

**ALMA MATER STUDIORUM - UNIVERSITÀ
DI BOLOGNA**

DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA

**PROGETTO E ATTIVITÀ PROGETTUALE IN SISTEMI
DIGITALI M**

EYE TRACKER

Luigi di Nuzzo - Daniele Foschi - Filippo Veronesi

Anno Accademico 2021/2022

Maggio 2022

Indice

1	Introduzione	1
2	Rete Neurale	2
2.1	Dataset	2
2.2	Training	3
2.3	TFLite	4
2.3.1	Metadata	4
3	Progetto Android	5
3.1	Nerd Mode	5
3.1.1	Bottone Analyze	7
3.2	Calibration	8
3.3	Game	9
4	Conclusioni	10

Elenco delle figure

1	labelmap.pbtxt	2
2	Testing GUI TkAgg	4
3	Homepage App	5
4	Nerd mode iniziale	6
5	Nerd mode con analyze attiva	7
6	Schermata Calibration	8
7	Schermata Game	9

1 Introduzione

Questo progetto prevede la realizzazione di un applicativo Android che, sfruttando una rete neurale, sia in grado di riconoscere il volto dell'utente e di questo ultimo, anche i propri occhi. Il sistema sfrutterà un modello di rete neurale addestrato in modo tale da essere in grado di riconoscere i visi e gli occhi di uno o più utenti grazie ad un dataset di 10 mila immagini di visi umani. L'applicativo permetterà di svolgere un filtraggio "live", cioè aprendo la camera frontale ed esterna dello smartphone e individuando real-time gli occhi dell'utente.

Inoltre, è stato implementato un gioco composto da un Quiz di 4 domande aiutato da un tool di calibrazione. L'utente tramite il riconoscitore di occhi è in grado di rispondere ad una certa domanda spostando, tramite gli occhi e il cellulare, un puntatore verso la risposta giusta posta ai 4 angoli del dispositivo. In caso di risposta corretta, viene posta un'ulteriore domanda, altrimenti si viene avvertiti della domanda sbagliata tramite un alert.

Tale caso di studio è un classico esempio di applicazione di Machine Learning e il software farà ricorso a una rete neurale convoluzionale (CNN). Tale scelta è dovuta al fatto che una rete neurale rappresenta il modo più comodo e pratico per problemi di object detection, come quello di questa attività in cui vengono individuati gli occhi.

L'applicativo, inoltre, è pensato per la piattaforma Android e quindi tale progetto pone attenzione anche all'uso di risorse in quanto dovrà funzionare su smartphone, ovvero dispositivi embedded.

2 Rete Neurale

L'obiettivo primario di questo task è produrre un modello di rete neurale addestrata in grado di riconoscere gli occhi di una o più persone. A tal scopo si è utilizzato un modello messo a disposizione dalla libreria *TensorFlow 2 Detection Model Zoo* ottimizzato per il training di modelli specifici per object detection di immagini. Visto l'ambito dei sistemi embedded nel quale il progetto complessivo si colloca si è optato per un modello SSD MobileNet¹, una CNN pensata per dispositivi mobile. MobileNet è il primo modello di computer vision pensato per dispositivi embedded basato su TensorFlow. MobileNet è sufficientemente leggera e veloce da essere eseguita su smartphone senza consumo di risorse eccessivo mantenendo comunque una precisione adeguata.

2.1 Dataset

Per il training del modello si è utilizzato un dataset pubblico rilasciato da Google (LINK DA INSERIRE) contenente 10 mila immagini di visi umani, dove ogni foto conteneva da uno a più visi umani.

Oltre alle immagini il dataset conteneva un utile file Excel che indicava le coordinate della posizione degli occhi delle persone all'interno dell'immagine.

Prima dell'addestramento della rete era però necessario convertire questi dati di input in file XML in modo tale da poterli poi convertire in TFRecord, utili per TensorFlow. Per questo motivo è stato implementato uno script Java in grado di creare un file XML per ogni foto del dataset con al suo interno le informazioni geografiche della posizione degli occhi. Inoltre, si è reso necessario compilare il file labelmap.pbtxt con al suo interno i nomi e gli id di nostri oggetti da riconoscere: nel nostro caso un solo item di id pari a 1 e con nome "Human eye".

Figura 1: labelmap.pbtxt

```
item {  
    id: 1  
    name: 'Human eye'  
}
```

¹SSD MobileNet V2 FPNLite 640x640

2.2 Training

Si è quindi proceduto all’addestramento della rete tramite Python usando TensorFlow e le API di Keras. Si è reso necessario modificare il file *pipeline.config* adattandolo alle nostre esigenze:

- *numclasses* che corrisponde al numero di item da identificare;
- *path* vari da adattare al nostro workspace, tra cui il path per i checkpoint, per i record di input e per il labelmap.

Durante il procedimento di training si sono tenuti controllati i valori della nostra rete. In particolare, abbiamo usato **Tensorboard**: un framework grafico utile a capire l’andamento della nostra rete, infatti abbiamo avuto modo di controllare ad ogni training le performance della nostra rete di machine learning.

TensorBoard fornisce la visualizzazione e gli strumenti necessari per la sperimentazione del machine learning:

- Monitoraggio e visualizzazione di metriche come perdita e precisione;
- Visualizzazione del grafico del modello (operazioni e livelli);
- Visualizzazione degli istogrammi di pesi, distorsioni o altri tensori man mano che cambiano nel tempo.

INSERIRE IMMAGINI GRAFICI

Si è ottenuto in output un modello addestrato e pronto all’uso.

Prima di testare direttamente su Android, abbiamo deciso prima di testare e analizzare la nostra nuova rete utilizzando la libreria Matplotlib, utile per visualizzare graficamente via shell i nostri risultati. In particolare abbiamo usato **Tkinter**, l’unico framework GUI incluso nella libreria standard di Python.

A fini di testing si è preso ad esempio il volto di un personaggio pubblico, quello di Albert Einstein, e anche un’immagine contenente più persone in modo tale da poter verificare la correttezza e la precisione della rete anche in condizioni più difficili.



(a) Test personaggio famoso

(b) Test multiplo

Figura 2: Testing GUI TkAgg

2.3 TFLite

Dopo esserci accertati che la rete funzionasse correttamente e avesse dei livelli di precisione sopra una certa soglia, si è ottenuto in output un modello addestrato e pronto all'uso, che poi è stato convertito e quantizzato in un formato adatto ai sistemi embedded, quello di TensorFlow Lite.

2.3.1 Metadata

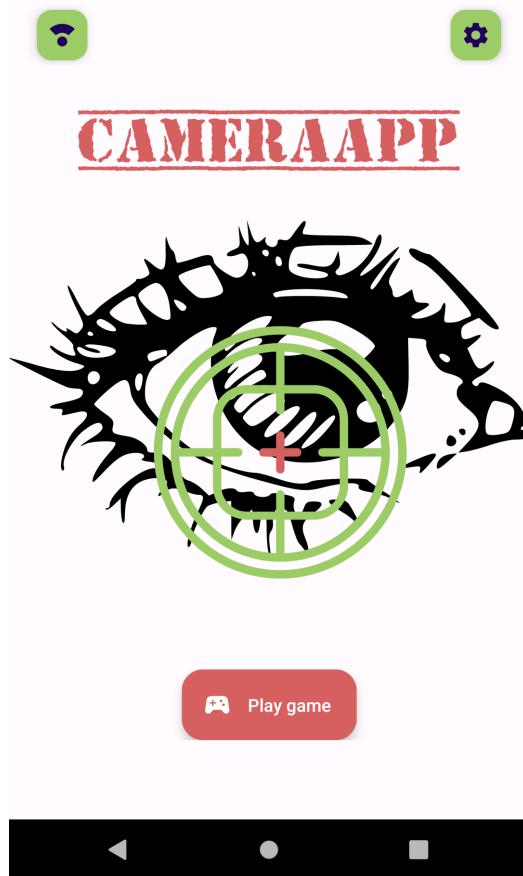
Per un corretto funzionamento della rete neurale all'interno di Android, tflite richiede che venga generato anche un file Metadata che contiene le informazioni necessarie per pre-processare le immagini. Questo file si rende necessario in quanto i nostri tensori di input sono di tipo kTfLiteFloat32.

Una volta generato questo nuovo file metadata.tflite siamo pronti ad importarlo all'interno della nostra applicazione Android come file di Machine Learning "ml".

3 Progetto Android

L'applicazione Android una volta avviata presenta la seguente interfaccia:

Figura 3: Homepage App



Sono da segnalare i 3 principali bottoni dell'applicazione:

- **Nerd Mode** in alto a destra;
- **Calibration** in alto a sinistra;
- **Play Game** al centro dello schermo.

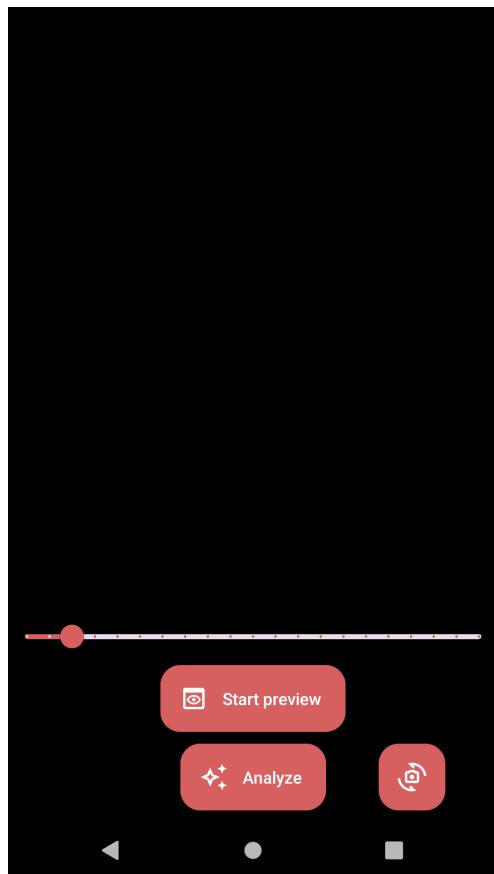
3.1 Nerd Mode

Modalità la cui utilità è quella di individuare real time gli occhi della persona, in particolare permette di vedere i boxes creati dalla rete neurale attorno agli occhi dell'utente.

Interfaccia che presenta 3 bottoni:

- **Preview** che permette di avviare la propria fotocamera frontale o esterna;
- **Analyze** che permette la visualizzazione dei boxes creati dalla rete neurale in real time;
- **Fotocamera** frontale/esterna per cambiare da una fotocamera all'altra.

Figura 4: Nerd mode iniziale

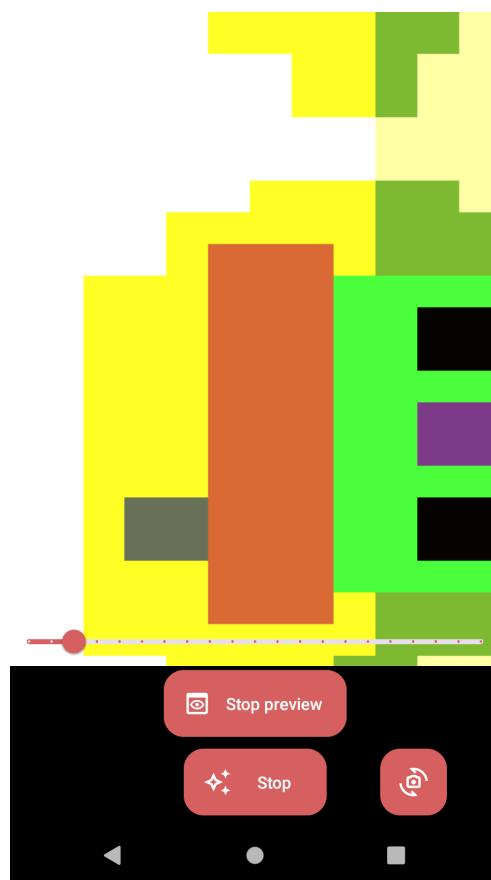


3.1.1 Bottone Analyze

Bottone premuto una volta avviata la *Preview*, ci permette di vedere per la prima volta la nostra rete addestrata al lavoro su Android. È stata inserita anche una barra di scorrimento in modo tale da impostare e mostrare real time a schermo gli occhi individuati solo sopra una certo livello di precisione della rete. (0% minimo - 100% massimo). Su Android se si inquadrano bene gli occhi, in media si ha una precisione del 80%.

Chiaramente più viene allontanato il telefono dalla faccia, più cala la precisione, ma comunque la rete individua sempre e comunque gli occhi a meno che non si provino a nascondere gli occhi o ci si metta di profilo.

Figura 5: Nerd mode con analyze attiva



3.2 Calibration

Modalità usata per calibrare al meglio la fotocamera e ottenere risultati i più veritieri possibili. Utile per svolgere al meglio il mini gioco descritto in seguito. Qui di sotto la sua schermata.

Figura 6: Schermata Calibration



3.3 Game

Modalità usata per completare un mini-gioco sfruttando le potenzialità della rete neurale sottostante. Tramite un mini gioco Quiz infatti, l'utente può rispondere alle domande del Quiz semplicemente spostando il cellulare verso uno dei 4 angoli dove è posizionata la risposta corretta: infatti viene riconosciuto l'occhio e viene visualizzato un puntatore che lo identifica sullo schermo. La domanda viene posta al centro dello schermo per motivi pratici.

I dati delle domande e delle risposte sono state recuperate da uno dei tanti servizi di Api pubblici che si trovano in rete: tramite appunto una richiesta Url, vengono ricevuti i dati casuali in formato JSON per una singola domanda, e una volta ricevuta la risposta, corretta o sbagliata che sia, viene generata un'altra richiesta in modo tale da poter continuare il mini gioco.

Figura 7: Schermata Game



4 Conclusioni

Gli obiettivi del progetto sono stati raggiunti a pieno e con risultati soddisfacenti.

L'applicativo è conforme alle aspettative e svolge i compiti adeguatamente. Le parti realizzate in Android risultano con prestazione adeguate.

Il modello addestrato restituisce un output corretto la maggior parte delle volte con una precisione di eye tracking del 95%.