

**UNIVERSITÀ DEGLI STUDI DI BARI   
“ALDO MORO”**

Dipartimento di Informatica

Corso di Laurea in Informatica

*Tesi di Laurea in Modelli e Metodi per la Sicurezza delle Applicazioni*

**HUMAN ACTIVITY RECOGNITION DA SMARTPHONE CON APPROCCI A.I.**

*Relatore*:  
Prof. Donato Impedovo

*Co-relatore*:  
Dott. Vincenzo Gattulli

*Laureando*:  
Gennaro Davide Paduanelli

Anno Accademico 2020/21

*“Sono le persone che nessuno*

*immagina che possano fare certe cose,*

*sono quelle che fanno cose*

*che nessuno può immaginare”*

Joan Clarke, dal film “The Imitation Game”, di Morten Tyldum.

**Abstract**

Con la presente tesi di laurea si vuole scoprire se le *Deep Belief Network* (DBN) siano un approccio valido per la risoluzione del problema riguardante il riconoscimento delle attività umane. Nello specifico si andrà a vedere la teoria dietro tale modello, il suo funzionamento, quali risultati sono stati ottenuti applicandolo su alcuni *dataset* di riferimento e un *dataset* prodotto utilizzando ShieldApp, una *app* sviluppata dal Dipartimento di Informatica per il monitoraggio e la salvaguardia dei giovani utilizzando i sensori montati sugli smartphone. Infine, i risultati del DBN verranno confrontati con i risultati ottenuti dai test di altre due intelligenze artificiali, una *Convolutional Neural Network* (CNN) e una *Bi-directional Long Short Term Memory* (BLSTM)*.*

Sommario

[Capitolo 1 – Introduzione 1](#_Toc83916120)

[1.1 **Cosa si intende per *Human Activity Recognition*** 1](#_Toc83916121)

[1.2 ***Smartphone*** 2](#_Toc83916122)

[1.3 Sensori 2](#_Toc83916123)

[1.3.1 Accelerometro 3](#_Toc83916124)

[1.3.2 Giroscopio 3](#_Toc83916125)

[1.3.3 Magnetometro 4](#_Toc83916126)

[1.3.4 Sensore di prossimità 5](#_Toc83916127)

[1.3.5 Sensore di luminosità 6](#_Toc83916128)

[1.3.6 GPS 6](#_Toc83916129)

[1.3.7 Altri sensori 7](#_Toc83916130)

[Capitolo 2 – Stato dell’arte 8](#_Toc83916131)

[2.1 Algoritmi di Deep Learning 8](#_Toc83916132)

[2.2 Convolutional Neural Network 8](#_Toc83916133)

[2.3 Recurrent Neural Network 9](#_Toc83916134)

[2.4 Long Short-Term Memory 10](#_Toc83916135)

[2.5 Bi-directional Long Short-Term memory 11](#_Toc83916136)

[2.6 Deep Belief Network 12](#_Toc83916137)

[2.7 Restricted Boltzmann Machine 13](#_Toc83916138)

[Capitolo 3 – Progettazione 15](#_Toc83916139)

[3.1 Strutturazione del progetto (introduzione) 15](#_Toc83916140)

[3.3 Progettazione della CNN 18](#_Toc83916141)

[3.4 Progettazione della Bi-LSTM 19](#_Toc83916142)

[3.5 Progettazione dellaDBN 19](#_Toc83916143)

[3.6 Dataset utilizzati 20](#_Toc83916144)

[3.6.1 UCIHAR 20](#_Toc83916145)

[3.6.2 KUHAR 21](#_Toc83916146)

[3.6.3 MotionSense 22](#_Toc83916147)

[3.6.4 ISLAB 23](#_Toc83916148)

[3.6.4.1 ShieldApp 23](#_Toc83916149)

[3.6.4.2 Come usare ShieldApp 24](#_Toc83916150)

[3.7 Preparazione dei dataset 26](#_Toc83916151)

[Capitolo 4 – Implementazione 28](#_Toc83916152)

[4.1 Python 28](#_Toc83916153)

[4.2 PyCharm 29](#_Toc83916154)

[4.3 Librerie utilizzate 31](#_Toc83916155)

[4.3.1 Matplotlib 31](#_Toc83916156)

[4.3.2 Mlxtend 32](#_Toc83916157)

[4.3.3 NumPy 32](#_Toc83916158)

[4.3.4 Pandas 33](#_Toc83916159)

[4.3.5 Scikit-learn 33](#_Toc83916160)

[4.3.6 Scipy 34](#_Toc83916161)

[4.3.7 Seaborn 34](#_Toc83916162)

[4.3.8 Tensorflow 35](#_Toc83916163)

[4.4 Funzioni utilizzate per la preparazione dei dati 35](#_Toc83916164)

[4.4.1 Misurazioni lungo gli assi X, Y, Z 36](#_Toc83916165)

[4.4.2 Magnitudine 36](#_Toc83916166)

[4.4.3 Angoli tra le misurazioni lungo l’asse e l’asse stesso 37](#_Toc83916167)

[Capitolo 5 – Sperimentazione 38](#_Toc83916168)

[5.1 Metriche utilizzate per la valutazione 38](#_Toc83916169)

[5.2 Impostazione della sperimentazione 39](#_Toc83916170)

[5.2 CNN 40](#_Toc83916171)

[5.2.1 Risultati ottenuti nella fase Instance-Based 42](#_Toc83916172)

[5.2.1.1 UCIHAR 42](#_Toc83916173)

[5.2.1.2 KUHAR 42](#_Toc83916174)

[5.2.1.3 MotionSense 43](#_Toc83916175)

[5.2.1.4 ISLAB 44](#_Toc83916176)

[5.2.1.5 Train: UCIHAR + KUHAR Test: MotionSense 44](#_Toc83916177)

[5.2.1.6 Train: MotionSense + KUHAR Test: UCIHAR 45](#_Toc83916178)

[5.2.1.7 Train: UCIHAR + MotionSense Test: KUHAR 46](#_Toc83916179)

[5.2.1.7 Train: UCIHAR + MotionSense Test: ISLAB 46](#_Toc83916180)

[5.2.2 Risultati ottenuti nella fase User-Based 47](#_Toc83916181)

[5.2.2.1 UCIHAR 47](#_Toc83916182)

[5.2.2.2 KUHAR 48](#_Toc83916183)

[5.2.2.3 MotionSense 49](#_Toc83916184)

[5.2.2.4 ISLAB 49](#_Toc83916185)

[5.2.2.5 Train: UCIHAR + KUHAR Test: MotionSense 50](#_Toc83916186)

[5.2.2.6 Train: MotionSense + KUHAR Test: UCIHAR 51](#_Toc83916187)

[5.2.2.7 Train: UCIHAR + MotionSense Test: KUHAR 51](#_Toc83916188)

[5.2.2.7 Train: UCIHAR + MotionSense Test: ISLAB 52](#_Toc83916189)

[5.3 Bi-LSTM 53](#_Toc83916190)

[5.3.1 Risultati ottenuti nella fase Instance-Based 54](#_Toc83916191)

[5.3.1.1 UCIHAR 54](#_Toc83916192)

[5.3.1.2 KUHAR 55](#_Toc83916193)

[5.3.1.3 MotionSense 55](#_Toc83916194)

[5.3.1.4 ISLAB 56](#_Toc83916195)

[5.3.1.5 Train: UCIHAR + KUHAR Test: MotionSense 57](#_Toc83916196)

[5.3.1.6 Train: MotionSense + KUHAR Test: UCIHAR 57](#_Toc83916197)

[5.3.1.7 Train: UCIHAR + MotionSense Test: KUHAR 58](#_Toc83916198)

[5.3.1.7 Train: UCIHAR + MotionSense Test: ISLAB 59](#_Toc83916199)

[5.3.2 Risultati ottenuti nella fase User-Based 59](#_Toc83916200)

[5.3.2.1 UCIHAR 59](#_Toc83916201)

[5.3.2.2 KUHAR 60](#_Toc83916202)

[5.3.2.3 MotionSense 61](#_Toc83916203)

[5.3.2.4 ISLAB 62](#_Toc83916204)

[5.3.2.5 Train: UCIHAR + KUHAR Test: MotionSense 62](#_Toc83916205)

[5.3.2.6 Train: MotionSense + KUHAR Test: UCIHAR 63](#_Toc83916206)

[5.3.2.7 Train: UCIHAR + MotionSense Test: KUHAR 64](#_Toc83916207)

[5.3.2.7 Train: UCIHAR + MotionSense Test: ISLAB 64](#_Toc83916208)

[5.4 DBN 65](#_Toc83916209)

[5.4.1 Risultati ottenuti nella fase Instance-Based 65](#_Toc83916210)

[5.4.1.1 UCIHAR 65](#_Toc83916211)

[5.4.1.2 KUHAR 66](#_Toc83916212)

[5.4.1.3 MotionSense 67](#_Toc83916213)

[5.4.1.4 ISLAB 67](#_Toc83916214)

[5.4.1.5 Train: UCIHAR + KUHAR Test: MotionSense 68](#_Toc83916215)

[5.4.1.6 Train: MotionSense + KUHAR Test: UCIHAR 69](#_Toc83916216)

[5.4.1.7 Train: UCIHAR + MotionSense Test: KUHAR 69](#_Toc83916217)

[5.4.1.7 Train: UCIHAR + MotionSense Test: ISLAB 70](#_Toc83916218)

[5.4.2 Risultati ottenuti nella fase User-Based 71](#_Toc83916219)

[5.4.2.1 UCIHAR 71](#_Toc83916220)

[5.4.2.2 KUHAR 72](#_Toc83916221)

[5.4.2.3 MotionSense 73](#_Toc83916222)

[5.4.2.4 ISLAB 73](#_Toc83916223)

[5.4.2.5 Train: UCIHAR + KUHAR Test: MotionSense 74](#_Toc83916224)

[5.4.2.6 Train: MotionSense + KUHAR Test: UCIHAR 75](#_Toc83916225)

[5.4.2.7 Train: UCIHAR + MotionSense Test: KUHAR 75](#_Toc83916226)

[5.4.2.7 Train: UCIHAR + MotionSense Test: ISLAB 76](#_Toc83916227)

[Capitolo 6 – Conclusioni e Sviluppi Futuri 77](#_Toc83916228)

[RIFERIMENTI 79](#_Toc83916229)

[RINGRAZIAMENTI 82](#_Toc83916230)

# Capitolo 1 – Introduzione

## **Cosa si intende per *Human Activity Recognition***

La *Human Activity Recognition (HAR)* è una branca dell’informatica che si occupa di *pattern* *recognition* applicata al riconoscimento e alla classificazione di dati raccolti da sensori collegati a persone, o di immagini che le raffigurano, durante lo svolgimento di attività fisiche. Le possibili aree di applicazione di tale studio sono: lo studio del comportamento umano, la creazione di assistenti per ambienti in cui vivere, sicurezza basata sulla sorveglianza dei comportamenti delle persone e il riconoscimento dei movimenti per scopi agonistici sportivi.

L’*HAR* ha ottenuto un’importanza significativa nella comunità di ricerca data la sfida che rappresenta, in quanto si cerca di riuscire a classificare le attività utilizzando un tempo di raccolta ed elaborazione dei dati sempre più ristretto. Nella predizione delle attività umane, prima di tutto si raccolgono i dati dai soggetti, mentre svolgono specifiche attività, utilizzando dei sensori; successivamente un algoritmo di *machine* *learning* viene allenato su tali dati, in modo da poter ottenere una regola di generalizzazione che verrà applicata a dati mai visti prima. L’*HAR* ha anche molte applicazioni per il miglioramento della qualità della vita delle persone più anziane. Ad esempio, attraverso continui monitoraggi un sistema *HAR* efficace può suggerire dei trattamenti di tipo medico ad hoc per il paziente, per il riconoscimento delle parti doloranti, se per esempio si tende a camminare in maniera anomala a causa di una malformazione fisica. Attraverso l’utilizzo di algoritmi specializzati, si può giungere al corretto tipo di attività fisica da svolgere, così come riconoscere quali siano le condizioni di malattia dei soggetti.

Inoltre, una delle sfide più grandi dell’*activity prediction* è quella di riuscire a generalizzare il modello in modo da essere utilizzato su diversi tipi di problemi, sensori e attività. I segnali raccolti durante le attività possono variare di molto da persona a persona, persino la stessa persona che svolge la medesima attività potrebbe restituire valori differenti. Alla stessa maniera attività differenti, possono presentare dei segnali molto simili e venire confusi tra di loro dal modello durante la fase di *training* dello stesso. Altre sfide possono essere rappresentate dal voler ridurre il costo computazionale del riconoscimento delle attività in modo che anche *device* meno potenti come i dispositivi *wearable* siano capaci di raccogliere dati in maniera accurata, riconoscere attività complesse che vengono svolte nell’arco della giornata, il tutto elaborato in locale, così da mantenere la *privacy* degli utenti. Gli approcci classici per il riconoscimento delle attività umane richiedono l’estrazione delle *feature* specifiche per il riconoscimento del determinato problema; così da ottenere solo quelle *feature* utili all’algoritmo di *machine learning* che si è scelto. Fortunatamente, il *Deep Learning* rende tale compito più semplice imparando autonomamente quali sono le *feature* utili per l’esecuzione del compito.

## ***Smartphone***

Gli smartphone, con la loro diffusione, sono man mano diventati un’estensione del corpo umano e si tende a portarli sempre con sé. Ci si aspetta che il numero degli smartphone raggiungerà circa i 4 miliardi di unità attive in tutto il globo nel 2021 [1]. Grazie all’avanzamento delle capacità computazionali, gli smartphone risultano essere sempre più performanti, e pertanto, possono essere utilizzati per migliorare la vita quotidiana.

Generalmente, questi *dispositivi* presentano un’ampia gamma di sensori quali ad esempio: accelerometro, magnetometro, giroscopio, GPS, sensore di prossimità, microfono, fotocamera, ecc. Per di più i costi per i servizi che forniscono una connessione ad *Internet* diventano sempre più economici, permettendo a chiunque di poterla utilizzare senza limitazioni. Tutti questi fattori rendono possibile con facilità le condizioni delle strade e del traffico [2], [3] [4], monitorare la qualità del sonno, tenere il conto del numero di passi personale, interazioni sociali etc.

Inoltre, questi dispositivi sono dotati di sistemi di accessibilità che rendono possibile il loro utilizzo in maniera facilitata anche a soggetti diversamente abili, sia nella mobilità che dal punto di vista cognitivo.

## Sensori

Generalmente, quando si parla di *HAR* il tipo di monitoraggio che si può effettuare è utilizzando due tipi di sensori:

* **Sensori esterni:** i sensori vengono posti in determinati luoghi o oggetti in cui l’utente interagirà con essi. Solitamente, questo tipo di soluzioni si trovano in ambienti controllati come le case domotiche. Ad esempio, è possibile trovare dei *tag RFID* in oggetti di uso comune, in modo da permettere all’ambiente di riconoscere il tipo di attività che l’abitante sta svolgendo (se sta consumando un pasto, o guardando la tv, o se si trova in un punto specifico della casa).
* **Sensori personali:** questo è il caso di sensori presenti direttamente sul corpo del soggetto. Possono essere sistemi a sé stanti e *single purpose*, vedi i sensori cardiaci per monitorare l’attività sportiva, gli *sportwatch* chemonitoranoi movimenti effettivi della persona e ne tengono traccia. Oppure si tratta di sensori presenti all’interno di uno *smartphone* come: accelerometro, giroscopio, GPS, fotocamere, sensori di prossimità, ecc.

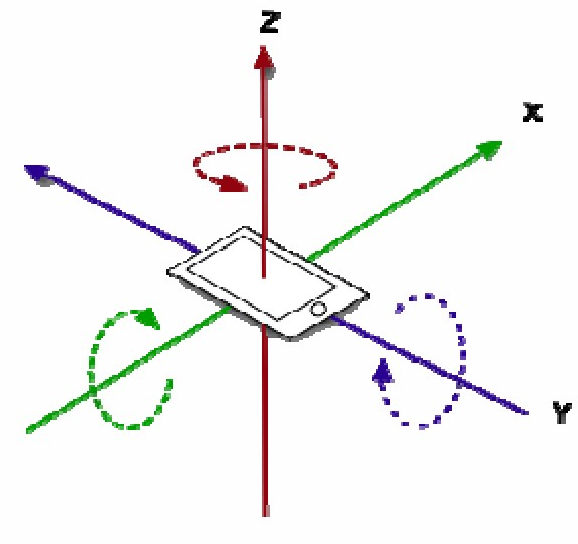
Di seguito si andrà ad analizzare nello specifico tutti i sensori presenti negli smartphone e soprattutto quelli che sono stati utili alla nostra sperimentazione.

## Accelerometro

L’accelerometro è un dispositivo capace di misurare le vibrazioni o l’accelerazione lungo i tre assi spaziali della struttura o dell’oggetto, in questo caso lo smartphone, su cui è posto [5]. L’energia delle vibrazioni viene trasformata in segnale elettrico, che aumenta di intensità in maniera proporzionale all’accelerazione dello stesso. L’accelerometro quindi, più che un sensore è un trasduttore.

L’accelerazione è una grandezza vettoriale che rappresenta la variazione della velocità nell’unità di tempo. Nel Sistema Internazionale l’unità di misura del modulo dell’accelerazione è indicata in m/s2 (metro al secondo quadro).

Negli *smartphone* l’accelerometro che viene utilizzato è di tipo triassiale, capace cioè di effettuare le misurazioni sull’accelerazione del dispositivo lungo i tre assi, come mostrato in *figura 1.*

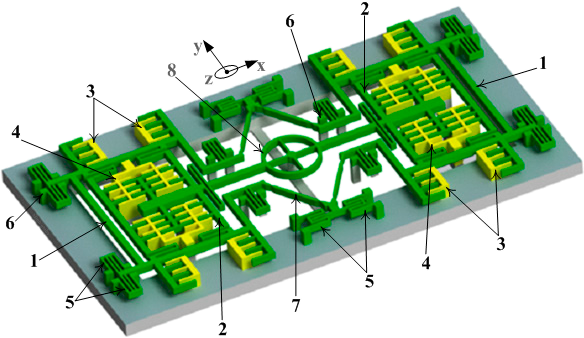
**

*Figura 1: Assi dell’accelerometro e del giroscopio negli smartphone* [6]

## Giroscopio

Il giroscopio è un sensore capace di misurare l’orientamento dell’oggetto su cui è posto e più nello specifico della sua velocità angolare. Un giroscopio è composto da un disco che ruotando ha la possibilità di assumere qualsiasi orientamento in maniera totalmente autonoma. Generalmente, durante la rotazione di questo disco non avvengono variazioni nell’orientamento rispetto all’asse su cui ruota, grazie alla conservazione del movimento angolare [5].

Negli *smartphone* vengono solitamente adottati i cosiddetti *Inexpensive vibrating structure microelectromechanical systems* o *MEMS*; il loro aspetto è simile a quello di un circuito integrato e sono capaci di restituire in *output* segnali sia analogici che digitali. Questo tipo di giroscopi sono stati definiti dall’ente *IEEE* come dei *Coriolis vibratory gyroscope*. In *figura 2* è possibile vedere la struttura di un giroscopio di tipo *vibrating MEMS.*



*Figura 2: Struttura di un giroscopio di tipo vibrating MEMS*

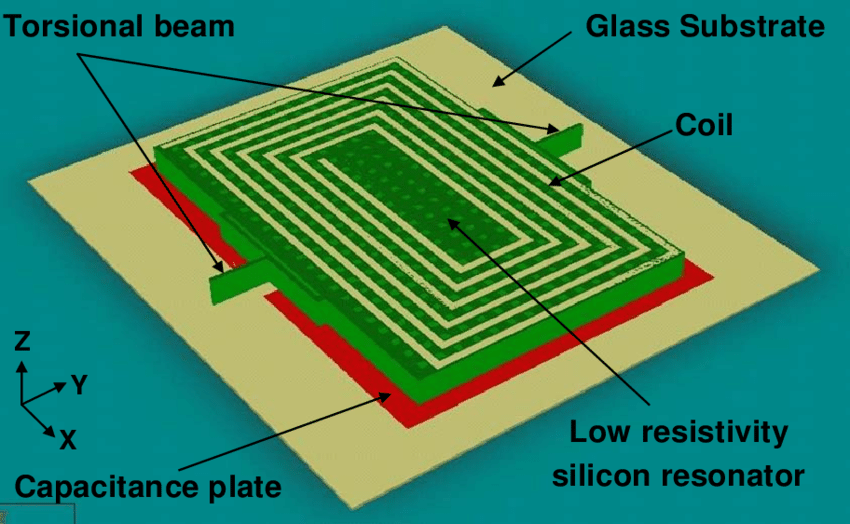
## Magnetometro

Il magnetometro è un sensore capace di percepire i campi magnetici e di misurarne l’intensità. Alcuni magnetometri sono in grado di misurare la direzione, la potenza o variazioni riguardanti il campo magnetico in un luogo particolare. Un esempio di magnetometro è la bussola, che però misura la direzione del campo magnetico che generano i poli della Terra. Altri magnetometri misurano il momento di un dipolo magnetico di un determinato materiale.

I magnetometri vengono largamente utilizzati per la misurazione del campo magnetico generato dalla Terra, controllarne variazioni e per fare degli studi sulle anomalie magnetiche che si verificano. Tali sensori hanno anche impieghi in ambito militare.

Vengono anche impiegati per la rilevazione di metalli, ad esempio i *metal* *detector* presenti in aeroporti, centri commerciali, tribunali, ecc.

Con i recenti sviluppi tecnologici si è stati in grado di miniaturizzare tali sensori, per poi riuscire a inserirli in apparecchi molto piccoli come gli *smartphone.*



*Figura 3: Schema di un magnetometro MEMS*

## Sensore di prossimità

I sensori di prossimità sono sensori in grado di percepire la vicinanza di oggetti senza che questi li tocchino effettivamente. Il loro funzionamento consiste nell’emettere dei campi elettromagnetici, solitamente raggi infrarossi, e ne controllano se ci sono state variazioni nell’intensità del segnale di ritorno. Di questa tipologia di sensori ne esistono di due tipi:

* Sensori di prossimità capacitivi, utili se si deve riconoscere la prossimità di un oggetto fatto in plastica;
* sensori di prossimità induttivi, se si deve riconoscere la prossimità di oggetti fatti di metallo.

I sensori di prossimità possono avere un’elevata affidabilità e funzionare per molto tempo, grazie proprio all’assenza di parti meccaniche e il non dover necessariamente entrare in contatto con l’oggetto per poterlo rilevare.

Questi sensori di prossimità vengono utilizzati molto negli *smartphone*, soprattutto per rendere il dispositivo capace di capire se l’utente tiene il telefono in tasca, e quindi innescare comportamenti che evitino la rilevazione di tocchi accidentali, se ha poggiato il telefono all’orecchio durante una telefonata, così da spegnere lo schermo ed evitare che surriscaldandosi possa dare fastidio all’utente. Alcuni *smartphone* implementano anche la possibilità di poter effettuare *gesture* con le mani sullo schermo, che poi verranno colti ed interpretati dal sensore di prossimità, ad esempio alcuni smartphone della *Samsung* permettono di effettuare uno *screenshot* della schermata semplicemente passando una mano sull’intera superficie dello schermo.



*Figura 4: Punto in cui di solito vengono posti i sensori di prossimità negli smartphone*

## Sensore di luminosità

Si tratta di un sensore fotosensibile, quindi capace di percepire la quantità di luce presente nell’ambiente, tale capacità viene sfruttata per poter impostare la luminosità dello schermo dello *smartphone* basandosi sull’ambiente circostante. Se si utilizzerà il *dispositivo* in un ambiente poco luminoso lo schermo sistemerà la luminosità sul valore più basso così da non accecare l’utilizzatore. Nel caso invece l’ambiente sia molto luminoso, lo *smartphone* imposterà la luminosità al massimo in modo da rendere più facile la lettura dello schermo.

## GPS

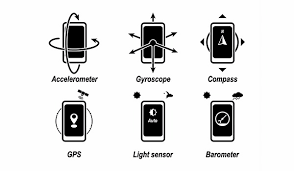
Un *Global Positioning System*, o più semplicemente *GPS,* è un dispositivo capace di ricevere informazioni da un satellite geostazionario (*GNSS)* che gli permettano di calcolare la posizione del dispositivo sulla superficie terrestre. Utilizzando dei software appositi si è anche in grado di visualizzare la propria posizione su una mappa, in modo da poter sfruttare tale tecnologia per avere indicazioni sul percorso da seguire per poter andare dal punto A al punto B.

Inizialmente questo tipo di tecnologia è stata sviluppata per scopi militari dagli Stati Uniti, tuttavia, negli anni ’80 il governo americano ha reso possibile l’utilizzo di tale tecnologia per scopi civili.

## Altri sensori

Oltre ai sensori elencati in precedenza gli *smartphone* possono essere corredati di altri sensori, come termometri, barometri, lettore *NFC,* cardiofrequenzimetro, sensori che misurano la pressione sanguigna, lettori di impronte digitali. Molti di questi sensori, come lettore di impronte digitali o lettore *NFC* non trovano alcun utilizzo per quanto riguarda il campo di studio *dell’HAR.* Inoltre, non sono disponibili in tutti gli *smartphone* in vendita, alcuni come il cardiofrequenzimetro, sono presenti solo nei top di gamma delle marche di *smartphone* più famose.

Quindi è necessario trovare approcci che si possano utilizzare anche da utenti che posseggano smartphone più economici.



*Figura 5: Altri sensori presenti sugli smartophone*

# Capitolo 2 – Stato dell’arte

## 2.1 Algoritmi di Deep Learning

Quando si parla di *Deep Learning* ci si riferisce ad una sottocategoria del *Machine Learning*, ed indica quella branca delle Intelligenze Artificiali in cui la struttura degli algoritmi si ispira alla struttura e alla funzione del cervello umano.

Il *Deep Learning* risulta essere una delle versioni migliorate del *Machine Learning* per via dei multipli strati di astrazione che spesso vengono impiegati nella realizzazione dei vari modelli appartenenti a tale categoria. Tali strati servono ad effettuare attività di *feature selection* e di filtraggio dei dati [7]. Il modello viene utilizzato per estrapolare delle regole generali da dei dati che gli vengono passati in *input,* in modo tale da poter essere in grado di classificare autonomamente dei nuovi dati mai visti dal modello. Questa fase iniziale è detta ***fase di training***. Successivamente, il modello allenato sarà in grado di fare predizioni su dati che non ha mai visto prima con una certa *accuracy*.

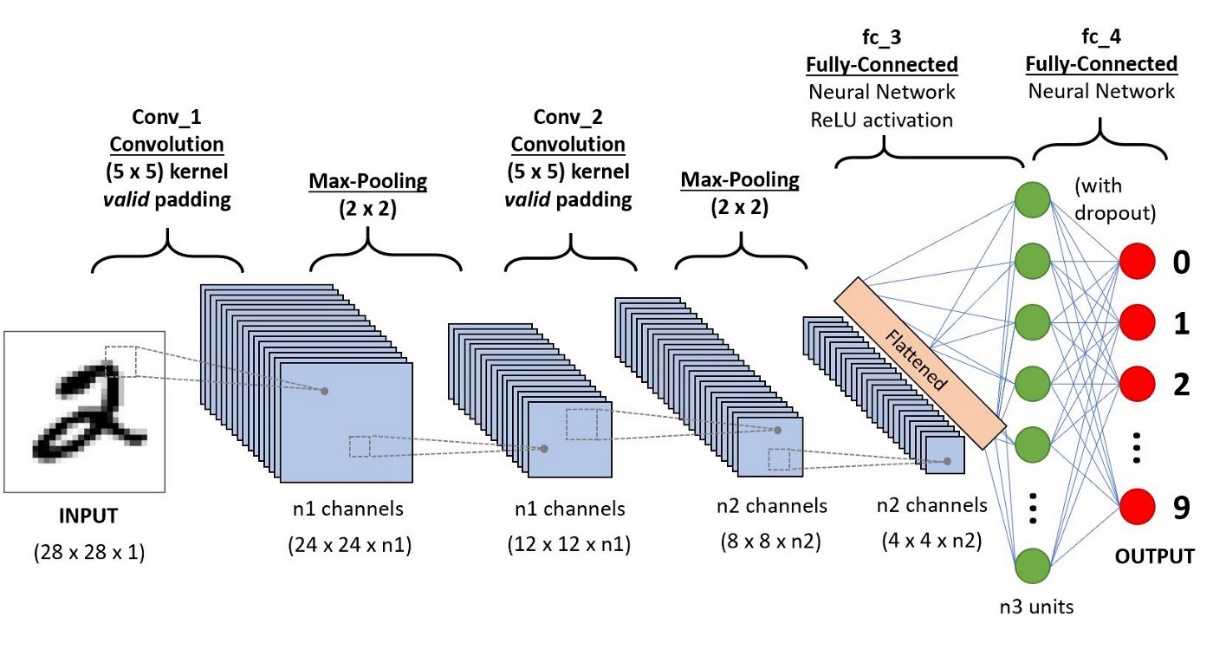
Alcuni dei vari *layer,* che compongono l’intelligenza artificiale, sono detti *abstraction.* L’*output* di un *layer* diventa l’*input* del layer successivo. I **neuroni**, cioè i singoli nodi che compongono i vari *layer*, si occupano di effettuare tutti i calcoli necessari per poter elaborare gli *input* del proprio *layer* e preparare i dati per il *layer* successivo. Ad ognuno di questi neuroni viene associato un peso, detto *weight*, e al variare di tale peso, il modello sarà in grado di effettuare predizioni con una accuratezza diversa. La causa della variazione dell’accuratezza del modello è dovuta alle variazioni apportate dal modello stesso ai pesi di ogni suo singolo neurone.

Generalmente, quando si leggono articoli riguardanti l’*HAR,* gli algoritmi che più vengono utilizzati per via delle loro performance sono la CNN e il Bi-LSTM, in quanto per la loro struttura e capacità di categorizzazione dei dati sono in grado di restituire risultati più alti ed affidabili.

## 2.2 Convolutional Neural Network

La *Convolutional Neural Network* viene largamente utilizzata per l’analisi delle immagini, tuttavia, ha dimostrato di poter essere utilizzata con ottimi risultati per il riconoscimento di altri tipi di segnali come *speech recognition, text analysis* e *human activity prediction* [8]*.* Nell’HAR, l’utilizzo della CNN porta a due benefici principali, se confrontata con gli altri modelli: la *local dependency* e la *scale invariance* [9]*.*

L’architettura di un *CNN* è molto simile alla struttura dei neuroni presenti nel cervello umano e più nello specifico è ispirata all’organizzazione della corteccia visiva. Nel cervello umano i singoli neuroni rispondono agli stimoli presenti solo in un’area ristretta del campo visivo, conosciuto come Campo ricettivo. Una serie di tali aree sovrapposte serve a coprire l’intera area visiva.

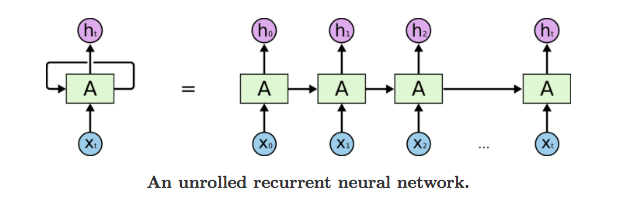


*Figura 6: Architettura di una Convolutional Neural Network*

Un elemento importante che differenzia le *CNN* dalle altre reti neurali è la presenza di *layer* convoluzionali nella rete, il che porta a una condivisione dei pesi assegnati ai singoli neuroni nell’intera rete. Le *CNN* presentano, inoltre, dei *pooling layer* in cui avvengono operazionidi *downsampling* sulle *feature* passate, solitamente effettuano una sorta di media sui dati oppure operazioni di *max pooling*, per ridurre le dimensioni spaziali delle *feature map* così da alleggerire il costo computazionale della rete.

## 2.3 Recurrent Neural Network

Una *Recurrent Neural Network (RNN)* è una classe di reti neurali artificiali dove le connessioni tra i nodi della rete formano un grafo orientato secondo una sequenza temporale. Questo permette alla rete di assumere un atteggiamento dinamico che si adatta in base alla variazione dei dati nel tempo. Le *RNN*, derivano dalle *feedforward neural networks*, possono utilizzare i loro stati interni (le memorie) per processare sequenze di *input* di lunghezza variabile [10] [11] [12]. Questa caratteristica rende tali reti applicabili a compiti di natura variabile, come il riconoscimento della grafia umana oppure *speech recognition* [13] [14].

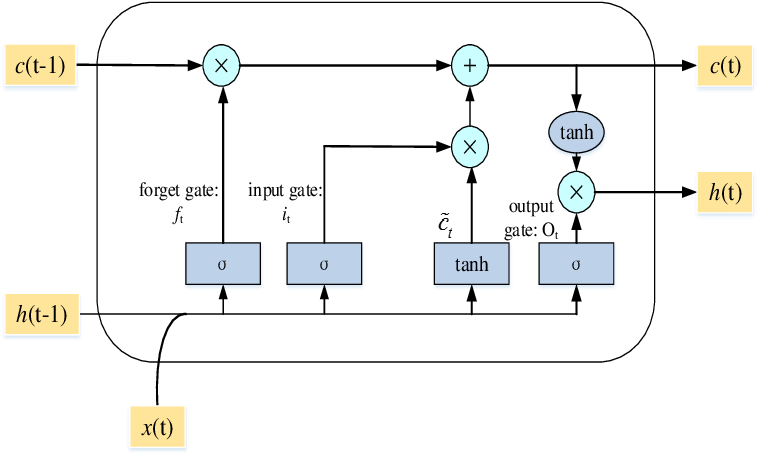


*Figura 7: Struttura di una RNN*

## 2.4 Long Short-Term Memory

La *Long Short-Term Memory (LSTM)* è un algoritmo appartenente al *deep learning* che mira ad evitare il problema del *vanishing gradient.* La *LSTM* è normalmente composta da *recurrent gate* chiamati “*forget gates”* [15]. Essa è capace di prevenire la *backpropagation* degli errori. Tuttavia, gli errori si possono ripercuotere sui *layer* precedenti attraverso un numero illimitato di *layer* virtuali. La *LSTM* è capace di imparare come svolgere compiti che necessitano di memoria di eventi che sono accaduti in migliaia o addirittura milioni di passi, in termini di tempo, precedenti. [16].

La *LSTM* può imparare a riconoscere linguaggi di tipo *context-sensitive* a differenza di altri modelli basati sugli *Hidden Markov Models* e algoritmi simili [17].



*Figura 8: Struttura di una LSTM*

Immagine che contiene notte

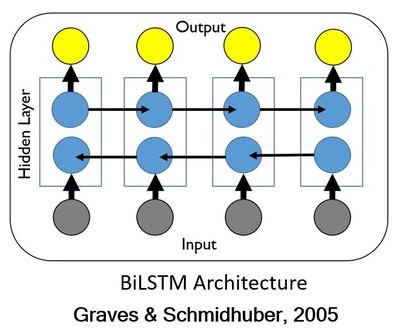
Descrizione generata automaticamente

*Figura 9: Struttura di un Hidden Markov Model*

## 2.5 Bi-directional Long Short-Term memory

Le *RNN* bidirezionali utilizzano una sequenza finita per predire o etichettare ogni elemento della sequenza, basandosi sul contesto passato e futuro della sequenza di elementi che sta analizzando. Questo processo viene svolto concatenando gli *output* di due *RNN*, la prima *RNN* processa la sequenza di dati procedendo da sinistra verso destra, la seconda rete si muove al contrario, da destra verso sinistra.

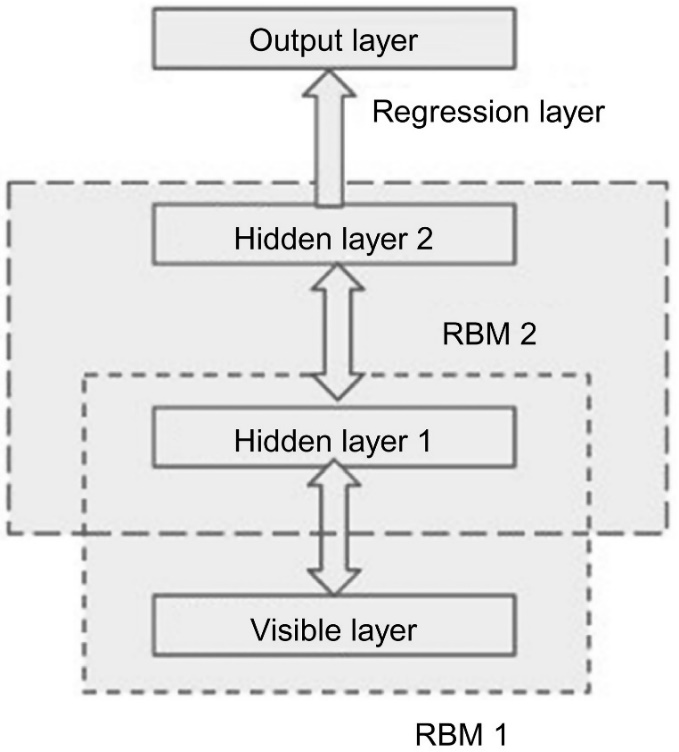
Tale tecnica si è dimostrata particolarmente utile quando combinata con *LSTM RNN* [18][19]*.*



*Figura 10: Struttura di una Bi-LSTM*

## 2.6 Deep Belief Network

Una *Deep Belief Network (DBN)* viene considerata come un approccio robusto e veloce facente parte delle tecniche di *Deep Learning* [20]*.* Questa tipologia di rete è composta da delle *Restricted Boltzmann Machines (RBM)*, utili all’allenamento, che utilizzano delle *back-propagation neural network* per la messa a punto dei pesi della rete. Per di più, la *DBN* risulta essere un ottimo classificatore per il problema dell’HAR, soprattutto se in gioco ci sono molte classi da distinguere.

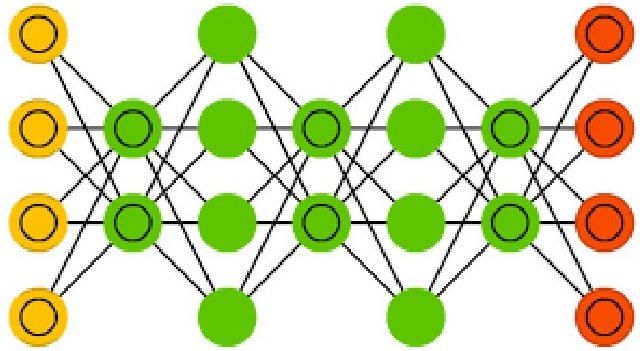


*Figura 11: Architettura di una DBN contenente due RBM* [21]

Una *DBN* è quindi un modello grafico generativo composto da diversi *layer*, chiamati *hidden units*, i quali presentano connessioni fra i *layer,* ma non tra le unità di ogni *layer* [22]*.* Quando allenata su un insieme di esempi senza supervisione, una *DBN* può imparare a ricostruire probabilisticamente i suoi *input*. In seguito, i *layer* si comportano come dei *feature* *detector* [22].

Dopo questo *learning* *step*, un *DBN* può essere ulteriormente allenato con supervisione per poter essere in grado di classificare [23]. I *DBN* possono essere visti come un insieme di semplici, reti non supervisionate come le *Restricted* *Boltzmann* *machine* [22] o *autoencoder* [24], dove ogni *hidden* *layer* della sottorete viene utilizzato come il *layer* visibile della successiva.

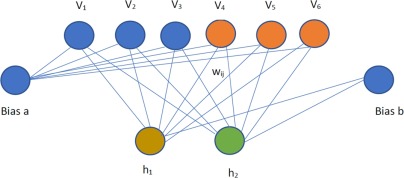
L’aver scoperto che è possibile addestrare ogni *layer* di una *DBN* utilizzando un approccio *greedy*, ha portato allo sviluppo del primo algoritmo efficace, che avesse un’architettura uguale a quella del *DBN* e che appartenesse al *Deep Learning* [25].



*Figura 12: Rappresentazione del grafico di una DBN* [26]

## 2.7 Restricted Boltzmann Machine

Un RBM è un modello generativo indiretto con un input layer visibile e un hidden layer, esso presenta connessioni tra i vari layer ma non all’interno degli stessi. Questa composizione porta ad una procedura di training non supervisionato su ogni layer, dove la *contrastive* *divergence* *(CD)* viene applicata ad ogni sotto rete a turno, partendo dal layer più basso. Il layer visibile più basso è il training set.



*Figura 13: Struttura delle RBM* [27]

Il metodo per l’allenamento delle *RBM* proposto da Geoffrey Hinton viene utilizzato assieme al modello “*Product of Expert”* nell’allenamento dei modelli *RBM*, è chiamato *contrastive divergence* [28]*.* Tale tecnica consiste nell’attuare un’approssimazione del metodo della *maximum likelihood* che idealmente verrebbe applicata per il bilanciamento dei pesi assegnati ai singoli neuroni [29] [30]. Durante la fase di *training* di una singola *RBM*, l’aggiornamento dei pesi viene effettuata tramite l’utilizzo della tecnica del *gradient descent* attraverso l’utilizzo della seguente equazione:

Immagine che contiene testo

Descrizione generata automaticamente

*Figura 14: Equazione del gradient descent utilizzato nelle RBM*

Dove *p(v)* rappresenta la probabilità di un vettore visibile, il cui valore è dato da:

Immagine che contiene testo, orologio, clipart

Descrizione generata automaticamente

*Figura 15: Formula che calcola la probabilità al vettore visibile di una RBM*

Z è la funzione di partizione, utilizzata per normalizzare, ed *E(v, h)* è la funzione di energia assegnata allo stato della rete. Valori più bassi di tale funzione di energia indicano una configurazione della rete meno “desiderabile”.

# Capitolo 3 – Progettazione

## 3.1 Strutturazione del progetto (introduzione)

La tesi si divide in tre fasi: ricerca, progettazione (slide 14), implementazione e scelte sulle feature e spiegare tutto.

* Nella fase di ricerca è stato necessario trovare dei *paper* che fornissero tutte le informazioni riguardanti gli algoritmi da utilizzare e i giusti dataset per il corretto svolgimento dello studio. Gli strumenti utilizzati per la ricerca dei *paper* sono stati due siti web molto rinomati dalla comunità scientifica, [*www.scopus.com*](http://www.scopus.com)e [*www.ieeexplore.com*](http://www.ieeexplore.com). Grazie all’utilizzo di determinate parole chiave si è stati in grado di ottenere tutta la materia prima necessaria per una comprensione più profonda dell’argomento trattato. E, inoltre, si è stati in grado di ottenere i tre *dataset* che verranno poi utilizzati per la misurazione delle *performance* dei tre algoritmi.
* Nella fase di implementazione si è dapprima capito come strutturare l’intero progetto, e poi si è proceduti con la scrittura del codice vero e proprio. Inizialmente, invece del *DBN,* si è provato ad utilizzare un altro tipo di approccio, gli *Hidden Markov Models,* che però non hanno ottenuto un’accuratezza abbastanza alta e quindi si è preferito cambiare tipo di approccio ed è stato utilizzato il *DBN.*
* Nella fase di sperimentazione sono stati dapprima raccolti i dati per creare un nuovo dataset utilizzando l’app creata dal Dipartimento di Informatica, successivamente, sono stati effettuati gli addestramenti delle singole Intelligenze Artificiali in un primo momento sui *dataset* da soli e poi ecombinandoli tra di loro per verificare l’attendibilità degli approcci. Per di più per verificare la bontà degli algoritmi si è deciso di svolgere due tipi di test, su ogni dataset, il primo di tipo I*nstance-Based Split* eil secondo *User-Based Split.* Nel primo si utilizzano parte dei dati di un solo utente per la fase di *training* e la parte restante per il *test*, nell’ultimo, invece, si utilizzano dati di più utenti per la fase di *training* e dati di utenti mai visti prima dall’algoritmo per la fase di *test.* Più avanti vedremo nello specifico come questi dati sono stati ripartiti e quali risultati hanno restituito gli approcci.

Il progetto di tesi è diviso in quattro cartelle principali, che in ordine alfabetico sono: *checkpoint, dataset, dbn\_libraries* e *models.*

Immagine che contiene testo

Descrizione generata automaticamente

*Figura 16: Albero delle cartelle e dei file appartenenti al progetto di Tesi.*

Nelle cartella *checkpoint* troviamo un *save state* di ogni algoritmo allenato. Una volta che uno dei tre modelli viene addestrato, il programma salva un’istantanea dell’intelligenza addestrata, in modo da poterla riutilizzare subito dopo senza doverla addestrare nuovamente. Il salvataggio viene separato per tipo di intelligenza addestrata, tuttavia, viene sovrascritto di volta in volta e quindi se si vuole tenere traccia dell’algoritmo addestrato per ogni dataset è necessario implementare una funzione che permetta di separare i vari stati.

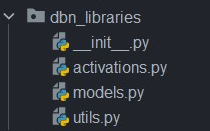
Nella cartella *dataset* vi sono contenuti tutti i dataset utilizzati per la sperimentazione. Per quanto riguarda il *KUHAR dataset,* vi sono sei sottocartelle, una per ogni tipo di attività che è stata registrata dai ricercatori. In ogni ulteriore sottocartella è possibile trovare un file contenente i dati di ogni partecipante. Stesso dicasi per *MotionSense e ISLAB.* In *UCIHAR,* invece, è possibile trovare i dati già pronti per l’elaborazione in *test e train.*

Immagine che contiene testo

Descrizione generata automaticamente

*Figura 17: Dettaglio del contenuto della cartella dataset*

In *dbn\_libraries* vi sono le librerie utilizzate per l’implementazione del *DBN*, trovate nella fase di ricerca su *GitHub* al seguente link: <https://github.com/albertbup/deep-belief-network>. È stato necessario utilizzare questo tipo di libreria perché non è presente nella versione di *tensorflow* utilizzata.



*Figura 18: Contenuto della cartella dbn\_libraries*

Nell’ultima cartella, *models*, vi sono le implementazioni di ogni Intelligenza Artificiale.

## 3.3 Progettazione della CNN

È stata utilizzata una 2D CNN in modo da sfruttare al massimo il punto di forza della CNN. Una *deep* CNN pensata per l’HAR in genere presenta un’architettura simile con la differenza che presenta molteplici *layer* convoluzionali, una trasformazione non lineare, *max pooling* e, infine, un *fully connected layer* [31]*.*

I *layer* convoluzionali utilizzano un insieme di *kernel* che generano automaticamente una *feature map*, la quale viene utilizzata come *input* dai *layer* successivi. Inoltre, sono presenti delle funzioni di approssimazione, le quali servono a semplificare le *feature map* il più possibile, come la *rectified linear unit (RELU),* la funzione sigmoidea e la *Batch Normalization (BN)* in modo da rendere l’addestramento più stabile e più veloce. La *BN* consiste nel normalizzare gli *activation vector* presi dall’*hidden layer* applicando ad essi la media e la varianza della *batch* corrente. Tale step di normalizzazione viene applicato o subito prima oppure subito dopo aver utilizzato la funzione non-lineare.

Il *layer* per il *Dropout* serve a regolarizzare l’algoritmo, in modo da evitare che il training dell’Intelligenza Artificiale sia affetto da *overfitting,* cioè che le grande quantità di dati vada a “bloccare” l’algoritmo in un massimo locale e che quindi non riesca a classificare bene*.* Successivamente, è stato messo un *Fully Connected layer*, il quale solitamente viene posto prima dell’*Output layer*, poiché combina i risultati del *layer* precedente per calcolare lo score di ognuna delle classi riconosciute.

L’O*utput layer* è responsabile della generazione dei risultati ottenuti dalla classificazione, basandosi sui dati prodotti dai *Fully Connected layer*. Prima di fare tale operazione, l’*Output layer* si occupa di applicare una *softmax function* per l’assegnazione delle previsioni effettuate dal *CNN* ad ogni classe predetta.



*Figura 19: Grafo dei layer appartenenti alla CNN*

## 3.4 Progettazione della Bi-LSTM

Anche per la *Bi-LSTM* è stato necessario utilizzare dei *Fully Connected* *layer* così da poter mettere a disposizione del modello un numero elevato di neuroni e ottenere delle prestazioni maggiori. Una rete neurale che presenta un elevato numero di neuroni è capace di comprendere correlazioni più complesse tra i dati passati e creare un modello più preciso e robusto. Tuttavia, un numero troppo elevato di neuroni può portare ad un *overfitting* del modello, in questo caso la rete ottiene risultati eccellenti sui dati di training ma non riesce a generalizzare bene sui dati e quindi si ottengono risultati scadenti nella fase di *test.* È quindi necessario trovare un equilibrio tra la complessità del modello e la complessità dei dati in *input*.

## 3.5 Progettazione dellaDBN

L’idea chiave dietro l’addestramento di una *DBN* è quella di allenare prima una sequenza di *RBM* cosicché i parametri ottenuti dall’addestramento delle due RBM influenzino sia il *probability vector* che gli *hidden states* del *DBN.*

## 3.6 Dataset utilizzati

I quattro dataset utilizzati nella sperimentazione, sono:

* UCIHAR
* KUHAR
* MotionSense
* ISLAB

I primi tre vengono utilizzati spesso durante le attività di ricerca, per quanto riguarda la sperimentazione che mira a risolvere il problema dell’HAR. Per ogni dataset viene riportato il numero di partecipanti, il tipo di sensori utilizzati, le attività svolte e in quale maniera sono state svolte, se in un ambiente simulato o in situazioni reali, il tipo di dati resi disponibili e come le attività sono state configurate prima di essere svolte.

## 3.6.1 UCIHAR

L’esperimento per la raccolta dei dati è stato svolto su un campione di 30 volontari, con età compresa tra i 19-48 anni. Ogni persona ha svolto un totale di sei attività, portando uno *smartphone* al polso.Lo smartphone utilizzato è un Samsung Galaxy S2 e le attività svolte sono le seguenti:

* Camminare;
* Salire le scale;
* Scendere le scale;
* Stare seduti;
* Stare in piedi;
* Stare stesi.

Utilizzando i sensori presenti all’interno dello *smartphone,* gli autori dell’esperimento hanno raccolto i dati relativi all’accelerazione tri-assiale e la velocità angolare nei tre assi ad una frequenza continua di campionamento pari a 50Hz. Per di più l’esperimento è stato anche registrato in modo da poter effettuare un *labeling* manuale dei dati.

I segnali dei sensori, accelerometro e giroscopio, sono stati pre-processati applicandovi dei *noise filters* e successivamente è stato effettuato un *sampling* dei dati con delle *sliding windows* di larghezza fissata pari a 2.56 secondi e con un *overlap* pari al 50% (quindi 128 letture per *window)*. I segnali provenienti dall’accelerometro, che presentano delle componenti di tipo gravitazionale e del movimento del corpo, sono state separate utilizzano di *Butterworth low-pass filter* per i parametri appartenenti all’accelerazione del corpo e a quella gravitazionale. Si assume che tale forza gravitazionale abbia solo componenti su basse frequenze. Quindi è stato applicato un ulteriore filtro che eliminasse tutti i rumori prodotti da frequenze a 0.3Hz. Per ogni *window*, viene quindi ottenuto un vettore calcolando le variabili appartenenti allo stesso dominio di tempo e frequenza.

Per ogni tabella ottenuta, gli autori forniscono:

* Accelerazione triassiale dell’accelerometro e l’accelerazione del corpo stimata;
* Velocità angolare triassiale del giroscopio;
* Un vettore contenente 561 *feature* contenenti sia la frequenza che l’orario di misurazione;
* Le *label* di ogni attività per ogni dato registrato;
* Un identificatore riguardante il soggetto che ha condotto l’esperimento.

## 3.6.2 KUHAR

Questo dataset contiene informazioni riguardanti 18 diversi tipi di attività, svolte da 90 partecipanti, divisi in 75 uomini e 15 donne, utilizzando i sensori di uno *smartphone* (accelerometro e giroscopio). Esso presenta 1945 *sample* riguardanti le attività raccolti direttamente dai partecipanti, e 20750 *subsample* estratti da loro. Le attività svolte sono:

* Rimanere in piedi;
* Rimanere seduti;
* Parlare gesticolando mentre si sta seduti;
* Parlare gesticolando mentre o si cammina o si sta in piedi;
* Alzarsi e sedersi ripetutamente;
* Stare stesi;
* Raccogliere un oggetto dal pavimento;
* Saltare ripetutamente;
* Fare squat;
* Camminare;
* Camminare all’indietro;
* Camminare in tondo;
* Correre;
* Salire le scale;
* Scendere le scale;
* Giocare a ping-pong;

Nel dataset è possibile trovare misurazioni triassiali sia dell’accelerometro che del giroscopio, entrambe con una frequenza di campionamento pari a 100Hz. Inoltre, è possibile sia trovare le misurazioni senza manipolazioni, sia le misurazioni su cui sono stati applicati dei *noise filter* per ridurre il disturbo sui segnali.

Nonostante la presenza di molte attività in questo dataset, si è deciso di prendere in considerazione solo le misurazioni che riguardano le attività:

* Stare in piedi;
* Stare seduti;
* Camminare;
* Salire le scale;
* Scendere le scale;
* Stare stesi.

## 3.6.3 MotionSense

Questo *dataset* contiene dati generati dall’accelerometro e dal giroscopio di un iPhone 6s tenuto dai partecipanti nella tasca anteriore del pantalone. È stata utilizzata *SensingKit*, un app che permette di registrare le informazioni dal *Core Morion framework* presente sui dispositivi iOS. Tutti i dati raccolti sono stati campionati con una frequenza pari a 50Hz. Per la raccolta dei dati si sono sottoposti un totale di 24 partecipanti, differenti per quanto riguarda sesso, età, peso e altezza ed ognuno di essi ha svolto 6 attività in 15 prove nello stesso ambiente e nelle stesse condizioni:

* Salire le scale;
* Scendere le scale;
* Camminare;
* Correre;
* Stare seduti;
* Stare in piedi.

## 3.6.4 ISLAB

Questo *dataset* è stato raccolto utilizzando ShieldApp, un app sviluppata dal Dipartimento di Informatica, capace di raccogliere un ampio ventaglio di dati registrati dai sensori presenti su uno smartphone. L’esperimento è stato condotto su un totale di 10 partecipanti, con un *range* di età che varia dai 18 ai 45 anni, con differenza di altezza, peso e sesso. Si è deciso di svolgere tale sperimentazione su 10 soggetti poiché in molte altre sperimentazioni trovate durante la fase di ricerca spesso si affermava che 10 partecipanti fosse il numero minimo di soggetti da registrare per avere una maggiore varietà nei dati raccolti.

Le misurazioni sono state effettuate utilizzando un Samsung Galaxy Note 9, che veniva posto nella tasca anteriore destra. Durante questa fase gli utenti svolgevano le attività e nel frattempo l’autore dell’esperimento si annotava l’orario in cui iniziava a svolgere l’attività. Tutto questo, per poi essere in grado di effettuare un *labeling* manuale dei dati. Le attività registrate sono:

* Camminare;
* Salire le scale;
* Scendere le scale;
* Stare stesi;
* Stare seduti;
* Stare in piedi.

Sui dati registrati dall’app non sono stati applicati alcun genere di filtri, poiché si voleva verificare se utilizzare dei dati privi di preelaborazione fosse possibile e quali risultati potevano restituire i vari algoritmi.

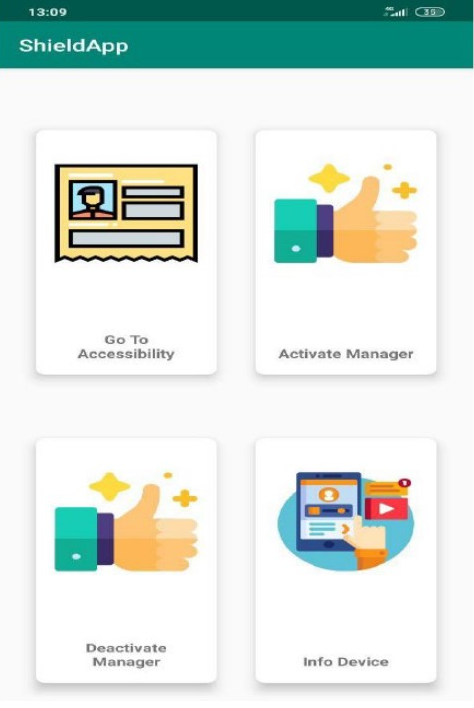
## 3.6.4.1 ShieldApp

Si tratta di un’applicazione sviluppata da un tesista del Dipartimento di Informatica, il cui scopo è quello di fornire ai genitori un mezzo tramite il quale salvaguardare i propri figli e verificare che non vi siano degli episodi di bullismo. L’app è capace di fornire i seguenti dati:

* La posizione dello smartphone tramite il GPS;
* Tutti i contatti presenti nella rubrica dello *smartphone* su cui è installata;
* Una lista di tutte le applicazioni installate sullo *smartphone;*
* Utilizza servizi che implementano un *KeyLogger* per registrare i tocchi che vengono effettuati sullo schermo e cosa si scrive;
* Utilizza i sensori sul cellulare per capire se si è caduti o l’avvenimento di fenomeni inusuali, registrando tutti gli eventi sospetti in un file .txt generato dal servizio di *KeyLogger.*

## 3.6.4.2 Come usare ShieldApp

Al primo avvio l’applicazione mostra la seguente schermata:



*Figura 20: Pagina principale di ShieldApp*

Il primo elemento che appare sullo schermo è una finestra di dialogo che contiene un messaggio di benvenuto e le istruzioni da seguire per poter attivare i servizi di *KeyLogger,* necessari per il corretto utilizzo dell’applicazione.

Una volta che la finestra di dialogo è scomparsa, lo schermo presenta quattro box differenti: uno per accedere alla sezione di accessibilità del dispositivo chiamato “Go to Accessibility”, uno per attivare l’app di Gestione del dispositivo, Active Manager, uno per disattivare il Manager di gestione del dispositivo, che permette di disinstallare l’app, e infine un box che mostra le informazioni riguardanti il dispositivo.

Dopo aver installato l’app si può decidere se renderla immediatamente non disinstallabile cliccando sul box “Active Manager”, oppure cliccare su “Go to Accessibility” per accedere alla sezione di accessibilità nelle impostazioni del dispositivo. All’interno di questa sezione è necessario trovare il servizio di *keylogger* tra i vari servizi scaricati e attivarlo. Dopo averlo attivato, una finestra apparirà e descriverà le azioni che il servizio farà.

Immagine che contiene testo

Descrizione generata automaticamente

*Figura 21: Avvertenza dei dati raccolti da ShieldApp*

Una volta che il servizio viene attivato, l’applicazione può essere chiusa; il servizio verrà eseguito in *background*, anche se il dispositivo viene spento o riavviato. Dal momento dell’attivazione e successivamente dello sblocco dello *smartphone*, il *keylogger,* sarà capace di intercettare qualsiasi tocco che avviene sullo schermo del cellulare. Tutte le informazioni catturate dal *keylogger* saranno registrate in un file di testo, che verrà successivamente mandato su un *server* per rendere il contenuto del file disponibile per l’analisi più tardi. Per quanto riguarda i sensori, tutte le registrazioni sui dati vengono fatte all’insaputa dell’utente; quindi, tutte le abitudini e i movimenti vengono costantemente registrati per mantenere un profilo sempre aggiornato per la sua salvaguardia.

I sensori che vengono utilizzati da ShieldApp sono:

* Accelerometro;
* Giroscopio;
* Magnetometro;
* Sensore di prossimità;
* Sensore di luminosità;
* GPS.

Ognuno dei valori registrati dai sensori vengono registrati in un file di testo che viene trovato sul *server* di riferimento.

## 3.7 Preparazione dei dataset

Più avanti verrà spiegato come i tre *dataset* sono stati caricati in memoria e come i dati siano stati preparati per l’addestramento delle Intelligenze Artificiali. In questa sezione, si andrà a vedere, invece, come si è deciso di combinare questi quattro *dataset.*

Per verificare, l’effettiva bontà dei tre approcci, e che quindi i risultati ottenuti non siano stato un mero caso fortuito, si è deciso di testare i tre algoritmi prima sui singoli dataset e poi sulle possibili combinazioni. In modo da avere in *input* nella fase di *training* due dataset e nella fase di *test* un terzo *dataset* differente. In questo modo, si verifica anche l’interoperabilità dei modelli su dati che sono stati elaborati in maniera differente. Come abbiamo visto in precedenza, sono stati applicati filtri diversi sui vari *dataset.* Qui di seguito vi è una tabella contenente tutte le varie combinazioni decise:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| Train | UCIHAR + KUHAR | MotionSense + KUHAR | UCIHAR + MotionSense | Dataset in cui le I.A. hanno ottenuto i risultati migliori |
| Test | MotionSense | UCIHAR | KUHAR | ISLAB |

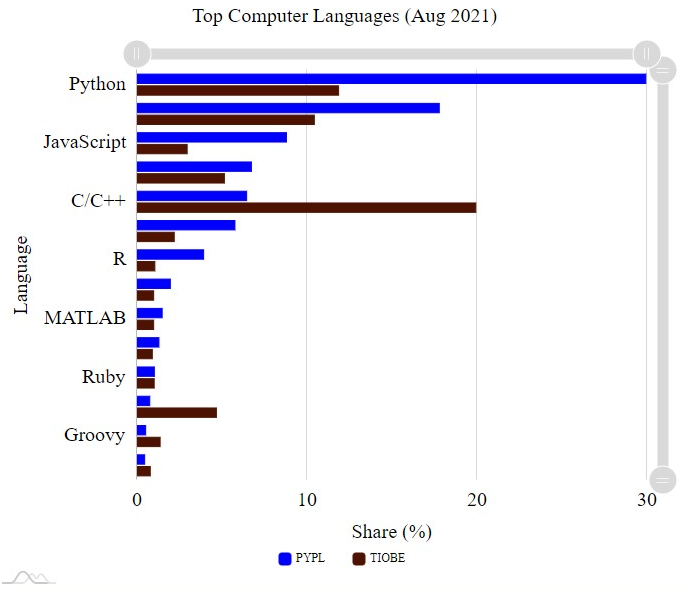
Inoltre, verranno effettuati una serie di test divisi in due categorie:

* *User-Based Split:* in questo tipo di test verranno usati i dati provenienti da diversi utenti. Ad esempio, nel caso in cui si effettua l’addestramento e il test sui singoli *dataset* si andranno ad utilizzare i dati provenienti da 5 utenti per il *training* degli algoritmi e i dati di 2 utenti per la fase di test degli stessi. Nel caso in cui, invece, si vada ad utilizzare più *dataset,* si prenderanno i dati di 5 utenti da ciascun dataset, per un totale di 10, per il training dei modelli e i dati di 4 utenti per il test.
* *Instance-Based Split:* in questa iterazione dei test, si andrà semplicemente a dividere i dati di un singolo utente in 70% per la fase di *training* e 30% per la fase di *test,* con unulteriore divisione dei dati per il *training* in 90% effettivi per l’addestramento e 10% per la *validation* dei dati.

# Capitolo 4 – Implementazione

## 4.1 Python

Se si vede tra la classifiche dei linguaggi di programmazione più utilizzati, *Python* risulta essere al primo posto, secondo la classifica *PYPL Index (Worldwide)* raggiungibile sul sito *www*.*statistictimes.com.* Tale notorietà è dovuta alla facilità di utilizzo di questo linguaggio di programmazione e ai vasti campi di applicazione dello stesso.



*Figura 22: Classifica dei linguaggi più utilizzati presenti su* [*www*.*statistictimes.com*](http://www.statistictimes.com)

*Python* è un linguaggio di programmazione dinamico orientato agli oggetti utilizzabile per molti tipi di sviluppo software. Offre un forte supporto all’integrazione con altri linguaggi e programmi, è fornito di una estesa libreria standard e di facile apprendimento. Risulta essere tra i linguaggi più utilizzati quando si parla di Intelligenze Artificiali, grazie alla marcata presenza di librerie facili da utilizzare e che implementano un’ampia gamma di algoritmi.



*Figura 23: Logo ufficiale di Python*

Al giorno d’oggi, *Python* viene utilizzato per qualsiasi tipo di sviluppo, da semplici programmi, a videogiochi fino allo sviluppo di algoritmi per il *Machine Learning.* Tale flessibilità nel campo di utilizzo è dovuta grazie alle librerie, facilmente reperibili e scritte da utenti e organizzazioni, che premettono lo sviluppo di qualsivoglia programma utilizzando poche linee di codice.

Il progetto di tesi è interamente scritto in *Python 3.9*, la decisione di tale versione è dovuta alla presenza di aggiornamenti critici per quanto riguarda l’interprete utilizzato dal linguaggio e ad alcune esigenze derivate dall’utilizzo di determinate librerie per lo sviluppo della *CNN, Bi-LSTM e DBN.*

## 4.2 PyCharm

*PyCharm* è un I*ntegrated Development Environment (IDE)* utilizzato per la programmazione in *Python*. È un programma sviluppato dalla compagnia ceca *JetBrains*. E presenta molti *tool* utili per la programmazione come: *code analysis, debugger* grafico, un’unità di test integrata e molteplici supporti per il *web development* e *data science.*



*Figura 24: Logo di PyCharm*

La versione utilizzata di *PyCharm* nel progetto è la 2021.2.2 *Community Edition,* rilasciata con la *Apache License*, che si differenzia dalla *Professional Edition* per alcune *feature* ma che risultano essere pressoché identiche nel funzionamento e nel supporto che prestano nello sviluppo. Le *feature* più importanti presenti in entrambe le versioni sono:

* Assistenza durante la fase di *Coding* e analisi del codice, con completamento automatico durante la scrittura, possibilità di evidenziare gli errori di sintassi;
* Presenza di *tool* per la facilitazione della navigazione nel progetto come: visualizzazione ad albero del progetto, possibilità di passare da un file all’altro con pochi click;
* *Refactoring* del codice facilitato: possibilità di rinominare simultaneamente file e variabili con pochi passaggi, rinominare variabili;
* *Debugger Python* integrato;
* *Unit Testing* integrato, con la possibilità di eseguire il programma linea per linea.

## 4.3 Librerie utilizzate

Le librerie utilizzate nel progetto, elencate in ordine alfabetico, sono:

* Matplotlib 3.4.2;
* Mlxtend 0.18.0;
* Numpy 1.19.5;
* Pandas 1.3.0;
* Scikit-learn 0.24.2;
* Scipy 1.7.0;
* Seaborn 0.11.1;
* Tensorflow 2.5.0.

## 4.3.1 Matplotlib

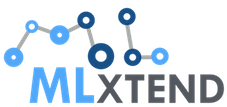
Matplotlib è una libreria *Python* utile per creare visualizzazioni grafiche statiche, animate e interattive. Essa offre, utilizzando poche linee di codice, la possibilità di realizzare dei grafici. Inoltre, fornisce delle API orientate agli oggetti che permettono di inserire grafici all’interno di applicativi utilizzando dei *toolkit* *GUI* generici, come *Qt* o *GTK.* È possibile utilizzare anche un’interfaccia grafica per la creazione dei grafici, chiamata *pylab*, basata su OpenGL progettata appositamente per assomigliare a quella di MATLAB.



*Figura 25: Logo di Matplotlib*

## 4.3.2 Mlxtend

È una libreria che contiene dei *tool* utili per l’adempimento di task che riguardano la *data analysis* e il *machine learning.*



*Figura 26: Logo di mlxtend*

## 4.3.3 NumPy

NumPy è una libreria *open source* che fornisce supporto per la creazione di matrici e array multidimensionali, insieme a una vasta collezione di funzioni matematiche di alto livello per poter operare efficientemente su queste strutture dati.

Esso si rivolge all’implementazione di riferimento *CPython* di *Python*, che è un interprete di *bytecode* non ottimizzante. Gli algoritmi matematici scritti per questa versione di *Python* spesso vengono eseguiti molto più lentamente degli equivalenti compilati. *NumPy* affronta il problema della lentezza in parte, fornendo *array* multidimensionali, funzioni e operatori che effettuano azioni in maniera efficiente sugli array. L’utilizzo di questi richiede la riscrittura del codice, principalmente i *loop* interni, utilizzando *NumPy.*



*Figura 27: Logo di Numpy*

## 4.3.4 Pandas

Pandas è una libreria *software* scritta per la manipolazione e l’analisi dei dati. In particolare, offre strutture dati e operazioni per manipolare tabelle numeriche e serie di dati temporali. È un software libero rilasciato sotto la licenza BSD. Il nome deriva dal termine “*panel* *data*”, termine econometrico per set di dati che include osservazioni su più periodi di tempo per gli stessi elementi.



*Figura 28: Logo di Pandas*

## 4.3.5 Scikit-learn

*Scikit-learn* è una libreria *open source* di apprendimento automatico. Contiene algoritmi di classificazione, regressione, *clustering* e *support vector machine*, regressione logistica, classificatore bayesiano, *k-mean* e *DBSCAN*. È stato progettato per operare con le librerie *NumPy e SciPy*.



*Figura 29: Logo di Scikit-learn*

## 4.3.6 Scipy

È una libreria *open source* di algoritmi e strumenti matematici per *Python.* Contiene moduli per l’ottimizzazione, l’algebra lineare, l’integrazione, funzioni speciali, *FTT*, elaborazione di segnali ed immagini. *SciPy* si trova sotto licenza *BSD*.

Immagine che contiene testo, luce

Descrizione generata automaticamente

*Figura 30: Logo di Scipy*

## 4.3.7 Seaborn

Si tratta di una libreria per la *data visualization* basata su *matplotlib*. Fornisce interfacce grafiche ad alto livello per disegnare grafici statistici accattivanti.



*Figura 31: Logo di Seaborn*

## 4.3.8 Tensorflow

È una libreria *open source* di supporto per l’implementazione di algoritmi di apprendimento automatico. Fornisce moduli sperimentali e ottimizzati, utili alla realizzazione di algoritmi di *machine* learning.



*Figura 32: Logo di Tensorflow*

## 4.4 Funzioni utilizzate per la preparazione dei dati

Prima di poter procedere con l’implementazione vera e propria delle tre Intelligenze Artificiali, è stato necessario effettuare delle operazioni di preparazione dei dataset. Inizialmente è stato necessario capire come caricare in memoria i dati presenti nei dataset, operazione conclusa con successo grazie all’ausilio di *pandas.* Sono state create delle funzioni ad hoc che permettessero di elaborare e modellare i dati nel formato più adeguato.

Le tabelle sono state create istanziando una classe di *pandas* chiamata *DataFrame.* Tale classe permette di creare una struttura dati simile a quella delle tabelle utilizzate in MySQL, con la possibilità di utilizzare operatori sia algebrici (somma, sottrazione, elevamento a potenza, ecc…) per la manipolazione dei dati, sia operatori di tipo logico per spostare, mostrare o eliminare dati con determinate caratteristiche. Inoltre, è anche possibile utilizzare operatori di *join* per poter unire i dati di due o più tabelle.

Inizialmente, le tabelle contenevano soltanto le misurazioni lungo i tre assi spaziali dei due sensori, accelerometro e giroscopio. Dopo una serie di esperimenti con i vari approcci, in particolare per la *DBN*, è stato necessario fare una serie di prove in cui si utilizzavano diversi tipi di *feature*. Nelle varie prove effettuate, si è notato che alcuni valori adottati, non erano molto utili nell’incremento della precisione e quindi è stato necessario eliminarne e aggiungerne altri. Alla fine si è giunti ad un totale di undici *feature*:

* velocità lungo l’asse X;
* velocità lungo l’asse Y;
* velocità lungo l’asse Z;
* magnitudine della velocità lungo i tre assi;
* accelerazione angolare sull’asse X;
* accelerazione angolare sull’asse Y;
* accelerazione angolare sull’asse Z;
* magnitudine dell’accelerazione angolare lungo i tre assi;
* angolo che si forma tra il vettore velocità lungo asse X e l’asse X stesso;
* angolo che si forma tra il vettore velocità lungo asse Y e l’asse Y stesso;
* angolo che si forma tra il vettore velocità lungo asse Z e l’asse Z stesso.

## 4.4.1 Misurazioni lungo gli assi X, Y, Z

Per quanto riguarda la velocità lungo gli assi X, Y e Z, queste *feature* si riferiscono alle misurazioni effettuate dall’accelerometro triassiale. Semplicemente, le misurazioni di ogni asse vengono inserite nella colonna corrispondente.

Lo stesso vale per le misurazioni dell’accelerazione angolare lungo i tre assi, con la differenza che queste *feature* si riferiscono alle misurazioni effettuate dal giroscopio.

Sono le *feature* necessarie ai modelli per poter classificare, si potrebbero utilizzare solo queste, ma per rendere il *DBN* al passo con il *CNN* e il *Bi-LSTM* è necessario introdurre altre *feature.*

## 4.4.2 Magnitudine

Nella preparazione dei dati sono state calcolate due magnitudini, una per ogni sensore utilizzato. È stato deciso di utilizzarla poiché durante la fase di ricerca si è letto in molti studi che l’utilizzo e l’inserimento di tale *feature* nella fase di addestramento portava a notevoli miglioramenti nella capacità del modello di classificare. Per calcolare la magnitudine si utilizza la seguente formula:

Dove *x, y, z* sono i valori della velocità o accelerazione angolare misurata lungo i rispettivi assi.

L’idea dietro tale *feature* è quella di ottenere un’unica variabile sensibile ai cambiamenti che avvengono lungo tutti i tre assi contemporaneamente. Nella raccolta dei dati per i *dataset* potrebbero esserci alcune attività che si assomigliano per valori lungo uno o due assi, e quindi potrebbero venire confuse e classificate alla stessa maniera dall’algoritmo. Invece, avendo la magnitudine si ha costantemente contezza delle variazioni sui tre assi con un’unica variabile.

Utilizzare questa *feature* ha portato ad un incremento di circa il 15% nella precisione ottenuta dal *DBN.*

## 4.4.3 Angoli tra le misurazioni lungo l’asse e l’asse stesso

Questa *feature* è necessaria per poter rendere più comprensibile all’algoritmo, cosa realmente sta succedendo all’utente. In particolare, avere a disposizione tale angolo permette di capire meglio la direzione verso la quale ci si sta movendo. La formula per calcolare tale *feature* è una semplice formula trigonometrica:

sin-1

in cui asse [x, y, z] prende il valore della velocità misurata dall’accelerometro lungo un singolo asse e al denominatore viene calcolata la radice quadrata della somma dei quadrati del valore misurato dall’accelerometro lungo lo stesso asse presente al numeratore e del valore misurato dall’accelerometro lungo un asse adiacente.

# Capitolo 5 – Sperimentazione

## 5.1 Metriche utilizzate per la valutazione

Prima di procedere con i risultati della sperimentazione, è necessario spiegare quali sono i parametri per la valutazione che sono stati utilizzati in questo lavoro. La prima metrica utilizzata è la *Classification Accuracy*, che corrisponde al rapporto tra il numero di predizioni corrette e il numero totale di valori passati in *input.*

Tuttavia, l’utilizzo di tale metrica ha senso solo se nel dati passati in *input* è presente un numero uguale di elementi appartenenti ad ogni classe. Se per esempio si ha che il 90% dei dati appartengono alla classe A e il restante 10% appartiene alla classe B, il modello in esame potrà ottenere tranquillamente un’accuratezza pari al 90% semplicemente classificando tutti i dati che sta esaminando come appartenenti alla classe A.

Successivamente, sono state utilizzate la P*recision* e il *Recall.* La Precisione corrisponde al numero di veri positivi, diviso per il numero di valori che sono stati classificati come appartenenti a quella classe dal modello.

Il *Recall*, invece, corrisponde al numero di positivi reali diviso per il numero di tutti i valori che appartengono a quella classe.

Inoltre, utilizzando queste due metriche, è stato possibile calcolare un’ulteriore metrica, l’*F1-Score.* L’*F1-Score* non è altro che una media armonica tra la *Precision* e il *Recall.* Solitamente l’*F1-Score* assume valori compresi tra [0, 1]. Essa serve per comprendere quanto preciso sia il classificatore, quante istanze sia in grado di classificare correttamente, così come quanto sia robusto, in altre parole, indica se tende a sbagliare un numero significativo di classificazioni. Avere una precisione molto alta ma un *recall* molto basso, ci fa capire che il classificatore sia estremamente preciso, ma che tende a sbagliare un grande numero di classificazioni che siano difficili effettuare. Più l’*F1-Score* si avvicina ad 1, meglio sono le prestazioni del nostro modello.

## 5.2 Impostazione della sperimentazione

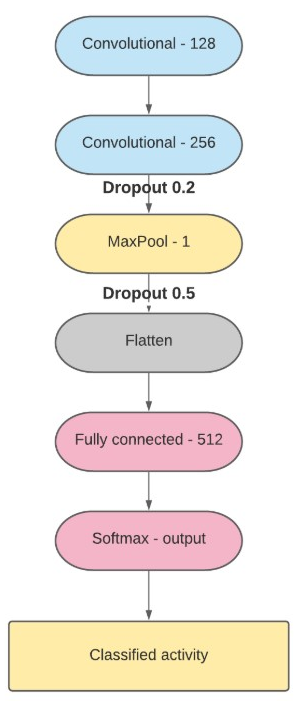
Come detto in precedenza, gli esperimenti sui vari dataset vengono divisi in due parti: User-Based Split e Instance-Based Split, e per entrambi i test verranno condotte sperimentazioni combinando in modi differenti i vari dataset:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| Train | UCIHAR + KUHAR | MotionSense + KUHAR | UCIHAR + MotionSense | UCIHAR + MotionSense |
| Test | MotionSense | UCIHAR | KUHAR | ISLAB |

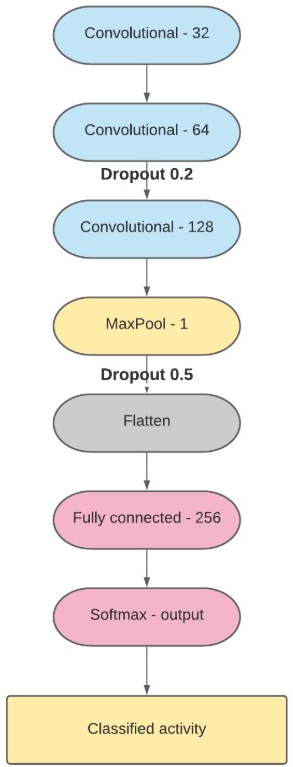
Nella quarta colonna è stato anticipato che i *dataset* in cui i modelli hanno performato meglio è stata proprio la combinazione tra *UCIHAR* e *MotionSense.*

## 5.2 CNN

Il modello è stato allenato utilizzando un ottimizzatore *Adam* con un valore pari a 0.001 per 100 *epoch*, con un *Batch size* pari a 64. Ogni *epoch* impiegava circa cinque minuti per completare il processo di *training*. Il tempo impiegato per la conclusione dell’addestramento dipende dal tipo di *hardware* utilizzato, se si addestrano i modelli utilizzando la scheda video anziché il processore, più altri fattori casuali dovuti al *seed* utilizzato nel programma.



*Figura 33: Struttura della CNN utilizzata nel test Instance-Based Split*

**

*Figura 34: Struttura della CNN utilizzata nel test User-Based Split*

## 5.2.1 Risultati ottenuti nella fase Instance-Based

Tutti i dati che verranno ritrovati di seguito si riferiscono solo alla tipologia di *training* di tipo *Instance-Based.*

## 5.2.1.1 UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo *l’UCIHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9861 | 0.9852 | 0.9623 | 0.9769 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.98 | 0.97 |
| Salire le scale | 1 | 0.95 |
| Scendere le scale | 1 | 0.95 |
| Stare seduti | 0.99 | 0.98 |
| Stare in piedi | 0.96 | 0.99 |
| Stare stesi | 0.95 | 0.94 |

## 5.2.1.2 KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo il *KUHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9662 | 0.9645 | 0.9525 | 0.9608 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.94 | 0.95 |
| Salire le scale | 0.95 | 0.99 |
| Scendere le scale | 0.97 | 0.92 |
| Stare seduti | 0.98 | 0.97 |
| Stare in piedi | 0.93 | 0.98 |
| Stare stesi | 0.99 | 0.94 |

## 5.2.1.3 MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo il *MotionSense* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9262 | 0.9463 | 0.9243 | 0.9302 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.92 | 0.91 |
| Salire le scale | 0.94 | 0.94 |
| Scendere le scale | 0.95 | 0.90 |
| Stare seduti | 0.93 | 0.91 |
| Stare in piedi | 0.96 | 0.95 |
| Stare stesi | 0.99 | 0.94 |

## 5.2.1.4 ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo l’*ISLAB* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9662 | 0.9672 | 0.9543 | 0.9609 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.99 | 0.95 |
| Salire le scale | 0.95 | 0.97 |
| Scendere le scale | 0.96 | 0.95 |
| Stare seduti | 0.94 | 0.96 |
| Stare in piedi | 0.95 | 0.93 |
| Stare stesi | 0.98 | 0.96 |

## 5.2.1.5 Train: UCIHAR + KUHAR Test: MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *UCIHAR + KUHAR* per la fase di *training* del modello e *MotionSense* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9936 | 0.9599 | 0.9387 | 0.9433 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.97 | 0.90 |
| Salire le scale | 0.96 | 0.94 |
| Scendere le scale | 0.93 | 0.95 |
| Stare seduti | 0.98 | 0.93 |
| Stare in piedi | 0.95 | 0.92 |
| Stare stesi | 0.96 | 0.94 |

## 5.2.1.6 Train: MotionSense + KUHAR Test: UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *MotionSense + KUHAR* per la fase di *training* del modello e *UCIHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9654 | 0.9734 | 0.9874 | 0.9821 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.98 | 0.99 |
| Salire le scale | 0.99 | 0.97 |
| Scendere le scale | 0.96 | 0.98 |
| Stare seduti | 0.99 | 0.96 |
| Stare in piedi | 0.95 | 0.98 |
| Stare stesi | 0.93 | 0.99 |

## 5.2.1.7 Train: UCIHAR + MotionSense Test: KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *KUHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9723 | 0.9612 | 0.9589 | 0.9612 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.97 | 0.96 |
| Salire le scale | 0.99 | 0.98 |
| Scendere le scale | 0.93 | 0.93 |
| Stare seduti | 0.95 | 0.94 |
| Stare in piedi | 0.96 | 0.92 |
| Stare stesi | 0.97 | 0.99 |

## 5.2.1.7 Train: UCIHAR + MotionSense Test: ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *ISLAB* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9723 | 0.9821 | 0.9738 | 0.9789 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.99 | 0.97 |
| Salire le scale | 0.97 | 0.98 |
| Scendere le scale | 0.96 | 0.96 |
| Stare seduti | 0.99 | 0.94 |
| Stare in piedi | 0.97 | 0.99 |
| Stare stesi | 0.95 | 0.96 |

## 5.2.2 Risultati ottenuti nella fase User-Based

Tutti i dati che verranno ritrovati di seguito si riferiscono solo alla tipologia di *training* di tipo *User-Based.*

## 5.2.2.1 UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo *l’UCIHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9663 | 0.9673 | 0.9542 | 0.9609 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.96 | 0.92 |
| Salire le scale | 0.99 | 0.96 |
| Scendere le scale | 0.93 | 0.97 |
| Stare seduti | 0.96 | 0.94 |
| Stare in piedi | 0.97 | 0.93 |
| Stare stesi | 0.94 | 0.95 |

## 5.2.2.2 KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo il *KUHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9812 | 0.9772 | 0.9989 | 0.9879 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.98 | 0.99 |
| Salire le scale | 0.99 | 1 |
| Scendere le scale | 0.95 | 0.98 |
| Stare seduti | 0.96 | 0.99 |
| Stare in piedi | 0.97 | 1 |
| Stare stesi | 0.98 | 1 |

## 5.2.2.3 MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo il *MotionSense* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9090 | 0.9234 | 0.9033 | 0.9132 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.91 | 0.89 |
| Salire le scale | 0.93 | 0.90 |
| Scendere le scale | 0.92 | 0.91 |
| Stare seduti | 0.94 | 0.88 |
| Stare in piedi | 0.90 | 0.92 |
| Stare stesi | 0.92 | 0.89 |

## 5.2.2.4 ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando solo l’*ISLAB* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9882 | 0.9763 | 0.9664 | 0.9713 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.99 | 0.95 |
| Salire le scale | 0.98 | 0.96 |
| Scendere le scale | 0.97 | 0.96 |
| Stare seduti | 0.96 | 0.97 |
| Stare in piedi | 0.97 | 0.94 |
| Stare stesi | 0.97 | 0.95 |

## 5.2.2.5 Train: UCIHAR + KUHAR Test: MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *UCIHAR + KUHAR* per la fase di *training* del modello e *MotionSense* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9888 | 0.9662 | 0.9290 | 0.9822 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.95 | 0.91 |
| Salire le scale | 0.96 | 0.93 |
| Scendere le scale | 0.97 | 0.93 |
| Stare seduti | 0.94 | 0.92 |
| Stare in piedi | 0.99 | 0.94 |
| Stare stesi | 0.93 | 0.91 |

## 5.2.2.6 Train: MotionSense + KUHAR Test: UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *MotionSense + KUHAR* per la fase di *training* del modello e *UCIHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9998 | 0.9832 | 0.9735 | 0.9783 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.99 | 0.97 |
| Salire le scale | 0.98 | 0.96 |
| Scendere le scale | 0.98 | 0.99 |
| Stare seduti | 0.97 | 0.96 |
| Stare in piedi | 1 | 0.98 |
| Stare stesi | 0.99 | 0.97 |

## 5.2.2.7 Train: UCIHAR + MotionSense Test: KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *KUHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9882 | 0.9763 | 0.9664 | 0.9713 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.98 | 0.97 |
| Salire le scale | 0.99 | 0.96 |
| Scendere le scale | 0.96 | 0.95 |
| Stare seduti | 0.95 | 0.93 |
| Stare in piedi | 0.97 | 0.99 |
| Stare stesi | 0.97 | 0.96 |

## 5.2.2.7 Train: UCIHAR + MotionSense Test: ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *CNN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *ISLAB* per i test sull’addestramento.

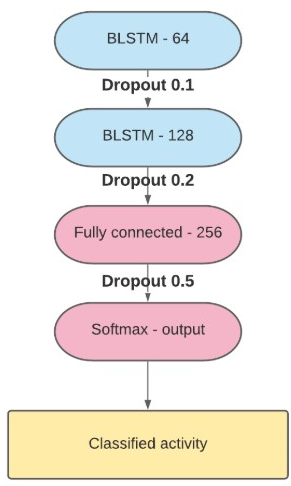
|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9843 | 0.9723 | 0.9883 | 0.9853 |

La *precision e recall* sono riportati per ogni classe:

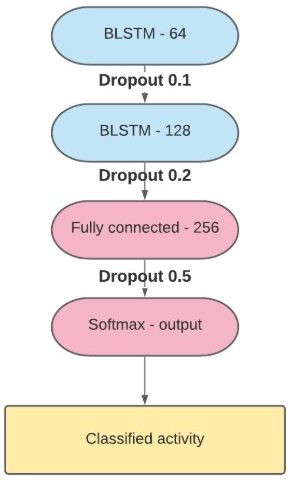
|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.98 | 0.99 |
| Salire le scale | 0.96 | 0.98 |
| Scendere le scale | 0.98 | 0.97 |
| Stare seduti | 0.99 | 0.97 |
| Stare in piedi | 0.96 | 1 |
| Stare stesi | 0.95 | 0.99 |

## 5.3 Bi-LSTM

Anche in questo caso, il modello è stato allenato utilizzando un ottimizzatore *Adam* con un valore pari a 0.001 per 100 *epoch*, con un *Batch size* pari a 16. Ogni *epoch* impiegava circa tre minuti per completare il processo di *training*.



*Figura 35: Struttura della Bi-LSTM utilizzata nel test Instance-Based Split*

**

*Figura 36: Struttura della Bi-LSTM utilizzata nel test User-Based Split*

## 5.3.1 Risultati ottenuti nella fase Instance-Based

Tutti i dati che verranno ritrovati di seguito si riferiscono solo alla tipologia di *training* di tipo *Instance-Based.*

## 5.3.1.1 UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo *l’UCIHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9182 | 0.9321 | 0.9217 | 0.9254 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.93 | 0.92 |
| Salire le scale | 0.94 | 0.94 |
| Scendere le scale | 0.95 | 0.93 |
| Stare seduti | 0.92 | 0.90 |
| Stare in piedi | 0.93 | 0.91 |
| Stare stesi | 0.93 | 0.92 |

## 5.3.1.2 KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo il *KUHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.8992 | 0.8932 | 0.9143 | 0.9036 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.88 | 0.91 |
| Salire le scale | 0.89 | 0.89 |
| Scendere le scale | 0.90 | 0.93 |
| Stare seduti | 0.88 | 0.90 |
| Stare in piedi | 0.87 | 0.92 |
| Stare stesi | 0.89 | 0.91 |

## 5.3.1.3 MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo il *MotionSense* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.8888 | 0.8512 | 0.8386 | 0.8445 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.86 | 0.83 |
| Salire le scale | 0.87 | 0.85 |
| Scendere le scale | 0.85 | 0.86 |
| Stare seduti | 0.87 | 0.79 |
| Stare in piedi | 0.84 | 0.84 |
| Stare stesi | 0.85 | 0.83 |

## 5.3.1.4 ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo l’*ISLAB* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9065 | 0.9288 | 0.9364 | 0.9309 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.93 | 0.93 |
| Salire le scale | 0.91 | 0.94 |
| Scendere le scale | 0.94 | 0.91 |
| Stare seduti | 0.93 | 0.93 |
| Stare in piedi | 0.95 | 0.96 |
| Stare stesi | 0.92 | 0.90 |

## 5.3.1.5 Train: UCIHAR + KUHAR Test: MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *UCIHAR + KUHAR* per la fase di *training* del modello e *MotionSense* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.8999 | 0.9245 | 0.9000 | 0.9189 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.92 | 0.89 |
| Salire le scale | 0.90 | 0.90 |
| Scendere le scale | 0.87 | 0.87 |
| Stare seduti | 0.93 | 0.92 |
| Stare in piedi | 0.91 | 0.93 |
| Stare stesi | 0.92 | 0.87 |

## 5.3.1.6 Train: MotionSense + KUHAR Test: UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *MotionSense + KUHAR* per la fase di *training* del modello e *UCIHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9693 | 0.9232 | 0.9012 | 0.9121 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.92 | 0.90 |
| Salire le scale | 0.93 | 0.89 |
| Scendere le scale | 0.94 | 0.87 |
| Stare seduti | 0.93 | 0.93 |
| Stare in piedi | 0.90 | 0.91 |
| Stare stesi | 0.92 | 0.89 |

## 5.3.1.7 Train: UCIHAR + MotionSense Test: KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *KUHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9076 | 0.9232 | 0.9143 | 0.9036 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.92 | 0.91 |
| Salire le scale | 0.95 | 0.90 |
| Scendere le scale | 0.89 | 0.89 |
| Stare seduti | 0.93 | 0.93 |
| Stare in piedi | 0.91 | 0.91 |
| Stare stesi | 0.92 | 0.90 |

## 5.3.1.7 Train: UCIHAR + MotionSense Test: ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *ISLAB* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9332 | 0.9412 | 0.9233 | 0.9399 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.95 | 0.93 |
| Salire le scale | 0.96 | 0.92 |
| Scendere le scale | 0.94 | 0.92 |
| Stare seduti | 0.95 | 0.94 |
| Stare in piedi | 0.92 | 0.91 |
| Stare stesi | 0.94 | 0.93 |

## 5.3.2 Risultati ottenuti nella fase User-Based

Tutti i dati che verranno ritrovati di seguito si riferiscono solo alla tipologia di *training* di tipo *Instance-Based*

## 5.3.2.1 UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo *l’UCIHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.8992 | 0.8932 | 0.9143 | 0.9036 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.88 | 0.91 |
| Salire le scale | 0.89 | 0.94 |
| Scendere le scale | 0.90 | 0.90 |
| Stare seduti | 0.91 | 0.91 |
| Stare in piedi | 0.92 | 0.93 |
| Stare stesi | 0.87 | 0.92 |

## 5.3.2.2 KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo il *KUHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9120 | 0.9465 | 0.8930 | 0.9189 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.95 | 0.89 |
| Salire le scale | 0.95 | 0.91 |
| Scendere le scale | 0.92 | 0.87 |
| Stare seduti | 0.93 | 0.90 |
| Stare in piedi | 0.94 | 0.89 |
| Stare stesi | 0.93 | 0.90 |

## 5.3.2.3 MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo il *MotionSense* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.8798 | 0.8680 | 0.8239 | 0.8406 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.87 | 0.80 |
| Salire le scale | 0.89 | 0.84 |
| Scendere le scale | 0.82 | 0.82 |
| Stare seduti | 0.83 | 0.84 |
| Stare in piedi | 0.90 | 0.83 |
| Stare stesi | 0.80 | 0.81 |

## 5.3.2.4 ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando solo l’*ISLAB* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9132 | 0.9367 | 0.9442 | 0.9397 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.93 | 0.94 |
| Salire le scale | 0.92 | 0.92 |
| Scendere le scale | 0.93 | 0.91 |
| Stare seduti | 0.94 | 0.96 |
| Stare in piedi | 0.91 | 0.95 |
| Stare stesi | 0.94 | 0.94 |

## 5.3.2.5 Train: UCIHAR + KUHAR Test: MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *UCIHAR + KUHAR* per la fase di *training* del modello e *MotionSense* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.8990 | 0.9367 | 0.8720 | 0.9436 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.94 | 0.89 |
| Salire le scale | 0.94 | 0.86 |
| Scendere le scale | 0.93 | 0.90 |
| Stare seduti | 0.95 | 0.88 |
| Stare in piedi | 0.92 | 0.87 |
| Stare stesi | 0.94 | 0.89 |

## 5.3.2.6 Train: MotionSense + KUHAR Test: UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *MotionSense + KUHAR* per la fase di *training* del modello e *UCIHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9573 | 0.9122 | 0.8949 | 0.9034 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.91 | 0.93 |
| Salire le scale | 0.92 | 0.92 |
| Scendere le scale | 0.93 | 0.80 |
| Stare seduti | 0.90 | 0.87 |
| Stare in piedi | 0.92 | 0.90 |
| Stare stesi | 0.91 | 0.89 |

## 5.3.2.7 Train: UCIHAR + MotionSense Test: KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *KUHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9132 | 0.9367 | 0.9442 | 0.9397 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.93 | 0.94 |
| Salire le scale | 0.94 | 0.94 |
| Scendere le scale | 0.92 | 0.96 |
| Stare seduti | 0.95 | 0.92 |
| Stare in piedi | 0.96 | 0.95 |
| Stare stesi | 0.90 | 0.94 |

## 5.3.2.7 Train: UCIHAR + MotionSense Test: ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *Bi-LSTM* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *ISLAB* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9452 | 0.9643 | 0.9333 | 0.9546 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.99 | 0.94 |
| Salire le scale | 0.97 | 0.92 |
| Scendere le scale | 0.96 | 0.93 |
| Stare seduti | 0.95 | 0.95 |
| Stare in piedi | 0.95 | 0.91 |
| Stare stesi | 0.96 | 0.95 |

## 5.4 DBN

Nel nostro caso si è deciso di allenare gli RBM con dieci *epoch* la cuidurata media era all’incirca di due minuti per ogni *epoch.* Successivamente,

## 5.4.1 Risultati ottenuti nella fase Instance-Based

Tutti i dati che verranno ritrovati di seguito si riferiscono solo alla tipologia di *training* di tipo *Instance-Based*

## 5.4.1.1 UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo *l’UCIHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9326 | 0.8065 | 0.7942 | 0.7995 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.80 | 0.79 |
| Salire le scale | 0.82 | 0.81 |
| Scendere le scale | 0.79 | 0.78 |
| Stare seduti | 0.79 | 0.79 |
| Stare in piedi | 0.81 | 0.80 |
| Stare stesi | 0.82 | 0.79 |

## 5.4.1.2 KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo il *KUHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9233 | 0.8045 | 0.7912 | 0.8051 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.80 | 0.78 |
| Salire le scale | 0.81 | 0.81 |
| Scendere le scale | 0.82 | 0.80 |
| Stare seduti | 0.79 | 0.79 |
| Stare in piedi | 0.79 | 0.79 |
| Stare stesi | 0.80 | 0.82 |

## 5.4.1.3 MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo il *MotionSense* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9866 | 0.8555 | 0.8395 | 0.8472 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.85 | 0.81 |
| Salire le scale | 0.86 | 0.80 |
| Scendere le scale | 0.84 | 0.85 |
| Stare seduti | 0.85 | 0.82 |
| Stare in piedi | 0.88 | 0.84 |
| Stare stesi | 0.82 | 0.83 |

## 5.4.1.4 ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo l’*ISLAB* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9647 | 0.8598 | 0.8423 | 0.8475 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.86 | 0.84 |
| Salire le scale | 0.87 | 0.83 |
| Scendere le scale | 0.85 | 0.81 |
| Stare seduti | 0.86 | 0.88 |
| Stare in piedi | 0.89 | 0.85 |
| Stare stesi | 0.84 | 0.83 |

## 5.4.1.5 Train: UCIHAR + KUHAR Test: MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando *UCIHAR + KUHAR* per la fase di *training* del modello e *MotionSense* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9222 | 0.8399 | 0.8298 | 0.8345 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.83 | 0.82 |
| Salire le scale | 0.82 | 0.84 |
| Scendere le scale | 0.84 | 0.83 |
| Stare seduti | 0.85 | 0.84 |
| Stare in piedi | 0.82 | 0.82 |
| Stare stesi | 0.82 | 0.82 |

## 5.4.1.6 Train: MotionSense + KUHAR Test: UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando *MotionSense + KUHAR* per la fase di *training* del modello e *UCIHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9444 | 0.8297 | 0.7999 | 0.8201 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.83 | 0.79 |
| Salire le scale | 0.84 | 0.81 |
| Scendere le scale | 0.85 | 0.80 |
| Stare seduti | 0.82 | 0.82 |
| Stare in piedi | 0.84 | 0.83 |
| Stare stesi | 0.83 | 0.80 |

## 5.4.1.7 Train: UCIHAR + MotionSense Test: KUHAR

Di seguito riportati vi sono i risultati ottenuti *DBN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *KUHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9666 | 0.8954 | 0.8829 | 0.8901 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.89 | 0.89 |
| Salire le scale | 0.90 | 0.90 |
| Scendere le scale | 0.91 | 0.91 |
| Stare seduti | 0.87 | 0.87 |
| Stare in piedi | 0.88 | 0.88 |
| Stare stesi | 0.89 | 0.89 |

## 5.4.1.7 Train: UCIHAR + MotionSense Test: ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *ISLAB* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9732 | 0.9120 | 0.9075 | 0.9090 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.92 | 0.88 |
| Salire le scale | 0.91 | 0.91 |
| Scendere le scale | 0.90 | 0.90 |
| Stare seduti | 0.93 | 0.89 |
| Stare in piedi | 0.95 | 0.90 |
| Stare stesi | 0.90 | 0.91 |

## 5.4.2 Risultati ottenuti nella fase User-Based

Tutti i dati che verranno ritrovati di seguito si riferiscono solo alla tipologia di *training* di tipo *Instance-Based.*

## 5.4.2.1 UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo *l’UCIHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9066 | 0.7855 | 0.7795 | 0.7775 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.78 | 0.77 |
| Salire le scale | 0.79 | 0.76 |
| Scendere le scale | 0.80 | 0.80 |
| Stare seduti | 0.81 | 0.82 |
| Stare in piedi | 0.83 | 0.82 |
| Stare stesi | 0.79 | 0.83 |

## 5.4.2.2 KUHAR

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo il *KUHAR* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9442 | 0.8429 | 0.8141 | 0.8163 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.85 | 0.81 |
| Salire le scale | 0.86 | 0.82 |
| Scendere le scale | 0.84 | 0.79 |
| Stare seduti | 0.85 | 0.80 |
| Stare in piedi | 0.85 | 0.81 |
| Stare stesi | 0.86 | 0.80 |

## 5.4.2.3 MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo il *MotionSense* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9745 | 0.8732 | 0.8345 | 0.8365 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.88 | 0.83 |
| Salire le scale | 0.89 | 0.82 |
| Scendere le scale | 0.86 | 0.84 |
| Stare seduti | 0.87 | 0.83 |
| Stare in piedi | 0.88 | 0.85 |
| Stare stesi | 0.87 | 0.84 |

## 5.4.2.4 ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando solo l’*ISLAB* come *dataset.*

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9510 | 0.8627 | 0.8551 | 0.8589 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.86 | 0.85 |
| Salire le scale | 0.87 | 0.86 |
| Scendere le scale | 0.85 | 0.84 |
| Stare seduti | 0.84 | 0.86 |
| Stare in piedi | 0.87 | 0.85 |
| Stare stesi | 0.86 | 0.85 |

## 5.4.2.5 Train: UCIHAR + KUHAR Test: MotionSense

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando *UCIHAR + KUHAR* per la fase di *training* del modello e *MotionSense* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9390 | 0.8294 | 0.8278 | 0.8283 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.83 | 0.81 |
| Salire le scale | 0.83 | 0.83 |
| Scendere le scale | 0.85 | 0.82 |
| Stare seduti | 0.84 | 0.80 |
| Stare in piedi | 0.84 | 0.82 |
| Stare stesi | 0.80 | 0.83 |

## 5.4.2.6 Train: MotionSense + KUHAR Test: UCIHAR

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando *MotionSense + KUHAR* per la fase di *training* del modello e *UCIHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9332 | 0.8414 | 0.8249 | 0.8331 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.85 | 0.83 |
| Salire le scale | 0.86 | 0.84 |
| Scendere le scale | 0.84 | 0.82 |
| Stare seduti | 0.85 | 0.83 |
| Stare in piedi | 0.86 | 0.84 |
| Stare stesi | 0.87 | 0.83 |

## 5.4.2.7 Train: UCIHAR + MotionSense Test: KUHAR

Di seguito riportati vi sono i risultati ottenuti *DBN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *KUHAR* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9510 | 0.8627 | 0.8551 | 0.8589 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.86 | 0.85 |
| Salire le scale | 0.87 | 0.86 |
| Scendere le scale | 0.86 | 0.85 |
| Stare seduti | 0.89 | 0.87 |
| Stare in piedi | 0.84 | 0.84 |
| Stare stesi | 0.88 | 0.86 |

## 5.4.2.7 Train: UCIHAR + MotionSense Test: ISLAB

Di seguito riportati vi sono i risultati ottenuti dal *DBN* durante l’addestramento e il test utilizzando *UCIHAR* + *MotionSense* per la fase di *training* del modello e *ISLAB* per i test sull’addestramento.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F1 Score |
| 0.9653 | 0.9354 | 0.9435 | 0.9399 |

La *precision e recall* sono riportati per ogni classe:

|  |  |  |
| --- | --- | --- |
| Attività | *Precision* | *Recall* |
| Camminare | 0.94 | 0.95 |
| Salire le scale | 0.95 | 0.95 |
| Scendere le scale | 0.92 | 0.96 |
| Stare seduti | 0.94 | 0.93 |
| Stare in piedi | 0.93 | 0.94 |
| Stare stesi | 0.94 | 0.93 |

# Capitolo 6 – Conclusioni e Sviluppi Futuri

Con questo lavoro di tesi si è voluto ricercare quali erano gli approcci più gettonati per la risoluzione del problema della *Human Activity Recognition* e in cosa consistessero gli approcci da applicare in scenari reali. Si è voluto effettivamente controllare se fosse possibile classificare i dati raccolti da uno *smartphone* posto in una posizione più naturale, la tasca anteriore dei pantaloni, e quanto fosse effettivamente accurata la classificazione dei dati registrati in questa maniera. Si è dimostrato che è possibile utilizzare dati registrati da *ShieldApp* e non dover necessariamente applicare dei filtri sui dati per poter essere utilizzabili. Il che porta ad alleggerire ulteriormente il costo computazionale. In ultima analisi, si è voluto capire quali fossero le *feature*, tra le molteplici elencate ed utilizzate in letteratura, che effettivamente restituissero ottimi risultati.

Partendo da tutte queste premesse e dalla conoscenza accumulata leggendo le varie ricerche ritrovate, è stato possibile capire cosa fosse necessario per la configurazione dell’esperimento. Sono stati usati quattro *dataset* tutti con caratteristiche differenti, in range d’età, modalità di registrazione dei dati e persino la tipologia di attività registrate. Sono state utilizzate anche strategie differenti per la suddivisione dei dati presenti nei vari *dataset*, quello *Istance-Based* e lo *User-Based­,* in questa maniera si è voluto controllare sia la validità dei dati sui singoli utenti, sia la validità dei modelli e la loro capacità di classificare dati registrati da soggetti in momenti differenti. Dato che come è stato detto in precedenza, uno dei punti critici dell’HAR è proprio il riuscire a creare modelli che siano in grado di generalizzare il piùpossibile e che siano applicabili a qualsiasi tipologia di utente.

I risultati hanno mostrato come la *Convolutional Neural Network* abbia ottenuto sempre i risultati migliori, ma che approcci come il *Bidirectional Long Short-Term Recurrent Neural Network* e il *Deep Belief Network* siano comunque stati in grado di tenergli test, restituendo risultati discreti e comunque accettabili. Uno dei maggiori punti di forza del *DBN* è sicuramente la possibilità di essere allenato e utilizzato, richiedendo una potenza computazionale meno proibitiva della *CNN* del *Bi-LSTM.* Si è potuto anche osservare come l’utilizzo delle *feature* che si sono scelte e delle due strategie di addestramento dei modelli non abbia portato a grandi differenze nell’accuratezza degli stessi. Tendenzialmente, però, i sistemi riuscivano a classificare meglio quando si utilizzava l’approccio di tipo *User-Based*. Sicuramente aver incluso il calcolo degli angoli tra le *feature* e la magnitudine hanno aiutato molto i modelli nel riuscire a classificare i dati che venivano loro sottoposti.

Un altro punto importante è stata la creazione di un *dataset* utilizzando *ShieldApp*, le difficoltà riscontrate in alcuni casi, l’applicazione non effettuava misurazioni in alcuni momenti e quindi è stato necessario ripetere gli esperimenti più volte per poter essere in grado di raccogliere tutti i dati necessari.

Tra i lavori futuri che è possibile svolgere per continuare questo lavoro di tesi, vi sono sicuramente ampliare il tipo di attività registrate nell’*ISLAB*, utilizzando *smartphone* top di gamma per effettuare le misurazioni, poiché è stato possibile notare che i problemi maggiori riscontrati con l’utilizzo dell’applicazione, sono incorsi nel momento in cui le misurazioni veniva effettuate da smartphone di fascia medio-bassa.

Si potrebbe rendere gli algoritmi capaci di classificare più attività, magari utilizzando ancora meno *feature* e che siano capaci di funzionare utilizzando meno potenza computazionale.

Un’ulteriore lavoro futuro è sicuramente provare a classificare le attività, utilizzando frequenze di campionamento più basse dei 50Hz e capire quanto la frequenza di campionamento impatti sulla precisione dei modelli.

# RIFERIMENTI

|  |  |
| --- | --- |
| [1] | D. Kaldra, «Overriding FINTECH,» *Proc. Int. Conf. Digitization (ICD),* pp. 254-259, Nov. 2019. |
| [2] | P. K. Garg, R. D. Garg e P. P. Tasgaonkar, «“Vehicle detection and traffic estimation with sensors technologies for intelligent transportation systems,”,» *Sens. Imag., vol. 21, no. 1,* pp. 1-28, Dec. 2020. |
| [3] | U. Chaudhary, A. Patel, A. Patel e M. Soni, «“Survey paper on automatic vehicle accident detection and rescue system”,» *Data Science and Intelligent Applications,* p. 319–324, 2020. |
| [4] | M. Bernas, B. Placzek, W. Korski, P. Loska, J. Smyla e P. Szymala, «“A survey and comparison of low-cost sensing technologies for road traffic monitoring”,» *Sensors, vol. 18, no. 10,* p. 3243, Sep. 2018. |
| [5] | M. Krichen, «Anomalies Detection Through Smartphone Sensors: A Review,» *IEEE SENSORS JOURNAL,* vol. 21, 2021. |
| [6] | Z. R. Arkham, E. N. Lukito, Widyawani e Kurnianingsihi, «Fall Detection System Using Accelerometer and Gyroscope Based on Smartphone,» *I st International Conference on Infonnation Technology, Computer and Electrical Engineering (ICIT ACEE),* 2014. |
| [7] | C. Chakraborty e B. Gupta, «Adaptive Filtering Technique for Chronic Wound Analysis under TeleWound Network,» *Journal of Communication, Navigation, Sensing and Services,* pp. 57-76, 2016. |
| [8] | S. Lee, S. Yoon e H. Cho, «Human activity recognition from accelerometer data using convolutional neural network,» *International Conference on Big Data and Smart Computing,* pp. 131-134, 2017. |
| [9] | M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, W. Pang e J. Zhang, «Convolutional neural networks for human activity recognition using mobile sensors,» *International Conference on Mobile Computing, Applications and Services,* pp. 197-205, 2014. |
| [10] | S. Dupond, «A thorough review on the current advance of neural network structures,» *Annual Reviews in Control,* vol. 14, pp. 200-230, 2019. |
| [11] | O. I. Abiodun, A. Jantan, A. E. Omalare, K. V. Dada, N. A. Mohamed e H. Arshad, «State-of-the-art in artificial neural network applications: A survey,» *Heliyon,* 2018. |
| [12] | A. Tealab, «Time series forecasting using artificial neural networks methodologies: A systematic review,» *Future Computing and Informatics Journal,* p. 334–340, 2018. |
| [13] | H. Sak, A. Senior e F. Beaufays, «Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling,» 2014. |
| [14] | X. Li e X. Wu, «Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition,» 2014. |
| [15] | F. Gers, N. N. Schraudolph e J. Schmidhuber, «Learning Precise Timing with LSTM Recurrent Networks,» *Journal of Machine Learning Research,* p. 115–143, 2017. |
| [16] | J. Schmidhuber, «Deep Learning in Neural Networks: An Overview,» *Neural Networks,* p. 85–117, 2015. |
| [17] | F. Gers e J. Schmidhuber, «LSTM recurrent networks learn simple context-free and context-sensitive languages,» *IEEE Transactions on Neural Networks,* p. 1333–1340, 2001. |
| [18] | A. Graves e J. Schmidhuber, «Framewise phoneme classification with bidirectional LSTM and other neural network architectures,» *Neural Networks,* 2005. |
| [19] | T. Thireou e M. Reczko, «Bidirectional Long Short-Term Memory Networks for Predicting the Subcellular Localization of Eukaryotic Proteins,» *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* p. 441–446, 2007. |
| [20] | S. N. Gowda, «Human Activity Recognition based on Deep Belief Network Classifier and Combination of Local and Global Features,» *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* pp. 1589-1594, 2017. |
| [21] | S. Abirami e P. Chitra, «Advances in Computers,» 2020. |
| [22] | G. Hinton, «Deep belief networks,» *Scholarpedia,* 2009. |
| [23] | G. E. Hinton, «A fast learning algorithm for deep belief nets,» *Neural Computation,* p. 1527–1554, 2006. |
| [24] | Y. Bengio , P. Lamblin , D. Popovici e H. Larochelle , «Greedy Layer-Wise Training of Deep Networks,» 2007. |
| [25] | Y. Bengio, «Learning Deep Architectures for AI,» *Foundations and Trends in Machine Learning,* pp. 1-127, 2009. |
| [26] | A. Mahmoodzadeh, «Human Activity Recognition based on Deep Belief Network Classifier and Combination of Local and Global Features,» *Department of Electrical Engineering, Shiraz Branch, Islamic Azad University, Shiraz, Iran,* 2020. |
| [27] | J. Ren e X. Huang, «Learning Control,» 2021. |
| [28] | G. Hinton, «Training Product of Experts by Minimizing Contrastive Divergence,» *Neural Computation,* p. 1771–1800, 2002. |
| [29] | G. Hinton, «A Practical Guide to Training Restricted Boltzmann Machines,» *Tech. Rep.,* 2010. |
| [30] | A. Fischer e C. Igel, «Training Restricted Boltzmann Machines: An Introduction,» *Pattern Recognition,* pp. 25-39, 2014. |
| [31] | A. Ignatov, «Real-time human activity recognition from accelerometer data using convolutional neural networks,» *Applied Soft Computing,* vol. 62, pp. 915-922, 2018. |

## RINGRAZIAMENTI

Un ringraziamento va a mio padre e a mia madre, che da sempre mi supportano nei momenti di difficoltà, per avermi insegnato a combattere e a non darmi mai per vinto, anche quando sembra che non possa farcela. A mio fratello, che riesce a capirmi come nessuno meglio di lui e sa sempre come prendermi. Ai miei due amici di sempre, Antonello e Alessandro, le persone di cui più mi fido e che più stimo di tutte. E anche a tutte le persone, famigliari e amici, che hanno contribuito a rendermi la persona che sono oggi. A chi c’è stato, a chi non c’è più e a chi ci sarà sempre.