Report

**Introduzione**

L'obiettivo del progetto realizzato è il riconoscimento di esercizi adatti agli anziani attraverso l'utilizzo di librerie che si occupano della Computer Vision, in particolare OpenCV, OpenPose e Tensorflow.

Prima di realizzare ed eseguire gli script presenti all'interno del progetto, è necessaria l'installazione delle dipendenze, ovvero strumenti necessari al corretto funzionamento degli stessi (tra questi vi sono quelli citati precedentemente).

Il primo step è stato quello di individuare quali esercizi potessero essere adatti per gli anziani e, dopo una ricerca approfondita su internet, abbiamo concordato sui seguenti quattro esercizi per la “ginnastica dolce”, utili per l’addestramento del sistema:

* circonduzione delle braccia in piedi;
* core stability;
* sollevamento di una gamba da seduti;
* portare un ginocchio al petto da steso.

Successivamente abbiamo raccolto e realizzato dieci video per ogni esercizio di cui otto sono stati dedicati al training set e gli ultimi due video per il test set (i filmati considerati hanno 30 fps).  
A seguire, i video sono stati convertiti in una sequenza di frame numerati (attraverso l’esecuzione di uno script python) e salvati in cartelle dedicate contrassegnate dal nome dell’esercizio e da un numero. Infine, sono stati scritti due file di testo (“*valid\_images.txt*”), relativi al training set e al test set, che specificano per ogni cartella il nome e l’intervallo dei frame contenuti, ovvero il numero corrispondente al frame iniziale e al frame finale.

Immagine che contiene testo

Descrizione generata automaticamente

Nel file di configurazione del progetto inizialmente sono stati inseriti il formato dei frame, il formato dei documenti di testo che vengono generati dopo il riconoscimento dello scheletro della persona presente all’interno del frame, il numero di frame adiacenti per l’estrazione delle features, il numero dei timesteps e sono state definite le classi che verranno considerate nel progetto.



**s1\_get\_skeletons\_from\_training\_testing\_imgs.py**

Il codice si occupa di prendere in input ciascuna cartella all'interno del training set e del test set e, per ciascuno dei frame all'interno di essa, identifica lo scheletro della persona nell'immagine e salva in un documento di testo le seguenti informazioni:

- un numero identificativo associato a ciascuna classe(esercizio)

- un numero identificativo associato ad ogni video

- un numero identificativo associato ad ogni frame del video

- la stringa relativa alla classe

- la directory del video e il nome del frame in questo modo: <video\_directory>/<nome\_frame>.jpg

- per ciascun joint dello scheletro vengono salvate le relative coordinate

Il riconoscimento dello scheletro viene effettuato utilizzando la libreria OpenPose, in particolare scegliendo il modello e la dimensione dell'immagine.

Le immagini di cui effettuare il riconoscimento vengono definite dal file "*valid\_images.txt*" contenuto in entrambi i set.

Il file di configurazione in questo codice viene utilizzato per prendere i path relativi alle cartelle da cui raccogliere i frame e definire le directory dove salvare i file txt realizzati.

Funzione da citare: *process\_images*

 input:

* *images\_loader*: Lettore di immagini, prende il riferimento di ciascuna immagine di cui effettuare il riconoscimento
* *DST\_DETECTED\_SKELETONS\_FOLDER*: directory dove sono presenti tutte le cartelle associate a ciascun video (all'interno contengono i frame)

         elaborazione:

Per ciascun frame contenuto in ciascuna cartella si effettua il riconoscimento dello scheletro della persona contenuta all'interno del frame attraverso il modello offerto da OpenPose. I joint e le relative connessioni vengono disegnate sul frame che viene visualizzato a schermo, successivamente le informazioni definite precedentemente vengono salvate in un file txt dedicato.

**s2\_put\_skeletons\_txts\_into\_train\_test\_csv.py**

Il codice si occupa di importare le informazioni contenute nei file di testo relativi a ciascun frame all'interno di due file csv: “*train\_set.csv*” e “*test\_set.csv*”.

Il file di configurazione viene usato per avere il riferimento alle directory in cui sono salvati i file di testo relativi ai frame e per salvare i file csv nella cartella *data\_proc/*

Funzione da citare: *write\_on\_csv*

input:

* + *num\_skeletons*: numero di file di testo di ciascun frame
  + *csvpath*: directory in cui è presente il file csv
  + *skeletons\_folder*: directory in cui sono presenti i file di testo relativi a ciascun frame

elaborazione:

Scrive nella prima riga del file csv i nomi delle colonne da riempire. Per ogni file di testo associato al frame prende le informazioni al suo interno e le copia nel file csv

**s2\_put\_skeletons\_txts\_into\_train\_test\_csv\_interpolation.py**

il codice riprende il file precedente ma prima della scrittura sui file csv effettua l’interpolazione sui dati per diminuire la presenza di valori nulli.

funzioni da citare: *interpolation*

input:

*data*: dataset completo

output:

*data*: dataset su cui è stata effettuata l’operazione di interpolazione

elaborazione:

Per ogni valore all’interno del dataset si controlla se questo sia un valore nullo; in caso sia presente si raccolgono in un array tutti i valori che siano della stessa categoria e dello stesso video. Se la percentuale di valori non nulli supera il 60%, viene realizzata la funzione di interpolazione e si calcolano i nuovi valori che andranno a sostituire quelli nulli presenti nell’array, compreso quello considerato inizialmente.

**s3\_from\_train\_test\_csv\_to\_overlap.py**

Il codice effettua l'operazione di overlapping su training set e sul test set salvando i nuovi dataset e i relativi set che contengono la feature da predire nei rispettivi file csv.

Dal file di configurazione si fa riferimento ai path dei csv che contengono train set e test set e si definiscono i path dei nuovi file csv realizzati.

funzioni da citare:

*get\_matrix*

input:

* + *data*: dataset completo
  + *idx\_row*: indice della riga da cui partire per elaborare la matrice

output:

* + *matrix*: matrice da 60 righe e 36 colonne
  + *offset*: valore che indica lo spostamento del vecchio indice di riga da cui realizzare la nuova matrice

elaborazione:

Partendo dall'indice di riga fornito, la funzione raccoglie nel dataset un numero di righe <= 60, le quali devono essere associate ai frame dello stesso video. In caso le righe raccolte siano 60 queste vengono utilizzate per creare una matrice da 60 righe e 36 colonne e l'offset viene posto a 30, ovvero la metà del numero di timesteps. In caso contrario la matrice per raggiungere il numero di righe definito precedentemente viene riempita con righe di 36 zeri e l'offset è calcolato contando le righe di valori non nulli.

*overlap*:

input:

*data*:dataset completo

output

* + *data\_overlap*: dataset su cui è stato effettuato l'overlap
  + *data\_y*: dataset che riporta, per ogni riga di *data\_overlap*, il relativo numero identificativo della classe

elaborazione:

Partendo dal dataset iniziale vengono create le matrici da inserire nel nuovo dataset (*data\_overlap*) e i relativi id di classe (*data\_y*)

**s4\_evaluation\_model\_performance\_lstm.py**

Il codice carica i dataset dai relativi file csv: per i dataset che contengono features non predette questi vengono sottoposti ad una operazione di *reshape*, mentre si effettua l'operazione di hot encoding tramite la funzione *to\_categorical* su quelli con la features da predire.

Successivamente si calcola l'accuratezza del modello dopo l'addestramento.

Il modello considerato è il LSTM.

La libreria *Tensorflow* viene utilizzata per la configurazione del modello e l'operazione di hot encoding.

Dal file di configurazione si fa riferimento ai path dei csv che contengono train set e test set su cui è stata effettuata l'operazione di overlap e i relativi set con la feature da predire e si definiscono i path del modello e del grafico creati.

funzione da citare:

*evaluate\_model*

input:

* + *trainX*: training set delle features non predette
  + *trainy*: training set delle feature da predire
  + *testX*: test set delle features non predette
  + *testy*: test set delle feature da predire

output:

*accuracy*: accuratezza del modello

elaborazione:

Il modello viene creato utilizzando i seguenti parametri:

* + - numero delle features
    - numero degli output
    - numero dei timesteps

Il modello viene addestrato su *trainX* e *trainy* e viene mostrato e salvato un grafico che ne illustra l'errore e l'accuratezza nelle cento epoche.

Infine il modello viene salvato in un file formato *pickle* e si estrae l'accuratezza valutando il modello su *testX* e *testy*.

**s4\_evaluation\_model\_performance\_conv\_lstm.py**

Il codice carica i dataset dai relativi file csv: per i dataset che contengono features non predette questi vengono sottoposti ad una operazione di *reshape*, mentre si effettua l'operazione di hot encoding tramite la funzione *to\_categorical* su quelli con la features da predire.

Successivamente si calcola l'accuratezza del modello dopo l'addestramento.

Il modello considerato è il Convolutional LSTM.

La libreria *Tensorflow* viene utilizzata per la configurazione del modello e l'operazione di hot encoding.

Dal file di configurazione si fa riferimento ai path dei csv che contengono train set e test set su cui è stata effettuata l'operazione di overlap e i relativi set con la feature da predire e si definiscono i path del modello e del grafico creati.

funzione da citare:

*evaluate\_model*

input:

* + *trainX*: training set delle features non predette
  + *trainy*: training set delle feature da predire
  + *testX*: test set delle features non predette
  + *testy*: test set delle feature da predire

output:

*accuracy*: accuratezza del modello

elaborazione:

Il modello viene creato utilizzando i seguenti parametri:

* + - numero delle features
    - numero degli output

Il modello viene addestrato su *trainX* e *trainy* e viene mostrato e salvato un grafico che ne illustra l'errore e l'accuratezza nelle cento epoche.

Infine il modello viene salvato in un file formato *pickle* e si estrae l'accuratezza valutando il modello su *testX* e *testy*.

**s4\_evaluation\_model\_performance\_cnn\_lstm.py**

Il codice carica i dataset dai relativi file csv: per i dataset che contengono features non predette questi vengono sottoposti ad una operazione di *reshape*, mentre si effettua l'operazione di hot encoding tramite la funzione *to\_categorical* su quelli con la features da predire.

Successivamente si calcola l'accuratezza del modello dopo l'addestramento.

Il modello considerato è il CNN-LSTM.

La libreria *Tensorflow* viene utilizzata per la configurazione del modello e l'operazione di hot encoding.

Dal file di configurazione si fa riferimento ai path dei csv che contengono train set e test set su cui è stata effettuata l'operazione di overlap e i relativi set con la feature da predire e si definiscono i path del modello e del grafico creati.

funzione da citare:

*evaluate\_model*

input:

* + *trainX*: training set delle features non predette
  + *trainy*: training set delle feature da predire
  + *testX*: test set delle features non predette
  + *testy*: test set delle feature da predire

output:

*accuracy*: accuratezza del modello

elaborazione:

Il modello viene creato utilizzando i seguenti parametri:

* + - numero delle features
    - numero degli output

Il modello viene addestrato su *trainX* e *trainy* e viene mostrato e salvato un grafico che ne illustra l'errore e l'accuratezza nelle cento epoche.

Infine il modello viene salvato in un file formato *pickle* e si estrae l'accuratezza valutando il modello su *testX* e *testy*.

Qui di seguito sono riportati i grafici che mostrano l’andamento dell’errore e dell’accuratezza durante le cento epoche per i modelli:

* LSTM

<GRAFICO>

* Convolutional LSTM

<GRAFICO>

* CNN-LSTM

<GRAFICO>

**s5\_run.py**

Lo script visualizza in real-time il flusso della videocamera, mostra i joint riconosciuti della persona presente nel filmato e usa uno dei tre modelli implementati per il riconoscimento degli esercizi(il nome dell’esercizio verrà visualizzato a schermo). Vi è inoltre la possibilità di effettuare il riconoscimento partendo da una cartella di immagini o da un video pre-caricato.

La libreria OpenPose viene utilizzata per il riconoscimento dei joint della persona presente nel filmato.

Dal file di configurazione si fa riferimento al path da cui caricare il modello per il riconoscimento degli esercizi, mentre lo script in output fornisce il video(registrato durante il funzionamento della videocamera) e una cartella di documenti di testo (ognuno dei quali fornisce le coordinate dei joint riconosciute nel frame dedicato) i cui path sono definiti nel config file.

funzioni da citare:  *Draw\_result\_img*

Input:

* *img\_disp*: frame della videocamera
* *humans*: oggetto che fa riferimento agli scheletri da visualizzare a video
* *skeleton\_detector*: oggetto che permette di “disegnare” sul frame i joint riconosciuti
* *label\_class*: il nome dell’esercizio predetto

output:

*img\_disp*: immagine che visualizza i joint riconosciuti e il nome dell’esercizio predetto

Elaborazione:

Funzione che ridimensiona la finestra per la visualizzazione, disegna lo scheletro delle persone e visualizza il nome dell’esercizio su una barra bianca posta al di sopra della videocamera.