



UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di laurea in Informatica

Integrazione di classificatori di ADL in App Android

Relatore: Prof.ssa Daniela Micucci

Correlatore: Prof. Marco Mobilio

Relazione della prova finale di:
Gabriele De Rosa
Matricola 829835

Anno Accademico 2019-2020

Sommario

Il riconoscimento dell'attività umana (Human Activity Recognition, HAR) è un campo molto attivo della ricerca e molte tecniche di *deep learning* sviluppate negli ultimi anni hanno dimostrato di essere affidabili per la classificazione delle attività di vita quotidiana (Activities of Daily Living, ADLs).

Malgrado le avanzate tecnologie messe in campo, spesso si incorre in limiti e problemi che sono da attribuire ad aspetti puramente organizzativi, di gestione e significato dei dati raccolti.

La seguente trattazione tenta di descrivere quanto svolto durante l'esperienza di stage presso l'Università degli Studi di Milano - Bicocca, durante il quale si è cercato di sviluppare un classificatore di ADLs in grado di analizzare i dati sensoriali ottenuti da un'applicazione Android.

Indice

1	Introduzione	2
2	App	3
3	API	4
4	Server	5
5	Classificazione	6
5.1	Caricamento dei dati	6
5.2	Apprendimento e Test	7
5.2.1	Preparazione dei dati	7
5.2.2	Creazione della rete neurale	9
5.2.3	Testing della rete neurale	10
5.3	Predizioni	10
6	Conclusioni	11
	Riferimenti	12
	Siti	12

Capitolo 1

Introduzione

Capitolo 2

App

Capitolo 3

API

Capitolo 4

Server

Capitolo 5

Classificazione

Il cuore del progetto riguarda la classificazione delle attività mediante i dati ottenuti.

Ho optato per l'utilizzo di Keras [1]. Si tratta di una libreria open-source per le reti neurali che astrae lo sviluppo rendendolo più comprensibile, pur mantenendo pieno supporto alle librerie di più basso livello (es. Tensorflow [2]) su cui si basa.

5.1 Caricamento dei dati

Il passaggio che precede l'apprendimento è il recupero dei dati collezionati nei file CSV.

Archive	Index	X Axis	Y Axis	Z Axis	Timestamp	Phone Position	Activity
8e9c147c-6fa0-466f-993b-f726a786933c	2	1.5047760009765625	4.826629638671875	7.0322265625	1592314676719	2	0
8e9c147c-6fa0-466f-993b-f726a786933c	3	1.76776123046875	5.1774139404296875	8.53094482421875	1592314676721	3	0
8e9c147c-6fa0-466f-993b-f726a786933c	4	1.7921142578125	5.622589111328125	9.804702758789062	1592314676724	4	0
8e9c147c-6fa0-466f-993b-f726a786933c	5	1.8303985595703125	5.4080047607421875	12.129501342773438	1592314676790	5	0
8e9c147c-6fa0-466f-993b-f726a786933c	7	3.29852294921875	3.7346649169921875	17.15325927734375	1592314676797	6	0
8e9c147c-6fa0-466f-993b-f726a786933c	8	3.431658837890625	2.5201263427734375	16.86737060546875	1592314676799	7	0
8e9c147c-6fa0-466f-993b-f726a786933c	9	2.33697509765625	2.123687744140625	15.2672119140625	1592314676870	8	0
8e9c147c-6fa0-466f-993b-f726a786933c	10	1.0340576171875	2.21173095703125	12.205001831054688	1592314676873	9	0
8e9c147c-6fa0-466f-993b-f726a786933c	11	-0.002960205078125	2.9204254150390625	8.565078735351562	1592314676876	10	0
8e9c147c-6fa0-466f-993b-f726a786933c	12	-0.40704345703125	3.8975830078125	6.2041015625	1592314676879	11	0
8e9c147c-6fa0-466f-993b-f726a786933c	13	0.1982269287109375	4.698699951171875	5.07843017578125	1592314676950	12	0
8e9c147c-6fa0-466f-993b-f726a786933c	14	1.20355224609375	5.3319854736328125	5.075714111328125	1592314676956	13	0
8e9c147c-6fa0-466f-993b-f726a786933c	15	1.9846649169921875	5.592987060546875	6.3537445068359375	1592314676961	14	0

Figura 5.1: Esempio di dati contenuti nel file CSV

Dopo la sola lettura è già possibile ottenere una chiara visualizzazione grafica della suddivisione per attività dei dati presenti.

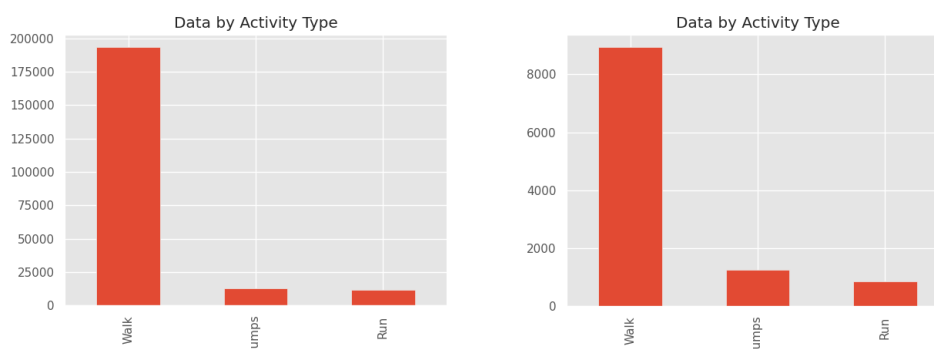


Figura 5.2: Visualizzazione della suddivisione dei dati per i due sensori

I dataset sono relativi ai sensori attivi sull'applicazione, ovvero accelerometro e giroscopio. Entrambi contengono la stessa tipologia di informazioni (i valori dei tre assi, il timestamp e il posizionamento del dispositivo). Nelle procedure seguenti considererò quindi il singolo dataset, ricordando però di dover applicare i passaggi indistintamente ad entrambi.

5.2 Apprendimento e Test

L'intera mole di dati necessita un partizionamento per differenziare i dati che saranno utilizzati per l'*apprendimento* e quelli che saranno utilizzati per il *test*.

Personalmente ho scelto una semplice suddivisione che prevede l'utilizzo di $\frac{4}{5}$ dei dati per il *train* e la parte restante ($\frac{1}{5}$) per il *test*.

È indispensabile non sovrapporre queste frazioni se non si vuole ottenere una valutazione dell'efficienza falsata.

5.2.1 Preparazione dei dati

Ci si aspetta che, fornita una serie di caratteristiche (i tre assi x, y, z, il valore temporale e la posizione del dispositivo), la rete neurale dia in risposta un'etichetta rappresentante l'attività associata.

Dobbiamo quindi organizzare i dati in nostro possesso in modo da renderlo possibile.

Trasformazione del valore temporale

Tra le caratteristiche abbiamo i *timestamps* che però rappresentano il tempo assoluto, ovvero il momento esatto di svolgimento dell'attività durante la raccolta dei dati.

Il momento esatto non è in alcun modo rilevante nella classificazione, ma a partire da questo è possibile ricavare il tempo trascorso tra l'acquisizione di una tripla di dati (x, y, z) e l'acquisizione di quella immediatamente successiva. Posso presumere una somiglianza in tali distanze temporali durante lo svolgimento di una uguale attività.

Normalizzazione dei dati

La rete neurale accetta in ingresso valori compresi tra 0 e 1. Eseguo una semplice normalizzazione sui dati delle caratteristiche.

```
# Normalize features for data set (values between 0 and 1)
df['x-axis'] = df['x-axis'] / df['x-axis'].max()
df['y-axis'] = df['y-axis'] / df['y-axis'].max()
df['z-axis'] = df['z-axis'] / df['z-axis'].max()
df['timestamp'] = df['timestamp'] / df['timestamp'].max()
df['phone-position'] = df['phone-position'] / number_of_phone_positions
```

Listing 1: Banale normalizzazione dei dati

Creazione dei segmenti e delle etichette

La parte principale dell'intero adattamento risulta essere quella che suddivide i dati in un formato che possa realizzare l'associazione tra una serie di caratteristiche e una etichetta.

Per realizzare ciò i record sono presi a gruppi, anche sovrapposti (così come si vede in figura 5.3). Ogni raggruppamento sarà caratterizzato da

- un segmento contenente i record con le sole caratteristiche
- l'etichetta più frequente

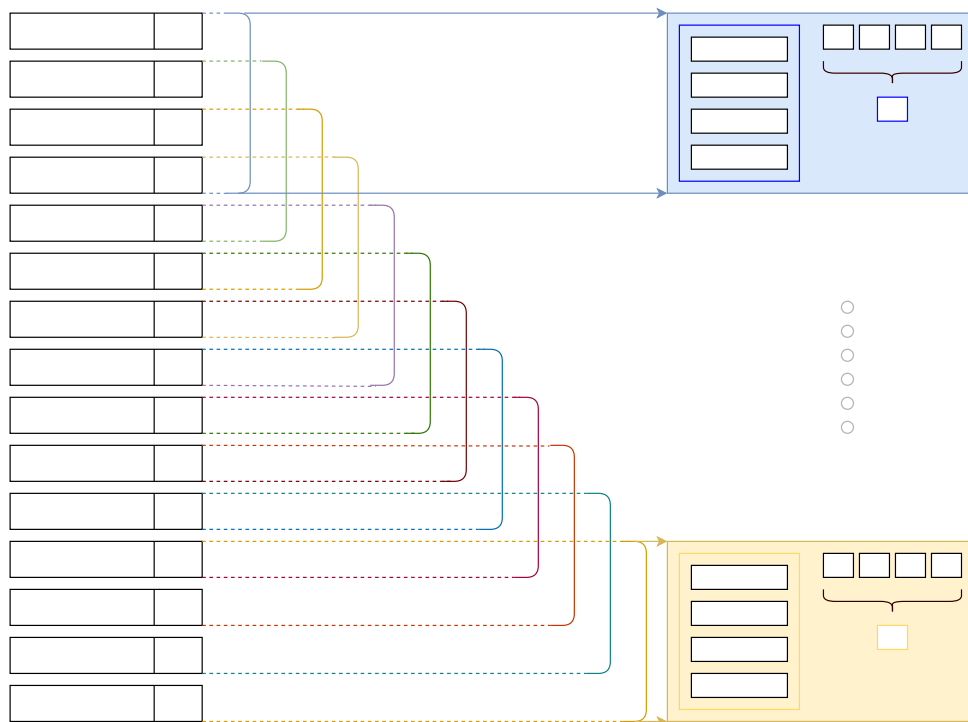


Figura 5.3: Creazione dei segmenti e delle etichette

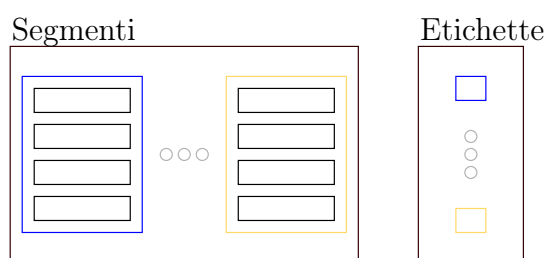


Figura 5.4: Risultato dopo la creazione dei segmenti e delle etichette

Alla fine del processo ho quindi ottenuto una serie di segmenti di dati a cui posso singolarmente associare una determinata etichetta identificativa.

5.2.2 Creazione della rete neurale

Una volta generati i dati nel formato supportato da *Keras* procedo alla creazione di una rete neurale che abbia

- in input il formato dei dati appena generato
- 5 strati di 100 nodi connessi
- in output il calcolo di probabilità per ogni classe

```
# Create DNN
model_m = Sequential()
model_m.add(Dense(100, activation='relu')) # Layer 1
model_m.add(Dense(100, activation='relu')) # Layer 2
model_m.add(Dense(100, activation='relu')) # Layer 3
model_m.add(Dense(100, activation='relu')) # Layer 4
model_m.add(Dense(100, activation='relu')) # Layer 5
model_m.add(Flatten())
model_m.add(Dense(num_classes, activation='softmax'))
```

Listing 2: Creazione della DNN

Per poi procedere all'apprendimento.

```
# Fit the model
model_m.fit(segments_train,
            labels_train,
            batch_size=BATCH_SIZE,
            epochs=EPOCH,
            validation_split=0.20,
            verbose=1)
```

Listing 3: Apprendimento della rete neurale

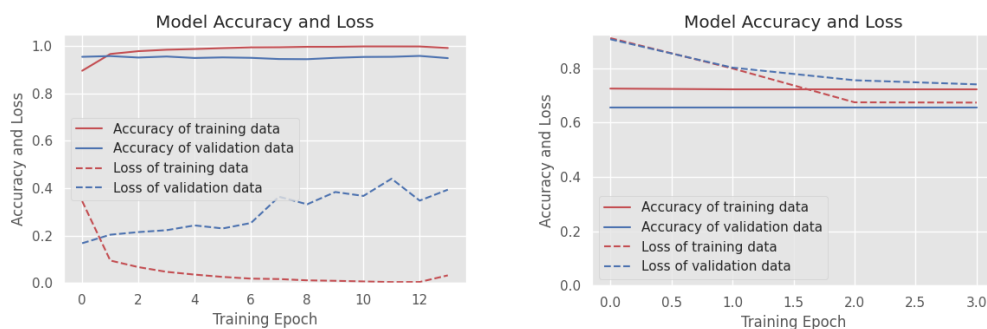


Figura 5.5: Statistiche del modello ottenuto

5.2.3 Testing della rete neurale

Il test della rete neurale creata avviene mediante l'uso dei dati che avevamo precedentemente separato dal dataset iniziale.

L'intento è quello di effettuare una predizione con i dati di test, di cui però si conoscono già i risultati. Sarà quindi immediato trovare l'efficienza del modello creato mediante un banale confronto tra i dati ipotizzati dalla rete neurale e i risultati corretti.

La qualità dei dati può essere visualizzata mediante una *matrice di confusione* che fornisce una rappresentazione grafica del confronto appena descritto.

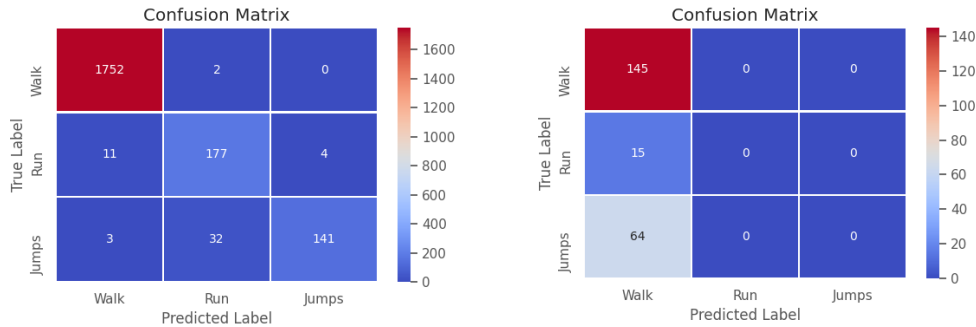


Figura 5.6: Statistiche del modello ottenuto

5.3 Predizioni

In maniera del tutto equivalente a quanto già visto durante l'apprendimento, per effettuare una predizione sfruttando la rete neurale allenata forniremo in input una serie di segmenti e attenderemo che il classificatore ci fornisca in risposta una serie di etichette ipotizzate.

Capitolo 6

Conclusioni

Riferimenti

Siti

- [1] *Keras*. URL: <https://github.com/keras-team/keras> (cit. a p. 6).
- [2] *Tensorflow*. URL: <https://github.com/tensorflow/tensorflow> (cit. a p. 6).