# Face recognition

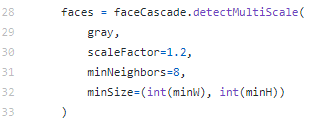
## Face detection

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:

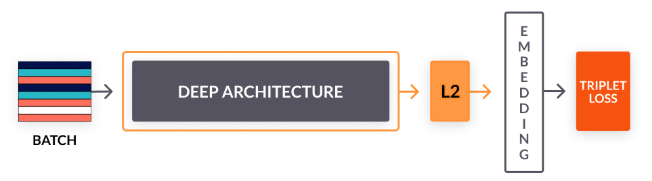


* **gray**: l’immagine in scala di grigi
* **scaleFactor**: parametro che specifica di quanto la dimensione dell’immagine deve essere ridotta. Viene utilizzato per creare la piramide della scala
* **minNeighbors**: è un parametro che specifica il numero di vicino che ogni rettangolo candidato deve avere 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.

## Raccolta dati e addestramento del modello

Queste due fasi, avvengono contemporaneamente. Quando viene identificato un volto, il frame viene processato. Durante questa fase vengono processate circa inserire il numero preciso immagini.

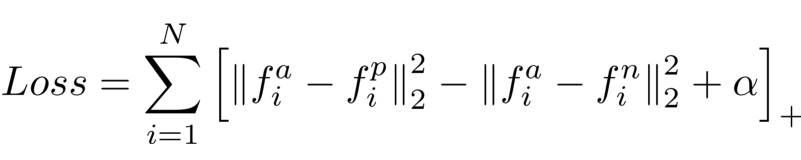
In questo progetto, viene utilizzato l’algoritmo di Google Facenet, un framework molto usato per l’implementazione di moderne applicazioni di riconoscimento facciale. FaceNet utilizza una tecnica chiamata “apprendimento one-shot”. La sua rete è composta da un livello di input batch e una rete neurale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

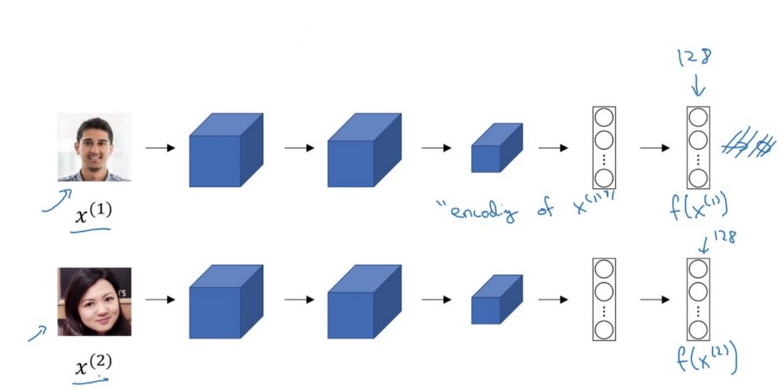
L’ultimo stadio dell’architettura FaceNet è la “perdita di tripletta” (triplet loss), che minimizza la distanza tra un’ancora e un positivo noto (somiglianza tra i due volti), massimizzando al contempo la distanza tra l’ancora e un negativo noto (dissomiglianza).



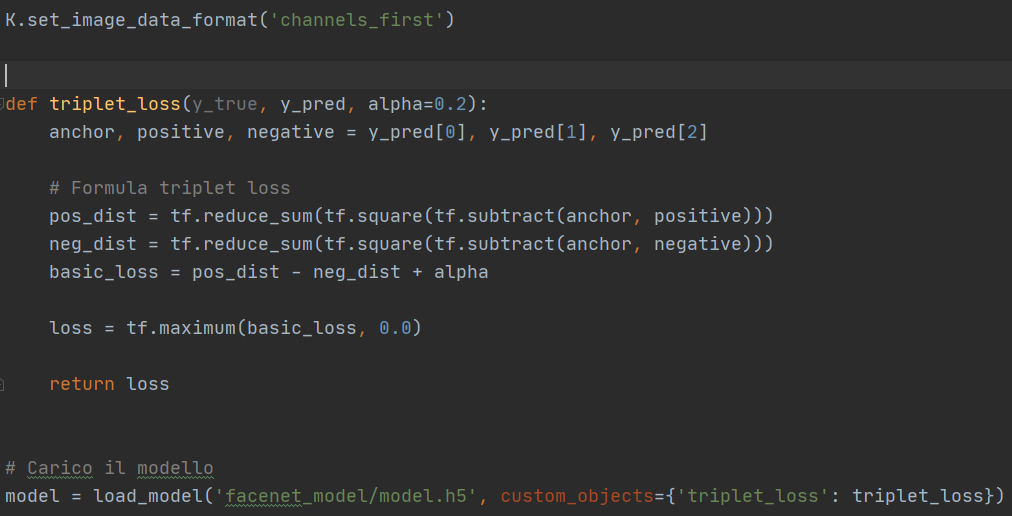
* f(a) si riferisce alla codifica di output dell’ancora.
* f(p) si riferisce alla codifica in uscita del positivo.
* f(n) si riferisce alla codifica in uscita del negativo.
* ɑ è una costante utilizzata per assicurarsi che la rete non tenti di ottimizzare verso f(a) - f(p) = f(a) - f(n) = 0.
* [...]+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



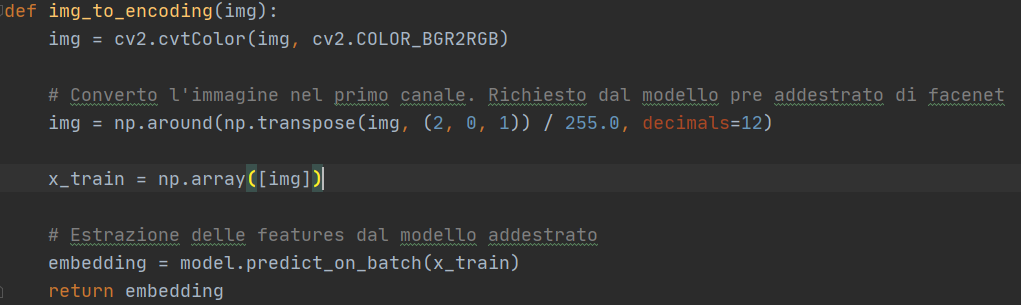
### Implementazione



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 dimensioni 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.

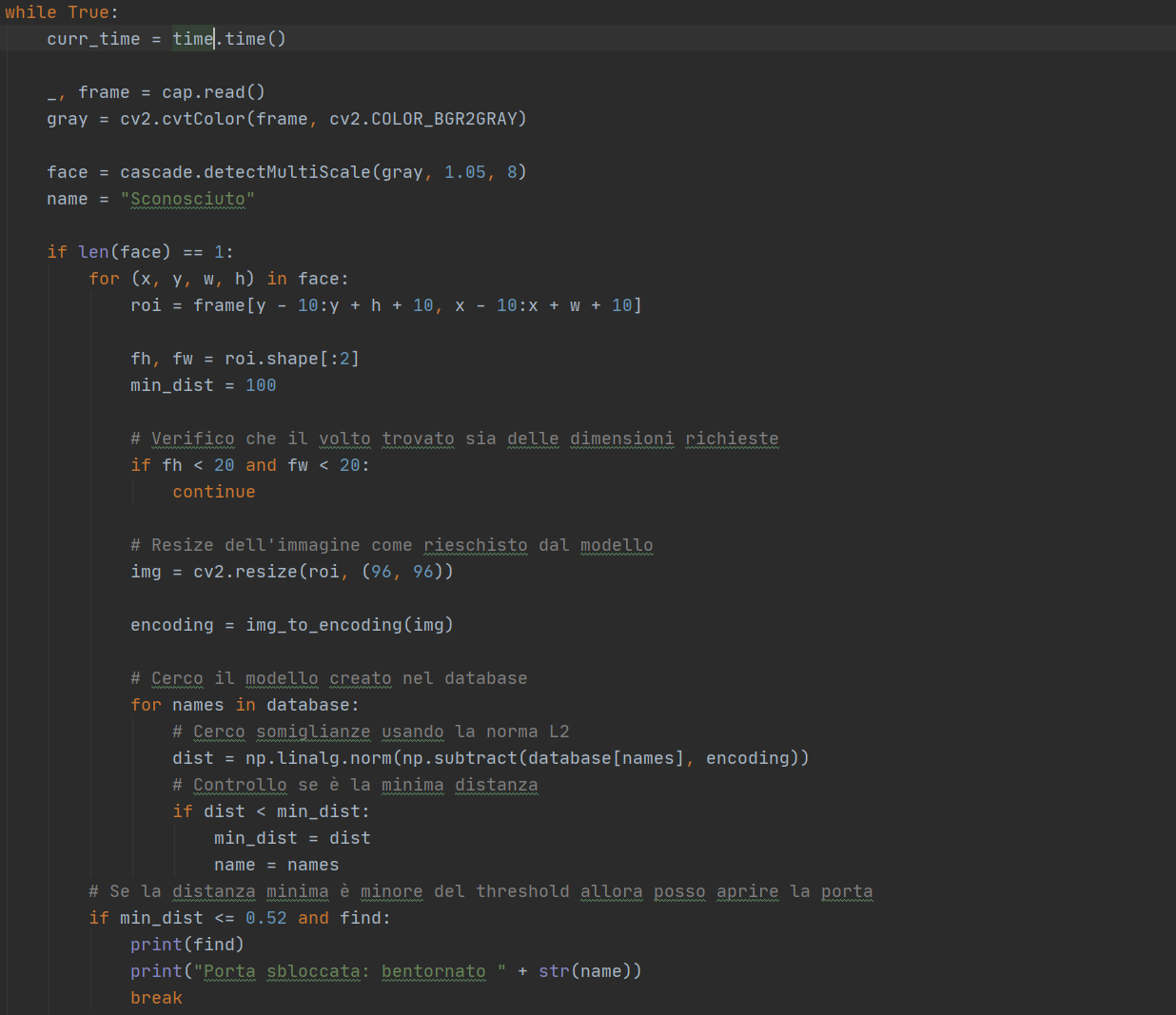
Adesso, è possibile compilare il modello di riconoscimento facciale usando Keras.



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.

## Riconoscimento del 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.