Smart Ringbell

D’Amelio Amedeo

Lillo Alessandro

Casi d’uso

1. Utente in fase di adding: l’utente immette campioni di tipo audiovisivo tali da generare un modello da utilizzare durante la fase di test.
2. Utente in fase di modifica dei dati salvati: l’utente può modificare i dati già immessi in un modello sovrascrivendoli. La sovrascrittura permetterà di ricreare un modello completamente nuovo eliminando il precedente
3. Utente in fase di eliminazione di dati salvati: l’utente può eliminare un modello precedentemente creato immettendo lo username corrispondente
4. Utente riconosciuto in fase di test: l’utente immette un campione audiovisivo che verrà confrontato con i modelli presenti nel data set e riconosciuto attraverso un confronto sia con i modelli generati utilizzando i campioni sonori sia con i modelli generati utilizzando i campioni video.
5. Utente non riconosciuto in fase di test: l’utente immette un campione audiovisivo che verrà confrontato e rigettato con i modelli presenti nel data set. Precisamente, entrambi i campioni non supereranno una soglia di somiglianza con i modelli presenti del data set tale da poter rivendicare un’identità. È inviata una notifica al dispositivo android dell’utente proprietario di casa informandolo della presenza di un potenziale sconosciuto alla sua porta. Attraverso l’utilizzo dell’app android sarà possibile visualizzare la foto del possibile sconosciuto e decidere se aprire o meno la porta a distanza. La foto scattata al possibile sconosciuto sarà cancellata appena presa una decisione. Per rispettare la privacy dell’utenza non sarà neanche possibile fare screenshots durante la visualizzazione della foto. Una volta che l’utente proprietario avrà deciso se lasciare o meno aperta la porta arriverà un messaggio di conferma sia sullo schermo del proprio dispositivo android sia al citofono vero e proprio
6. Utente riconosciuto parzialmente in fase di test: l’utente immette un campione audiovisivo che verrà confrontato e riconosciuto parzialmente con i modelli presenti nel dataset. Ciò implica che l’utente avrà superato solo uno dei due riconoscimenti previsti per l’accesso. Il riconoscimento parziale impedisce un rigetto assoluto dell’utente che potrà accedere solo grazie ad una conferma avvenuta da parte dell’utente proprietario di casa. Attraverso l’utilizzo dell’app android sarà possibile visualizzare la foto della persona alla porta e decidere se aprire o meno quest’ultima a distanza. La foto scattata sarà cancellata appena presa una decisione. Per rispettare la privacy dell’utenza non sarà neanche possibile fare screenshots durante la visualizzazione dell’immagine. Una volta che l’utente proprietario avrà deciso se lasciare o meno aperta la porta arriverà un messaggio di conferma sia sullo schermo del proprio dispositivo android sia al citofono vero e proprio

Manuale Tecnico

Smart Ringbell nasce dall’intenzione di applicare tecniche biometriche all’ambito della sicurezza domestica. Si presenta come un citofono smart che prevede l’immissione di due tipi diversi di tratto biometrico: la voce e il volto. Attraverso un riconoscimento di questi ultimi sarà possibile consentire l’accesso alla casa solo a chi sarà registrato come utente nel sistema. Il riconoscimento per i tratti avverrà in maniera separata ed indipendente in modo tale che se una delle due metodologie di riconoscimento non dovesse andare a buon fine, la “compromessa” non infici con le performance del sistema. In situazioni di dubbio o rigetto la possibilità che la porta si apra non sarà esclusa a priori, ma sarà lasciata alla discrezione dell’utente padrone di casa attraverso l’utilizzo di una apposita app android.

All’apertura dell’applicativo sarà visualizzato un menu dal quale si potrà scegliere tra le seguenti azioni:

* Aggiungi un utente
* Modifica un utente
* Elimina un utente
* Suona il campanello (Recognition)
* Esci dall’applicativo

Raccolta dei dati

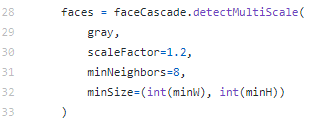
All’avvio della fase di raccolta dei dati sarà chiesto all’utente di scegliere un nome univoco con cui saranno salvati i suoi modelli.

Volto: Il rilevamento di oggetti che utilizzano i cascade classifiers (classificatori a cascata) si basano su Haar che è un metodo di rilevamento di oggetti molto efficace. È un approccio che si basa sul machine learning. La “cascade function” viene allenata attraverso molte immagini positive (contengono ciò che vogliamo identificare) e negative (immagini che non contengono l’elemento che vogliamo identificare). Fatto questo, la funzione viene utilizzata per rilevare gli oggetti in altre immagini.

Dopo aver allenato il classificatore, bisogna estrarre le features. Per fare questo vengono utilizzate le funzioni di Haar. Ogni feature è un singolo valore ottenuto sottraendo la somma dei pixel sotto il rettangolo bianco dalla somma dei pixel sotto il rettangolo nero. Tutte le dimensioni e posizioni possibili, vengono utilizzate per calcolare tutte le features. I calcoli da effettuare sono tantissimi anche se si ha una finestra molto piccola. Ad esempio, una finestra 24\*24 produce oltre 160000 funzioni. Per ridurre il tempo di computazione, viene utilizzata un’immagine integrale. L’immagine integrale è una struttura dati per il calcolo rapido della somma dei valori in un sottoinsieme rettangolare di una griglia.

Per la realizzazione del progetto non è stato creato un classificatore ad-hoc, ma è stato utilizzato il classificatore “haarcascade\_frontalface\_default.xml” fornito dalla libreria OpenCV. Questo classificatore è in grado di identificare solo volti posizionati di fronte alla videocamera. È stato utilizzato questo classificatore, perché è abbastanza “leggero” e non ha bisogno di un hardware molto performante per dare risultati in tempi accettabili.

La chiamata alla funzione di classificazione contiene dei parametri molto importanti:

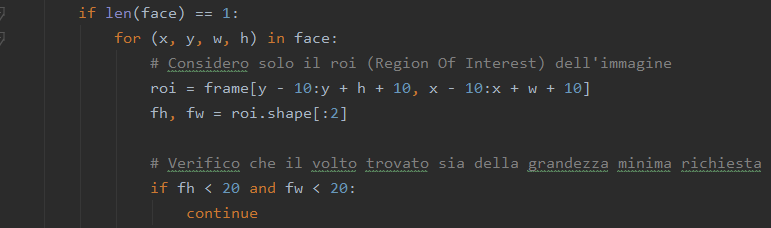


* gray: l’immagine in scala di grigi
* scaleFactor: parametro che specifica di quanto la dimensione dell’immagine deve essere ridotta. Viene utilizzato per creare la piramide della scala
* minNeighbors: è un parametro che specifica il numero di vicino che ogni rettangolo candidato deve avere per conservarlo. Un numero più alto fornisce meno falsi positivi.
* minSize: è la grandezza minima che il rettangolo deve avere affinché venga considerata una faccia.

Voce: l’acquisizione dei dati è avvenuta attraverso la realizzazione di un dataset personale basato su un numero di audio tale da poter costruire un modello ottenuto da un numero discreto di individui. A questi ultimi è stato richiesto di muoversi in ambienti sia privi sia con rumore per riuscire a testare al meglio l’algoritmo. Al fine di ottenere una qualità migliore nei modelli, agli utenti è stato richiesto di inserire un numero minimo di tre audio. Una volta registrati questi ultimi, all’utente sarà richiesto se voglia inserirne altri per migliorare la precisione del modello. Una volta generato quest’ultimo, i file audio saranno automaticamente cancellati al fine di rispettare i dati sensibili dell’utente. Le registrazioni avvengono per un totale di 5 secondi ciascuna, per impedire l’utilizzo di audio troppo corti e che quindi comprometterebbero la correttezza del modello, e con una frequenza di campionamento pari a 44100 Hz, una frequenza molto utilizzata nell’ambito dell’analisi audio poiché dato che il campo uditivo umano va da circa 20 Hz a 20000 Hz, si sarebbe dovuta utilizzare una frequenza superiore ai 40 Khz. Inoltre, i segnali devono essere passa-basso filtrati prima del campionamento per evitare aliasing. Mentre un filtro ideale passa-basso potrebbe perfettamente passare frequenze inferiori a 20 kHz e altrettanto perfettamente tagliare frequenze al di sopra di 20 kHz, un filtro tale ideale è teoricamente impossibile, quindi in pratica una banda di transizione diventa necessaria, ove le frequenze sono parzialmente attenuate. Il più ampio questa banda di transizione è, più facile e più economico è quello di fare un filtro anti-aliasing che è un filtro utilizzato per limitare la larghezza di banda del segnale. La frequenza di campionamento 44.1 kHz consente una banda di transizione 2.05 kHz. La registrazione dei file wav avviene attraverso l’utilizzo della funzione rec della libreria sounddevice.

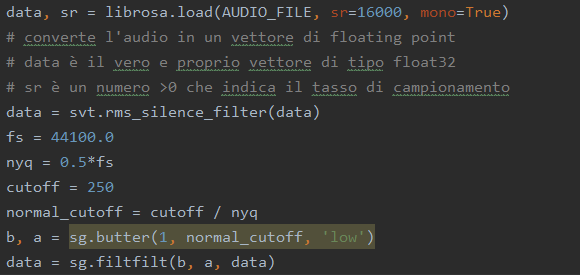
Pre-processing

Volto: Mentre il campione dell’utente viene ottenuto attraverso l’impiego di 30 frames, la fase di pre-processing si applica sul singolo frame, al quale viene attuata una conversione in bianco e nero al fine di controllare ulteriormente se sia presente o meno un volto ed evitare ancora una volta falsi positivi. Una volta fatto ciò, si attua un resize dell’immagine su scala 96x96 poiché è la grandezza richiesta dal modello utilizzato. Infine, il frame viene convertito in scala RGB





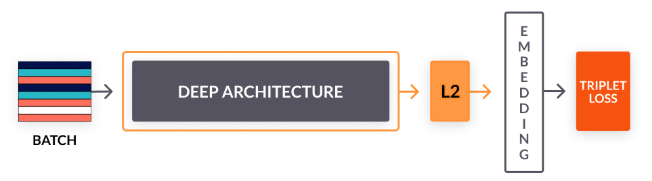
Voce: sugli audio sono stati applicati rimozione del silenzio e riduzione del rumore. La rimozione del silenzio avviene attraverso l’utilizzo della funzione rms\_silence\_filter appartenente alla libreria speaker\_verification\_toolkit la quale adatta la soglia di treshold sulla base dei primi frame di audio. La funzione di noise removal è stata implementata attraverso l’utilizzo di un butterworh filter applicato attraverso la libreria scipy.signal. Il filtro si applica ottenendo un grado di smoothing del grafico rappresentante l’audio, nel nostro caso è 1 per permettere uno smooth lento, e una frequenza di cutt-off normalizzata ottenuta scegliendo una frequenza di cut-off e dividendola per la frequenza di Nyquist. Prima di applicare queste due tecniche il dato viene letto attraverso la funzione load appartenente alla libreria librosa che si occupa di convertire l’audio in un vettore di floating point. Dalla suddetta funzione otteniamo anche un intero maggiore di zero che indica il tasso di campionamento del nostro file wav.



Training

Volto: Quando viene identificato un volto, il frame viene processato. Durante questa fase vengono processate circa inserire il numero preciso immagini.

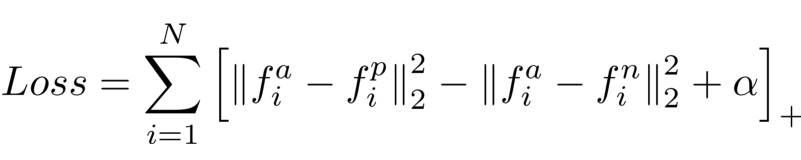
In questo progetto, viene utilizzato l’algoritmo di Google Facenet, un framework molto usato per l’implementazione di moderne applicazioni di riconoscimento facciale. FaceNet utilizza una tecnica chiamata “apprendimento one-shot”. La sua rete è composta da un livello di input batch e una rete neurale convoluzionale profonda (CNN) seguita da normalizzazione L2.



La rete neurale convoluzionale di FaceNet si basa sui pixel dell’immagine come caratteristiche (features), anziché estrarli manualmente. L’idea principale alla base dell’algoritmo è quella di rappresentare un volto come un vettore a 128 dimensioni, mappando le caratteristiche di input ai vettori.

Poiché questi vettori sono rappresentati nello spazio vettoriale condiviso, è possibile utilizzare la distanza vettoriale per calcolare la somiglianza tra due vettori. Questa è una tecnica per calcolare quanto sono simili due facce.

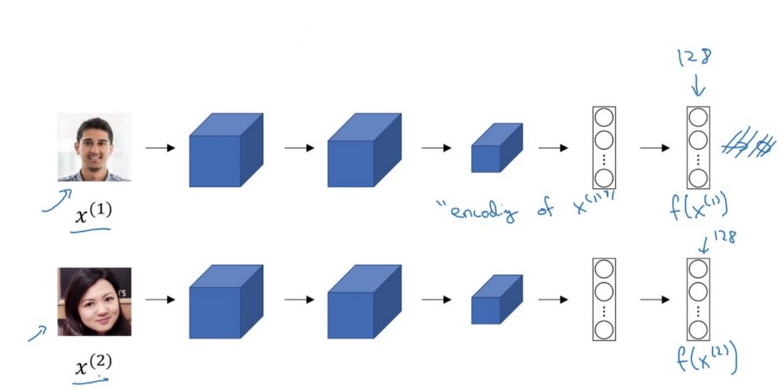
L’ultimo stadio dell’architettura FaceNet è la “perdita di tripletta” (triplet loss), che minimizza la distanza tra un’ancora e un positivo noto (somiglianza tra i due volti), massimizzando al contempo la distanza tra l’ancora e un negativo noto (dissomiglianza).



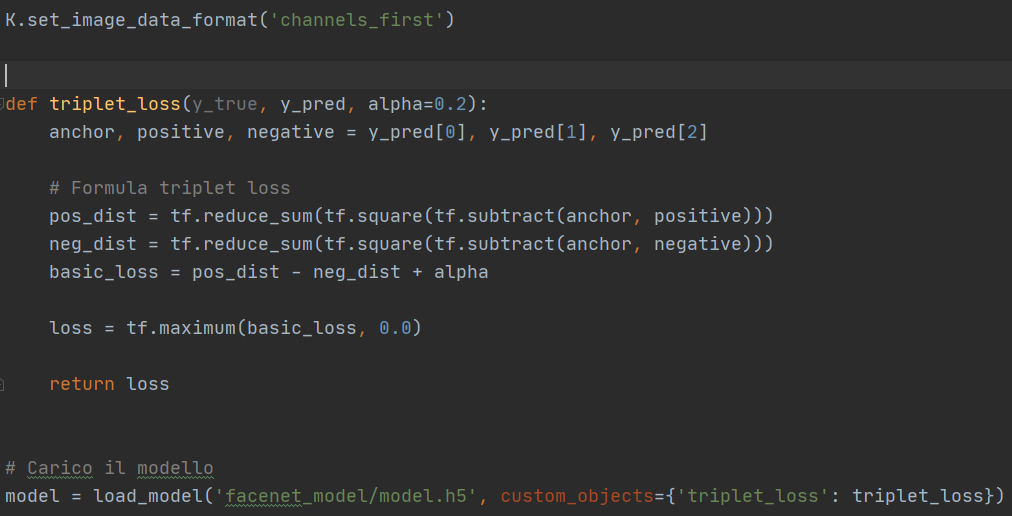
* f(a) si riferisce alla codifica di output dell’ancora.
* f(p) si riferisce alla codifica in uscita del positivo.
* f(n) si riferisce alla codifica in uscita del negativo.
* ɑ è una costante utilizzata per assicurarsi che la rete non tenti di ottimizzare verso f(a) - f(p) = f(a) - f(n) = 0.
* [...]+ è uguale a max(0, somma).

### Rete Siamese

FaceNet è una rete siamese. Una rete siamese è un tipo di architettura di rete neurale che impara a distinguere tra due input. Ciò consente loro di apprendere quali immagini sono simili e quali no. Le reti siamesi sono costituite da due reti neurali identiche, ognuna con gli stessi pesi esatti. innanzitutto, ogni rete prende come input una delle due immagini ottenute in ingresso. Quindi, gli output degli ultimi layer di ciascuna rete vengono inviati a una funzione che determina se le immagini contengono la stessa identità. In FaceNet questo viene fatto calcolando la distanza tra le due uscite.



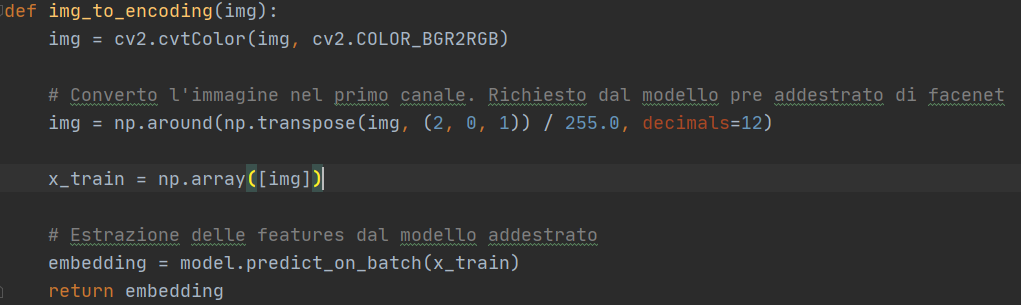
### Implementazione



Iniziamo inizializzando la rete con una forma di input di (3, 96, 96). Ciò significa che i canali Rosso-Verde-Blu (RGB) sono la prima dimensione del volume dell’immagine messa nella rete. Tutte le immagini che vengono inviate alla rete devono essere immagini di 96x96 pixel.

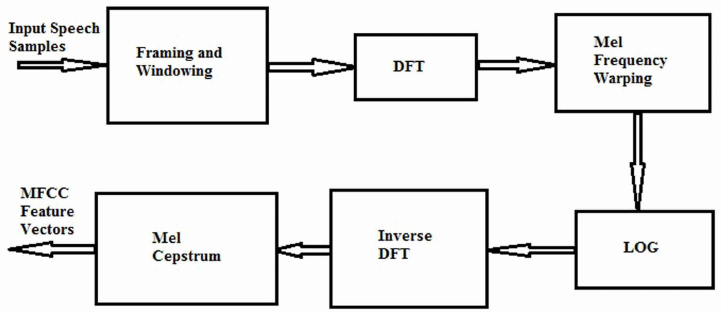
Successivamente, viene definita la funzione di “Perdita della Tripletta” definita precedentemente.

Adesso, è possibile compilare il modello di riconoscimento facciale usando Keras.

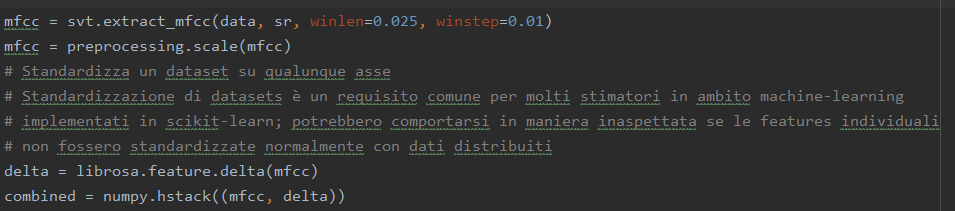


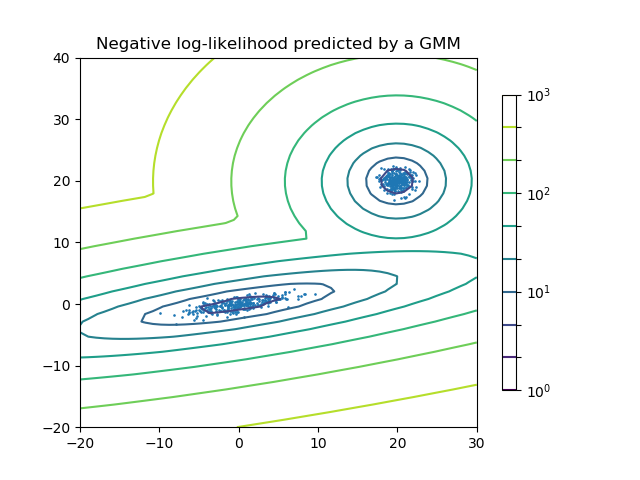
Per ogni immagine acquisito dalla webcam, i dati dell’immagine vengono convertiti in una codifica di 128 numeri float. Questo è fatto tramite la funzione img\_to\_encoding.

La funzione prende come parametro di ingresso l’immagine che viene convertita in RGB; questa, viene trasmessa alla rete di riconoscimento del volto. La funzione ritorna l’output dato dalla rete, che risulta essere la codifica dell’immagine. Il risultato viene poi inserito nel database.

Voce: in questa fase ci concentriamo su due caratteristiche fondamentali per il nostro algoritmo: Mel frequency Cepstral coefficients (MFCC) e la sua derivata, chiamata Delta-MFCC. I MFCC sono coefficienti che collettivamente costruiscono e rendono tale un mel-frequency cepstrum, il quale si presenta come una rappresentazione dello spettro di potenza su scala short-term di un suono basata sulla trasformata lineare coseno di uno spettro di potenza su scala logaritmica. I MFCC si concentrano su una serie di calcoli che usano il Cepstrum con un asse di frequenza non lineare che segue la scala Mel. Per ottenere melcepstrum, il segnale audio è diviso in finestre usando dapprima una finestra di analisi, poi attraverso l’utilizzo della trasformata di Fourier. L’algoritmo continua mappando le potenze dello spettro ottenuto sulla scala mel usando finestre triangolari. Infine, si attua il logaritmo delle potenze per ognuna delle frequenze mel e facciamo la trasformata discreta coseno: gli mfcc saranno l’amplitude dello spettro risultante.

Il calcolo degli MFCC avviene grazie alla funzione extract\_mfcc appartenente alla libreria speaker\_verification\_toolkit; il calcolo della derivata avviene per mezzo della funzione delta appartenente alla libreria librosa. Si sottolinea l’utilizzo della funzione scale appartenente alla libreria sklearn, la quale attua una standardizzazione del dataset lungo ogni asse, requisito comune per molti stimatori in ambito machine-learning. Se le feature non fossero standardizzate gli stimatori potrebbero dare risultati inaspettati. I coefficienti vengono infine combinati e salvati in un’apposita cartella.



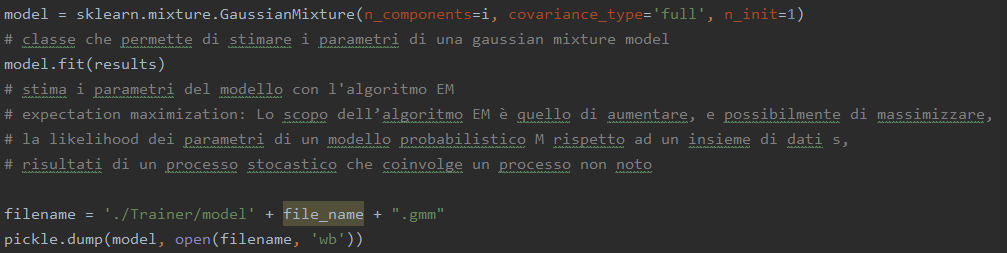
Successivamente un Gaussian Mixture Model è generato a partire dagli MFCC memorizzati per ogni singolo utente. Un GMM è un modello probabilistico che assume che tutti i data points sono generati da una combinazione di un numero finito di distribuzioni gaussiane con parametri sconosciuti. Ogni distribuzione gaussiana è spiegabile attraverso i seguenti parametri:

* Una media μ che definisce il centro
* Una covarianza Σ che definisce la larghezza.
* Una probabilità mista π che definisce quanto grande o piccola sarà la funzione gaussiana

Ciò è attuato attraverso l’utilizzo della funzione GaussianMixture appartenente alla libreria sklearn, la quale opera su una matrice contenente tutti i coefficienti appartenenti ad un singolo utente, e della funzione fit che immette i dati nella matrice. Infine, il file viene salvato per mezzo della funzione dump appartenente alla libreria pickle. La funzione fit stima i parametri del modello grazie all’algoritmo EM (expectation maximization) il quale stima la massima somiglianza statistica attraverso un procedimento iterativo al fine di ottimizzare problemi. Si basa su due passaggi fondamentali: la fase di expectation prevede di stimare le variabili mancanti, la fase di maximization tenta di ottimizzare i parametri del modello per spiegare al meglio i dati.

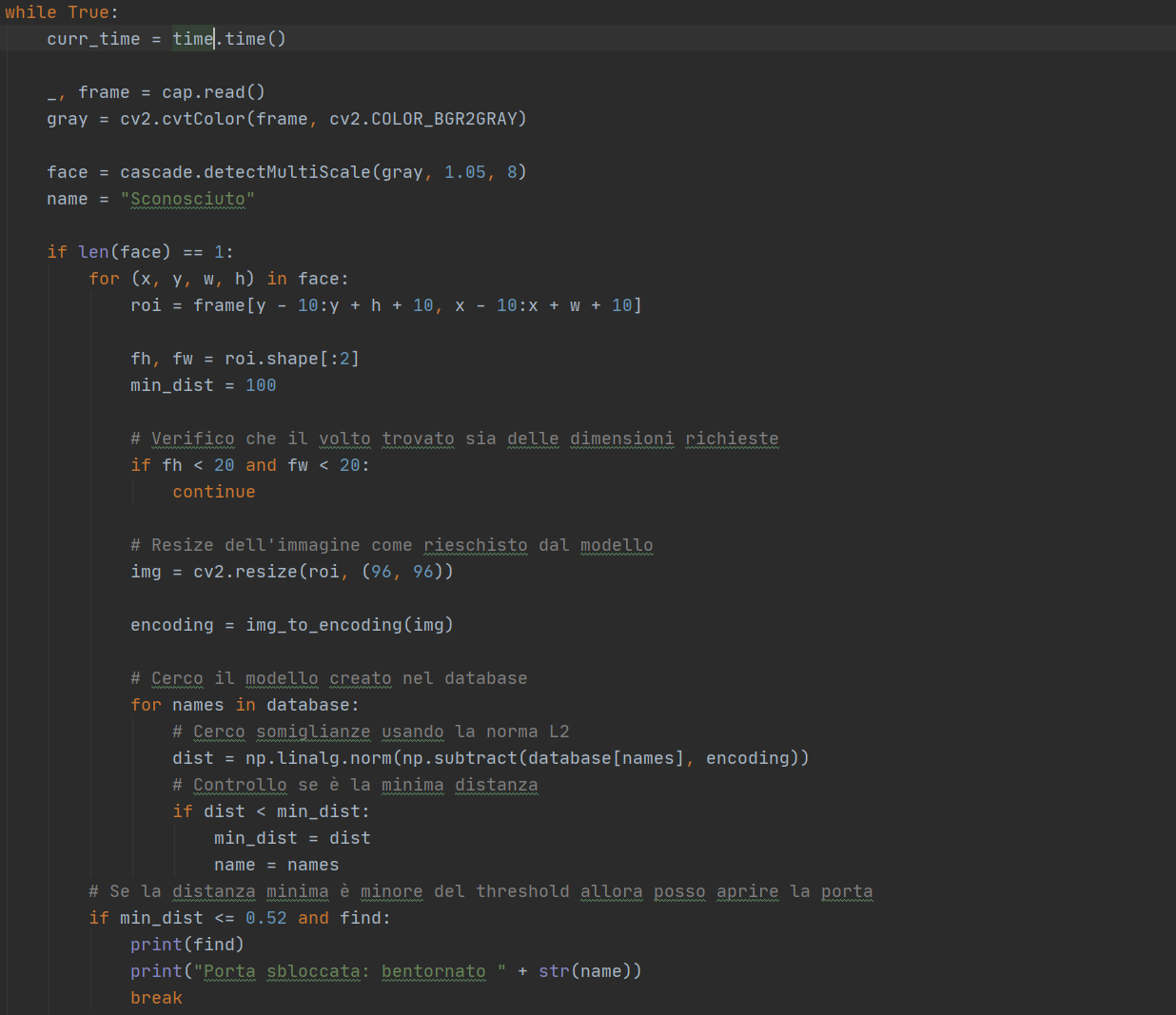
Ci sono molte tecniche per stimare i parametri di un Gaussian Mixture Model, una stima per massima somiglianza è il più comune. Nel caso in cui il nostro dataset sia compreso tra molti punti generati da diversi processi. Questi punti hanno una distribuzione di probabilità gaussiana ma il dato è combinato e le distribuzioni sono abbastanza simili che non è ovvio ci sia corrispondenza tra le distribuzioni in un determinato punto. I processi sono soliti generare un data point che rappresenti una variabile latente. Questo influenza il dato ma non è osservabile. Per questo l’algoritmo EM è un approccio appropriato da usare per stimare i parametri della distribuzione. Nell’algoritmo EM, lo step E stimerebbe il valore per la variabile lantente per ogni data point e lo step M ottimizzerebbe i parametri della distribuzione di probabilità per ottenere la migliore densità dei dati. Il processo è ripetuto finchè un buon set di valori latenti e una massima somiglianza sono ottenuti. Alcune parti sono di approfondimento

Sklearn.mixture è un package che permette di stimare Gaussian Mixture Models a partire da dati quali il numero di componenti e il tipo di covarianza.



Recognition

Volto:

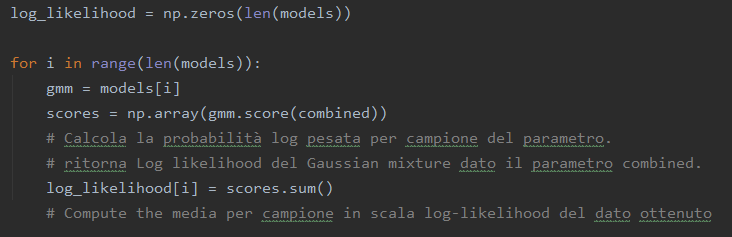


La funzione acquisisce un frame dalla videocamera, il quale viene elaborato dalla rete neurale.

Per trovare l’individuo viene esaminato il database e viene calcolata la distanza tra l’immagine acquisita e quella di ciascun individuo presente nel database. L’individuo con la distanza minima dalla nuova immagine viene quindi scelto come candidato più importante. Infine, dobbiamo determinare se l’immagine candidata e la nuova immagine contengano o meno la stessa persona. Dal momento che alla fine del nostro ciclo abbiamo determinato solo l’individuo più probabile.

Se la distanza è superiore a 0.52, l’utente non è presente nel database, altrimenti, abbiamo trovato una corrispondenza.

Il valore 0.52 è stato raggiunto attraverso una serie di tentativi. Con questo valore abbiamo pochi falsi positivi e qualche false rejection.

Voce: Durante la fase di test i GMM sono letti e, per ognuno di essi, è ottenuta una somma di score mostrata su scala logaritmica e basata sul confronto tra i modelli presenti ed un audio immesso in input da un utente talvolta registrato talvolta sconosciuto al sistema. Lo score è ottenuto grazie alla funzione omonima appartenente al Modello. Lo score maggiore appartiene a chi, secondo il sistema, appartiene con più probabilità la voce immessa.

Se lo score maggiore dovesse superate il valore di 62 avremo una corrispondenza. Il valore 62 è stato raggiunto attraverso una serie di tentativi su un data set personale formato da modelli creati appositamente.

Per mostrare all’utente dei dati sensati sia è stato attuato uno scale attraverso la scala minmax che converte il valore più grande a 1, il più piccolo a 0 e gli altri valori in proporzione, sia la scala standard che standardizza features rimuovendo la media e scalando all’unità di varianza. Lo score standard di un campione x è calcolato con la formula z = (x - u) / s dove u è la media dei campione di training ed s è la deviazione standard dei campioni di training. Le normalizzazioni sono ottenute attraverso le funzioni scale e minmax\_scale appartenenti alla libreria sklearn.preprocessing