose e le connessioni che intercorrono fra essi;
- mostrare le informazioni sulla classe riconosciuta tramite la fotocamera, in particolare l'etichetta (nome) della classe e il suo punteggio di confidenza (di cui si parlerà nella sezione successiva).

Tali impostazioni vengono salvate in un file memorizzato localmente sul dispositivo e persistente al ciclo di vita dell'applicazione stessa, secondo il meccanismo delle *Shared Preferences* [11].

4.1.2 Diagrammi dei processi

Per descrivere efficacemente il processo che permette il riconoscimento della posa in tempo reale a partire dalla fotocamera del dispositivo, si farà uso dei *sequence diagram* di UML[5], opportunamente sintetizzati e snelliti al fine di migliorarne la visualizzazione.

Acquisizione dei frame

Fintanto che il ciclo di vita della LivePreviewActivity è attivo, ovvero fintanto che l'utente rimane nella schermata di classificazione della posa, l'oggetto CameraSource, definito a partire dalla classe *Camera* di Android [9], processa ogni nuovo frame che perviene dalla fotocamera del dispositivo. L'immagine acquisita è un vettore di byte che viene convertito in un *ByteBuffer* [14] e, successivamente, in un oggetto *InputImage*, classe compatibile con le funzionalità di ML Kit. È compito del *PoseDetectorProcessor* dare il via al meccanismo di tracciamento e classificazione della posa. Quanto detto è schematizzato nel diagramma in Figura 4.3.

Classificazione della posa

Il PoseDetectorProcessor riceve, dunque, l'immagine catturata dalla fotocamera; questa viene, a sua volta, convertita in una posa corporea tramite la libreria MediaPipe (supportata nello sviluppo Android, come anticipato) che restituisce un oggetto di tipo *Pose*. La classe Pose contiene i metodi e gli attributi per la rappresentazione in oggetti di pose corporee; tali oggetti vengono passati al *PoseClassifier*, che implementa l'algoritmo k-NN (Sezione 4.2.2) per il riconoscimento. Infine, i risultati della classificazione vengono inviati al GraphicOverlay, introdotto precedentemente, che li visualizza sullo schermo. La Figura 4.4 riassume quanto spiegato.

[4] I metadati sono dati che forniscono informazioni su altri dati; in particolare, sono descrizioni o dettagli che spiegano, classificano, gestiscono o forniscono contesto aggiuntivo sui dati principali.

[5] UML, acronimo di Unified Modeling Language (Linguaggio di Modellizzazione Unificato), è uno standard di settore per la creazione di modelli visivi di software orientati agli oggetti.

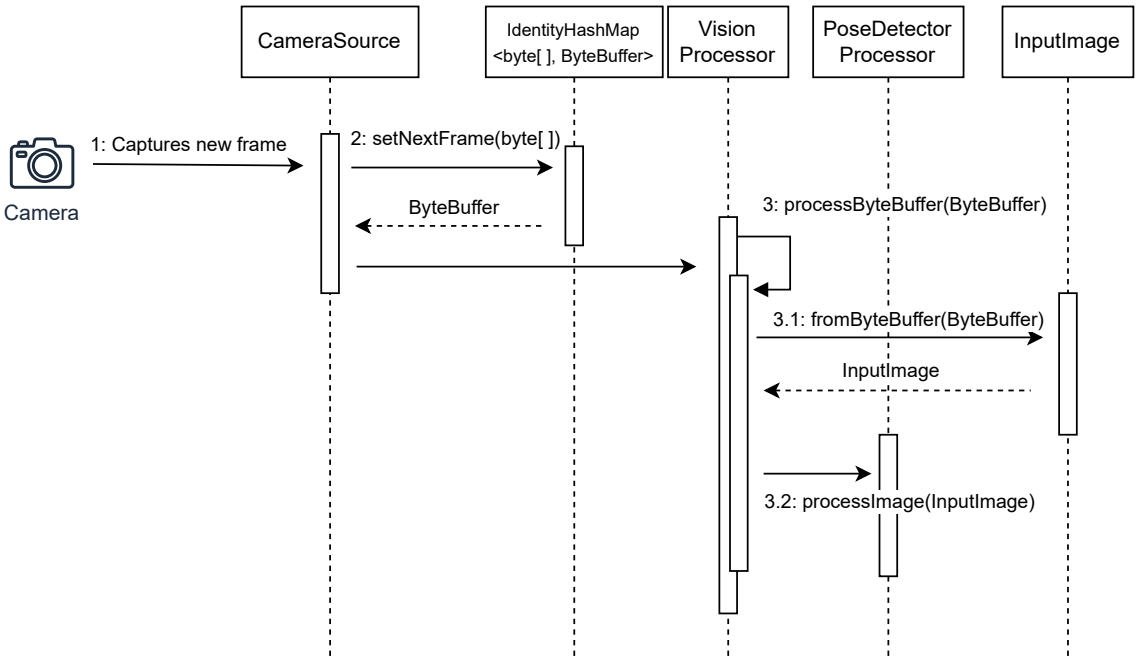

Figura 4.3: Diagramma di sequenza relativo all'acquisizione dei frame

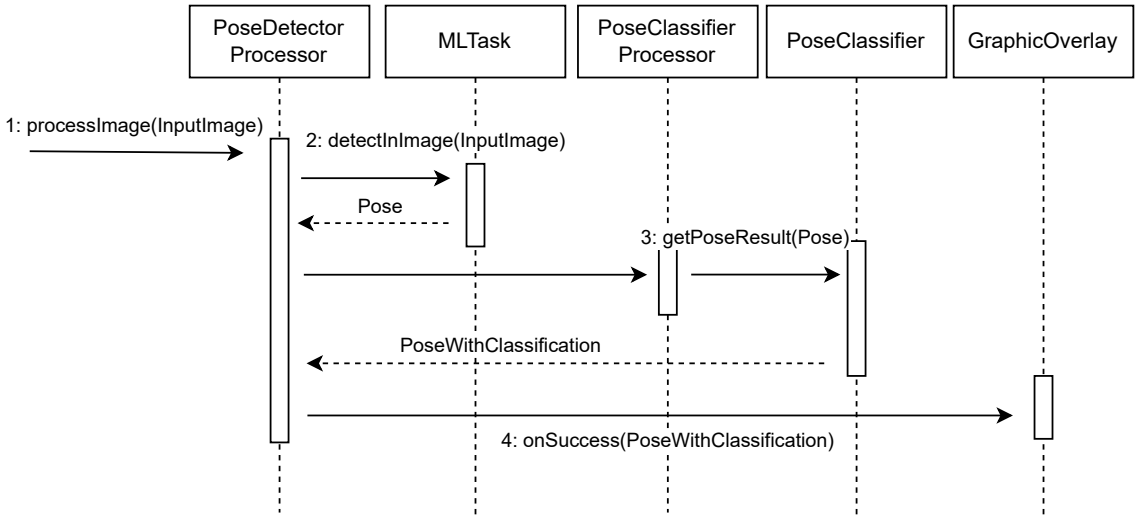

Figura 4.4: Diagramma di sequenza relativo alla classificazione della posa

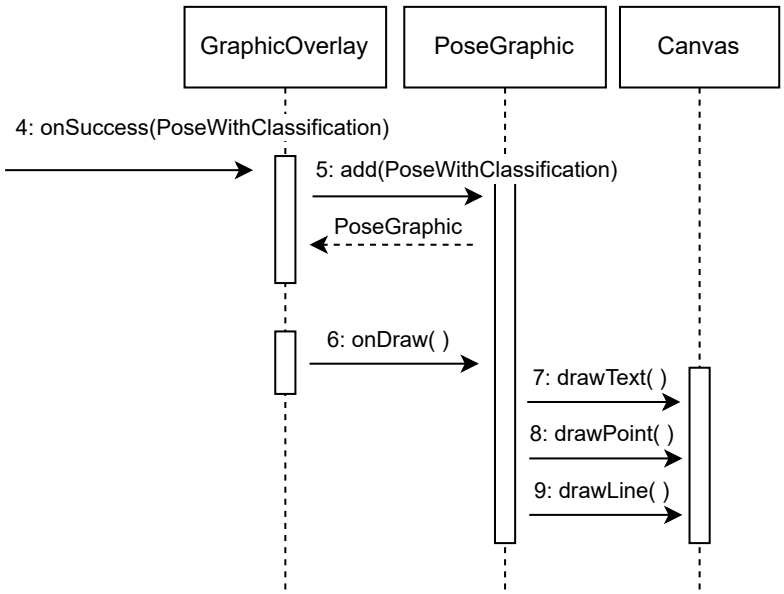

Figura 4.5: Diagramma di sequenza relativo alla restituzione dei risultati

Restituzione dei risultati

L'ultimo passaggio è quello della produzione del feedback per l'utente. Il classificatore produce due distinti risultati del task, ovvero l'etichetta della classe di posa riconosciuta e il livello di confidenza (approfondito nella sezione successiva) associato a tale classe. Entrambi vengono incorporati in un oggetto della classe *PoseGraphic* che si occupa di gestire la loro visualizzazione sullo schermo. A tale scopo si sfrutta una *View* [12] che, alla frequenza di aggiornamento del display, richiama la funzione *onDraw()* per disegnare su quest'ultimo:

- il testo contenente i risultati della classificazione;
- i punti scheletrici identificati da MediaPipe (viene visualizzata esclusivamente una parte dei 33 punti, per migliorare la visibilità del feedback);
- le linee colorate che congiungono i vari punti.

La resa grafica dei risultati verrà presentata nella Sezione 4.3.

4.2 Classificazione in real-time

L'algoritmo k-NN, come anticipato, valuta la somiglianza tra ogni nuova posa tracciata e tutte quelle presenti nel training set. A tale scopo è necessario utilizzare un indice di somiglianza che distingua in maniera efficace una classe di posa dall'altra, accomunando quelle analoghe.

Le sezioni che seguono descrivono i processi che stanno alla base del meccanismo di riconoscimento della posa: la normalizzazione dei dati e la classificazione.

4.2.1 Normalizzazione della posa

Tutte le pose del training set e ciascuna di quelle che perviene dalla fotocamera del dispositivo vengono sottoposte a un processo di normalizzazione[6], in modo da evitare che le stesse pose tracciate su due corporature umane differenti vengano etichettate distintamente. La tecnica utilizzata è la *min-max*, la cui formula generale per la normalizzazione di una variabile X è:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}.$$

In particolare:

- X è il valore originale della variabile;
- X_{min} e X_{max} sono, rispettivamente, il valore minimo e massimo che la variabile può avere;
- X_{norm} è il valore normalizzato.

Nel contesto applicativo, considerando che si stanno trattando punti tridimensionali, applicare questa formula vuol dire effettuare un cambiamento di origine e di scala. Il Listato 4.2 include le funzioni che implementano la normalizzazione; risulta chiaro che, per ogni posa (intesa come insieme di punti tridimensionali):

1. vengono calcolate le coordinate del punto medio tra i fianchi del soggetto inquadrato, considerato punto "centrale" e nuova origine (X_{min}) della posa;
2. ogni altro punto è traslato rispetto alla nuova origine ($X - X_{min}$);
3. viene calcolata la distanza[7] di ogni punto dall'origine e viene selezionata la distanza massima ($X_{max} - X_{min}$), escludendo da tale misura la coordinata artificiale z;
4. le variabili vengono scalate del fattore $\frac{1}{X_{max} - X_{min}}$ appena misurato.

```
public List<PointF3D> normalize(List<PointF3D> landmarks) {
    List<PointF3D> normalizedLandmarks = new ArrayList<>(landmarks);
    PointF3D hipsCenter = average(landmarks.get(PoseLandmark.LEFT_HIP),
    ↪ landmarks.get(PoseLandmark.RIGHT_HIP));
    subtractAll(hipsCenter, normalizedLandmarks);
    multiplyAll(normalizedLandmarks, 1 / getPoseSize(normalizedLandmarks,
    ↪ hipsCenter));
    return normalizedLandmarks;
}

public float getPoseSize(List<PointF3D> landmarks, PointF3D center) {
    float maxDistance = 0;
    for (PointF3D landmark : landmarks) {
        float distance = norm2D(subtract(center, landmark));
        if (distance > maxDistance) maxDistance = distance;
    }
    return maxDistance;
}
```

Listato 4.2: Funzioni per la normalizzazione delle pose

[6] La normalizzazione dei dati è una tecnica comune del Machine Learning, utilizzata per portare tutte le variabili di un dataset su una scala comune. Questo processo è importante perché molte tecniche di analisi sono sensibili alla scala delle variabili, le quali, se non normalizzate, non avrebbero un impatto equo durante l'addestramento del modello.

[7] Misurata come distanza tra punti in uno spazio 2D, nonché norma del vettore che congiunge i due punti.

Viso	Mani
nose	left_pinky
left_eye_inner	right_pinky
left_eye	left_index
left_eye_outer	right_index
right_eye_inner	left_thumb
right_eye	right_thumb
right_eye_outer	
left_ear	
right_ear	
left_mouth	
right_mouth	

Tabella 4.1: Punti scheletrici non utili alla classificazione

Successivamente, vengono rimossi tutti i punti scheletrici utili a MediaPipe per il solo tracciamento del corpo ma non per la classificazione, come i punti del viso o delle mani. In particolare, vengono scartati 17 punti su 33, elencati nella Tabella 4.1.

La valutazione della somiglianza tra due pose utilizzerà come indice la distanza[8] tra i 16 punti rimanenti, presi a coppie.

4.2.2 Implementazione dell'algoritmo k-NN

Il Listato 4.3 mostra l'implementazione, semplificata per favorirne la leggibilità, dell'algoritmo k-NN per la classificazione delle pose; esso segue diversi step fondamentali, descritti di seguito.

1. Viene normalizzata, come descritto nella sezione precedente, la posa da classificare e viene creata una *PriorityQueue*[9] [15].

2. Il primo passaggio consiste nel selezionare k elementi del training set che, messi a confronto con la nuova posa, non presentino alcun attributo notevolmente differente; ciò aiuta le prestazioni dell'algoritmo, evitando di considerare, nel passaggio successivo, pose quasi identiche alla nuova se non per uno, o più, punti orientati in maniera completamente diversa. A tale scopo, per ogni elemento del training set importato dal file CSV (Capitolo 3) e opportunamente normalizzato, si seleziona il punto scheletrico che più si discosta dal corrispondente nella nuova posa. È compito della PriorityQueue mantenere gli elementi che presentano differenze maggiori all'infuori della coda, tramite la rimozione dell'elemento in testa; infatti, la coda stessa è ordinata per valori decrescenti della distanza massima evidenziata.

3. Il secondo passaggio si occupa di selezionare, a partire dalla coda appena popolata, i k elementi più simili alla nuova posa. Si seguono i medesimi step calcolando, però, la media delle differenze tra l'i-esimo punto e il corrispettivo della posa da classificare. Nuovamente, gli elementi che più si discostano rimangono fuori dalla coda.

[8] Ciò è possibile grazie alla normalizzazione, che rende il procedimento indipendente dalle dimensioni delle immagini utilizzate per generare il training set.

[9] Struttura dati Java simile a una coda, la quale garantisce, però, che l'elemento occupante la sua testa sia l'ultimo secondo l'ordinamento definito dall'utente.

4. Si giunge, quindi, a una PriorityQueue contenente le classi di posa che maggiormente si avvicinano al nuovo elemento. L'ultimo passaggio incrementa la confidenza di tali classi.

```
public ClassificationResult classify(Pose pose) {

    ClassificationResult result = new ClassificationResult();
    List<PointF3D> landmarks = extractPoseLandmarks(pose);
    List<PointF3D> n_landmarks = normalize(landmarks);

    PriorityQueue<Pair<PoseSample, Float>> maxDistances = new PriorityQueue
    ↪ <>(30, (o1, o2) -> -Float.compare(o1.second, o2.second));
    for (PoseSample poseSample : poseSamples) {
        List<PointF3D> n_sample = normalize(extractPoseLandmarks(poseSample));
        float originalMax = 0;
        for (int i = 0; i < n_landmarks.size(); i++) {
            originalMax = max(originalMax, maxAbs(subtract(n_landmarks.get(i),
    ↪ n_sample.get(i))));
        }
        maxDistances.add(new Pair<>(poseSample, originalMax));
        if (maxDistances.size() > maxDistanceTopK) maxDistances.poll();
    }

    PriorityQueue<Pair<PoseSample, Float>> meanDistances = new PriorityQueue
    ↪ <>(10, (o1, o2) -> -Float.compare(o1.second, o2.second));
    for (Pair<PoseSample, Float> sampleDistances : maxDistances) {
        PoseSample poseSample = sampleDistances.first;
        List<PointF3D> n_sample = normalize(extractPoseLandmarks(poseSample));
        float originalSum = 0;
        for (int i = 0; i < n_landmarks.size(); i++) {
            originalSum +=
                sumAbs(subtract(n_landmarks.get(i), n_sample.get(i)));
        }
    float meanDistance = originalSum / n_landmarks.size();
    meanDistances.add(new Pair<>(poseSample, meanDistance));
    if (meanDistances.size() > meanDistanceTopK) meanDistances.poll();
    }

    for (Pair<PoseSample, Float> sampleDistances : meanDistances) {
        String className = sampleDistances.first.getClassName();
        result.incrementClassConfidence(className);
    }
    return result;
}
```

Listato 4.3: Funzione per la classificazione delle pose

La confidenza, appena introdotta, è un valore numerico associato a ogni classe che indica quante volte quest'ultima è presente tra quelle più simili alla nuova posa da classificare; dunque, maggiore è il numero di elementi della coda finale etichettati con una certa classe, più alta sarà la confidenza della classe stessa. I valori di k sono predefiniti: 30 per il primo passaggio e 10 per il secondo.

Per ogni frame, si riesce, quindi, a ottenere la classe di posa riconosciuta, ovvero quella con il valore di confidenza maggiore; tale valore e l'etichetta della classe vengono utilizzati nella produzione del feedback a schermo.

4.3 Risultati ottenuti

4.3.1 EMA Smoothing

Si tenga in considerazione che il valore di confidenza associato a ogni classe di posa viene sottoposto a un processo di smoothing a media mobile esponenziale (EMA - Exponential Moving Average). Le tecniche di smoothing sono utilizzate nello studio delle serie temporali e hanno lo scopo di "addolcire" un insieme di dati cercando di ridurne il rumore e ricostruirne al meglio il trend; a tal scopo, si tende a smorzare le variazioni brusche tra la misura di un valore in un istante e il successivo.

Dunque, il valore di confidenza di una certa classe dipende anche dai valori dei frame precedenti, con una finestra temporale di 10 frame, ampliabile a piacere. A differenza della media mobile semplice, che assegna lo stesso peso a tutti i punti dati in una finestra temporale, l'EMA assegna pesi decrescenti ai punti dati in base al loro ordine di arrivo. Questo significa che i frame più recenti hanno un peso maggiore rispetto a quelli più vecchi. La formula di calcolo dell'EMA, infatti, è la seguente:

$$EMA_t = \alpha \cdot X_t + (1 - \alpha) \cdot EMA_{t-1}$$

- EMA_t è il valore di confidenza smussato del frame attuale, da calcolare;
- X_t è il valore di confidenza reale del frame attuale;
- EMA_{t-1} è il valore di confidenza smussato del frame precedente;
- α è il fattore di smoothing, compreso tra 0 e 1, che determina il peso relativo dei nuovi dati (un valore più alto di α dà maggiore importanza ai dati più recenti).

Il Listato 4.4 mostra l'algoritmo di EMA Smoothing [4] grazie al quale, per ogni classe di posa, si genera il relativo valore di confidenza, smussato rispetto alle 10 osservazioni precedenti.

```
public ClassificationResult getSmoothedResult(ClassificationResult
    ↪ classificationResult) {

    if (window.size() == windowSize) window.pollLast();
    window.addFirst(classificationResult);
    Set<String> allClasses = new HashSet<>();
    for (ClassificationResult result : window) {
        allClasses.addAll(result.getAllClasses());
    }

    ClassificationResult smoothedResult = new ClassificationResult();
    for (String className : allClasses) {
        float ema = window.getLast().getClassConfidence(className);
        for (ClassificationResult result : window) {
            ema = alpha * result.getClassConfidence(className) + (1-alpha) *
    ↪ ema;
        }
        smoothedResult.putClassConfidence(className, ema);
    }
    return smoothedResult;
}
```

Listato 4.4: Funzione che implementa l'EMA Smoothing

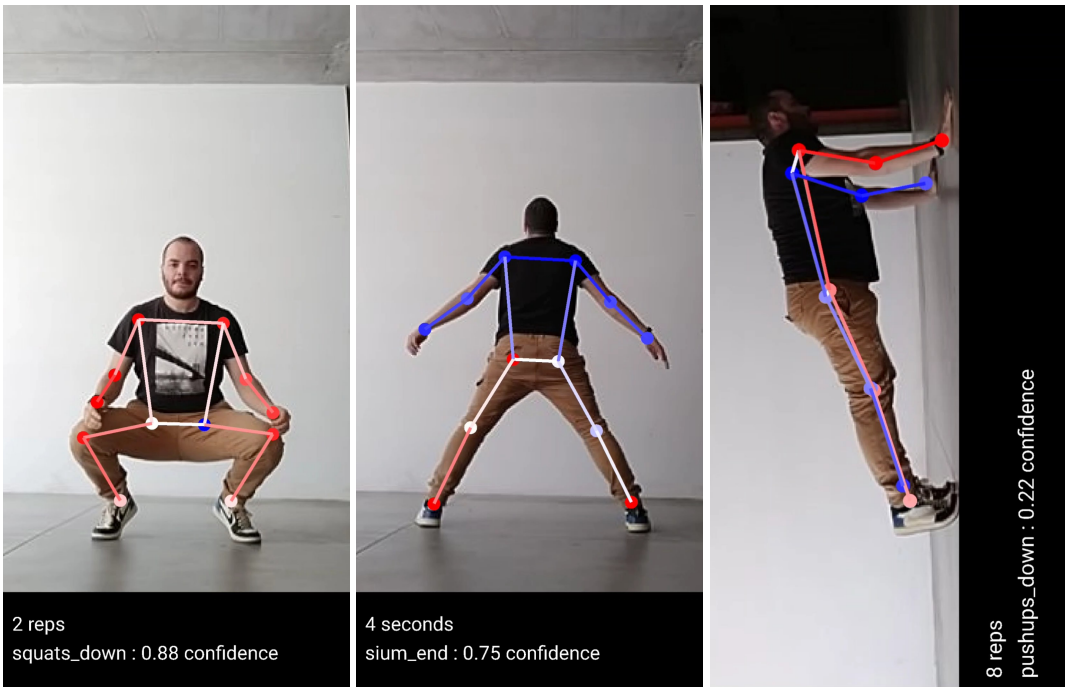

Figura 4.6: Risultati visivi della classificazione

4.3.2 Resa visiva dell'applicazione

Quanto descritto riassume i processi fondamentali che permettono all'applicazione di soddisfare i requisiti elencati nel Capitolo 1. In aggiunta, sono presenti due ulteriori meccanismi.

- *Conteggio delle ripetizioni.* L'applicazione consente di contare quante ripetizioni di una posa vengono effettuate dal soggetto inquadrato; ciò è possibile enumerando quante volte la classe di posa di cui si sta effettuando il conteggio supera un certo valore di confidenza. Tale valore può essere aumentato o diminuito a piacere, per rendere il contatore più o meno sensibile.

- *Cronometraggio delle posizioni.* La seconda possibilità è quella di misurare quanto a lungo il soggetto inquadrato mantiene una certa posa; il meccanismo è analogo al precedente; l'unica differenza sta nel fatto che vengono conteggiati i secondi durante i quali la classe di posa selezionata presenta un valore di confidenza superiore a un certo livello.

La Figura 4.6 mostra come si presenta la schermata LivePreviewActivity durante il riconoscimento della posa.

CAPITOLO 5

Sviluppi futuri

In questo capitolo, che conclude quanto presentato, si analizzano i miglioramenti da apportare all'applicazione realizzata e quali siano le principali funzionalità da aggiungere. Nella seconda parte ci si focalizzerà sui possibili sviluppi futuri del progetto, approfondendo il mercato odierno della realtà aumentata.

5.1 Nuove implementazioni e miglioramenti

Il core del prodotto realizzato si basa su tecniche e strumenti integrati in ML Kit il quale, attualmente, è ancora in una fase *beta*[1] di sviluppo e necessiterà di modifiche e miglioramenti da parte di Google stesso. Ciò implica che anche l'applicazione realizzata ha grandi margini di miglioramento prima di poter rappresentare un prodotto adeguato alla distribuzione.

5.1.1 Applicazione mobile

Di seguito sono approfondite alcuni punti deboli dell'applicazione mobile oggetto della presente tesi, essi includono difetti evidenziati durante l'utilizzo della stessa o mancanze già note nel corso della progettazione.

Mancanza di una versione iOS

Come anticipato, per le tempistiche legate allo sviluppo del progetto, è stato scelto Android come sistema operativo mobile rispetto a iOS. Tuttavia, è stata più volte ribadita l'importanza di disporre di un prodotto compatibile con entrambe le piattaforme per non precludersi una buona quantità di dispositivi mobili in circolazione.

Fortunatamente, ML Kit fornisce anche per iOS due SDK ottimizzati per il rilevamento della posa ed è completamente integrato in Swift e Objective-C. Un primo miglioramento del progetto in esame, dunque, è sicuramente quello di estendere il supporto ai dispositivi Apple.

[1] Quando un prodotto è in fase "beta" di sviluppo, potrebbe non essere completamente stabile e contenere funzionalità incomplete. Durante questa fase, l'obiettivo principale è quello di ottenere feedback dagli utenti per perfezionare il prodotto prima del lancio ufficiale.

Performance grafiche

Durante l'utilizzo dell'app è facile accorgersi del fatto che la dimensione delle immagini di input, dunque della preview della fotocamera visualizzata sullo schermo, è molto limitata. Infatti, per garantire una fluidità accettabile nella schermata di riconoscimento delle pose, si deve tarare la dimensione dell'anteprima di ciò che si sta classificando.

Guardando una sequenza di frame (ovvero, un video), l'occhio umano percepisce la sensazione di movimento a partire da 24 fps [20]. Se da una parte, vista la buona disponibilità di processori grafici sul mercato, quest'ultima può sembrare una quota facilmente raggiungibile, dall'altra si deve tenere in considerazione che i dispositivi sui quali viene eseguita l'applicazione non dispongono di una potenza di calcolo sufficiente a ottenere tali risultati. A riprova di ciò, il dispositivo[2] su cui è stata testata l'applicazione ha registrato un picco massimo di 18 fps con una dimensione delle immagini di input impostata a 480x360 pixel, dunque, ancora distante da performance accettabili per una buona *UX*[3].

In aggiunta, la qualità dei risultati della classificazione dipende dalla qualità dell'immagine di input; in particolare, affinché ML Kit rilevi con precisione la posa, il soggetto inquadrato deve essere di almeno 256x256 pixel, dunque è bene assicurarsi che il soggetto occupi la maggior parte possibile dell'immagine.

Il secondo orizzonte di crescita del prodotto è, quindi, quello di ottimizzare le funzioni di classificazione e produzione di feedback a schermo, in modo da garantire un'esperienza d'uso piacevole.

5.1.2 Valutazione delle prestazioni

In Data Mining, è spesso necessario misurare le prestazioni del classificatore scelto per etichettare il dataset di test perché tale misura fornisce una stima della sua accuratezza; quest'ultima può anche essere utilizzata per confrontare le performance di diversi classificatori, nello stesso dominio, e per valutare se la scelta effettuata è corretta oppure è necessario operare un cambio di rotta. Tuttavia, per fare ciò, le etichette di classe dei record di test devono essere note.

Holdout e Cross-Validation

I due metodi di valutazione delle performance più celebri sono, sicuramente, l'*Holdout* e la *Cross-Validation* [17].

Nel metodo Holdout, il dataset etichettato viene ulteriormente suddiviso in due set disgiunti, il training set e il *validation set*. Un modello di classificazione viene, quindi, indotto dal training set e le sue prestazioni vengono valutate sul validation set. La proporzione di dati riservati per l'addestramento e la validazione è tipicamente a discrezione degli analisti. L'accuratezza del classificatore può essere calcolata valutando quanti dati sono stati correttamente etichettati (Capitolo 3); ciò è possibile grazie al fatto che i due set contengono elementi già classificati. Il metodo Holdout ha diverse limitazioni ben note. In primo luogo, sono disponibili meno esempi etichettati per l'addestramento perché alcuni dei record sono trattenuti per la validazione; di conseguenza, il modello indotto potrebbe non essere così buono come quando tutti gli esempi del dataset di partenza sono utilizzati per l'addestramento. In secondo luogo, il modello potrebbe dipendere fortemente dalla

[2] Trattasi di un Android di fascia bassa, precisamente uno Xiaomi Redmi Note 8T (*willow*).

[3] "UX" è l'abbreviazione di "User Experience". Si riferisce alla percezione complessiva che un utente ha durante l'interazione con un prodotto, servizio, o sistema, specialmente in termini di facilità d'uso, efficienza e soddisfazione.

composizione dei due dataset: minore è la dimensione del training set e peggiori saranno le prestazioni del modello; d'altra parte, se il validation set è troppo piccolo, la precisione stimata è meno affidabile.

Una tecnica alternativa è la Cross-Validation, in cui ogni record è utilizzato lo stesso numero di volte per l'addestramento ed esattamente una volta per la validazione. L'approccio generale (*k-fold cross-validation*) consiste nel suddividere i dati in *k* partizioni di dimensioni uguali. Durante ogni esecuzione, una delle partizioni è scelta per la validazione, mentre le altre sono utilizzate per l'addestramento. Questa procedura viene ripetuta *k* volte in modo che ogni partizione sia utilizzata per la validazione esattamente una volta e l'accuratezza è calcolata considerando tutte le *k* esecuzioni. A differenza di quanto accade nell'Holdout, qui i validation set sono mutualmente esclusivi e coprono efficacemente l'intero set di dati. Lo svantaggio di questo approccio è che è computazionalmente dispendioso.

Valutazione del modello di classificazione delle pose

Per valutare le performance del modello di classificazione realizzato per il progetto in questione, sarebbe stato necessario disporre di un'enorme quantità di immagini raffiguranti pose corporee già etichettate, in modo da poter applicare una delle metodologie presentate poc'anzi. Data la difficoltà di reperire questo tipo di materiale multimediale entro le tempistiche di sviluppo, non è stato possibile studiare le prestazioni del classificatore k-NN, e ciò rappresenta un'ulteriore direzione di miglioramento del progetto. Tuttavia, il riconoscimento della posa corporea, come ormai chiaro, include due step differenti: il tracciamento del corpo umano attraverso i punti scheletrici e la successiva classificazione delle pose. Il team di sviluppo di MediaPipe si è occupato di fornire, congiuntamente alle API utilizzate per il tracciamento, le prestazioni del modello che permette tale task.

Il modello prende il nome di *"MediaPipe BlazePose GHUM 3D"* [3], risale al 2021 ed è stato oggetto di tre studi differenti. A partire da un dataset contenente 1400 immagini di pose corporee di soggetti provenienti da 14 regioni del mondo differenti (dunque, 100 esempi per regione) è stata valutata la bontà del modello rispetto:

- alla regione geografica di provenienza del soggetto;
- al genere del soggetto;
- al colore della pelle del soggetto, utilizzando la scala Fitzpatrick[4].

Le Tabelle 5.1, 5.2 e 5.3 riportano l'accuratezza del modello, misurata attraverso l'indice *PDJ*[5] corredata con la deviazione standard[6] (σ) relativa.

5.2 Opportunità di sviluppo

Quanto detto finora evidenzia le mancanze e le possibilità di miglioramento dell'applicazione mobile e del processo di classificazione; ciò non toglie che siano numerosi gli orizzonti di sviluppo del progetto, soprattutto in un mercato in costante evoluzione e sempre più attento a quella che è la realtà aumentata.

[4]La scala Fitzpatrick è uno schema di classificazione numerica per il colore della pelle umana dal tono più chiaro (1) al più scuro (6).

[5]"Average percentage of detected joints", ovvero in che percentuale i punti scheletrici vengono correttamente tracciati, su un totale di 33 punti.

[6]La deviazione standard è una misura di dispersione o variabilità in un insieme di dati statistici. Rappresenta quanto i valori di un insieme di dati si discostano dalla loro media aritmetica; dunque, fornisce un'indicazione della dispersione dei dati attorno alla media.

Regione	**PDJ**	σ
Australia and New Zealand	94.1	8.3
Caribbean	94.8	8.3
Europe	90.3	12.6
Nohern Africa	94.7	8.2
South America	95.0	8.3
Southeastern Asia	94.5	8.1
Western Asia	94.9	7.8
Central America	95.4	6.5
Central Asia	94.3	8.7
Eastern Asia	91.3	12.5
Middle Africa	94.6	9.5
Nohern America	93.4	8.6
Southern Africa	92.9	10.0
Southern Asia	93.4	9.0
Media	93.8	-

Tabella 5.1: PDJ rispetto alla provenienza geografica

Genere	**% del dataset**	**PDJ**	σ
Maschio	45.9	94.7	9.0
Femmina	54.1	93.1	9.5
Media	-	93.9	-

Tabella 5.2: PDJ rispetto al genere

Tono della pelle	**% del dataset**	**PDJ**	σ
1 - *palest*	1.3	96.3	2.5
2 - *light colored*	9.5	91.4	10.1
3 - *golden honey*	34.3	93.9	9.3
4 - *moderate brown*	36.2	94.3	9.0
5 - *dark brown*	14.2	94.5	9.3
6 - *darkest brown*	4.5	96.1	7.1
Media	-	94.2	-

Tabella 5.3: PDJ rispetto al colore della pelle

Figura 5.1: Formazione e addestramento medico in un ambiente AR

5.2.1 Telemedicina

Come approfondito nella Sezione 1.2, nel campo del tracciamento del corpo, sono numerosi i prodotti esistenti orientati alla telemedicina[7]. La possibilità di integrare il sistema implementato in un'applicazione mobile apre diverse vie in questo campo. Infatti, la sinergia tra AR e telemedicina offre opportunità significative per migliorare la fornitura di servizi medici.

Consultazioni mediche virtuali

La telemedicina utilizza la connettività digitale per consentire consultazioni mediche a distanza. L'integrazione della realtà aumentata permette ai medici di visualizzare dati clinici in tempo reale, sovrapponendoli alla vista della telecamera del paziente. Questo può includere la proiezione di dati vitali, risultati di test o immagini diagnostiche direttamente nel campo visivo del professionista medico.

Formazione e addestramento medico

La realtà aumentata può essere utilizzata durante la formazione e l'addestramento degli studenti, i quali possono avere accesso a simulazioni virtuali e dettagliate di procedure chirurgiche (Figura 5.1), anatomia o diagnosi, migliorando la loro preparazione pratica in un ambiente privo di rischi per i pazienti.

Chirurgia assistita

Nel campo della chirurgia, la realtà aumentata può essere utilizzata per fornire assistenza durante le procedure. I chirurghi possono ricevere informazioni cruciali direttamente sul campo operatorio attraverso visori AR, migliorando la precisione e la sicurezza delle operazioni.

Terapie fisiche e riabilitazione

Nell'ambito della riabilitazione, la realtà aumentata può rendere le sessioni più coinvolgenti e personalizzate. I pazienti possono svolgere esercizi virtuali che vengono

[7]La telemedicina è una forma di assistenza medica che utilizza le tecnologie dell'informazione e della comunicazione per fornire servizi sanitari a distanza, invece di richiedere la presenza fisica del paziente e del professionista sanitario nello stesso luogo. I servizi di telemedicina possono includere la consulenza virtuale, la trasmissione di dati medici (come risultati di test o immagini diagnostiche), la gestione remota di malattie croniche, la prescrizione e la consegna di farmaci online, e molte altre applicazioni, come quella sviluppata.

sovrapposti al loro ambiente reale, facilitando la partecipazione e il monitoraggio delle prestazioni da parte dei professionisti sanitari. L'applicazione realizzata ha le potenzialità per soddisfare appieno questo tipo di task.

Monitoraggio remoto

Nel monitoraggio remoto delle condizioni mediche, la realtà aumentata può contribuire a migliorare la comprensione del paziente riguardo alla propria situazione. Ad esempio, un'applicazione AR potrebbe fornire informazioni in tempo reale sulle dosi di farmaci o i sintomi da monitorare direttamente sullo schermo del dispositivo mobile di chi sta affrontando la cura.

5.2.2 Videogiochi e *serious game*

Da due decenni a questa parte, le tecnologie della Computer Graphics[8] sono state orientate allo sviluppo della più grande fetta di mercato disponibile: quella dell'intrattenimento [21]; questa include la cinematografia e, soprattutto, i videogiochi. Il settore dei videogiochi, in Italia, dove si contano 14,2 milioni di appassionati, con un'età media di 29,8 anni [19], è sempre in maggiore crescita. Oltre alla parte di intrattenimento, il videogioco sta entrando negli ambiti culturali ed educativi, come i musei o, perfino, la scuola.

I giochi che supportano la realtà aumentata sono sempre più diffusi e le funzionalità integrate nell'applicazione realizzata trovano corrispondenze in alcuni prodotti già commercializzati.

Esplorazione del mondo reale

Alcuni videogiochi AR consentono ai giocatori di esplorare il mondo reale mentre interagiscono con elementi virtuali sovrapposti al loro ambiente attraverso lo schermo del dispositivo mobile. Questo può includere la caccia a creature fantastiche (si faccia riferimento a Pokémon Go, già citato), la risoluzione di enigmi o l'accesso a missioni basate sulla posizione del giocatore.

Multiplayer

La realtà aumentata ha aperto la strada a esperienze di gioco multiplayer e sociali che coinvolgono giocatori fisicamente distanti. Gli amici possono collaborare o competere in sfide virtuali sovrapposte al mondo reale, rendendo l'esperienza di gioco più sociale e connessa. Ad esempio, la realtà aumentata è stata utilizzata per creare giochi di caccia al tesoro in cui i giocatori seguono indizi nel mondo reale, risolvono enigmi e cercano oggetti virtuali attraverso la fotocamera dei propri dispositivi.

Fitness e benessere

Gli sviluppatori stanno sfruttando la realtà aumentata per creare giochi che promuovono l'attività fisica e il benessere. Ad esempio, alcuni giochi incoraggiano gli utilizzatori a muoversi nel mondo reale per raggiungere obiettivi o partecipare a sfide di fitness. Tale soluzione rappresenterebbe la normale evoluzione di un prodotto come quello realizzato nel progetto in esame.

[8] La Computer Graphics (CG) è un campo multidisciplinare che coinvolge la creazione, la manipolazione e la rappresentazione visuale di dati attraverso l'uso di algoritmi, software e hardware specializzati. Questo campo si occupa della produzione di immagini digitali e animazioni che possono essere visualizzate su schermi, proiettate o stampate.

Figura 5.2: Formazione dei lavoratori basata su AR

Istruzione e apprendimento

Alcuni giochi AR mirano a educare mentre intrattengono, offrendo esperienze di apprendimento basate sulla realtà aumentata. È il caso dei *serious game* ("giochi seri"), ovvero videogiochi progettati con l'obiettivo principale di insegnare, formare o trasmettere informazioni. In tal senso, l'integrazione della realtà aumentata può arricchire ulteriormente l'esperienza educativa, rendendola più coinvolgente e interattiva, tramite:

- simulazioni pratiche più realistiche all'interno dei serious game medici, già introdotti indirettamente nella sezione precedente;
- l'interazione con oggetti del mondo reale attraverso il dispositivo, come succede nei serious game di Ingegneria, durante task di prototipazione[9] di prodotti ancora in stato embrionale;
- l'interazione a distanza di sicurezza con macchinari industriali, nella formazione dei lavoratori sulle procedure (Figura 5.2).

È, dunque, chiaro che gli ambiti di sviluppo sono numerosi, e ciò gioca a favore di una possibile futura evoluzione del progetto descritto.

[9]La prototipazione è il processo di creazione di un modello iniziale o un prototipo di un prodotto o sistema, al fine di testarne le funzionalità, raccogliere feedback e apportare eventuali miglioramenti prima della realizzazione definitiva. Questo approccio consente di identificare e risolvere problemi potenziali in una fase precoce dello sviluppo, riducendo così i rischi e i costi associati alla produzione completa.

Conclusioni

Questo documento di tesi ha riassunto, nei cinque capitoli presentati, il processo di realizzazione di un sistema per il tracciamento della posa corporea, compatibile con dispositivi mobili.

Nella prima parte, ovvero nei Capitoli 1 e 2, è stato descritto il task da portare a termine, delineando quanto ci fosse di già implementato nel mercato delle applicazioni mobile e quali tecnologie potevano aiutare lo sviluppo del prodotto in questione.

La parte centrale, che include i Capitoli 3 e 4, ha presentato il meccanismo di classificazione delle pose corporee, dalla costruzione di un training set adeguato all'addestramento di un modello, all'implementazione di un algoritmo di Machine Learning per il riconoscimento effettivo.

Il Capitolo 5, a chiusura del documento, ha delineato quali siano gli orizzonti di sviluppo dell'applicazione realizzata, i quali, in particolar modo, vertono intorno alla telemedicina e ai sistemi di realtà aumentata rivolti all'intrattenimento e all'istruzione.

È opportuno evidenziare che quanto realizzato si trova in una fase primordiale e sarà, in futuro, oggetto di rinnovamento in vista di nuove versioni più estese, affidabili, efficienti o, semplicemente, migliori.

Questa conclusione, benché trovata da povera gente, c'è parsa così giusta, che abbiam pensato di metterla qui, come il sugo di tutta la storia. La quale, se non v'è dispiaciuta affatto, vogliatene bene a chi l'ha scritta, e anche un pochino a chi l'ha raccomodata. Ma se in vece fossimo riusciti ad annoiarvi, credete che non s'è fatto apposta.

Alessandro Manzoni
I promessi sposi

Bibliografia

[1] Al-Ansi, A. M., Jaboob, M., Garad, A. e Al-Ansi, A. (2023), «Analyzing augmented reality (AR) and virtual reality (VR) recent development in education», *Social Sciences & Humanities Open*, vol. 8 (1). (Citato a pagina 4)

[2] Bazarevsky, V. e Grishchenko, I. (2020), «On-device, Real-time Body Pose Tracking with MediaPipe BlazePose», *Google Research*. (Citato a pagina 15, 19 e 25)

[3] Bazarevsky, V., Grishchenko, I. e Bazavan, E. G. (2021), «Model Card BlazePose GHUM 3D», *Google Research*. (Citato a pagina 40)

[4] Burgstahler, L. e Neubauer, M. (2002), «New Modifications of the Exponential Moving Average Algorithm for Bandwidth Estimation», *Institute of Communication Networks and Computer Engineering, University of Stuttgart*. (Citato a pagina 36)

[5] Chowdhary, K. R. (2020), *Natural Language Processing*, p. 603–649, Springer. (Citato a pagina 13)

[6] Giordano Angelini (2023), «RehabBodyTracking», URL `https://github.com/giordanoangelini/RehabBodyTracking`. (Citato a pagina 24 e 28)

[7] Gonzalez, R. C. e Woods, R. E. (2017), *Digital image processing*, Pearson, 4th ed. (Citato a pagina 10)

[8] Google (2023), «Android - Activity», URL `https://developer.android.com/reference/android/app/Activity`. (Citato a pagina 28)

[9] Google (2023), «Android - Camera», URL `https://developer.android.com/reference/android/hardware/Camera`. (Citato a pagina 30)

[10] Google (2023), «Android - ConstraintLayout», URL `https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout`. (Citato a pagina 29)

[11] Google (2023), «Android - SharedPreferences», URL `https://developer.android.com/reference/android/content/SharedPreferences`. (Citato a pagina 30)

[12] Google (2023), «Android - View», URL `https://developer.android.com/reference/android/view/View`. (Citato a pagina 32)

[13] Google (2023), «Google for Developers - ML Kit», URL https://developers.google.com/ml-kit. (Citato a pagina 9)

[14] Java (2023), «Java Platform SE 8 - ByteBuffer», URL https://docs.oracle.com/javase/8/docs/api/java/nio/ByteBuffer.html. (Citato a pagina 30)

[15] Java (2023), «Java Platform SE 8 - PriorityQueue», URL https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html. (Citato a pagina 34)

[16] Kumar, V., Tan, P.-N. e Steinbach, M. (2005), *Introduction to Data Mining*, cap. Introduction, Pearson. (Citato a pagina 19 e 21)

[17] Kumar, V., Tan, P.-N. e Steinbach, M. (2005), *Introduction to Data Mining*, cap. Classification: Basic Concepts, Decision Trees, and Model Evaluation, Pearson. (Citato a pagina 22 e 39)

[18] Kumar, V., Tan, P.-N. e Steinbach, M. (2005), *Introduction to Data Mining*, cap. Classification: Alternative Techniques, Pearson. (Citato a pagina 23)

[19] Palesse, D. (2023), «Tutti pazzi per i videogiochi, un giro d'affari da 2,2 miliardi di euro», *ANSA*. (Citato a pagina 43)

[20] Wixted, J., Gallistel, R., Medin, D., Yantis, S., Pashler, H. e Stevens, S. S. (2002), *The Stevens' Handbook of Experimental Psychology. Sensation and Perception*, cap. Motion perception, Wiley. (Citato a pagina 39)

[21] Zingaretti, P. (2004), *Fondamenti di Computer Graphics*, Pitagora. (Citato a pagina 43)

Siti web consultati

- Exer Health - https://www.exer.ai/
- Kaia Health - https://kaiahealth.com/
- Mirror AR - https://www.mirrorar.io/
- Unity Docs - https://docs.unity3d.com/
- Exer Health - https://www.exer.ai/
- Apple ARKit - https://developer.apple.com/documentation/arkit/
- Google ML Kit - https://developers.google.com/ml-kit/

Ringraziamenti

Un **GRAZIE**, in ordine alfabetico, a tutte le persone che ho incontrato lungo una strada durata cinque anni, con la certezza che in futuro, rileggendo la mia storia, mi ricorderò di voi.

Adelmo	Albano	Alessandra	Alessandro	Alessio	Alex
Alice	Amedeo	Amelie	Andrea	Angelo	Anna
Annalaura	Antonio	Asia	Assunta	Attilio	Azzurra
Barbara	Beatrice	Carlo	Carol	Cecilia	Celeste
Chiara	Christian	Cristian	Cristiana	Cristiano	Cristina
Daniel	Daniela	Daniele	Danilo	Davide	Denis
Diego	Diletta	Domenico	Edoardo	Elda	Elena
Eleonora	Elisa	Emanuela	Emanuele	Enrico	Erica
Eugenio	Fabio	Fabrizio	Facundo	Federica	Federico
Filippo	Fioralba	Fiorenzo	Flavia	Francesca	Francesco
Franco	Frank	Gabriele	Giacomo	Giada	Gianmarco
Gioia	Giorgia	Giorgio	Giovanni	Giulia	Giulio
Giuseppe	Greta	Guendalina	Guido	Iacopo	Ilaria
Ilenia	Irene	Itala	Ivo	Jacopo	Jessica
Kim	Laura	Leonardo	Lorenzo	Luca	Lucia
Ludovica	Luigina	Luna	Manfredo	Manuel	Marco
Maria	Maria Vittoria	Marino	Mario	Marta	Martina
Martino	Marvin	Massimiliano	Massimo	Matt	Matteo
Matthias	Mattia	Maurizio	Mauro	Melissa	Monica
Morena	Morgan	Nadia	Nerio	Niccolò	Nicholas
Nicole	Nicolò	Niko	Noemi	Omar	Paolo
Pedro	Peppino	Pietro	Pino	Randy	Rebecca
Riccardo	Rita	Roberta	Roberto	Rocco	Rolando
Sabrina	Salvatore	Samuele	Sandra	Sandy	Sara
Sasha	Saverio	Sergio	Silvia	Simona	Simone
Siria	Sofia	Sonia	Sophie	Stefania	Stefano
Stephanie	Susanna	Tahirih	Tatiana	Teresa	Tiziano
Tomassina	Tommaso	Umberto	Valentina	Valeria	Valerio
Veronica	Viola	Vittorio	Walter		