Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Valutazione Comparativa di Algoritmi di Machine Learning per l'Analisi Temporale e Multimodale per la Classificazione di Processi Riabilitativi in Telemedicina

Relatore:
Dott. Stefano Pio Zingaro

Presentata da:
Gabriele Evangelista

Sessione 2
Anno Accademico 2022/2023

01001001 01100110 01010111 01101001 01110011 01101000
01100101 01110011 01010111 01100101 01110010 01100101
01001000 01101111 01110010 01110011 01100101 01110011

Sommario

La telemedicina e la riabilitazione sono processi che stanno diventando e diventeranno sempre più connessi fra loro. Tantissimi episodi di terapia vengono svolti dal paziente in casa e, dunque, vi è un impellente bisogno di creare nuovi metodi per aiutare chi è in cura a raggiungere i migliori risultati possibili in un ambiente domestico senza l'utilizzo di sensori. Grazie ai dati forniti dall'Istituto Ortopedico Rizzoli ho creato due dataset, uno basato su video di pazienti che eseguono 5 diversi esercizi fisici con i relativi score e l'altro formato dai dati demografici di questi soggetti. L'obiettivo della tesi è classificare le ripetizioni automaticamente con 5 diversi punteggi sviluppando due tipi di reti neurali: quelle specializzate su dati di serie temporali che adoperano l'architettura InceptionTime; quelle tabulari che, sfruttando i dati demografici forniti dai medici, sono state utilizzate anche come reti multimodali. A questo punto lo studio si domanda se e quanto i dati demografici possano aiutare nel miglioramento della classificazione, confrontando i risultati anche con un modello di machine learning tradizionale: il Random Forest. Le reti neurali hanno ottenuto performance altissime, in particolar modo quelle allenate sulle serie temporali, risultando estremamente utili per un utilizzo concreto. Viene dimostrato inoltre come, i dati demografici forniti, non siano stati utili ai fini della classificazione.

Indice

1 Introduzione **1**

2 Revisione della letteratura **7**

2.1 Riabilitazione attraverso il Deep Learning 7

2.2 Time Series Classification 11

2.3 Multimodal Learning 13

3 Materiali e metodo **17**

3.1 Progetto di Intelligenza Artificiale 17

3.2 Reti Neurali Time Series 21

3.3 Reti Neurali Multimodali e Tabulari 30

3.4 Modifica al Progetto di IA 35

4 Risultati sperimentali **39**

5 Conclusioni **47**

A Tabelle non aggregate **51**

Bibliografia **51**

Elenco delle figure

1.1 Tecnica del leave-one-out cross-validation 6

3.1 Panel fornito dai medici 18
3.2 Uso di OpenCV e Mediapipe per fare il tracking delle articolazioni 19
3.3 Risultati sperimentali 21
3.4 Tracking di tutte e 9 le articolazioni 22
3.5 Architettura di InceptionTime 23
3.6 Dati spaziali di un video 26
3.7 Miglioramento di un paziente tra le due sessioni 26
3.8 Aumento del numero di frame da 73 a 156 27
3.9 Diminuzione del numero di frame da 377 a 156 27
3.10 Modifica del numero di frame sul numero medio, massimo e minimo di frame di tutti i video dello stesso esercizio 28

Elenco delle tabelle

3.1 Esempio dei risultati della funzione `get_ts_features` 32
3.2 Esempio di concatenzione dei dataframe demografici 33

4.1 Esercizi e numeri di frame 40
4.2 Performance delle reti time series 41
4.3 Tempo di allenamento e validazione delle reti time series . . . 41
4.4 Performance delle reti multimodali 42
4.5 Tempo di allenamento e validazione delle reti multimodali . . 43
4.6 Performance delle reti tabulari senza dati demografici 43
4.7 Tempo di allenamento e validazione delle reti tabulari senza dati demografici 44
4.8 Performance dei Random Forest 44
4.9 Tempo di allenamento e validazione dei Random Forest 45
4.10 Performance dei Random Forest con dati demografici 45
4.11 Tempo di allenamento e validazione dei Random Forest con dati demografici 45
4.12 Tabella riassuntiva performance 46

A.1 Accuracy non aggregate delle reti multimodali, parte 1 51
A.2 Accuracy non aggregate delle reti multimodali, parte 2 52
A.3 Tempi non aggregati delle reti multimodali, parte 1 52
A.4 Tempi non aggregati delle reti multimodali, parte 2 53
A.5 Accuracy non aggregate delle reti tabulari senza dati demografici 54
A.6 Tempi non aggregati delle reti tabulari senza dati demografici 55

Capitolo 1

Introduzione

La riabilitazione consente a una persona di qualsiasi età di partecipare ad attività quotidiane che sono importanti e significative, come l'istruzione, il lavoro, le attività ricreative e la cura della famiglia, migliorando la sua indipendenza e le sue capacità. Può essere vista come un insieme di interventi progettati per ottimizzare la mobilità e ridurre le disabilità in individui con problemi fisici. I cambiamenti nelle condizioni di salute e nelle caratteristiche della popolazione, come l'allungamento dell'aspettativa di vita a scapito dell'aumento delle malattie croniche e delle disabilità, predicono un grande aumento della domanda di servizi di riabilitazione in tutto il mondo [6]. Chiunque può avere bisogno di un processo di riabilitazione a un certo punto della propria vita, in seguito magari di un intervento chirurgico, di una malattia o di un infortunio, ma anche a causa di un declino delle proprie funzionalità dovuto all'avanzare dell'età. Attraverso queste cure, per esempio, persone affette da Parkinson (come mostra lo studio [1]) o sopravvissute a un infarto possono migliorare la forza dei propri muscoli, perfezionando la precisione dei movimenti volontari facendo allenamenti fisici mirati assegnati da dottori e fisioterapisti. Il programma degli esercizi viene svolto nella maggior parte del tempo a casa del paziente [15], con visite periodiche in strutture ortopediche per poter valutare e misurare i progressi e dunque l'andamento della cura, potendo così modificare eventualmente il piano di allenamento.

Vi è però una grossa mancanza di metodi affidabili per valutare la qualità delle prestazioni degli esercizi a domicilio. Ciò non permette ai fisioterapisti di avere una buona visione dei progressi di un soggetto tra una visita e l'altra in clinica e non consente loro di fornire feedback correttivi o aiuti.
Al giorno d'oggi esistono numerose tecniche di telemedicina che permettono di risolvere molti dei problemi appena indicati. Un esempio molto moderno è quello riportato in [23], in cui, in piena emergenza Covid-19, medici di tutta Italia hanno dovuto rispondere in fretta e adottare tecniche di telemedicina per poter seguire i pazienti in una situazione difficile. Ciò che risalta dallo studio è che la tecnologia è pronta e che lo scoglio più grande da superare sono le abitudini dei medici di medicina fisica e riabilitativa e dei pazienti.
Rimane tuttavia una difficoltà da dover superare a causa di un incessante bisogno di processi di riabilitazione in tutto il mondo. La necessità di cure e programmi (soprattutto nei paesi a basso e medio reddito [13]) non fa che crescere e continua a rimanere largamente insoddisfatto. Vi è un urgente bisogno di medici e fisioterapisti che possano seguire i pazienti per poter misurare la qualità del recupero.
Per poter sopperire a questa mancanza di medici, io e altre due colleghe di corso, abbiamo sviluppato un progetto di Intelligenza Artificiale il cui obiettivo era quello di predire automaticamente vari punteggi riguardo l'esecuzione di esercizi di riabilitazione di pazienti. I dati necessari sono stati estratti a partire da dei panel compilati da dottori dell'Istituto Ortopedico Rizzoli di Bologna. I *modelli* sviluppati, ovvero le funzioni matematiche il cui scopo è quello di scoprire i pattern in un set di dati di addestramento e che, quindi, permettono di ottenere gli output desiderati a partire da un certo input, sono tutti algoritmi di *machine learning*, un sottoinsieme dell'intelligenza artificiale che si dedica a sviluppare sistemi che imparano o aumentano le loro prestazioni in base ai dati che elaborano. Le performance di questi modelli, tuttavia, non sono stati eccessivamente incoraggianti.
In letteratura vi sono molti studi che sfruttano il *deep learning*, un sottoinsieme del machine learning basato sulle reti neurali [18]. Queste sono un

modello di calcolo ispirato vagamente al cervello umano basato su neuroni che calcolano una semplice combinazione lineare per poi applicare al risultato una *funzione di attivazione* che decide se il neurone viene "acceso" o se rimane "spento". Quando molti di questi elementari neuroni vengono collegati tra loro, specie in più livelli formando strutture sempre più complesse, formano le reti neurali "profonde", da cui il nome dell'apprendimento. Il motivo principale per cui l'utilizzo delle neural networks è in continua crescita esponenziale è che sono in grado di trovare dai dati *feature* (proprietà) "nascoste", o che non sono conosciute, in maniera automatica e incrementale. Non vi è dunque bisogno che un esperto del dominio in questione sia presente per pre-elaborare i dati e selezionare le feature utili. Revisionando la letteratura è possibile trovare molte ricerche in cui gli autori hanno sviluppato reti neurali più o meno complesse per poter predire e classificare automaticamente il miglioramento dei pazienti grazie alla riabilitazione.
Molti di questi studi hanno a che fare con serie di dati temporali, o *time series data*. Questi sono una serie di dati formati da punti registrati in diversi istanti di tempo equidistanti tra loro. I time series data, che hanno un ordine naturale basato sul tempo, sono coinvolti in quasi tutti i task che richiedono un qualche tipo di ragionamento umano. Ogni problema che implichi la classificazione di dati con un certo senso di ordine può essere visto come un problema di *time series classification*. Esiste una moltitudine di ricerche scientifiche che si concentra sul processing e la classificazione di questo tipo di dati e, ovviamente, i modelli più performanti sono quelli che sfruttano le reti neurali. Un esempio tipico di task è la classificazione di attività di una persona basata su dati estratti da sensori come accelerometri [2]. La prelevazione di informazioni da sensori prende il nome di metodo *sensor-based*, che si contrappone al metodo *vision-based*. Quest'ultimo, infatti, utilizza algoritmi di computer vision, invece che hardware specifico, per riconoscere i punti d'interesse, come le articolazioni di una persona. Nonostante il continuo miglioramento dei sensori e delle loro perfomance, i loro costi possono essere proibitivi in molti studi, come quello di questa tesi, o in molte zone

del mondo non ancora sviluppate.

Lo stato dell'arte per il processing di dataset time series è attualmente un modello *ensemble* chiamato *HIVE-COTE* [21], ovvero formato da più modelli che vengono usati per predire un risultato. Tutti gli studi presi in considerazione durante la revisione della letteratura rivolti alla riabilitazione, però, hanno dei problemi comuni, tra cui esaminare pochi pazienti, prendere in input un numero basso di feature e, soprattutto, non considerare diverse modalità.

Il modo in cui qualcosa si verifica, viene percepito o viene registrato è chiamato *modalità*. Testi, suoni, immagini e video sono esempi di tipi di media che possono essere chiamati modalità [8] e, quando impariamo a rappresentare le informazioni da più di una modalità, il task relativo è multimodale. È facilmente comprensibile, dunque, che i dati multimodali forniscono più informazioni di quelli unimodali perché rappresentano un oggetto da diverse prospettive, spesso aggiungendo o migliorando le informazioni. Inoltre, utilizzando dati provenienti da diverse modalità, l'apprendimento multimodale può affrontare alcuni problemi presenti nell'elaborazione dei dati unimodali, come il rumore, l'ambiguità e l'incompletezza. *Speech recognition*, ovvero cercare di convertire le parole di un audio in testo usando sia feature acustiche che linguistiche; *emotion recognition*, ossia rilevare le emozioni di una persona a partire dalla sua voce e dalle sue espressioni facciali; *image captioning*, cioè creare una descrizione in linguaggio naturale di un'immagine a partire dal suo contenuto e *video summarization*, che prevede la creazione di un breve riassunto a partire da un video sfruttando le informazioni dei suoi frame e dell'audio sono tutti esempi di multimodal learning. Ovviamente, non mancano ricerche scientifiche di multimodal classification nel campo della riabilitazione. Molti di questi si concentrano su pazienti sopravvissuti a ictus, ma ne esistono altrettanti in diversi ambiti, come il riconoscimento automatico degli stati affettivi di un paziente per poter migliorare l'aderenza a terapie, la rilevazione delle fasi del sonno per monitorarlo o la classificazione di dati multimodali estratti da un elettroencefalogramma. Nessuno degli

studi revisionati, però, ha preso in considerazione tutte e tre le seguenti caratteristiche: avere un bacino di pazienti abbastanza numeroso, sfruttare un alto numero di feature e dare in input i dati a una rete neurale.
Questo lavoro di tesi nasce proprio per raggiungere tale scopo, imponendosi due intenti ulteriori. Il primo è quello di creare una rete neurale che possa analizzare in automatico i dati time series estratti dai video delle esecuzioni degli esercizi dei pazienti e attribuire loro un voto, in modo da fornire un feedback istantaneo che possa aiutare gli individui e i dottori a capire come migliorarsi tra una visita in clinica e la successiva. Il secondo è quella di sviluppare una rete neurale multimodale che sfrutti sia i dati estratti dai video dei soggetti e sia alcuni dei loro dati demografici (forniti sempre dai dottori dell'Istituto Ortopedico Rizzoli), per poi comparare i risultati delle due reti e decretarne un "vincitore". Dunque, questo studio aspira inoltre a comprendere se alcuni dati pregressi dei pazienti possono essere discriminanti e utili per la corretta classificazione dei video.
La rete neurale multimodale è stata ideata in modo da prendere in input dei dati tabulari che includono i dati demografici e informazioni estratte a partire dai dati time series attraverso tecniche di aggregazione (come la media o la deviazione standard), ottenendo alla fine un totale di circa 170 feature.
Per entrambe le reti neurali, i modelli sono stati validati con la *leave-one-out cross-validation*. Questa prevede che, durante l'allenamento, il train set sia formato da tutti i sample meno uno, il quale viene usato per testare il modello con un dato mai visto prima. A ogni iterazione i si ottiene uno $score_i$ e lo score finale si può facilmente calcolare con una semplice media aritmetica (seguendo l'esempio in Figura 1.1 dove la grandezza del dataset è pari a 6):

$$score = \frac{score_1 + score_2 + score_3 + score_4 + score_5 + score_6}{6}$$

Invece di dividere il dataset in training e test set una sola volta, con la cross-validation questo viene fatto più volte in modo da poter ridurre l'effetto della casualità nella suddivisione dei dati sui risultati. Il contro della leave-one-out cross-validation è che allena n modelli, dove n è la dimensione del dataset, e dunque il tempo di training può essere molto lungo. Per questo non è par-

Figura 1.1: Tecnica del leave-one-out cross-validation

ticolarmente indicata per dataset grandi, dove è invece preferibile la *K-Fold cross validation*, una sua generalizzazione in cui, a ogni iterazione, il test set è formato da K sample [30].
Le performance prese in considerazione sono l'*accuracy*, calcolata attraverso una funzione apposita delle librerie usate, e il tempo di allenamento e validazione dei modelli.
Nei capitoli successivi verranno mostrati alcuni articoli interessanti da cui questa tesi ha preso spunto e di cui ha cercato di superare i problemi, verranno descritti in dettaglio i metodi e i materiali utilizzati per creare le diverse reti neurali e sarà presente una discussione dei risultati di queste, comparando pro e contro delle soluzioni. Infine, nelle conclusioni verranno ricapitolati i punti essenziali dello studio di tesi e presentati alcuni possibili sviluppi futuri di questo lavoro.

Capitolo 2

Revisione della letteratura

In questo capitolo vengono riportati gli articoli scientifici correlati allo studio di tesi, evidenziando in particolare punti di forza e critiche ai metodi utilizzati. La ricerca è stata concentrata sulle tecniche di modelling di reti neurali applicati a contesti riabilitativi, su reti neurali specializzate nel processamento di dati di serie temporali e sulla creazione di modelli multimodali.

2.1 Riabilitazione attraverso il Deep Learning

Con il termine *riabilitazione* si intende un processo nel corso del quale si porta una persona con disabilità a raggiungere il miglior livello di autonomia possibile sul piano fisico, funzionale, sociale, intellettivo e relazionale, con la minor restrizione delle sue scelte operative.
La riabilitazione si basa su due tipi di attività: attività sanitarie, che mirano a valutare, diagnosticare, curare e ridurre la disabilità e le difficoltà presenti nelle attività quotidiane (come spostarsi, camminare, parlare, comunicare, lavarsi, nutrirsi, lavorare, etc.), e attività di riabilitazione sociale, ovvero azioni che si occupano di favorire la partecipazione alla vita sociale del disabile il più possibile, al fine di limitare la situazione di handicap.
Durante il recupero post-operatorio o il trattamento di diverse patologie

muscolo-scheletriche, è necessaria la partecipazione a programmi di riabilitazione. Molti degli attuali sistemi sanitari mondali sono organizzati in modo tale da conseguire una prima fase del programma riabilitativo in una struttura ospedaliera sotto la diretta supervisione di un medico, per poi proseguire con una seconda parte seguita in un ambiente ambulatoriale in cui i pazienti eseguono una serie di esercizi prescritti nella propria abitazione [20]. Secondo [15], oltre il 90% di ciascun episodio di terapia muscolo-scheletrica viene svolta dal paziente in casa, tipicamente come "home exercise program" (HEP), e, escluse le visite periodiche con un personal trainer nelle quali possono ottenere una valutazione riguardo lo svolgimento degli esercizi e determinare i progressi, i pazienti si ritrovano non di rado a dover utilizzare dispense cartacee, non interattive e, spesso, di bassa qualità per essere guidati nell'esecuzione degli esercizi. I medici riportano bassi livelli di aderenza dei pazienti ai regimi di esercizio raccomandati nella riabilitazione domiciliare causati da questo e tanti altri motivi, tra cui il non aver abbastanza tempo per effettuare la cura, orari di lavoro proibitivi, riduzione dell'autostima, credere che il trattamento non sia efficace o vedere risultati positivi a breve termine e sentire che non ci sia bisogno di continuare la terapia [16].
Le tecniche di *deep learning* possono offrirci aiuti considerevoli nel cercare di risolvere i problemi presenti nella riabilitazione moderna, le quali sono anche più efficaci delle tecniche di *machine learning* tradizionali. In letteratura, infatti, esistono diversi tipi di tecniche orientate ai dati che si allenano grazie a essi per poter imparare a eseguire diversi task. I modelli machine learning tradizionali, cioè quelli utilizzati particolarmente prima dell'avvento e della crescita esponenziale dell'utilizzo del deep learning, sono allenati su dataset relativamente piccoli e hanno bisogno di poco tempo per l'allenamento dal momento che creano correlazioni semplici e lineari tra gli esempi in input. Alcuni degli algoritmi più utilizzati sono i *Decisional Trees*, *Random Forest*, *Naive Bayes*, *K-Nearest Neighbours*, *Support Vector Machine* e *K-Means*.
Il deep learning, invece, è una tecnica di machine learning che si basa sulle reti neurali artificiali, un modello di calcolo formato da "neuroni" artificia-

li che si ispira in modo approssimativo a una rete neurale biologica, usato per cercare di risolvere problemi di intelligenza artificiale. I campi in cui si può applicare sono moltissimi, tra cui (ma non limitati a) computer vision, speech recognition, natural language processing, machine translation, bio-informatica, analisi di immagini mediche e scienze climatiche. L'apprendimento può essere supervisionato (in cui l'obiettivo dell'apprendimento è quello di produrre una funzione in grado di "apprendere" dai risultati forniti durante la fase di training e in grado di avvicinarsi a dei risultati desiderati per tutti gli esempi non forniti), non supervisionato (che consiste nel fornire al sistema informatico una serie di input che verranno riclassificati e organizzati sulla base di caratteristiche e pattern comuni per cercare di effettuare ragionamenti e previsioni sugli input successivi) e semi-supervisionato (in cui una piccola parte di dati sono accoppiati al risultato da ottenere, mentre la restante maggior parte dei dati non ha alcuna informazione sull'output desiderato).

Ma perché usare il deep learning e non altre tecniche di apprendimento automatico? Le reti neurali offrono diversi vantaggi: soltanto le "shallow" features possono essere apprese in base alle competenze di esperti [32] mentre le reti neurali permettono di apprendere automaticamente nuove features e relazioni non indicate, creando correlazioni complesse e non-lineari tra i dati. Inoltre, la maggior parte delle tecniche tradizionali focalizzano l'apprendimento su dati statici, mentre le attività di riabilitazione, come molte altre attività non correlate all'ambito medico, sono costituite da stream di dati [29]. C'è dunque bisogno, in questo campo applicativo, di un metodo di apprendimento robusto e incrementale.

Di seguito verranno elencati alcuni esempi di come i modelli di deep learning possono fornire un grande aiuto per i pazienti, ma verranno anche sottolineati alcuni aspetti critici degli studi in letteratura.

I movimenti funzionali degli arti superiori consentono agli esseri umani di svolgere le attività della vita quotidiana e, nell'ambito della riabilitazione, i medici cercano di accelerare il ripristino motorio allenando proprio le attività

funzionali (movimenti specifici e volitivi). È stato infatti riscontrato che i pazienti con lieve compromissione utilizzano un maggior numero di movimenti funzionali per portare a termine le attività quotidiane [28]. A oggi, però, la misurazione dei movimenti funzionali nella riabilitazione è una sfida tecnica ostacolata da imprecisione o impraticabilità: non esiste infatti un linguaggio universale per caratterizzare questo tipo di movimenti o i suoi elementi costitutivi fondamentali [26, 28]. Per questo gli autori di [26] hanno sviluppato "PrimSeq", un framework basato sul deep learning per identificare e contare automaticamente le primitive funzionali nell'allenamento riabilitativo in seguito a un infarto. Il loro approccio è stato quello di usare un modello sequence-to-sequence formato da tre fasi: la prima corrisponde alla cattura dei movimenti della parte superiore del corpo durante la riabilitazione sfruttando unità di misura inerziale indossabili (IMUs), la seconda consiste nella generazione di sequenze primitive a partire dai dati IMU attraverso il modello allenato e, infine, la terza comprende il conteggio delle primitive attraverso un apposito algoritmo. Anche se i risultati possono essere incoraggianti, vi sono alcuni problemi: per configurare correttamente l'IMU è stato necessario un tempo medio di 12.9 $\pm$ 4.1 minuti e, inoltre, non sempre si ha una disposizione economica tale che permette l'acquisto di sensori simili.

Le limitazioni di stampo economico si ritrovano in numerosissimi studi in letteratura che prendono il nome di *metodi sensor-based*. Questi sono molto utilizzati in quanto i sensori hanno fatto grossi progressi nella loro capacità di comunicare in modo wireless abbattendo i consumi e nella loro capacità di processare quantità sempre più elevate di dati. Inoltre, a differenza dei metodi *vision-based*, non vi è alcuna preoccupazione per quanto riguarda la privacy degli utenti e la complessità computazionale della creazione dei modelli è generalmente più bassa, non dovendo processare video o immagini ma dati tabulari [14].

Gli autori di [33] hanno usato i dati forniti da alcuni sensori, tra cui dei "*wearable human activity recognition folder*" (WHARF) e un DA14583 IoT (Dialog Semiconductor, Taipei, Taiwan), per sviluppare una rete neurale

convoluzionale multipath, costituita a sua volta da due reti neurali convoluzionali, con lo scopo di riconoscere gli esercizi di riabilitazione. La prima rete convoluzionale è usata per catturare la distribuzione dei dati raccolti dai sensori per i movimenti del corpo negli esercizi di riabilitazione fisica; la seconda estrae le probabilità di transizione degli stati nascosti della rete come features discriminative di diversi movimenti.
Rehab-net è un deep learning framework per classificare efficacemente tre movimenti del braccio umano, tra cui estensione, flessione e rotazione dell'avambraccio che, nel tempo, possono fornire una misura del progresso riabilitativo [25]. Il framework è stato in grado di eseguire l'estrazione automatica di caratteristiche su dati di accelerazione del polso di pazienti sopravvisuti a ictus, cosa che diversi classificatori (tra cui K-nearest neighbours, decision trees, support vector machine e Naive Bayes classifier) non sono riusciti a fare in autonomia, ma hanno avuto bisogno di esperti che scegliessero con cura la feature da estrarre.

2.2 Time Series Classification

Con dati time series si intende una sequenza di dati indicizzati in ordine temporale. In genere sono sequenze di dati catturate in punti successivi ugualmente distanziati nel tempo per osservare e tracciare le differenze di un fenomeno col passare del tempo. Possiamo distinguere due tipi di serie [10]:

- Serie temporali univariate: un insieme ordinato $X = [x_1, \ldots, x_n]$ di n valori reali
- Serie temporali multivariate: un insieme ordinato $X = [X^1, \ldots, X^m]$ formato da m serie temporali univariate con $X^i \in \mathbb{R}^n$

Dal momento che i time series data hanno un ordine naturale basato sul tempo, essi sono presenti nella maggior parte dei fenomeni che accadono nel mondo, siano essi processi naturali (meteo) o artificiali, come il mercato azionario o le metriche di evoluzione della salute di un paziente in cura [17]. A

causa della loro elevata presenza, è stato necessario sviluppare architetture in grado di classificare o etichettare questo tipo di dati.
Una categoria di modelli particolarmente efficiente è quella degli *ensemble models*. Questi combinano gli output di più modelli addestrati separatamente in un unico output, ottenendo un modello finale migliore riducendo la varianza degli output. Solitamente, tutti i classificatori di base dell'ensemble sfruttano lo stesso algoritmo (ad esempio, in una Random Forest, tutti i classificatori che la costituiscono sono dei Decision Trees). Nella classificazione time series, invece, sono stati usati modelli che combinano diversi tipi di algoritmi e, facendo ciò, è possibile imparare una rappresentazione più ampia dei dati [7].
Il primo di questi modelli è stato il "Collective of Transformation-Based Ensembles" (COTE) [3]. È formato da due ensembles che contengono a loro volta numerosi classificatori: il primo contiene dei K-Nearest Neighbours, Naive Bayes, Decision Trees, Support Vector Machines, Random Forest con 100 alberi, Rotation Forest con 10 alberi e una Rete Bayesiana; il secondo contiene 8 classificatori per dataset di diversi domini. In seguito, una gerarchia piatta che combina tutti i classificatori è risultata la migliore strategia di ensemble e, seguendo questa idea, è nato il modello che spesso viene chiamato Flat-COTE, che combina 35 classificatori su quattro diverse rappresentazioni di dati.
Il modello considerato lo stato dell'arte per la time series classification è il "Hierarchical Vote Collective of Transformation-based Ensembles" (HIVE-COTE) [22]. Costituito da 5 moduli, ha vinto 45 benchmark su 85 dataset e si è classificato tra i primi 3 classificatori su 83 problemi presi in considerazione in [21], battendo il predecessore Flat-COTE.
Un altro modello ensemble molto promettente è InceptionTime, un insieme di Deep Convolutional Neural Network, ispirati all'architettura di Inception-v4. Questo ha ottenuto la stessa precisione di HIVE-COTE, riuscendo però a essere più scalabile: può apprendere infatti da una quantità di dati che il suo contendente non può gestire in un periodo di tempo accettabile [11].

Concentrando le ricerche sugli studi sulla riabilitazione, una quantità molto elevata di ricerche hanno come dati delle serie temporali. In [19], i ricercatori hanno mostrato che è possibile identificare i movimenti goal-oriented durante l'esecuzione di attività di vita quotidiana e i movimenti mal eseguiti in esercizi di riabilitazione selezionati, consentendo così la creazione di un feedback adeguato. Hanno reclutato 20 soggetti sopravvissuti a ictus e li hanno equipaggiati di unità a misura inerziale su sei assi bilateralmente sul polso. È stato chiesto loro dapprima di eseguire compiti motori e poi una serie di esercizi che assomigliavano a diversi tipi di attività di vita quotidiana e di esercizi di riabilitazione che potevano eseguire da soli a casa. Per individuare le caratteristiche dei dati più importanti per la classificazione, hanno utilizzato l'algoritmo "minimal redundancy maximal-relevance". Di queste, le prime 15 feature sono state date in input a un Logisitic Regressor per classificare le diverse attività. In questo studio non è dunque stata usata una rete neurale e il numero di feature è abbastanza limitato. Inoltre, come in molti altri studi su dati time series, il dataset è composto esclusivamente da informazioni raccolte da un tipo di sensore e viene perciò escluso l'utilizzo di diversi tipi di dati provenienti dall'osservazione di diverse modalità.

2.3 Multimodal Learning

Gli umani possono percepire ciò che è presente intorno a loro in tanti modi diversi. Possiamo percepire gli oggetti, i suoni, la consistenza dei materiali, gli odori in natura e così via. Ciascuno dei nostri sensi ci fornisce una *modalità* con cui interagire con ciò che ci circonda. In termini generali, una modalità è il modo in cui qualcosa accade o viene sperimentato. Un problema è dunque categorizzato come *multimodale* quando questo presuppone l'utilizzo di più modalità di ricerca per essere risolto. Per estensione, quindi, il *multimodal machine learning* ha come obiettivo quello di sviluppare modelli che possono processare e analizzare informazioni sfruttando dati catturati in diverse modalità.

Il riconoscimento vocale, per esempio, è una delle moltissime applicazioni in cui l'apprendimento automatico multimodale è stato utilizzato per migliorare l'approccio al problema. Sono stati utilizzati infatti, oltre a dati puramente audio, anche immagini di segnali visivi del parlato. Altri esempi di problemi multimodali sono la descrizione di un media, in cui il task è quello di produrre una rappresentazione a parole di un'immagine, di un video o di un audio; la ricerca multimediale di dati in cui si possono applicare medoti di ricerca multimodali; il rilevamento di eventi, in cui bisogna classificare alcune azioni catturate da diversi tipi di media [4]. Ovviamente, l'apprendimento multimodale può essere usato nei contesti di riabilitazione e seguiranno alcuni esempi di studi in letteratura. Verrà inoltre posta l'enfasi su alcune limitazioni comuni che il lavoro di tesi ha superato.

Riconoscendo automaticamente gli stati affettivi dei pazienti, gli scenari di riabilitazione virtuale possono essere controllati per favorire interazioni empatiche e motivanti con i pazienti, per migliorare l'aderenza alla terapia e per adattare l'esercizio alle capacità fisiche e psicologiche. Tuttavia, è ancora difficile rilevare con precisione gli stati affettivi, fisici e/o cognitivi dei pazienti quando si utilizzano dati reali. Gli autori di [27] hanno sfruttato le co-occorrenze e le relazioni reciprocamente esclusive tra gli stati affettivi per supportare il riconoscimento automatico sviluppando un'architettura formata da un classificatore multi-label, chiamato Circular Classifier Chain, unito a un set di classificatori multimodali, chiamato Fusion, che sfrutta un Semi-Naive Bayesian Classifier. La prima versione del dataset comprende dati di pressione del dito, da cui sono state estratte esclusivamente 3 feature (pressione, velocità di pressione e accelerazione di pressione), di movimento della mano, da cui sono state estratte 5 feature (velocità, accelerazione e localizzazione differenziata per gli assi x, y e z), e di espressioni facciale, da cui sono state estratte 20 feaure, per un totale di 5 pazienti. La quantità di feature estratte e il numero di soggetti analizzati nello studio sono però piuttosto ridotti.

In [5], gli autori hanno prodotto degli algoritmi machine-learned per rilevare

le diverse fasi del sonno in quanto è sempre più evidente l'esistenza di un'importante connessione tra ictus e sonno. Infatti, uno studio ha mostrato che le persone con insonnia hanno una probabilità più alta del 54% di avere un ictus nei quattro anni seguenti rispetto a quelle senza insonnia della stessa età [31]. Le modalità dei sensori dello studio sulla correlazione sonno-ictus sono state quattro e in totale, da essi, sono state estratte 73 feature. Per l'apprendimento sono stati sviluppati modelli di machine learning supervisionati sfruttando quattro algoritmi: Balanced Bagging, Balanced Random Forest, Gradient Boosting e XGBoost. Anche in questo studio troviamo un numero ridotto di pazienti, nonostante la quantità di feature estratte sia stata più ampia rispetto alla ricerca mostrata precedentemente. Inoltre, l'utilizzo di metodi tradizionali di machine learning invece che di deep neural network porta alla serie di problemi evidenziati nella prima sezione del capitolo, tra cui l'utilizzo di feature "indicate" da un esperto e non imparate automaticamente dalla rete, rischiando così facendo di ignorare l'esistenza di importanti peculiarità dei dati a noi ancora ignote.
In [24] viene presentato un sistema basato su smartphone che utilizza l'Internet-of-Things, il machine learning e sistemi di tecnologie intelligenti per aiutare i sopravvissuti all'ictus a migliorare la riabilitazione dell'arto superiore. Gli smartphone possono fornire diversi sensori multimodali: il giroscopio e il sensore di orientamento possono produrre buone misure della rotazione; l'accelerometro può fornire una misura a tre dimensioni dei movimenti. I sopravvissuti all'ictus eseguono l'addestramento imposti dai medici attraverso il loro cellulare, quindi il server riceve i dati delle azioni di riabilitazione attraverso la rete internet e classifica e valuta l'accuratezza delle azioni. Per l'apprendimento è stato utilizzato l'algoritmo di Dynamic Time Warping, che calcula la similarità di due time series, e il classificatore adoperato è stato il K-Nearest Neighbour (e non una rete neurale). Non è specificato il numero di soggetti presi in esame, mentre le feature prese in considerazione sono state 9 (ancora una volta un numero abbastanza ridotto).
È, infine, importante notare come vi siano stati numerosi studi di multimo-

dal classification in cui vengono analizzati, insieme a immagini, anche dati demografici dei pazienti per migliorare le performance. In [9], sono elencate diverse ricerche in cui si cerca di capire, a partire da immagini dermoscopiche e da alcuni dati relativi all'individuo, se il paziente ha un cancro alla pelle. Le informazioni non visive variano da 2 a 8 e comprendono sempre l'età e il sesso del soggetto. I risultati delle 11 ricerche prese in considerazione mostrano come le immagini rimangono gli input principali per ottenere una buona diagnosi, ma suggeriscono anche che i dati dei pazienti possono migliorare le prestazioni delle reti convoluzionali.

Capitolo 3

Materiali e metodo

3.1 Progetto di Intelligenza Artificiale

Il progetto del corso di Intelligenza Artificiale è stato proposto in collaborazione con l'Istituto Ortopedico Rizzoli di Bologna. L'obiettivo dello studio è stato quello di sviluppare un metodo in grado di predire in automatico 5 punteggi riguardo la riabilitazione dei pazienti dell'istituto in due momenti diversi, rispettivamente prima e dopo la riabilitazione. Per fare ciò, i dottori hanno ripreso 32 soggetti, di ambo i sessi e tra i 35 e gli 80 anni, mentre eseguivano 5 diversi tipi di esercizi fisici: elevazione anteriore, abduzione, extra rotazione, intra rotazione ed elevazione e rotazione superiore. Per ciascuna delle due sessioni di riprese (prima e dopo la riabilitazione), sono state registrate più o meno 10 ripetizioni per ogni esercizio. In seguito, i dottori hanno assegnato 5 punteggi relativi alla qualità delle esecuzioni, i cui valori vanno da 1 a 5 (dove 5 è il valore massimo). I voti sono: **obiettivo** (è stato raggiunto l'obiettivo primario dell'esercizio?), **ampiezza** (è completa l'ampiezza del movimento?), **capo** (è corretta la postura del capo?), **spalla** (è corretta la postura della spalla?) e **tronco** (è corretta la postura dl tronco?). Io e il mio gruppo siamo partiti dai panel in formato *.xlsx* (fogli di lavoro di Microsoft Excel) forniti dai dottori dell'Istituto Rizzoli contenenti i dati dei pazienti con le relative votazioni. Queste schede erano "esteticamente" chia-

re ma difficili da analizzare e processare automaticamente utilizzando degli script. Per risolvere il problema, li abbiamo semplificati ed esportati in file con estensione *.csv*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Punteggi	1= Per nulla	2= Poco	3= Abbastanza	4= Molto	5= Moltissimo		1.ELEV ANT				Punteggi	1= Per nulla	2= Poco	3= Abbastanza
2	REP SES 1	Obiettivo	Ampiezza	Capo	Spalla	Tronco	Score	E' stato raggiunto l'obbiettivo primario dell'esercizio?				REP SES 2	Obiettivo	Ampiezza	Capo
3	1	4	3	5	4	5	21	E' completa l'ampiezza del movimento?				1	5	4	5
4	2	4	3	5	4	5	21	E' corretta la postura del capo?				2	5	4	5
5	3	4	3	5	4	5	21	E' corretta la postura della spalla?				3	5	4	5
6	4	4	3	5	4	5	21	E' corretta la postura del tronco?				4	5	4	5
7	5	4	3	5	4	5	21					5	5	4	5
8	6	4	3	5	4	5	21					6	5	4	5
9	7	4	3	5	4	5	21					7	5	4	5
10	8	4	3	5	4	5	21					8	5	4	5
11	9	4	3	5	4	5	21					9	5	4	5
12	10	4	3	5	4	5	21					10	5	4	5
13							210								
14	Punteggi	1= Per nulla	2= Poco	3= Abbastanza	4= Molto	5= Moltissimo		2.ABDUZ				Punteggi	1= Per nulla	2= Poco	3= Abbastanza
15	REP SES 1	Obiettivo	Ampiezza	Capo	Spalla	Tronco	Score	E' stato raggiunto l'obbiettivo primario dell'esercizio?				REP SES 2	Obiettivo	Ampiezza	Capo
16	1	3	2	5	5	5	20	E' completa l'ampiezza del movimento?				1	5	4	5
17	2	3	2	5	5	5	20	E' corretta la postura del capo?				2	5	4	5
18	3	3	2	5	5	5	20	E' corretta la postura della spalla?				3	5	4	5
19	4	3	2	5	5	5	20	E' corretta la postura del tronco?				4	5	4	5
20	5	3	2	5	5	5	20					5	5	4	5
21	6	3	2	5	5	5	20					6	5	4	5
22	7	3	2	5	5	5	20					7	5	4	5
23	8	3	2	5	5	5	20					8	5	4	5
24	9	3	2	5	5	5	20					9	5	4	5
25	10	3	2	5	5	5	20					10	5	4	5
26							200								

Figura 3.1: Panel fornito dai medici

Una volta ottenuti i dati target, dovevamo estrapolare le feature per poter allenare dei modelli di machine learning tradizionali. Per far ciò abbiamo sviluppato un sistema che traccia la posizione delle articolazioni interessate, diverse da esercizio a esercizio, prendendo in input i video delle esecuzioni delle 5 diverse attività motorie. La libreria python ***mediapipe*** di Google permette proprio, tra le varie cose, di tracciare selettivamente alcune articolazioni umane a partire dai video. Abbiamo scelto le seguenti articolazioni per i dati esercizi:

- Esercizio 1: naso (testa), spalle, polsi e anche
- Esercizio 2, 3, 4 e 5: naso (testa), spalle, gomiti, polsi e anche

Una volta estratte le posizioni delle varie articolazioni (su entrambi gli assi x e y) abbiamo calcolato una media delle posizioni in 3 diversi momenti: all'inizio del video, alla metà e alla fine.
Ottenute le feature con cui fare il training del modello, abbiamo effettuato un merge con i dati target dei corrispondenti video per ottenere un unico *.csv*. Ricercando tra la letteratura, ci siamo ispirati ad alcuni paper per la scelta di alcuni modelli da allenare per poi compararli e la selezione è composta da:

Figura 3.2: Uso di OpenCV e Mediapipe per fare il tracking delle articolazioni

- K-Nearest Neighbours: modello che, data una nuova osservazione, permette di identificare i K punti più vicini nel training set. L'osservazione viene classificata con la label condivisa dalla maggior parte dei K punti così trovati

- Decision Tree: albero decisionale in cui lo splitting, cioè la creazione di nodi e rami, viene decisa sulla base di misurazioni statistiche; tendenzialmente sono algoritmi iterativi con i quali si provano numerose volte i parametri d'interesse per poi scegliere l'albero che porta la previsione o la classificazione migliore. Inoltre sono facili da interpretare, capire e costruire

- Random Forest: si differenzia dal decision tree per come campiona lo spazio dei previsori; ha lo scopo, infatti, di creare dei sottoalberi decisionali indipendenti tra di loro restituendo come output la classe selezionata dal maggior numero di sottoalberi

- Support Vector Machine: modello che, dato un insieme di esempi per l'addestramento etichettati con la classe di appartenenza, mappa le

osservazioni di training in punti nello spazio in modo da massimizzare l'ampiezza del divario tra le diverse categorie

Per allenare i modelli precedentemente descritti abbiamo adottato un semplice splitting del dataset in cui il 30% delle osservazioni sono state usate per la fase di testing, evitando di usare tecniche di *cross validation* e abbiamo sfruttato le seguenti tecnologie:

- Linguaggio Python (versione 3.7), per la creazione di script
- Mediapipe di Google[1] (versione 0.8.10.1), libreria di computer vision con cui abbiamo tracciato le articolazioni del corpo a partire dai video dei pazienti
- NumPy[2] (versione 1.22.4), libreria utilizzata per gestire gli array multidimensionali
- Pandas[3] (versione 1.4.2), libreria sfruttata per processare e analizzare i file tabulari in formato *.csv*
- OpenCV[4] (versione 4.6.0.66), libreria usata per elaborare i file video
- Pickle (versione 4.0), libreria con la quale è possibile serializzare oggetti e, dunque, salvare i modelli creati dopo averli allenati

Il risultato migliore ottenuto, considerando come metrica la *F1 score weighted* (misura che tiene conto dello sbilanciamento delle classi), è stato 0.83. Facendo però la media di tutti i migliori score per ogni esercizio, il risultato è pari a 0.57. In generale abbiamo riscontrato che il modello con le migliori performance è stato il Random Forest, seguito da SVM, KNN e infine il Decision Tree.
Com'era prevedibile, avendo usato metodi che non sfruttano il deep learning,

[1] https://pypi.org/project/mediapipe/
[2] https://numpy.org/
[3] https://pandas.pydata.org/
[4] https://pypi.org/project/opencv-python/

i risultati faticano a essere discreti.

ES 1

F1 Score Macro

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.137	0.184	0.189	0.2	0.2	Forest
Ampiezza	0.279	0.336	0.36	0.602	0.602	Forest
Capo	0.345	0.319	0.349	0.364	0.364	Forest
Spalla	0.144	0.177	0.11	0.154	0.177	SVM
Tronco	0.393	0.36	0.109	0.198	0.393	KNN
Media	0.2596	0.275	0.223	0.304	0.304	Forest

F1 Score Weighted

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.211	0.267	0.291	0.308	0.308	Forest
Ampiezza	0.382	0.453	0.455	0.561	0.561	Forest
Capo	0.516	0.51	0.514	0.567	0.567	Forest
Spalla	0.354	0.393	0.246	0.38	0.393	SVM
Tronco	0.401	0.511	0.181	0.303	0.511	SVM
Media	0.3728	0.427	0.337	0.424	0.427	SVM

ES 2

F1 Score Macro

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.459	0.569	0.34	0.525	0.569	SVM
Ampiezza	0.465	0.357	0.483	0.489	0.489	Forest
Capo	0.251	0.301	0.213	0.325	0.325	Forest
Spalla	0.605	0.273	0.33	0.259	0.605	KNN
Tronco	0.205	0.175	0.437	0.411	0.437	Tree
Media	0.397	0.335	0.361	0.402	0.402	Forest

F1 Score Weighted

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.664	0.779	0.468	0.741	0.779	SVM
Ampiezza	0.754	0.593	0.798	0.806	0.806	Forest
Capo	0.433	0.43	0.353	0.421	0.433	KNN
Spalla	0.743	0.419	0.558	0.497	0.743	KNN
Tronco	0.405	0.345	0.689	0.664	0.689	Tree
Media	0.6	0.513	0.573	0.626	0.626	Forest

ES 3

F1 Score Macro

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.087	0.158	0.106	0.113	0.158	SVM
Ampiezza	0.099	0.299	0.096	0.108	0.299	SVM
Capo	0.306	0.373	0.345	0.311	0.373	SVM
Spalla	0.594	0.75	0.673	0.528	0.75	SVM
Tronco	0.478	0.278	0.237	0.312	0.478	KNN
Media	0.313	0.372	0.291	0.274	0.372	SVM

F1 Score Weighted

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.123	0.177	0.102	0.154	0.177	SVM
Ampiezza	0.098	0.246	0.1	0.094	0.246	SVM
Capo	0.796	0.786	0.785	0.81	0.81	Forest
Spalla	0.748	0.779	0.769	0.631	0.779	SVM
Tronco	0.624	0.402	0.32	0.544	0.624	KNN
Media	0.478	0.478	0.415	0.447	0.478	

ES 4

F1 Score Macro

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.46	0.309	0.243	0.176	0.46	KNN
Ampiezza	0.45	0.444	0.382	0.256	0.45	KNN
Capo	0.416	0.349	0.33	0.546	0.546	Forest
Spalla	0.267	0.22	0.259	0.275	0.275	Forest
Tronco	0.241	0.369	0.333	0.483	0.483	Forest
Media	0.367	0.338	0.309	0.347	0.367	KNN

F1 Score Weighted

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.593	0.418	0.375	0.274	0.593	KNN
Ampiezza	0.574	0.537	0.499	0.355	0.574	KNN
Capo	0.663	0.528	0.623	0.668	0.668	Forest
Spalla	0.341	0.33	0.324	0.36	0.36	Forest
Tronco	0.332	0.574	0.286	0.471	0.574	SVM
Media	0.501	0.477	0.421	0.426	0.501	KNN

ES 5

F1 Score Macro

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.202	0.346	0.319	0.311	0.346	SVM
Ampiezza	0.338	0.346	0.356	0.478	0.478	Forest
Capo	0.329	0.306	0.327	0.329	0.329	
Spalla	1	0.454	0.442	1	1	
Tronco	1	1	1	1	1	
Media	0.574	0.49	0.489	0.624	0.624	Forest

F1 Score Weighted

Misura	KNN	SVM	Tree	Forest	MAX	Winner
Obiettivo	0.2	0.552	0.555	0.477	0.555	Tree
Ampiezza	0.494	0.693	0.525	0.711	0.711	Forest
Capo	0.961	0.81	0.955	0.961	0.961	
Spalla	1	0.908	0.884	1	1	
Tronco	1	1	1	1	1	
Media	0.731	0.793	0.784	0.83	0.83	Forest

Figura 3.3: Risultati sperimentali

È stato dunque immediatamente chiaro che avremmo dovuto sfruttare le potenzialità delle reti neurali per ottenere performance migliori.
Per ogni classificatore e per ogni voto è stato evidenziato in giallo il vincitore e, inoltre, abbiamo aggiunto come ultima riga la media di tutti i voti. Abbiamo riscontrato come, tra tutti, il classificatore che ha performato in generale meglio è stato il Random Forest.

3.2 Reti Neurali Time Series

Il dataset iniziale utilizzato per l'allenamento delle reti neurali è stato lo stesso, in quanto sono partito dagli stessi video degli stessi pazienti e dal medesimo panel di valutazioni fornito dai dottori dell'Istituto Rizzoli. Con esso però ho estratto un quantitativo di dati molto più elevato rispetto al progetto del corso. Per prima cosa, infatti, per ognuno dei 5 esercizi sono

state tracciate tutte le articolazioni dalle anche in su, per un totale di 9 punti, e dunque di 18 features (per ognuna delle 9 articolazioni sono presenti entrambe le coordinate spaziali X e Y. Inoltre, le coordinate delle parti del corpo sono state registrate a ogni singolo frame di ciascun video ed esportate nella loro interezza, senza combinarle tra loro (per esempio) con medie pesate o combinazioni lineari.

Figura 3.4: Tracking di tutte e 9 le articolazioni

Dato il dominio del problema, la rete neurale deve essere un modello specializzato nella classificazione di dati timeseries. Per poterla costruire ho sfruttato la libreria **TimeseriesAI (TSAI)**[5], la quale è basata su *fast.ai*[6] e permette di creare facilmente le reti scegliendo tra i diversi modelli, tra cui **InceptionTime**, l'ensemble model che ho usato nel progetto di tesi e di cui l'architettura viene mostrata in Figura 3.5.
Questa, per funzionare, richiede che l'input sia un array tridimensionale. Più precisamente, le tre dimensioni rappresentano, seguendo la documentazione della libreria:

[5] https://github.com/timeseriesAI/tsai
[6] https://www.fast.ai/

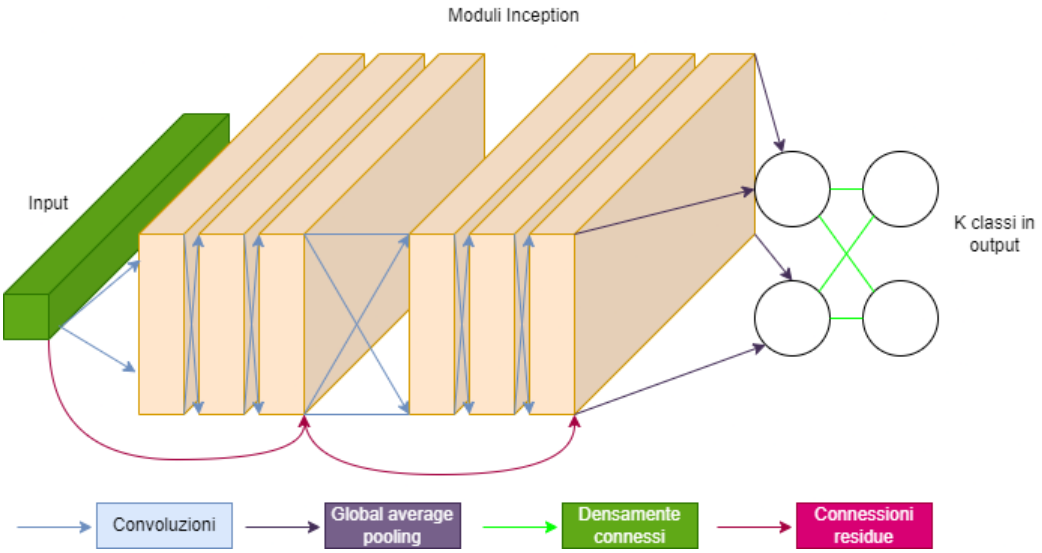

Figura 3.5: Architettura di InceptionTime

1. *Samples*, in questo caso ogni video dei pazienti;
2. *Variables*, ovvero le 18 coordinate spaziali precedentemente estratte;
3. *Sequence steps*, cioè i punti distanziati nel tempo

TSAI permette di creare modelli di classificazione o regressione sia per dataset univariati (se la seconda dimensione, rappresentata dalle variables, ha lunghezza 1), sia per dataset multivariati (nel caso in cui abbia lunghezza maggiore di 1, come nello studio di tesi).
Per iniziare ho scritto lo script python `timeseriesai_data_extractor.py` che ho usato per automatizzare il processo di estrazione dei keypoint di ogni articolazione per ciascun video, creando per ognuno un file `.csv` contenente tutte le informazioni necessarie. In totale sono stati analizzati 1443 video e le varie feature sono state estratte in 8411 secondi, più o meno 2 ore e 20 minuti.
Il dover avere un array multidimensionale contenente i vari dati dei diversi video è stato un problema da superare, in quanto l'ultima dimensione (le istanze nel tempo) ha una lunghezza distinta da video a video. Questo è riscontrabile perché ogni ripresa ha una durata differente e, dunque, i sequence

steps non possono combaciare. Per poter creare un input uniforme per i modelli, all'interno dello script `extracted_data_reader.py` ho implementato alcune funzioni che permettono di aggiustare il sequence steps di ciascun video durante l'estrazione dai video. Per non introdurre bias durante l'implementazione, ho reso possibile passare come argomento il numero di frame (e dunque il sequence steps) alla funzione `adjust_videos_frame_number`, il quale può assumere 4 tipi di valori:

- **mean**, indica che il numero di frame da estrarre deve essere pari al numero di frame medio di tutti i video dello stesso esercizio
- **max**, indica che il numero di frame da estrarre deve essere pari al numero di frame massimo di tutti i video dello stesso esercizio
- **min**, indica che il numero di frame da estrarre deve essere pari al numero di frame minimo di tutti i video dello stesso esercizio
- Un numero a scelta n, indica che verranno estratti esattamente n frame

Per poter estrarre il giusto numero di frame, lo script sfrutta le funzioni `increase_frame_number` e `decrease_frame_number` che permettono di ritornare i dati con la corretta lunghezza. Di seguito viene mostrata la funzione che diminuisce il numero di frame; quella che li aumenta funziona in modo molto simile.

```
def decrease_frame_number(video_data, frame_number):
    # calculate ratio of lengths between the processed video
    # and the chosen frame number
    ratio = len(video_data[0])/frame_number
    # final data for the processed video

    adjusted_video_data = [[], [], [], [], [], [], [], [], [],
        [], [], [], [], [], [], [], [], []]
    prev_i = 0
    i = 0
```

```
    while (i < len(video_data[0])):  # foreach frame and feature
        for j in range(len(video_data)):
            # append body data
            adjusted_video_data[j].append(video_data[j][i])
        i = prev_i + ratio # skip ceil(i+ratio) frames
        prev_i = i
        i = math.ceil(i)
    # check if there is still some frame missing
    if (len(adjusted_video_data[0]) < frame_number):
        i = math.floor(i - ratio) + 1
        for j in range(len(video_data)):
            adjusted_video_data[j].append(video_data[j][i])
    # check if too much frames were added
    elif (len(adjusted_video_data[0]) > frame_number):
        for j in range(len(video_data)):
            adjusted_video_data[j].pop()
    return np.array(adjusted_video_data)
```

Nello stesso script ho implementato anche diverse funzioni per permettere la visualizzazione dei dati timeseries, molto utile per capire meglio come gli esercizi cambiano nel tempo:

- `visualize_video_data`: semplice funzione che permette di visualizzare le features di un singolo video su grafo. La Figura 3.6 è un esempio del risultato

- `visualize_improvement`: funzione che permette di visualizzare il miglioramento nell'esecuzione di un eserizio da parte di un paziente tra una sessione e la successiva, mostrando la stessa ripetizione. L'esempio in Figura 3.7 mostra il miglioramento di un paziente nell'esercizio 1, elevazione anteriore (in cui per l'appunto bisogna alzare le braccia sopra la testa). È facilmente notabile come i valori Y (quindi l'altezza) di

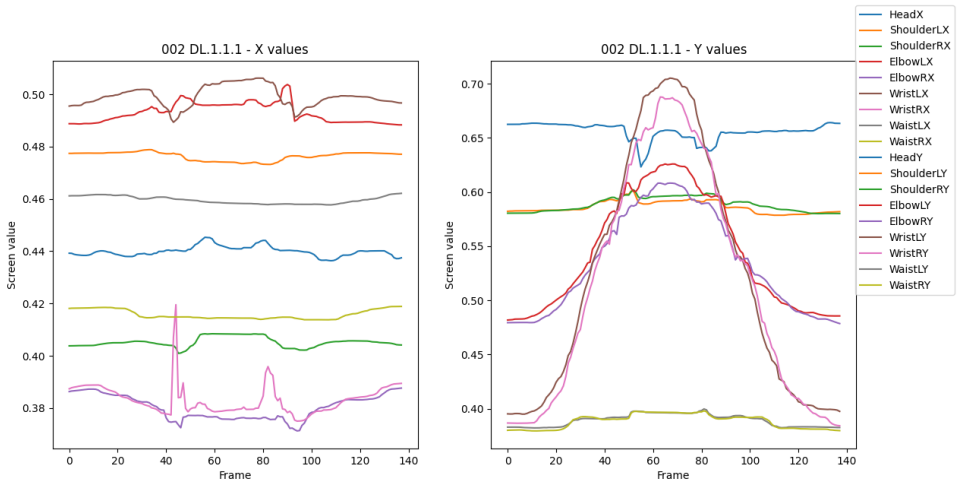

Figura 3.6: Dati spaziali di un video

spalle, gomiti e polsi raggiungano una misura più elevata significando una buona riabilitazione

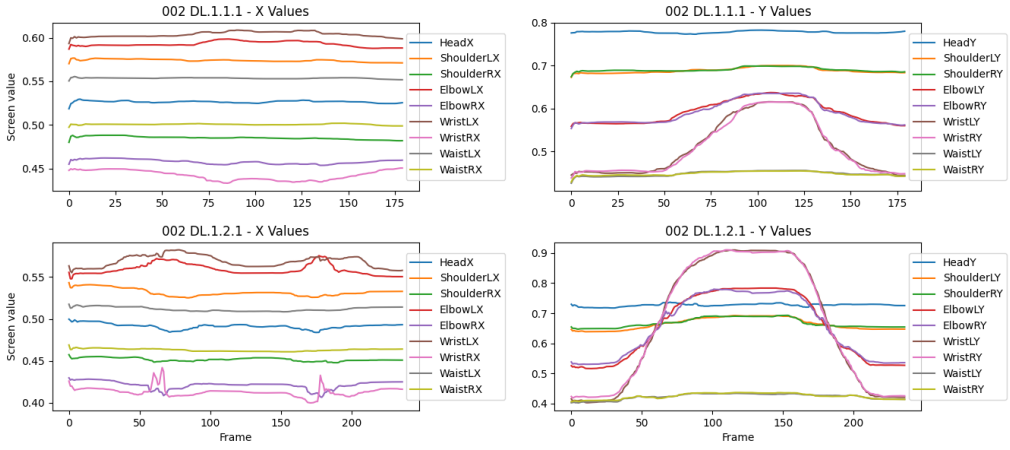

Figura 3.7: Miglioramento di un paziente tra le due sessioni

- `visualize_adjustment`: funzione che permette di visualizzare l'aggiustamento del numero di frame per un singolo video. Vengono mostrati i frame originali e i frame rispetto al numero medio, massimo e minimo di tutti i video dello stesso numero di esercizio. Come è possibile

notare grazie alle Figure 3.8, 3.9 e 3.10, l'aggiustamento non causa un cambiamento significativo del grafico

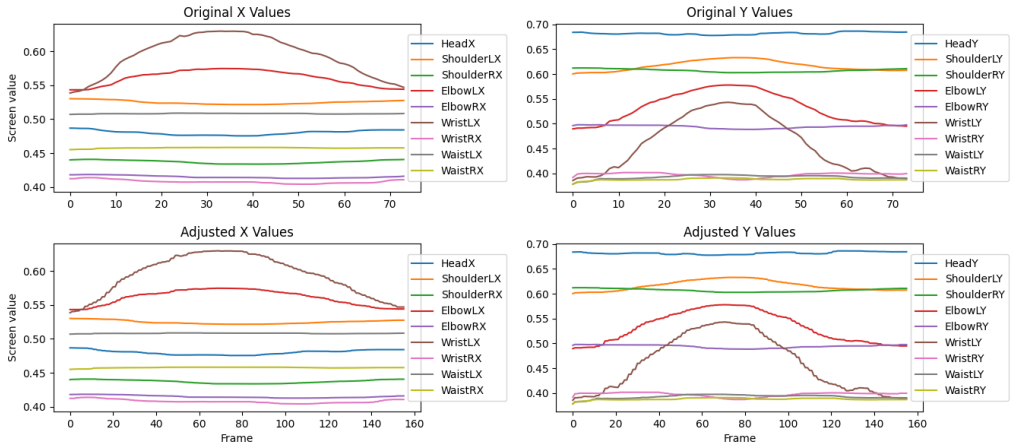

Figura 3.8: Aumento del numero di frame da 73 a 156

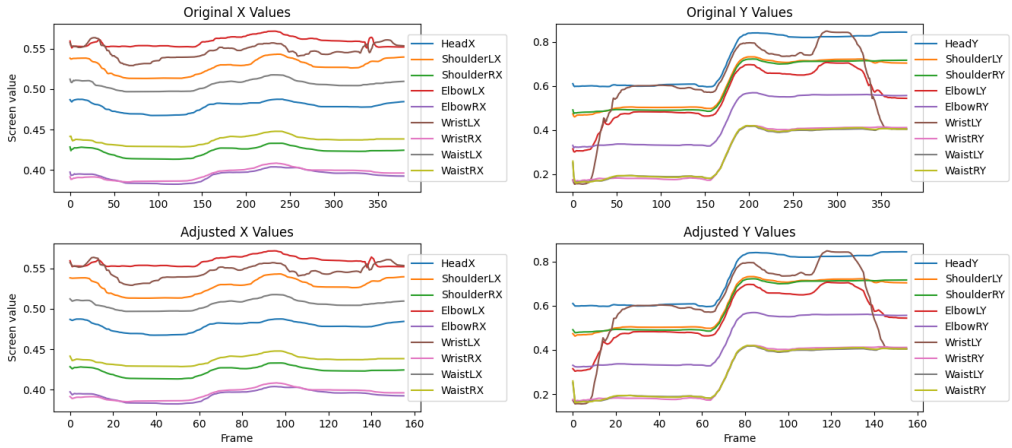

Figura 3.9: Diminuzione del numero di frame da 377 a 156

Ho continuato poi lo studio creando il notebook Jupyter `models_trainer.ipynb` con il quale creare i modelli. I diversi passaggi sono stati:

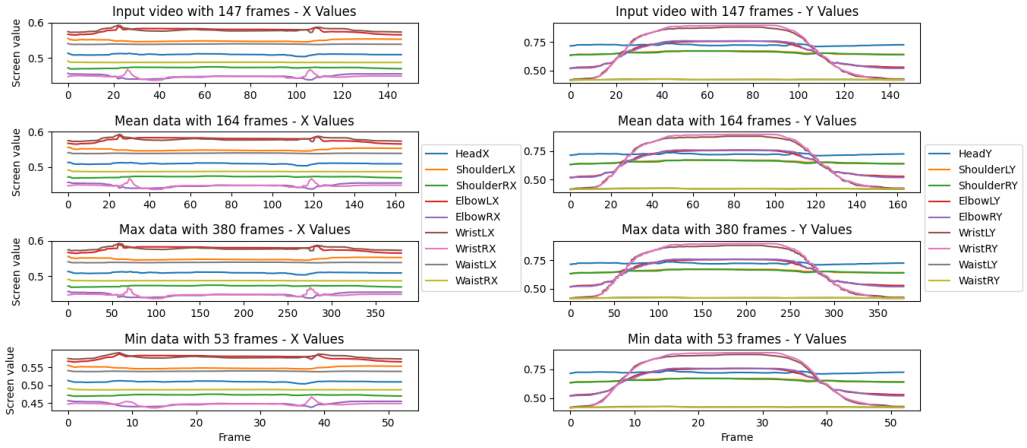

Figura 3.10: Modifica del numero di frame sul numero medio, massimo e minimo di frame di tutti i video dello stesso esercizio

1. Estrarre i dati di training con il codice sottostante. Da notare come le dimensioni delle X e delle Y combacino: la tripla di `X.shape` sta a significare che comprende 229 video, di cui ciascuno ha 18 feature (le 9 articolazioni con coordinate X e Y) e ognuno di esse ha 380 punti (numero di frame massimo dei video dell'esercizio 1); la tripla di `Y.shape` indica che per ognuno dei 229 video vi sono 5 target

```
X = np.array(extracted_data_reader.read_data_ex(
        ex_number=1, frame_number="max"))
Y = np.array(extracted_data_reader.read_target_ex(
        ex_number=1))
# dropping sum of the scores (last element of each list in Y)
y = np.array([y[:5] for y in Y])
# (X.shape, y.shape) = ((229, 18, 380), (229, 5))
```

2. Creare gli split per poter sfruttare la *leave-one-out cross-validation*. Per raggiungere tale scopo ho utilizzato la funzione `get_splits` di TSAI i cui parametri `n_splits` e `shuffle` vogliono indicare, rispettivamente, quanti gruppi di training-test sets creare (in questo caso tanti quanti

sono i sample nel dataset) e se mischiare casualmente il dataset ogni volta

```
splits = get_splits(y, n_splits=len(y), shuffle=False)
```

3. Creare, in ordine, un `TSDatasets`, un `TSDataLoaders` e un `Learner` che corrisponde al modello da voler utilizzare (tutti oggetti che mette a disposizione la libreria TSAI). Il modello in particolare si chiama `InceptionTimePlus` invece che `InceptionTime` dal momento che la libreria ha, come regola, quello di aggiungere il suffisso `Plus` ai modelli che possono predire liste di risultati (come appunto i 5 punteggi richiesti dal task di classificazione dello studio di tesi). Per l'allenamento sono state impostate 20 epoche

```
tfms = [None, TSClassification()]
accuracies = []
# leave-one-out cross validation
for split in splits:
    # create the datasets
    ds = TSDatasets(X, y, splits=split, tfms=tfms, inplace=True)
    # create the data loader from the datasets
    dls = TSDataLoaders.from_dsets(ds.train, ds.valid,
            bs=batch_size, batch_tfms=tfms, num_workers=0)
    # create learner of an InceptionTimePlus model
    learn = ts_learner(dls, 'InceptionTimePlus', metrics=accuracy)
    # fit the model
    learn.fit_one_cycle(epochs_number, 1e-3)
    accuracies.append(learn.validate()[1])
```

4. Calcolare la media di `accuracies`

```
np.mean(np.array(accuracies))
```

Per automatizzare tutto il processo di allenamento ho creato lo script `auto_trainer.py` che esegue esattamente le stesse istruzioni ma per ogni esercizio e per ogni numero di frame (medio, massimo e minimo).

3.3 Reti Neurali Multimodali e Tabulari

La rete multimodale costruita nello studio di tesi unisce le feature time series dei video dei pazienti con feature demografiche di ciascun soggetto registrato, fornite sempre dai dottori dell'Istituto Ortopedico Rizzoli. Queste ultime comprendono l'**età** (da un minimo di 35 anni a un massimo di 80), l'**altezza**, il **peso**, il **BMI** (Body Mass Index, dato biometrico definito come il rapporto tra peso e il quadrato dell'altezza di un individuo) e il **sesso**.
Il *CategoryEmbeddingModel* [12] è una rete standard feed-forward in cui le feature categoriche vengono passate a un layer di *Embeddings*. L'architettura usata in tesi è molto simile a essa: le variabili categoriche vengono passate attraverso il livello di *Embeddings* e poi a un potenziale *Dropout*, mentre le variabili continue vengono passate a un livello di *BatchNorm* unidimensionale; infine, entrambe le variabili processate vengono concatenate e passate attraverso una serie di *LinBnDrop* prima di arrivare al layer *Lineare* che corrisponde all'output aspettato. La multimodalità della rete risiede precisamente nella sua architettura. Le informazioni di contesto dei pazienti, ovvero i dati demografici elencati in precedenza, sono state trattate come dati categorici e, dunque, vengono processati dal modello passando attraverso il layer di Embeddings; le variabili continue, vale a dire quelle estratte dai video, sono processate direttamente, senza essere passate per lo stesso livello.
Dal momento che i dati demografici sono stati già consegnati in formato tabulare dai medici, è stato necessario soltanto trasformare i dati time series (già precedentemente estratti) in formato tabulare e accorpare i dati demografici con i relativi dati target. Lo script `tabdata_score_merger.py` si occupa proprio di eseguire quest'ultima unione: ripete i dati demografici del paziente per ogni ripetizione fatta all'interno dei suoi video e esporta un file

.csv contenente tutte le informazioni. Inoltre, per allenare le reti tabulari senza dati demografici è bastato semplicemente ignorare queste informazioni e utilizzare esclusivamente i dati time series aggregati.
Dato che la rete non deve più processare dati time series, ho usato la libreria `fast.ai` per la creazione della rete, nonostante `TSAI` sia stata ancora utile per l'estrazione delle feature dei video. Una differenza sostanziale rispetto ai modelli creati precedentemente è che il modello utilizzato per la rete multimodale, chiamato dalla libreria `TabModel`, non permette di predire più di un risultato. Questo ha impedito di creare modelli che potessero generare un unico output con tutti e 5 i voti necessari per la classificazione e ha forzato la creazione di un modello diverso per ogni esercizio, per ogni numero di frame e per ogni target, per un totale di 75 modelli.
Di seguito vengono elencati i passaggi seguiti per la creazione della rete multimodale e tabulare:

1. Come nella rete time series, per prima cosa vengono estratti i dati per l'allenamento. Questo viene fatto per ognuno dei 5 target. `TARGETS_LABELS` è una lista che ha valore `["goal", "head", "width", "shoulders", "trunk"]`

```
X = np.array(extracted_data_reader.read_data_ex(
        ex_number=1, frame_number="max))
Y = np.array(extracted_data_reader.read_target_ex(
        ex_number=1))
y = np.array([y[TARGETS_LABELS.index("goal")] for y in Y])
```

2. Viene poi usata la funzione `get_ts_features` di `TSAI` per poter estrarre diverse misure di aggregazione della serie temporale. Questa prende in input le feature X e il target y e ritorna un dataframe pandas che contiene, per ogni sample e per ogni coordinata delle articolazioni (sia valori X che Y), 10 misure di aggregazione, ottenendo così 180 colonne. Queste sono: *somma dei valori*, *mediana*, *media*, *lunghezza*, *deviazione*

standard, *varianza*, *radice della media quadratica*, *massimo*, *massimo assoluto* e *minimo*

```
ts_features = get_ts_features(X, y)
```

Un esempio di risultato viene riportato in Tabella 3.1. Il numero prefisso al nome dell'aggregazione è l'indice della feature di cui è stata fatta l'aggregazione

0__ sum_ values	0__ median	0__ mean	...	17__ maxi- mum	17__ absolu- te_ maxi- mum	17__ mini- mum
102.945	0.380	0.381	...	0.408	0.408	0.381
102.871	0.380	0.381	...	0.410	0.410	0.385
102.565	0.379	0.380	...	0.410	0.410	0.384

Tabella 3.1: Esempio dei risultati della funzione `get_ts_features`

3. Successivamente vengono concatenati tutti i dati demografici estratti dello stesso esercizio se si vuole allenare la rete multimodale (nell'esempio, l'esercizio numero 1), altrimenti si salta questo passaggio:

```
df = []
if use_demo:
    df_dir = f"final_tab_data/ex{ex_number}/"
    for file in sorted(os.listdir(df_dir)):
        df.append(pd.read_csv(os.path.join(df_dir, file)))
    df = pd.concat(df, axis=0, ignore_index=True)
```

Un esempio di risultato è la seguente Tabella 3.2.

4. Il passo successivo per le reti multimodali è fare il merge delle feature demografiche con quelle time series tabulari. Questo step viene saltato

Goal	Width	Head	Shoulders	Trunk	Score	id	age	height	weight	bmi	gender
3	3	5	5	5	21	001 SJ	35	182	91.0	27.5	M
3	3	5	5	5	21	001 SJ	35	182	91.0	27.5	M
3	3	5	5	5	21	001 SJ	35	182	91.0	27.5	M
3	3	5	5	5	21	001 SJ	35	182	91.0	27.5	M
3	3	5	5	5	21	001 SJ	35	182	91.0	27.5	M

Tabella 3.2: Esempio di concatenzione dei dataframe demografici

per le reti tabulari che non vengono allenate con le informazioni dei pazienti

```
if use_demo:
    ts_features.insert(loc=0, column='age',
                       value=df['age'])
    ts_features.insert(loc=0, column='height',
                       value=df['height'])
    ts_features.insert(loc=0, column='weight',
                       value=df['weight'])
    ts_features.insert(loc=0, column='bmi',
                       value=df['bmi'])
    ts_features.insert(loc=0, column='gender',
                       value=df['gender'])
```

5. Poi si prosegue specificando alcune impostazioni per costruire il modello tabulare e creando gli split per la leave-one-out cross-validation. I modelli multimodali specificano le variabili demografiche come variabili categoriche, mentre quelle tabulari senza informazioni sui pazienti avranno esclusivamente variabili continue

```
splits = get_splits(ts_features, n_splits=len(ts_features),
        shuffle=False)
procs = [Categorify, FillMissing]
cat_names = ['age', 'height', 'weight', 'bmi', 'gender']
            if use_demo else []
```

```
cont_names = list(ts_features.columns)[5:-1] if use_demo
             else list(ts_features.columns)[:-1]
y_names = 'target'
y_block = CategoryBlock()
```

Di seguito vengono spiegati i significati delle variabili create: `procs` sono le elaborazioni da eseguire sui dati, in questo caso le variabili poi definite come categoriche vengono categorizzate (trasformate cioè in numeri identificativi della classe appartenente) e vengono riempiti i valori mancanti, in caso ce ne siano, con valori di default; `cat_names` è la lista che indica quali sono le colonne categoriche e quelle, dunque, che verranno trasformate in numeri; `cont_names` sono le variabili continue, già numeriche; `y_names` indica la variabile dipendente, quella cioè da predire; `y_block` indica il tipo di task, in questo caso classificazione

6. Si allenano e validano i modelli con la tecnica del leave-one-out per 20 epoche, lo stesso valore dei modelli time series:

```
accs = []
# leave-one-out cross validation
for split in splits:
    to = TabularPandas(ts_features, cat_names=cat_names,
            cont_names=cont_names, y_names=y_names,
            splits=split, procs=procs, y_block=y_block,
            inplace=True)
    tab_dls = to.dataloaders(bs=32) # create dataloader
    # build model
    tab_model = build_tabular_model(TabModel, dls=tab_dls)
    # create learner
    learnTab = Learner(tab_dls, tab_model, metrics=accuracy)
    learnTab.fit(epochs_number) # fit
    # append accuracy
    accs.append(learnTab.validate()[1])
```

`TabularPandas` è un oggetto della libreria `fast.ai` che permette di trasformare un dataframe pandas in un dataset tabulare; questo viene poi trasformato in dataloader con il quale costruire il modello tabulare sfruttando la funzione `build_tabular_model`, che prende in input il modello da utilizzare e il dataloader; con questo si può creare il learner e allenare ciascun modello salvando, per ognuno, le sue performance.

7. Infine si può stampare l'accuratezza media:

   ```
   np.mean(np.array(accs))
   ```

Come nel caso delle reti timeseries, ho scritto lo script `auto_trainer_multi.py` che permette di allenare automaticamente i modelli per ogni esercizio, numero di frame e target.

3.4 Modifica al Progetto di IA

Dal momento che tutte le metriche delle reti neurali sono state calcolate attraverso la funzione della libreria `scikit_learn` `accuracy_score`, che è diversa dalla `f1_score` (weighted o no), per poter comparare i risultati del progetto ho ricalcolato le performance del Random Forest, il modello che ha performato meglio durante gli esperimenti. Ho usato il dataset delle informazioni aggregate estratte dai video come base per allenare il modello e ottenere risultati sensati, per poi aggiungerci anche i dati demografici in modo da poter fare utili considerazioni sulle performance. I punti principali sono stati:

1. Estrarre i dati dai video e le loro aggregazioni

   ```
   X = np.array(extracted_data_reader.read_data_ex(
           ex_number=1, frame_number="max"))
   Y = np.array(extracted_data_reader.read_target_ex(
           ex_number=1))
   # dropping sum of the scores (last element of each list in Y)
   ```

```
y = np.array([y[:5] for y in Y])
# getting ts features
ts_features = get_ts_features(X, y)
```

2. Per allenare i classificatori con anche i dati demografici bisogna aggiungere il seguente codice, identico a quello utilizzato per le reti multimodali con l'aggiunta di una modifica dei valori del sesso dei pazienti per renderli variabili numeriche

```
df = []
if use_demo:
    df_dir = f"final_tab_data/ex{ex_number}/"
    for file in sorted(os.listdir(df_dir)):
        df.append(pd.read_csv(os.path.join(df_dir, file)))
    df = pd.concat(df, axis=0, ignore_index=True)
    df['gender'] = df['gender'].map({'M': 0, 'F': 1})
    # merge timeseries data with demographic data
    ts_features.insert(loc=0, column='age',
                       value=df['age'])
    ts_features.insert(loc=0, column='height',
                       value=df['height'])
    ts_features.insert(loc=0, column='weight',
                       value=df['weight'])
    ts_features.insert(loc=0, column='bmi',
                       value=df['bmi'])
    ts_features.insert(loc=0, column='gender',
                       value=df['gender'])
# dropping target data
X = ts_features.iloc[:, :-5]
```

3. Allenare il modello con la leave-one-out cross-validation: in ordine, viene rimosso dal dataset l'esempio in test; viene creato il classificatore

e allenato sui dati in train; si predice il risultato per il dato in test e si calcola la metrica con la funzione `accuracy_score`

```
# leave-one-out cross validation
for i in range(len(y)):
    # get X features
    X_train = X.drop([i]).to_numpy()
    X_test = X.iloc[i,:].to_numpy().reshape(1,-1)
    # get y targets
    y_train = np.delete(y, i, axis=0)
    y_test = y[i].reshape(1,-1)
    # create RandomForest
    forest = RandomForestClassifier()
    # fit model
    forest.fit(X_train, y_train)
    # test model
    forest_y = forest.predict(X_test)
    forest_score = accuracy_score(y_test.flatten(),
            forest_y.flatten())
    accuracies.append(forest_score)
```

Anche in questo caso è stato creato uno script python per automatizzare il processo di allenamento dei modelli, chiamato `auto_trainer_randomforest.py` .

Capitolo 4

Risultati sperimentali

Di seguito verranno mostrati i risultati di ognuno dei modelli creati nelle sezioni precedenti del capitolo.
Innanzitutto viene riportata la semantica specifica dei seguenti termini che verranno usati da qui in avanti:

- *OA Avg Frame Models*: si riferisce all'Overall Accuracy del modello allenato impostando come numero di frame il valore medio del numero di frame di tutti i video dello stesso esercizio

- *OA Max Frame Models*: si riferisce all'Overall Accuracy del modello allenato impostando come numero di frame il valore massimo del numero di frame di tutti i video dello stesso esercizio

- *OA Min Frame Models*: si riferisce all'Overall Accuracy del modello allenato impostando come numero di frame il valore minimo del numero di frame di tutti i video dello stesso esercizio

- *Avg Frame Models (s)*: si riferisce al tempo in secondi occorso per l'allenamento e la validazione del modello con la leave-one-out cross-validation impostando come numero di frame il valore medio del numero di frame di tutti i video dello stesso esercizio

- *Max Frame Models (s)*: si riferisce al tempo in secondi occorso per l'allenamento e la validazione del modello con la leave-one-out cross-validation impostando come numero di frame il valore massimo del numero di frame di tutti i video dello stesso esercizio
- *Min Frame Models (s)*: si riferisce al tempo in secondi occorso per l'allenamento e la validazione del modello con la leave-one-out cross-validation impostando come numero di frame il valore minimo del numero di frame di tutti i video dello stesso esercizio

La Tabella 4.1 riporta inoltre il numero medio, massimo e minimo di frame di tutti i video dello stesso esercizio.

Esercizio	Numero di frame Medio	Numero di frame Massimo	Numero di frame Minimo
1	164	380	53
2	145	319	40
3	102	219	30
4	122	223	55
5	270	584	68

Tabella 4.1: Esercizi e numeri di frame

Di seguito, nella Tabella 4.2, vengono riportate le performance dei 15 modelli time series. Si può notare come in tutti i casi i modelli siano stati eccellentemente performanti, superando sempre una accuracy di 0.967 e arrivando addirittura a un massimo di 0.9942. Fissando il numero di esercizio, cambiare il numero di frame tra minimo, massimo e medio ha generato un cambiamento di performance molto basso. Tutti e tre gli indicatori di numero di frame hanno "vinto" 2 volte considerando tutti gli esercizi, dunque non si può decretare un vincitore assoluto. Ciò vuol dire che i modelli hanno performato bene a prescindere dal numero di frame considerato. Questo può avere anche un riscontro visivo: osservando il risultato della funzione `visualize_adjustments` (per esempio in Figura 3.10), l'andamento dei grafici è rimasto molto simile nonostante il cambiamento del numero di frame. Quello che cambia drasticamente al variare del numero di frame considerato

Esercizio	OA Avg Frame Models	OA Max Frame Models	OA Min Frame Models
1	0.9737	0.9677	0.9799
2	0.9864	0.9881	0.9847
3	0.9928	0.9880	0.9904
4	0.9917	0.9942	0.9942
5	0.9907	0.9889	0.9898

Tabella 4.2: Performance delle reti time series

è il tempo di allenamento e validazione dei modelli, visualizzabile in Tabella 4.3. Sia T_i^{min} il tempo di allenamento e validazione per il numero di frame minimo per l'esercizio i, T_i^{avg} quello per i frame medi e T_i^{max} per i frame massimi, si ha sempre che $T_i^{min} < T_i^{avg} < T_i^{max}$.

Esercizio	Avg Frame Models (s)	Max Frame Models (s)	Min Frame Models (s)
1	24483 (6.80h)	54939 (15.26h)	8993 (2.50h)
2	22527 (6.26h)	46028 (12.79h)	7053 (1.96h)
3	19791 (5.50h)	40657 (11.29h)	10381 (2.88h)
4	19141 (5.32h)	33295 (9.25h)	9497 (2.64h)
5	31027 (8.62h)	63427 (17.62h)	9480 (2.63h)

Tabella 4.3: Tempo di allenamento e validazione delle reti time series

Viene anche qui riscontrato ciò che era stato studiato in letteratura, ovvero che i modelli time series sono molto pesanti per l'allenamento: si va infatti da un minimo di 1 ora e 57 minuti circa a un massimo di 17 ore e 37 minuti. In totale, per allenare tutti e 15 i modelli è stato necessario un tempo pari a quasi 111 ore e 19 minuti, vale a dire all'incirca 4 giorni e mezzo. Infine, questi modelli contengono un totale di 982937 parametri, tutti allenabili.
Di seguito, invece, vengono riportate le perfomance dei modelli multimodali aggregati. Questo perché, come detto nella sezione precedente, il modello `TabModel` non permette la predizione di una lista di target. L'aggregazione delle misure di performance scelta è stata una semplice media aritmetica. In Appendice è possibile visualizzare le metriche di ogni singolo modello non

aggregato (precisamente nelle Tabelle A.1 e A.2).

Esercizio	OA Avg Frame Models	OA Max Frame Models	OA Min Frame Models
1	0.9284	0.6464	0.9600
2	0.8762	0.5916	0.9842
3	0.9336	0.7528	0.9888
4	0.9124	0.7800	0.9870
5	0.3316	0.3258	0.5046

Tabella 4.4: Performance delle reti multimodali

Questa volta possiamo decretare un vincitore assoluto: osservare la Tabella 4.4 rende chiaro che scegliere il numero di frame minimi degli esercizi è la strada giusta, in quanto questi modelli hanno performato meglio di tutti gli altri in ogni esercizio.
Anche in questo caso si sono raggiunte performance molto alte, con un picco di 0.9888, ma, per quanto riguarda l'ultimo esercizio, l'accuracy fatica a raggiungere un buon risultato arrivando a un massimo di 0.5046. Interessante è notare come le reti che considerano il numero di frame massimo abbiano sempre ottenuto metriche di precisione inferiori agli altri modelli, raggiungendo un massimo di 0.78 per l'esercizio 4, comunque non sufficiente per contrastarli.
Analizzando poi i tempi di allenamento e validazione dei modelli, grazie alla Tabella 4.5, possiamo notare come la proprietà dei tempi che valeva per le reti time series non vale anche per i modelli multimodali. Capita spesso, infatti, che i tempi non seguano la stessa caratteristica: per esempio, $T_1^{min} > T_1^{avg}$, $T_5^{min} > T_5^{max}$ o $T_2^{avg} > T_2^{max}$. Il lasso temporale per l'allenamento e la validazione è stato molto minore: si parte da un minimo di quasi 32 minuti per arrivare a un massimo di circa 77 minuti. In totale, la validazione e l'allenamento di tutti e 75 i modelli ha impiegato più o meno 639 minuti, ovvero 11 ore circa. L'esercizio per cui è stato richiesto maggior tempo per il processamento dei diversi numeri di frame è il primo, mentre il tempo maggiore trascorso per allenare tutti gli esercizi fissando il numero di frame è quello

usato dai modelli col numero di frame medi (pari a 234 minuti). Sempre in Appendice è possibile visualizzare i tempi sommati per ottenere le metriche appena discusse, precisamente nelle Tabelle A.3 e A.4. Per quanto riguarda il numero di parametri, i modelli sviluppati in questo studio di tesi ne hanno un totale di 64281, più di 10 volte in meno dei modelli time series.

Esercizio	Avg Frame Models (s)	Max Frame Models (s)	Min Frame Models (s)
1	2852 (48 min)	3241 (54 min)	3006 (50 min)
2	4629 (77 min)	2218 (37 min)	2215 (37 min)
3	2302 (38 min)	2251 (38 min)	2249 (37 min)
4	2341 (39 min)	2374 (40 min)	2441 (41 min)
5	1949 (32 min)	2126 (35 min)	2132 (36 min)

Tabella 4.5: Tempo di allenamento e validazione delle reti multimodali

La Tabella 4.6 mostra poi i risultati delle reti tabulari allenate senza i dati demografici dei pazienti. Si può notare come, anche in questo caso, i modelli con il numero di frame minimi degli esercizi hanno performato sempre molto meglio. Queste reti hanno raggiunto accuratezze molto alte ma quasi sempre inferiori a quelle delle reti multimodali, a esclusione dell'esercizio 5 in cui hanno ottenuto ben 12 punti percentuale in più. La Tabella A.5 riporta tutte le metriche non aggregate.

Esercizio	OA Avg Frame Models	OA Max Frame Models	OA Min Frame Models
1	0.7031	0.5057	0.9432
2	0.6568	0.5144	0.9686
3	0.7512	0.5456	0.9704
4	0.6880	0.6274	0.9212
5	0.5537	0.5491	0.6204

Tabella 4.6: Performance delle reti tabulari senza dati demografici

I tempi per l'allenamento e la validazione delle reti senza dati demografici sono stati sempre minori di quelli necessari per le reti multimodali, come visualizzabile in Tabella 4.7 (mentre in Tabella A.6 è possibile visualizzare i

tempi sommati). Inoltre questi modelli contengono un totale di 57625 parametri, quasi 10000 in meno delle reti multimodali.

Esercizio	Avg Frame Models (s)	Max Frame Models (s)	Min Frame Models (s)
1	2377 (40 min)	2338 (39 min)	2312 (39 min)
2	2079 (34 min)	2035 (33 min)	2061 (34 min)
3	2239 (37 min)	2246 (37 min)	2241 (37 min)
4	2339 (38 min)	2325 (38 min)	2279 (37 min)
5	1768 (29 min)	1764 (29 min)	1794 (29 min)

Tabella 4.7: Tempo di allenamento e validazione delle reti tabulari senza dati demografici

Come prevedibile, invece, le performance dei classificatori Random Forest, visibili in Tabella 4.8, faticano a essere accettabili. Il valore più alto raggiunto è pari a 0.625, mentre il minimo è 0.2568. Nonostante ciò, questi classificatori hanno performato meglio per l'esercizio 5 rispetto alle reti multimodali.

Esercizio	OA Avg Frame Models	OA Max Frame Models	OA Min Frame Models
1	0.4157	0.4148	0.4131
2	0.2568	0.2576	0.2568
3	0.3256	0.3248	0.3256
4	0.4390	0.4398	0.4390
5	0.6250	0.6250	0.6250

Tabella 4.8: Performance dei Random Forest

Anche i tempi di allenamento e validazione sono stati molto brevi (come ci si aspetterebbe da questo tipo di classificatore), quasi tutti al di sotto dei 2 minuti, com'è possibile osservare in Tabella 4.9.
I risultati in Tabella 4.10 invece sono relativi al Random Forest allenato con anche i dati demografici dei pazienti, che si mostrano quasi identici a quelli senza le informazioni relative ai soggetti presi in esame. Anche i tempi per l'allenamento e la validazione, visualizzabile in Tabella 4.11, sono quasi del tutto equivalenti aggiungendo o togliendo i dati demografici.

Esercizio	Avg Frame Models (s)	Max Frame Models (s)	Min Frame Models (s)
1	108	107	109
2	116	115	114
3	122	123	123
4	110	111	110
5	91	96	92

Tabella 4.9: Tempo di allenamento e validazione dei Random Forest

Esercizio	OA Avg Frame Models	OA Max Frame Models	OA Min Frame Models
1	0.4140	0.4140	0.4166
2	0.2576	0.2576	0.2576
3	0.3256	0.3248	0.3256
4	0.4390	0.4398	0.4398
5	0.6250	0.6269	0.6259

Tabella 4.10: Performance dei Random Forest con dati demografici

Esercizio	Avg Frame Models (s)	Max Frame Models (s)	Min Frame Models (s)
1	139	147	146
2	146	150	153
3	159	174	156
4	146	148	130
5	97	95	94

Tabella 4.11: Tempo di allenamento e validazione dei Random Forest con dati demografici

Viene mostrata infine la Tabella 4.12 comparativa di tutti i metodi studiati durante il lavoro di tesi. Per ognuno è riportato il dataset di partenza, il miglior modello considerando la media di tutti gli esercizi per numero di frame e il suo livello di accuracy. Si evince come i modelli migliori siano stati quelli specifici per l'elaborazione dei dati time series, seguiti dai modelli multimodali. La classifica viene terminata con i modelli di machine learning tradizionali, rispettando le previsioni. Possiamo inoltre notare come l'aggiunta dei dati demografici nei Random Forest e nelle reti tabulari non abbia in

alcun modo migliorato le performance, rendendo pertanto questi dati inutili ai fini della classificazione.

Dataset	Modello	Accuracy
Keypoint	Time Series Max Frame	0.9912
Keypoint + Dati demografici	Tabular Min Frame	0.8849
Keypoint	Tabular Min Frame	0.8847
Keypoint + Dati demografici	Random Forest Max Frame	0.4124
Keypoint	Random Forest Min Frame	0.4131

Tabella 4.12: Tabella riassuntiva performance

Capitolo 5

Conclusioni

In questo lavoro di tesi è stato presentato il progetto del corso d'Intelligenza Artificiale, il cui obiettivo è stato quello di creare modelli predittivi in grado di assegnare cinque score, con valori da 1 a 5, a dei video di pazienti che eseguivano esercizi fisici durante un periodo di riabilitazione. I grossi limiti e i risultati non convincenti del progetto del corso hanno spinto alla ricerca di nuove soluzioni più efficienti per affrontare il problema di classificazione automatica. Dopo un'attenta analisi delle varie tecniche utilizzate in letteratura per sopperire alle difficoltà che caratterizzano i modelli di machine learning tradizionali, ho deciso di seguire due percorsi distinti per vedere quale dei due avrebbe portato a dei risultati migliori.
Il primo prevede l'utilizzo di reti neurali che processano dati time series, ovvero quei dati che possono essere indicizzati nel tempo e rappresentano l'evoluzione di un fenomeno. I video degli esercizi registrati dai dottori dello IOR hanno rappresentato il dataset di partenza, in quanto i movimenti delle articolazioni dei pazienti sono stati elaborati ed esportati in modo da poterli poi utilizzare all'interno di reti neurali specializzate nel processamento di dati time series. Nonostante il modello di apprendimento profondo stato dell'arte sia il cosiddetto HIVE-COTE, il modello ensemble convoluzionale InceptionTime ha ottenuto performance simili a quelle di HIVE-COTE, permettendo però di essere più scalabile e di poter apprendere da una quantità

di dati molto più grande in meno tempo, ed è stato per questo che ho scelto di utilizzare quest'ultimo nello studio di tesi.
Il secondo percorso, invece, considera l'utilizzo di alcuni dati demografici dei pazienti, sempre forniti dai dottori dell'istituto ortopedico, in aggiunta alle informazioni di movimento delle articolazioni estrapolate dai video. In letteratura, l'apprendimento automatico che si allena a partire da diversi dati percepiti in modo diverso viene chiamato multimodale ed esistono svariate tecniche di multimodal learning. In questo studio di tesi si è scelto di usare il modello TabularModel che permette di gestire e analizzare in modo diverso le variabili categoriche da quelle numeriche. Le due modalità sono state, dunque, le seguenti: la prima sono alcune aggregazioni dei dati time series estratti dai video dei pazienti che compongono l'insieme delle variabili continue; la seconda sono le informazioni demografiche dei pazienti che vengono trattate come variabili categoriche. Queste ultime, all'interno del TabularModel, vengono passate inizialmente a un layer di Embeddings, a differenza delle variabili continue che invece vengono processate direttamente. Lo studio, a questo punto, si pone un ulteriore obiettivo: quello di scoprire se i dati demografici raccolti dai medici, cioè età, peso, altezzo BMI e sesso dei pazienti, possono avere influenze positive sulla previsione degli score. Per poter raggiungere tale traguardo sono stati allenati, in seguito, gli stessi modelli tabulari evitando però di dare loro in input i dati demografici dei pazienti, in modo da poter confrontare i risultati e decretare l'utilità delle informazioni aggiunte.
Inoltre, per poter comparare i risultati delle reti neurali con quelli del progetto del corso di Intelligenza Artificialie, ho rieseguito nello stesso modo il calcolo delle performance sul classificatore Random Forest, ovvero quello che aveva performato meglio, con e senza dati demografici.
Per ciascun approccio, sono stati allenati per ogni esercizio 3 modelli diversi, che consideravano un differente numero di frame a partire dai video (il numero minimo, medio e massimo di frame di tutti i video dello stesso esercizio). In totale, dunque, per ogni approccio sono stati allenati 15 modelli.

I risultati hanno dimostrato come le reti neurali specializzate nell'analisi dei dati time series siano state quelle che hanno performato meglio, ottenendo performance altissime. Si parte infatti da un'accuracy minima di 0.967 fino ad arrivare a un massimo di 0.994. Come appreso dalla revisione della letteratura, però, si è visto come questi modelli siano pesanti, anche a livello di tempo, per quanto riguarda l'allenamento e la validazione. Comprendendo 982937 parametri, questi modelli hanno impiegato svariate ore per terminare la fase di training. I modelli multimodali hanno raggiunto performance buone considerando come numero di frame quello minimo. Queste reti infatti hanno ottenuto accuracy a partire da 0.96 fino a 0.987, senza però contare l'esercizio 5 in cui ha performato male in confronto agli altri, raggiungendo un massimo di 0.504. Sono stati tuttavia molto più veloci durante la fase di allenamento e di validazione, mettendoci al massimo 77 minuti. Anche il numero di parametri totali, pari a 64281, indica che questi modelli sono molto più leggeri e sostenibili dei precedenti. I modelli tabulari allenati senza i dati demografici si sono comportati in maniera quasi analoga, performando molto bene considerando il numero di frame minimo. Escludendo sempre l'esercizio 5 che ha raggiunto l'accuratezza di 0.62, i modelli hanno ottenuto un picco di 0.97 e un minimo di 0.92. È molto importante notare che la media delle accuratezze dei modelli multimodali e tabulari senza dati demografici su tutti gli esercizi sono praticamente identiche e pari a 0.88.
Per quanto riguarda i classificatori Random Forest, invece, nonostante abbiano ottenuto sempre performance scarse, hanno performato meglio dei modelli multimodali e tabulari soltanto nell'esercizio 5 raggiungendo un'accuracy massima di 0.625. I Random Forest allenati con i dati demografici hanno ottenuto risultati quasi identici, anche se la media delle accuratezze su tutti gli esercizi maggiore è ottenuta dai modelli che considerano il numero di frame minimo invece che massimo.
Possiamo concludere dunque che per raggiungere ottime performance bisogna scegliere l'utilizzo di reti specializzate nell'elaborazione di dati time series come InceptionTime, nonostante siano più pesanti. I modelli tabulari e mul-

timodali hanno ottenuto buone performance considerando il numero di frame minimi dei video per ogni esercizio e possono essere una scelta da prendere in esame nel caso si volesse prediligere una soluzione più leggera e sostenibile.
È molto importante inoltre sottolineare come lo studio di tesi abbia dimostrato come esista una ragionevole capacità delle reti neurali di identificare le discriminanti per l'assegnamento automatico dei punteggi anche solo elaborando i keypoint delle articolazioni estratti dai video. Nonostante per alcuni singoli esercizi i dati demografici dei pazienti si rivelano utili al fine di migliorare la classificazione, si può concludere che la media risultante delle accuratezze su tutti gli esercizi esplica l'importanza dello studio di fattori legati al movimento e non alla situazione pregressa del paziente e, dunque, delle informazioni demografiche usate in questo elaborato.
Infine, concludo questo studio presentando possibili lavori futuri che possano migliorare e rendere più comodo l'utilizzo dei modelli creati. Le reti sviluppate, infatti, sono utilizzabili attraverso un computer su cui sono presenti tutti i vari software elencati nel Capitolo 3. Pretendere che medici o, a maggior ragione, pazienti e persone non inerenti al mondo informatico dispongano di tali materiali è altamente irrealistico. Potrebbe ritornare dunque molto utile la creazione di una applicazione web utilizzabile da terzi, come per esempio i dottori dell'IOR, che metta a disposizione un'interfaccia con cui poter caricare i video di esecuzione degli esercizi e ottenere i risultati della classificazione automatica in risposta. Non vi sarebbe dunque bisogno di possedere software specifico, ma solo di un dispositivo connesso alla rete internet, dal momento che tutta l'infrastruttura per far funzionare le reti sarebbe appoggiata su un server. Inoltre, in questo studio, ciascun modello ha avuto bisogno della creazione di una rete neurale specifica che ne riconoscesse la qualità dei movimenti. Questo implica che per prevedere gli score di un nuovo video bisogna scegliere manualmente la rete neurale specializzata che produrrà l'output. Sarebbe opportuno sviluppare un nuovo sistema di classificazione automatico che permetta di identificare autonomamente il numero dell'esercizio e in seguito richiamare la previsione scegliendo la rete neurale corretta.

Appendice A

Tabelle non aggregate

Esercizio	Target	OA Avg Models	OA Max Models	OA Min Models
1	Goal	0.904	0.607	1.000
1	Width	0.891	0.616	0.996
1	Head	0.948	0.611	1.000
1	Shoulders	0.921	0.581	0.987
1	Trunk	0.978	0.817	0.817
2	Goal	0.902	0.581	0.996
2	Width	0.839	0.521	0.967
2	Head	0.890	0.585	0.996
2	Shoulders	0.856	0.614	0.966
2	Trunk	0.894	0.657	0.996
3	Goal	0.916	0.684	0.992
3	Width	0.912	0.704	0.980
3	Head	0.900	0.720	0.988
3	Shoulders	0.944	0.724	0.984
3	Trunk	0.996	0.932	1.000

Tabella A.1: Accuracy non aggregate delle reti multimodali, parte 1

Esercizio	Target	OA Avg Models	OA Max Models	OA Min Models
4	Goal	0.912	0.797	0.992
4	Width	0.885	0.697	0.967
4	Head	0.917	0.800	1.000
4	Shoulders	0.889	0.743	0.980
4	Trunk	0.959	0.863	0.996
5	Goal	0.269	0.315	0.412
5	Width	0.306	0.250	0.505
5	Head	0.273	0.231	0.384
5	Shoulders	0.306	0.301	0.495
5	Trunk	0.504	0.532	0.727

Tabella A.2: Accuracy non aggregate delle reti multimodali, parte 2

Esercizio	Target	Time Avg Models (s)	Time Max Models (s)	Time Min Models (s)
1	Goal	449	477	624
1	Width	528	552	657
1	Head	828	1081	768
1	Shoulders	608	694	447
1	Trunk	439	437	510
2	Goal	734	457	451
2	Width	450	451	457
2	Head	439	436	438
2	Shoulders	2574	439	435
2	Trunk	432	435	434
3	Goal	478	421	469
3	Width	469	471	469
3	Head	459	464	461
3	Shoulders	447	448	447
3	Trunk	449	447	451

Tabella A.3: Tempi non aggregati delle reti multimodali, parte 1

Esercizio	Target	Time Avg Models (s)	Time Max Models (s)	Time Min Models (s)
4	Goal	456	467	454
4	Width	456	456	543
4	Head	435	430	432
4	Shoulders	436	443	503
4	Trunk	558	578	509
5	Goal	360	365	362
5	Width	372	363	359
5	Head	429	415	443
5	Shoulders	440	421	375
5	Trunk	348	562	593

Tabella A.4: Tempi non aggregati delle reti multimodali, parte 2

Esercizio	Target	OA Avg Models	OA Max Models	OA Min Models
1	Goal	0.6026	0.4367	0.9476
1	Width	0.6725	0.4017	0.9170
1	Head	0.6856	0.4367	0.9432
1	Shoulders	0.8297	0.6376	0.9782
1	Trunk	0.7249	0.6157	0.9301
2	Goal	0.5932	0.4110	0.9915
2	Width	0.5932	0.4873	0.9322
2	Head	0.6949	0.5593	0.9873
2	Shoulders	0.7288	0.5847	0.9746
2	Trunk	0.6737	0.5297	0.9576
3	Goal	0.6360	0.4480	0.9640
3	Width	0.6680	0.4000	0.9640
3	Head	0.8760	0.7160	0.9960
3	Shoulders	0.7960	0.5880	0.9680
3	Trunk	0.7800	0.5760	0.9600
4	Goal	0.5767	0.5726	0.9212
4	Width	0.6681	0.5145	0.9004
4	Head	0.7552	0.6930	0.9336
4	Shoulders	0.6929	0.6390	0.9004
4	Trunk	0.7469	0.7178	0.9502
5	Goal	0.2222	0.2454	0.3194
5	Width	0.2546	0.2037	0.3426
5	Head	0.2917	0.2963	0.4398
5	Shoulders	1.0000	1.0000	1.0000
5	Trunk	1.0000	1.0000	1.0000

Tabella A.5: Accuracy non aggregate delle reti tabulari senza dati demografici

Esercizio	Target	Time Avg Models (s)	Time Max Models (s)	Time Min Models (s)
1	Goal	589	558	536
1	Width	449	445	448
1	Head	456	445	441
1	Shoulders	442	438	442
1	Trunk	440	450	443
2	Goal	460	431	420
2	Width	419	421	425
2	Head	417	413	431
2	Shoulders	394	386	381
2	Trunk	386	381	401
3	Goal	415	448	457
3	Width	452	468	457
3	Head	450	425	413
3	Shoulders	416	402	414
3	Trunk	504	501	506
4	Goal	511	483	462
4	Width	451	453	451
4	Head	454	461	459
4	Shoulders	469	464	457
4	Trunk	451	461	447
5	Goal	348	350	364
5	Width	349	352	351
5	Head	348	350	355
5	Shoulders	355	353	357
5	Trunk	365	356	364

Tabella A.6: Tempi non aggregati delle reti tabulari senza dati demografici

Bibliografia

[1] Giovanni Abbruzzese, Roberta Marchese, Laura Avanzino, and Elisa Pelosin. Rehabilitation for parkinson's disease: current outlook and future challenges. *Parkinsonism & related disorders*, 22:S60–S64, 2016.

[2] Luay Alawneh, Tamam Alsarhan, Mohammad Al-Zinati, Mahmoud Al-Ayyoub, Yaser Jararweh, and Hongtao Lu. Enhancing human activity recognition using deep learning and time series augmented data. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–16, 2021.

[3] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[4] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[5] Pin-Wei Chen, Megan K O'Brien, Adam P Horin, Lori L McGee Koch, Jong Yoon Lee, Shuai Xu, Phyllis C Zee, Vineet M Arora, and Arun Jayaraman. Sleep monitoring during acute stroke rehabilitation: Toward automated measurement using multimodal wireless sensors. *Sensors*, 22(16):6190, 2022.

[6] Alarcos Cieza, Kate Causey, Kaloyan Kamenov, Sarah Wulf Hanson, Somnath Chatterji, and Theo Vos. Global estimates of the need for

rehabilitation based on the global burden of disease study 2019: a systematic analysis for the global burden of disease study 2019. *The Lancet*, 396(10267):2006–2017, 2020.

[7] Johann Faouzi. Time series classification: A review of algorithms and implementations. *Machine Learning (Emerging Trends and Applications)*, 2022.

[8] Wenzhong Guo, Jianwen Wang, and Shiping Wang. Deep multimodal representation learning: A survey. *Ieee Access*, 7:63373–63394, 2019.

[9] Julia Höhn, Achim Hekler, Eva Krieghoff-Henning, Jakob Nikolas Kather, Jochen Sven Utikal, Friedegund Meier, Frank Friedrich Gellrich, Axel Hauschild, Lars French, Justin Gabriel Schlager, et al. Integrating patient data into skin cancer classification using convolutional neural networks: systematic review. *Journal of medical Internet research*, 23(7):e20708, 2021.

[10] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[11] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

[12] Manu Joseph. Pytorch tabular: A framework for deep learning with tabular data. *arXiv preprint arXiv:2104.13638*, 2021.

[13] Grace Kayola, Mataa M Mataa, Melody Asukile, Lorraine Chishimba, Mashina Chomba, Dominique Mortel, Aparna Nutakki, Stanley Zimba, and Deanna Saylor. Stroke rehabilitation in low-and middle-income

countries: Challenges and opportunities. *American Journal of Physical Medicine & Rehabilitation*, 102(2S):S24–S32, 2023.

[14] Nida Saddaf Khan and Muhammad Sayeed Ghani. A survey of deep learning based models for human activity recognition. *Wireless Personal Communications*, 120(2):1593–1635, 2021.

[15] Ravi Komatireddy, Anang Chokshi, Jeanna Basnett, Michael Casale, Daniel Goble, and Tiffany Shubert. Quality and quantity of rehabilitation exercises delivered by a 3-d motion controlled camera: A pilot study. *International journal of physical medicine & rehabilitation*, 2(4), 2014.

[16] Sabine Lang, Colin McLelland, Donnie MacDonald, and David F Hamilton. Do digital interventions increase adherence to home exercise rehabilitation? a systematic review of randomised controlled trials. *Archives of physiotherapy*, 12(1):1–12, 2022.

[17] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern recognition letters*, 42:11–24, 2014.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[19] Sunghoon I Lee, Catherine P Adans-Dester, Matteo Grimaldi, Ariel V Dowling, Peter C Horak, Randie M Black-Schaffer, Paolo Bonato, and Joseph T Gwin. Enabling stroke rehabilitation in home and community settings: a wearable sensor-based approach for upper-limb motor training. *IEEE journal of translational engineering in health and medicine*, 6:1–11, 2018.

[20] Yalin Liao, Aleksandar Vakanski, and Min Xian. A deep learning framework for assessing physical rehabilitation exercises. *IEEE Transac-*

tions on Neural Systems and Rehabilitation Engineering, 28(2):468–477, 2020.

[21] Jason Lines, Sarah Taylor, and Anthony Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1041–1046. IEEE, 2016.

[22] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):1–35, 2018.

[23] EDIZIONI MINERVA MEDICA. Telemedicine from research to practice during the pandemic.“instant paper from the field” on rehabilitation answers to the covid-19 emergency, 2020.

[24] Sheng Miao, Chen Shen, Xiaochen Feng, Qixiu Zhu, Mohammad Shorfuzzaman, and Zhihan Lv. Upper limb rehabilitation system for stroke survivors based on multi-modal sensors and machine learning. *IEEE Access*, 9:30283–30291, 2021.

[25] Madhuri Panwar, Dwaipayan Biswas, Harsh Bajaj, Michael Jöbges, Ruth Turk, Koushik Maharatna, and Amit Acharyya. Rehab-net: Deep learning framework for arm movement classification using wearable sensors for stroke rehabilitation. *IEEE Transactions on Biomedical Engineering*, 66(11):3026–3037, 2019.

[26] Avinash Parnandi, Aakash Kaku, Anita Venkatesan, Natasha Pandit, Audre Wirtanen, Haresh Rajamohan, Kannan Venkataramanan, Dawn Nilsen, Carlos Fernandez-Granda, and Heidi Schambra. Primseq: A deep learning-based pipeline to quantitate rehabilitation training. *PLOS digital health*, 1(6):e0000044, 2022.

[27] Jesús Joel Rivas, Maria del Carmen Lara, Luis Castrejon, Jorge Hernández-Franco, Felipe Orihuela-Espina, Lorena Palafox, Amanda Williams, Nadia Bianchi-Berthouze, and Luis Enrique Sucar. Multi-label and multimodal classifier for affective states recognition in virtual rehabilitation. *IEEE Transactions on Affective Computing*, 13(3):1183–1194, 2021.

[28] Heidi M Schambra, Avinash Parnandi, Natasha G Pandit, Jasim Uddin, Audre Wirtanen, and Dawn M Nilsen. A taxonomy of functional upper extremity motion. *Frontiers in neurology*, 10:857, 2019.

[29] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern recognition letters*, 119:3–11, 2019.

[30] Tzu-Tsung Wong. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern recognition*, 48(9):2839–2846, 2015.

[31] Ming-Ping Wu, Huey-Juan Lin, Shih-Feng Weng, Chung-Han Ho, Jhi-Joung Wang, and Ya-Wen Hsu. Insomnia subtypes and the subsequent risks of stroke: report from a nationally representative cohort. *Stroke*, 45(5):1349–1354, 2014.

[32] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pages 3995–4001. Buenos Aires, Argentina, 2015.

[33] Zheng-An Zhu, Yun-Chung Lu, Chih-Hsiang You, and Chen-Kuo Chiang. Deep learning for sensor-based rehabilitation exercise recognition and evaluation. *Sensors*, 19(4):887, 2019.

Ringraziamenti

A mamma e a papà devo molto più di quanto è possibile scrivere in questa piccola parte alla fine della mia tesi. A loro, che mi hanno sempre guidato e aiutato, che sono sempre stati accanto a me, pronti a tutto per il mio bene, devo un grazie immenso. Sono fiero della persona che sono e lo devo anche, e soprattutto, a loro.
Ringrazio infinitamente mio fratello, compagno di mille storie e avventure. Senza di lui, le mie giornate avrebbero avuto un sapore del tutto diverso e sarebbero state incomplete. Non vedo l'ora di formare una ciurma con te per andare a cercare il *One Piece* e, come da tradizione, riporto qui una frase per noi molto importante:

- *Bella vita la nostra, eh, fratello?*
- *La migliore. Possa non cambiare mai.*
- *E possa non cambiare noi.*

Sono immensamente grato a te, Fatimella mia, per essere entrata a far parte della mia vita. Non avrei mai pensato che qualcuno potesse diventare tanto importante per me in così poco tempo, ma poi ti sei presentata in una videochiamata e da lì è cambiato tutto. Porterò sempre nel cuore tutti i momenti passati con te, dai messaggi mandati a orari improbabili ancor prima di esserci mai visti (in cui sono nati la Lupetta e il Vampiretto), allo studio sfrenato e alle incazzature dovute a certi esami, ai bellissimi viaggi che abbiamo fatto e che continueremo a fare e ai pomeriggi passati a giocare a The Forest e a Raft o a guardare serie TV di un certo calibro. Ti ringrazio per essere ciò che sei e per rendermi felice ogni giorno e non vedo l'ora di scoprire cosa il

futuro ha in serbo per noi.
Ringrazio anche Lucia e Nando, che mi hanno da sempre accolto con grandissimo affetto e mi hanno fatto sentire sin dall'inizio come di famiglia, e Sofia e Salvatore, che, nonostante ho incontrato solo una volta, mi hanno fatto sentire calorosamente accolto.
Ringrazio il relatore Dott. Stefano Pio Zingaro per avermi accompagnato in questo percorso e avermi aiutato nella realizzazione di questo elaborato.