

# Riconoscimento facciale

Sistema di riconoscimento facciale per smartphone Android.

Andrea Lo Russo

Dipartimento di Ingegneria Informatica  
Università di Bologna  
andrea.lorusso6@studio.unibo.it  
0000925492

2021/2022

## Abstract

L'obiettivo del progetto proposto è integrare un sistema di riconoscimento facciale all'interno di un dispositivo mobile basato su Android. Il sistema implementato permette, sotto determinate condizioni, di inserire una o più immagini contenti un volto che saranno utilizzate come riferimento per il riconoscimento facciale. Il seguente elaborato descrive nel dettaglio la struttura del sistema e tutte le tecniche utilizzate per la sua implementazione.

Nel primo capitolo è descritta la rete neurale utilizzata per il riconoscimento facciale, nonché il dataset utilizzato. Al termine del capitolo sono invece presentati i risultati ottenuti, i limiti e i possibili miglioramenti. Il training del modello, implementato su un notebook jupyter, è stato effettuato in locale con una scheda NVIDIA GTX 1050 Ti.

Nel secondo capitolo è presentata l'applicazione Android con tutte le sue funzioni e i vincoli da rispettare per un corretto utilizzo. L'applicazione è stata realizzata con il supporto di Android Studio, il linguaggio prescelto per la sua implementazione è stato invece Kotlin. Tutti i test riguardanti l'applicativo sono stati effettuati su un dispositivo Asus, precisamente su uno ZenFone Max (M2) con Android nella versione 9.

Nel terzo e ultimo capitolo è descritto come il modello per il riconoscimento facciale è stato integrato all'interno dell'applicazione Android.

# Contents

<b>1 Siamese Neural Network</b>	<b>3</b>
1.1 Labeled Faces in the Wild . . . . .	4
1.2 Generazione triple . . . . .	4
1.3 Risultati, limiti e sviluppi futuri . . . . .	5
<b>2 Applicativo Android</b>	<b>7</b>
2.1 Primo Avvio e fragment About . . . . .	7
2.2 Fragment Settings . . . . .	8
2.3 Fragment Camera . . . . .	9
2.4 Fragment Profiles . . . . .	11
<b>3 Integrazione del modello</b>	<b>12</b>
3.1 Ambiente simulato . . . . .	12
<b>4 Conclusioni</b>	<b>13</b>

# 1 Siamese Neural Network

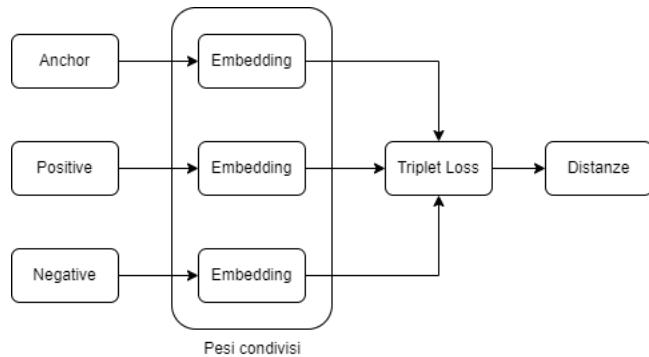
Il riconoscimento facciale è una degli argomenti più studiati nella computer vision. I migliori risultati in questo settore sono stati ottenuti grazie al deep learning, in particolar modo grazie ad una rete neurale chiamata siamese, che è stata per l'appunto utilizzata per la realizzazione di questo progetto.

Si tratta di una tipologia di architettura che si distingue per l'utilizzo di due o più sotto reti neurali identiche; dove per identiche si intende che condividono la medesima configurazione con parametri e pesi uguali. Ognuna di queste sotto reti, note come reti embedding, ha il compito di estrarre i vettori delle caratteristiche dalle immagini fornite come input. I vettori risultati saranno in seguito confrontati con un'analisi di similarità per determinare se le immagini di input appartengono o meno alla stessa classe.

Una configurazione possibile per implementare il riconoscimento facciale è quella che vede utilizzare tre sottoreti embedding e una particolare funzione loss chiamata Triplet Loss [Figure 1]. Dunque, per l'addestramento della rete, sarà necessario fornire tre diversi input, ovvero una tripla di immagini formata da:

1. un'immagine di riferimento, chiamata anchor image;
2. un'immagine di un'istanza diversa della stessa classe a cui appartiene l'immagine di riferimento, chiamata positive image;
3. un'immagine appartenente ad una classe diversa rispetto dall'immagine di riferimento, chiamata negative image.

Dato che in questo caso stiamo parlando di riconoscimento facciale, l'immagine anchor e l'immagine positive conterranno un'istanza diversa di volti appartenenti alla medesima persona, mentre l'immagine negative sarà l'immagine di un volto di un'altra persona.



**Figure 1:** Struttura del modello siamese utilizzato.

La funzione Triplet Loss ha il compito di diminuire la distanza fra i vettori delle caratteristiche associate alle immagini anchor e positive e, al contempo, aumentare la distanza fra i vettori delle caratteristiche associate alle immagini anchor e negative, ovvero raggiungere la seguente diseguaglianza:

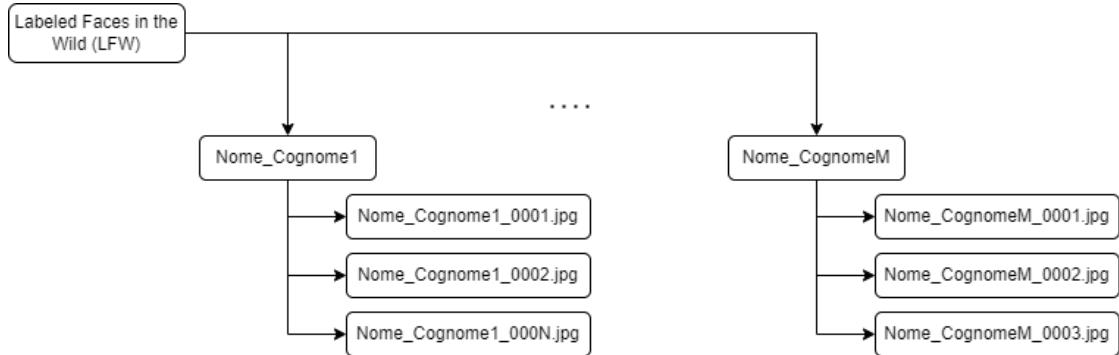
$$\|f(x_i^a) - f(x_i^p)\|^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|^2 \quad (1)$$

dove  $a$  sta per anchor image,  $p$  per positive image ed  $n$  per negative image,  $\alpha$  è invece un parametro chiamato margine che rappresenta la distanza fra i campioni positivi e i campioni negativi, mentre  $f$  è la funzione che mappa le immagini in vettori di caratteristiche, in sostanza rappresenta la funzione da imparare. In definitiva, lo scopo è minimizzare la seguente funzione Loss:

$$\sum_i^N [\|f(x_i^a) - f(x_i^p)\|^2 - \|f(x_i^a) - f(x_i^n)\|^2 + \alpha] \quad (2)$$

## 1.1 Labeled Faces in the Wild

Il dataset utilizzato per il progetto è chiamato Labeled Faces in the Wild (LFW) [1]. Si tratta di un dataset formato da 13233 immagini contenenti i volti di 5749 persone diverse. Le persone a cui sono associate due o più immagini sono invece 1680. La struttura del dataset [Figure 2] è molto semplice: ad ogni persona è associata una cartella in cui sono memorizzate un certo numero di immagini contenenti il volto della persona considerata.



**Figure 2:** Struttura del dataset LFW.

A partire dal dataset LFW è stato creato un secondo dataset chiamato LFW\_FF, dove FF sta per Frontal Face. La struttura del nuovo dataset riprende quella dell'originale, tuttavia, ogni immagine è stata preprocessata in modo tale da estrarre soltanto il volto. Ciò è stato reso possibile grazie all'utilizzo di un algoritmo di rilevamento dei volti fornito da OpenCV, chiamato Cascade Classifier [2]. In alcune immagini l'algoritmo non è stato in grado di rilevare il volto, in queste situazioni si è scelto semplicemente di escludere l'immagine dal dataset; più precisamente, sono state prese in considerazione soltanto le immagini al cui interno è stato possibile rilevare un numero di volti pari ad 1.

Il dataset LFW\_FF così generato è formato da 10759 immagini di 4934 persone diverse, mentre il numero di persone a cui sono associate due o più immagini è sceso a 1377. La dimensione del dataset è quindi diminuita, tuttavia la qualità in relazione al lavoro da svolgere è aumentata. Di seguito un'immagine estratta dal dataset originale LFW [Figure 3A] e la relativa immagine estratta dal dataset preprocessato LFW\_FF [Figure 3B].



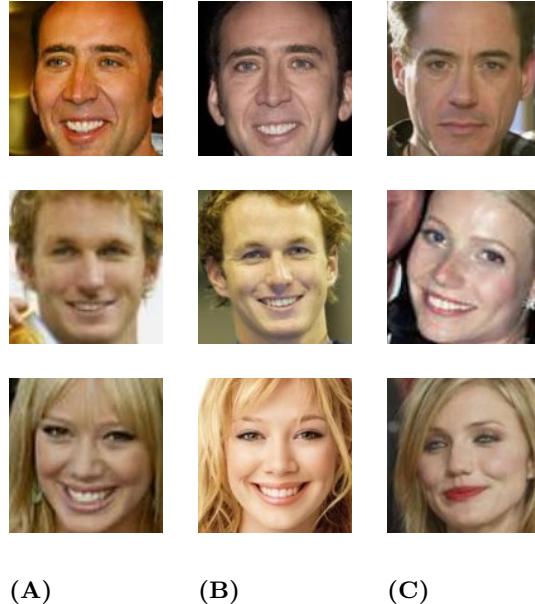
**Figure 3:** (A) Campione estratto dal dataset LFW. (B) Campione estratto dal dataset LFW\_FF.

Dal confronto precedente si evince che l'utilizzo di questa metodologia, oltre che mettere in evidenza il volto, permette di eliminare elementi di disturbo del background.

## 1.2 Generazione triple

A partire dal dataset *LFW\_FF* sono state estratte le triple di immagini necessarie per l'addestramento. Le triple sono state generate in modo casuale e salvate all'interno di una lista. Ogni tripla è descritta da una tupla di tre elementi contente i path delle immagini anchor, positive e negative. Ogni cartella contenente più di un'istanza di una persona è stata presa in considerazione per le immagini anchor e per le immagini positive. In particolare, ogni istanza è utilizzata come anchor, mentre l'immagine positiva è scelta casualmente fra le restanti istanze. Per completare la tripla viene inserito in modo causale il

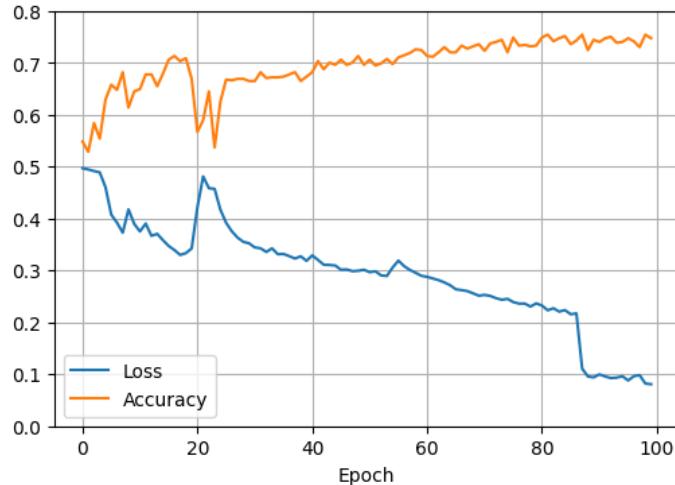
path di un’immagine negativa, ovvero non appartenente alla cartella processata. In questo caso sono considerate anche le cartelle contenenti un’unica istanza di una persona. Questa metodologia ha portato alla formazione di 7202 triple: 6282 utilizzate per il train set e le restanti 720 per il test set.



**Figure 4:** Tre esempi di triple. Nella colonna (A) sono presenti le immagini anchor, ovvero le immagini di riferimento. Nella colonna (B) sono presenti le immagini positive, ovvero immagini che appartengono alla stessa classe delle immagini anchor. Nella colonna (C) sono presenti le immagini negative, ovvero le immagini che appartengono ad una classe diversa rispetto all’immagine anchor.

### 1.3 Risultati, limiti e sviluppi futuri.

Dopo la generazione delle triple e la definizione degli iperparametri è stato avviato l’addestramento durato 100 epoche. Come già detto, le immagini utilizzate per l’addestramento appartengono al dataset LFW\_FF, ma prima di essere utilizzate sono state normalizzate e opportunamente ridimensionate. Di seguito è mostrato, per ogni epoca, l’andamento della funzione loss durante il training e l’andamento dell’accuratezza ottenuta sul test set.



**Figure 5:** Andamento della funzione loss durante il training e andamento della accuratezza sul test set.

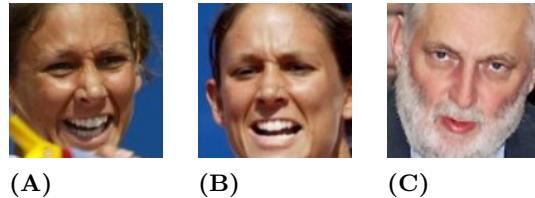
Come mostrato dal grafico [Figure 5], nelle fasi finale dell’addestramento il valore della funzione loss crolla improvvisamente raggiungendo un valore molto vicino allo zero, tuttavia il valore dell’accuratezza non

aumenta, o comunque aumenta di poco, raggiungendo un valore di circa 0.75. L'accuratezza raggiunta non risulta essere molto elevata e il motivo potrebbe essere imputabile alla generazione delle triple. Le triple possono essere infatti suddivise in tre categorie differenti:

1. easy negative:  $d(a, p) + \alpha < d(a, n)$
2. semi\_hard negatives :  $d(a, p) < d(a, n) < d(a, p) + \alpha$
3. hard negatives:  $d(a, n) < d(a, p)$

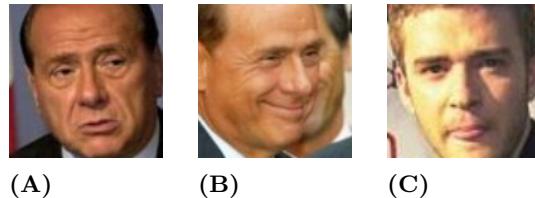
dove  $d$  rappresenta la distanza euclidea,  $a$  rappresenta il vettore delle caratteristiche associato all'immagine anchor,  $p$  quello associato all'immagine positive e  $n$  quello associato all'immagine negative. Le triple chiamate easy negative sono quelle che già soddisfano la disuguaglianza, allenare la rete su questa tipologia di triple non produce nessun miglioramento, tuttavia, il valore della funzione loss ad esse associato è pari a 0. Per questo motivo l'addestramento della rete dovrebbe avvenire escludendo le easy negative durante il training: nel presente lavoro non è stata fatta questa distinzione. Ulteriori miglioramenti potrebbero essere ottenuti effettuando un pre-processing più mirato alle immagini del dataset, in particolar modo effettuando l'allineamento dei volti; infatti, il dataset non risulta essere molto omogeneo da questo punto di vista essendo caratterizzato da volti con diverse inclinazioni ed espressioni. Sarebbe inoltre interessante arricchire il dataset o produrne uno nuovo contenente anche istanze della stessa persona cui volto è occluso dalla presenza di oggetti di uso comune come sciarpe, mascherine, occhiali da vista o da sole.

Di seguito sono commentati due esempi di predizione della rete. Nel primo esempio [Figure 6] la rete non ha difficoltà nel predire se le immagini appartengano o meno alla stessa classe. Infatti, nel caso delle immagini anchor e positive la distanza euclidea dei vettori generati dalla rete embedding risulta essere molto bassa, vicino allo zero. Al contrario, nel caso delle immagini anchor e negative la distanza euclidea dei vettori generati è molto alta, vicino a 9. Dunque settando un'opportuna soglia è possibile inferire se l'immagine considerata appartiene o meno alla stessa classe dell'immagine anchor.



**Figure 6:** (A) Immagine anchor, (B) immagine positive, (C) immagine negative. La distanza euclidea fra i vettori delle caratteristiche associati alle immagini anchor e positive (0.337) è tale da poter inferire che appartengono alla stessa classe. Al contrario la distanza fra i vettori associati alle immagini anchor e negative (8.590) è tale da poter inferire che non appartengono alla stessa classe.

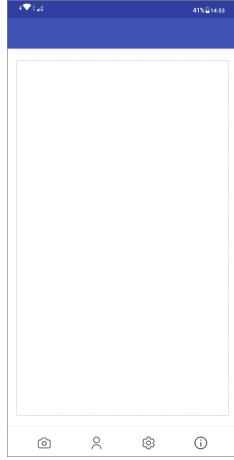
Nel secondo esempio [Figure 7], la distanza fra i vettori associati all'immagine anchor e all'immagine positive è maggiore della distanza fra i vettori associati all'immagine anchor e all'immagine negative. Questo probabilmente è causato dall'inclinazione dei volti e dalle espressioni facciali: molto simili fra l'immagine anchor e l'immagine negative, completamente differenti fra l'immagine anchor e l'immagine positive.



**Figure 7:** (A) Immagine anchor, (B) immagine positive, (C) immagine negative. La distanza euclidea fra i vettori delle caratteristiche associati alle immagini anchor e positive è sufficientemente bassa (2.604) da poter inferire, con un ragionevole dubbio, che appartengono alla stessa classe. La distanza fra i vettori associati alle immagini anchor e negative (0.572) è tale da poter inferire, erroneamente, che appartengono alla stessa classe.

## 2 Applicativo Android

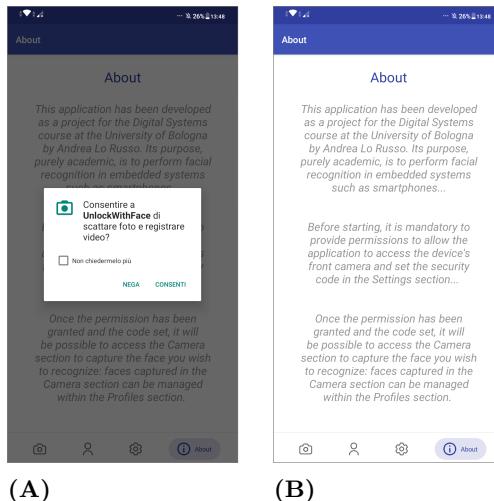
Lato Android, l'applicazione è composta da un'activity, chiamata MainActivity, all'interno della quale è presente un componente FragmentContainerView per la gestione dei diversi fragment che compongono l'applicativo. In totale sono presenti tre fragment principali più un quarto contenente alcune informazioni di carattere generale sull'applicazione e una guida rapida per il suo utilizzo [Figure 9B]. La navigazione fra i diversi fragment è resa possibile mediante l'utilizzo di una navigation bar [3, 4] posta nella parte inferiore della MainActivity [Figure 8]



**Figure 8:** Struttura della MainActivity. In basso è presente la navigation bar, mentre, la parte evidenziata con il rettangolo rappresenta il componente FragmentContainerView.

### 2.1 Primo Avvio e fragment About

Al primo avvio è necessario fornire i permessi per l'utilizzo della fotocamera, ciò è reso possibile attraverso una finestra di dialogo che consente di elargire o negare tale permesso. In assenza dei permessi, la fotocamera è inaccessibile per l'applicazione, rendendola di fatto inutilizzabile. Al contrario, forniti i permessi, il primo fragment visualizzato è chiamato About e, come già anticipato nella sezione precedente, si tratta di un semplice fragment contenente informazioni di carattere generale e una guida rapida all'uso. [Figure 9]



**Figure 9:** (A) Richiesta dei permessi per consentire all'applicazione di accedere alla fotocamera. (B) Layout fragment About.

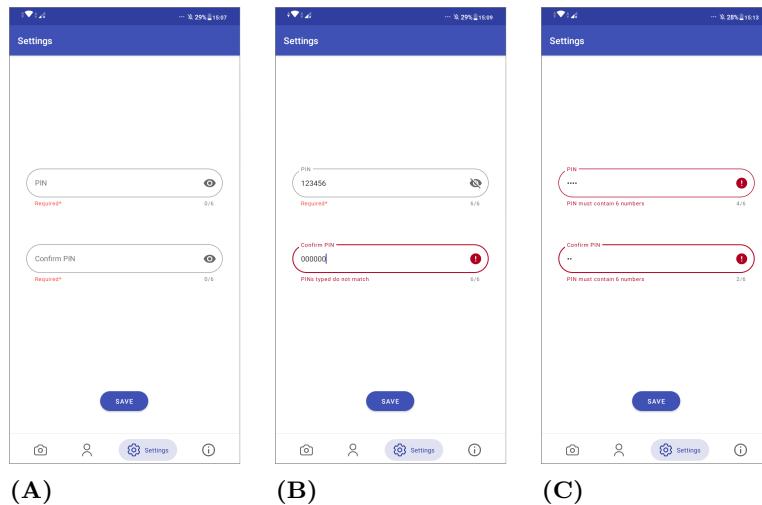
## 2.2 Fragment Settings

Accettare i permessi per l'utilizzo della fotocamera non è l'unico vincolo per il funzionamento dell'applicazione. Di fatti, come suggerito dalla guida rapida, per avere accesso a tutte le funzionalità dell'applicativo è necessario impostare un codice di sicurezza, ovvero un PIN.

L'utilizzo di un PIN è una prerogativa comune per tutti i sistemi di riconoscimento, principalmente per due motivi: in primo luogo consente l'accesso ad un qualsivoglia tipo di sistema, anche qualora il riconoscimento scelto come default dovesse fallire; in secondo luogo, non consente ai malintenzionati di effettuare cambiamenti nel paradigma del sistema di riconoscimento (come, ad esempio, l'inserimento di nuovi profili o la cancellazione di quelli preesistenti). Per utilizzare un codice di sicurezza all'interno di un'applicazione Android è necessario che questo sia persistente, ovvero non vada perso alla chiusura dell'applicazione. Dunque, deve essere memorizzato da qualche parte all'interno dell'applicazione stessa. Per realizzare questo comportamento è stato utilizzato il Jetpack DataStore che permette di memorizzare coppie chiave valore. Nel caso di un codice di sicurezza è sufficiente definire una sola coppia chiave valore e, all'occorrenza, impostare o modificare il valore associato alla chiave.

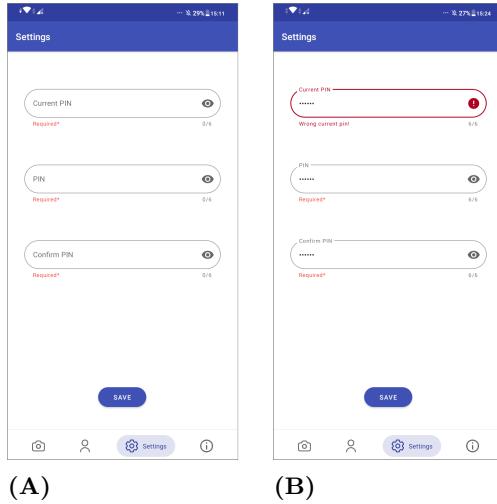
Il fragment che consente di impostare un codice di sicurezza è chiamato Settings. Alla sua apertura sono presenti due form, realizzati mediante l'utilizzo della View EditText in stile Material Design, e una View Button. Per impostare un PIN è dunque necessario compilare questi due form, denominati PIN e Confirm PIN, e cliccare sul Button, denominato invece SAVE. [Figure 10A]

L'impostazione del codice di sicurezza avrà successo soltanto se i due codici coincidono e se sono rispettati ulteriori due vincoli: il codice di sicurezza deve essere infatti formato esclusivamente da sei caratteri di tipo numerico. Se questi vincoli risultano essere rispettati, allora il PIN sarà impostato con successo, in caso contrario apparirà un diverso messaggio di errore a seconda del vincolo violato [Figure 10]. Da sottolineare che il vincolo numerico non può essere violato e di conseguenza non è stato associato nessun messaggio di errore.



**Figure 10:** (A) Layout del fragment Settings. (B) L'applicazione segnala che i PIN immessi non corrispondono. (C) L'applicazione segnala che il vincolo sulla lunghezza non è rispettato. Attualmente, l'unico compito di questo framgment è la gestione del PIN di sicurezza, tuttavia è stato denominato in modo più generale come Settings per lasciare spazio all'inserimento di ulteriori impostazioni in un secondo momento.

Impostato il codice di sicurezza il fragment settings sarà arricchito da un ulteriore form, denominato Current PIN. Infatti, a patto di conoscere il PIN impostato, è possibile sostituirlo con un nuovo codice di sicurezza. [Figure 11]

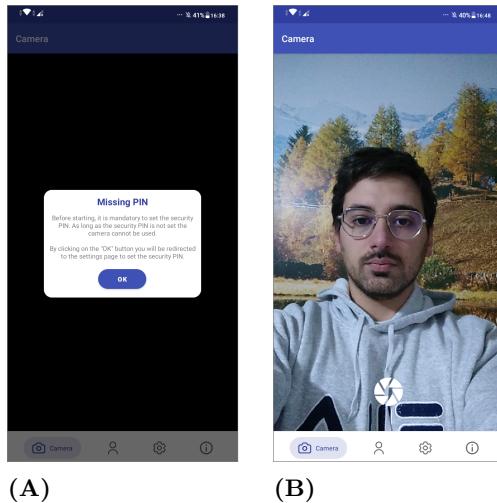


**Figure 11:** (A) Layout del fragment Settings dopo aver impostato un PIN. (B) L'applicazione segnala che il PIN attuale inserito è errato.

Infine, il Button Save non ha il solo scopo di salvare il PIN: se infatti il PIN è salvato con successo il fragment corrente (ovvero Settings) sarà sostituito con il fragment Camera.

### 2.3 Fragment Camera

Il fragment dedicato alla gestione della fotocamera permette di catturare e salvare frame, che saranno in seguito utilizzati come immagini di riferimento per effettuare il riconoscimento facciale. All'apertura, il fragment si presenta in maniera essenziale, infatti è presente soltanto una View per la visualizzazione dei frame acquisiti dalla fotocamera, che copre l'intera superficie del fragment, e da un Button di forma circolare per catturare i frame desiderati [Figure 12].



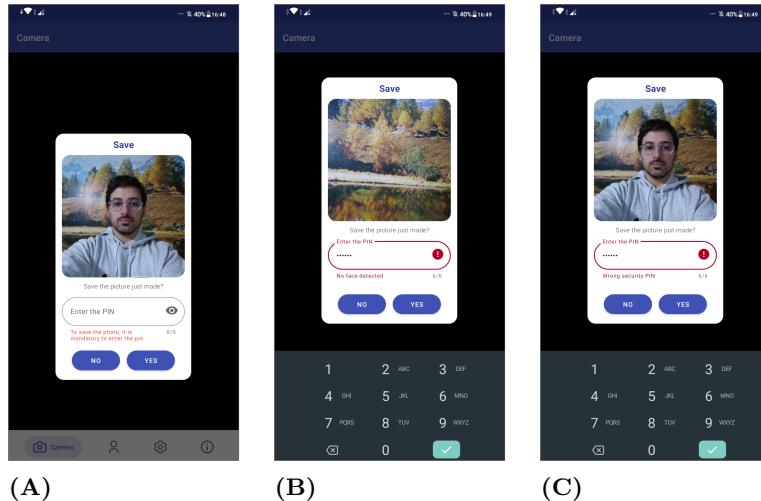
**Figure 12:** (A) L'applicazione segnala attraverso una finestra di dialogo che il PIN non è stato impostato. Fintanto che il PIN non è impostato non è possibile accedere al fragment Camera. Premendo il tasto OK della finestra di dialogo, il fragment corrente è sostituito con il fragment Settings per consentire all'utente di impostare il PIN e accedere al fragment Camera. (B) Layout del fragment camera.

Per implementare il comportamento appena descritto è possibile far riferimento alla libreria JetPack CameraX che, a seconda delle esigenze, mette a disposizione diversi usecases. In particolare:

1. la visualizzazione dei frame acquisiti è resa possibile dall'implementazione dello usecase Preview, che permette, fra le altre cose, di decidere se utilizzare la camera principale o quella frontale. Per lo scopo di questa applicazione, la camera frontale dello smartphone è stata impostata come predefinita;
2. la cattura dei frame è resa invece possibile dallo usecase ImageCapture, che si attiva alla pressione del button circolare descritto precedentemente.

Solitamente la pressione del Button comporta la cattura del frame e il suo salvataggio all'interno della galleria fotografica dello smartphone. Tuttavia, per rispondere alle esigenze dell'applicazione sono stati effettuati alcuni cambiamenti al comportamento predefinito. Infatti, alla pressione del button il frame catturato (ma non ancora salvato) sarà visualizzato su di una finestra di dialogo, insieme ad una EditText e a due Button [Figure 13A]. Per procedere con il salvataggio è necessario inserire nella EditText il codice di sicurezza impostato precedentemente nel fragment Settings e cliccare sul button YES. Tuttavia, il codice di sicurezza da solo non sarà sufficiente per confermare il salvataggio; per confermarlo, il frame catturato deve necessariamente contenere un numero di volti pari ad uno. Per verificare la presenza di un volto sul frame catturato è stato applicato un algoritmo di face detection fornito dal kit di Machine Learnig di Google [Figure 13B].

Se il frame rispetta i vincoli appena descritti, la pressione del button YES nella finestra di dialogo comporterà il salvataggio del volto ritagliato dal frame. Se al contrario i vicoli non sono rispettati o il codice di sicurezza è errato apparirà un messaggio di errore attinente all'errore commesso [Figure 13B, 13C].

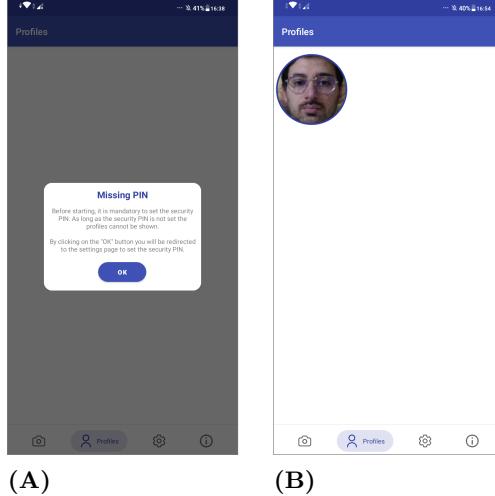


**Figure 13:** (A) Finestra di dialogo che mostra l'immagine appena catturata. (B) L'applicazione segnala che sull'immagine catturata non è stato rilevato alcun volto, pertanto il salvataggio non può essere portato a termine con successo. (C) L'applicazione segnala che il PIN inserito è errato, pertanto il salvataggio non può essere portato a termine con successo. Cliccando sul button NO la finestra di dialogo è dismessa, permettendo all'utente di catturare una nuova immagine.

Una seconda modifica riguarda la posizione di salvataggio del frame appena catturato; infatti, questo non sarà salvato all'interno della galleria immagini, ma all'interno dell'applicazione stessa dunque, sarà visibile soltanto dall'interno di quest'ultima. Se infatti le immagini di riferimento (anchor) fossero salvate all'interno della galleria potrebbero essere manipolate da un malintenzionato. Il salvataggio del frame all'interno dell'applicazione comporta la sostituzione del fragment Camera con il fragment Profiles.

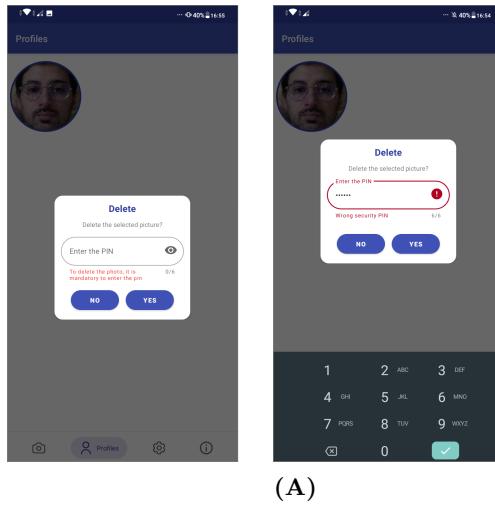
## 2.4 Fragment Profiles

I frame salvati all'interno dell'applicazione sono visualizzati nel fragment Profiles mediante l'utilizzo di una RecyclerView. Dunque, tutte le immagini salvate all'interno dell'applicazione sono visualizzate all'interno di una lista [Figure 14B].



**Figure 14:** (A) L'applicazione segnala attraverso una finestra di dialogo che il PIN non è stato impostato. Fintanto che il PIN non è impostato non è possibile accedere al fragment Profiles. Premendo il tasto OK della finestra di dialogo, il fragment corrente è sostituito con il fragment Settings per consentire all'utente di impostare il PIN e accedere al fragment Profiles. (B) Layout del fragment Profiles con all'interno un unico profilo, tuttavia possono essere inseriti ulteriori profili.

Ogni elemento della lista è cliccabile: il click produce l'apertura di una finestra di dialogo che consente la cancellazione del profilo previo inserimento del PIN. La conferma o la negazione di questa operazione è gestita attraverso l'utilizzo di due button all'interno della finestra di dialogo [Figure 15].



**Figure 15:** (A) Finestra di dialogo per la cancellazione di un profilo. (B) L'applicazione segnala che il PIN inserito non è corretto.

L'inserimento di questa funzionalità è stata resa necessaria dal fatto che le immagini non sono accessibili, e quindi eliminabili, dalla gallerie dello smartphone.

### 3 Integrazione del modello

Per poter effettuare il riconoscimento facciale, resta da integrare il modello nell'applicazione Android. Sfortunatamente Android non mette a disposizione API per la gestione dello sblocco del dispositivo a livello applicazione. Un comportamento simile potrebbe essere ottenuto mediante l'utilizzo di un Broadcaster Receiver per inviare un messaggio di sistema all'applicazione quando il dispositivo si risveglia. Tuttavia, per ottenere un comportamento ottimale il lavoro richiesto sarebbe stato eccessivo. Per questo motivo ho deciso di simulare lo sblocco del dispositivo attraverso l'utilizzo di un'activity denominata FakeHomeActivity.

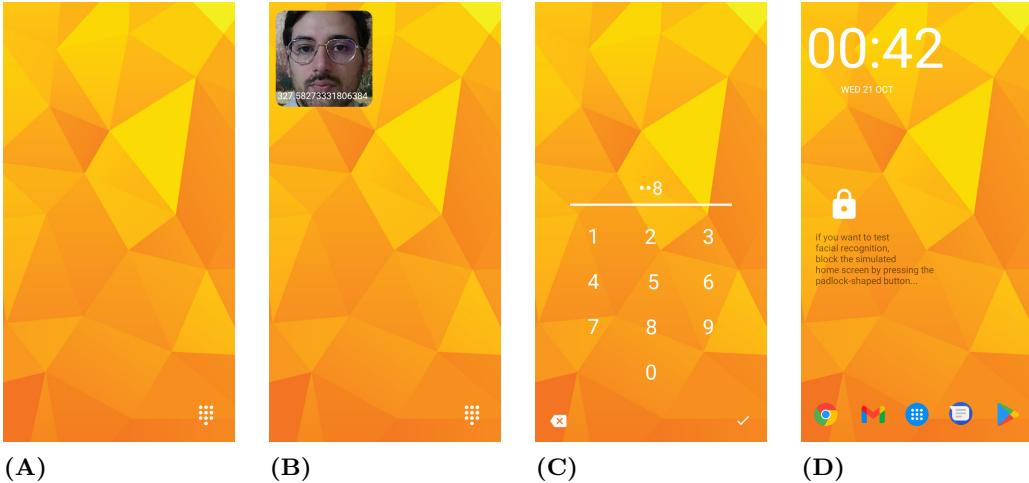
#### 3.1 Ambiente simulato

La FakeHome è stata nascosta all'interno dell'applicazione stessa. Per accedervi è necessario prima di tutto soddisfare due requisiti, ovvero aver impostato un PIN e avere a disposizione almeno un'immagine di riferimento nella sezione Profiles. Soddisfatti questi due requisiti, è possibile entrare nell'ambiente simulato cliccando sulla TextView About nel fragment omonimo.

Per la realizzazione dell'ambiente simulato sono stati utilizzati tre fragment, uno per il riconoscimento facciale, uno per l'inserimento del PIN e l'ultimo per simulare la pagina iniziale di un dispositivo dopo essere stato sbloccato.

All'avvio della simulazione, il primo fragment visualizzato è quello chiamato FaceRecognition [Figure 16A, 16B]. All'interno di questo fragment è stato istanziata la rete embedding, che non fa altro che prendere in ingresso un'immagine, opportunamente prepreoccesata, e fornire come output il vettore delle caratteristiche associato all'immagine di input. Il fragment è stato sviluppato ancora una volta grazie alle funzionalità di CameraX. Come per il fragment Camera, è stato sfruttato lo usecase Preview che permette di visualizzare i frame acquisiti dalla camera frontale, ancora una volta impostata come predefinita. Tuttavia, in questo caso, seppur presenti, i frame acquisiti non sono visualizzati [Figure 16A], ma analizzati mediante lo usecase ImageAnalysis. Per effettuare il riconoscimento facciale è necessario rilevare i volti acquisiti dalla camera frontale: ancora una volta il rilevamento dei volti è stato implementato grazie al kit di Machine Learning di Google. Se nei frame acquisiti il numero di volti rilevati è diverso da uno, allora l'analisi sui frame non è approfondita. In caso contrario il volto rilevato viene ritagliato e ridimensionato per poter essere utilizzato come input del modello embedding. Estratto il vettore delle caratteristiche dal volto rilevato, questo viene confrontato, attraverso l'utilizzo della distanza euclidea, con i vettori delle caratteristiche estratti dalle immagini anchor della sezione Profiles. Se il vettore estratto da uno dei frame acquisiti è sufficientemente simile ad uno dei vettori estratti dalle immagini della sezione Profiles, allora il riconoscimento avrà successo e il dispositivo sarà sbloccato. Caso in cui il fragment per il riconoscimento facciale è sostituito con il fragment che simula la pagina iniziale [Figure 16D]. Per avere un riscontro visivo, i frame preprocessati sono visualizzati dinamicamente insieme alla distanza euclidea fra i vettori, entrambe le View non sono visualizzate se il numero di volti rilevati è diverso da uno [Figure 16B].

Se il riconoscimento facciale fallisce è possibile premere il Button a forma di tastierino numerico in basso a destra che non fa altro che rimpiazzare il fragment corrente per il riconoscimento facciale con il fragment dedicato all'inserimento del PIN [Figure 16C]. Si tratta di un fragment che contiene semplicemente un tastierino numerico per inserire il PIN impostato nella sezione Settings. Se il PIN inserito coincide con il PIN impostato, il dispositivo sarà sbloccato e, come per il fragment FaceRecognition, sarà rimpiazzato dal fragment che simula la pagina iniziale [Figure 16D].



**Figure 16:** (A) Layout all'avvio della simulazione: In questo stato il dispositivo simulato risulta bloccato. (B) Per avere un riscontro visivo, i frame preprocessati sono visualizzati dinamicamente insieme alla distanza euclidea fra i vettori, entrambe le View non sono visualizzate se il numero di volti rilevati è diverso da uno. Da sottolineare che in questo caso le immagini non sono state normalizzate, per questo motivo la distanza euclidea aumenta rispetto a quella vista nel capitolo 1, tuttavia, modificando opportunamente la soglia, i risultati sono identici. (C) Layout del fragment dedicato all'inserimento del PIN. (D) Layout della Home simulata una volta che il dispositivo è stato sbloccato: premendo il Button a forma di lucchetto è possibile riavviare la simulazione daccapo, in altre parole bloccare il dispositivo simulato.

## 4 Conclusioni

Il sistema sviluppato permette dunque di simulare lo sblocco di un dispositivo tramite l'utilizzo del riconoscimento facciale. Sia l'applicativo Android che il modello presentano margini di miglioramento.

Per quanto riguarda il modello, come già descritto nella sezione 1.4, gli aspetti migliorabili riguardano principalmente la generazione delle triple e il preprocessing del dataset. Inoltre per aumentare l'accuratezza e diminuire i tempi di allenamento è possibile sfruttare le potenzialità del Transfer Learning.

Per quanto riguarda l'applicazione Android il primo ostacolo da superare riguarda il suo effettivo utilizzo come sistema di riconoscimento senza far ricorso ad un ambiente simulato; infatti, come descritto nella prefazione del capitolo 3, Android non dispone di API per la gestione dello sblocco del dispositivo, dunque è necessario aggirare in qualche modo questo limite. Inoltre potrebbero essere aggiunti ulteriori vincoli per l'acquisizione delle immagini di riferimento, come ad esempio il livello di luminosità e la posizione del volto. Attualmente l'unico vincolo affinché l'acquisizione vada a buon fine è il numero di volti rilevati, che deve essere necessariamente uguale a uno.

## Riferimenti

- [ 1 ] Labeled Faces in the Wild (<http://vis-www.cs.umass.edu/lfw/>)
- [ 2 ] Cascade Classifier ([https://docs.opencv.org/3.4/db/d28/tutorial\\_cascade\\_classifier.html](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html))
- [ 3 ] Interfaccia applicazione (<https://github.com/ismaeldivita/chip-navigation-bar>)
- [ 4 ] Icône (<https://www.flaticon.com/>)