ALMA MATER STUDIORUM –UNIVERSITÀ DI BOLOGNA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria Informatica

Progetto di Sistemi Digitali M

SIGNTRACKER

Autore:
Antonio Sarchione

Anno Accademico 2022-2023

Indice

Capitolo 1

Introduzione

SignTracker è un progetto che prevede la realizzazione di un applicativo Android che, tramite l'utilizzo di due reti neurali, si renda in grado di riconoscere con un'elevata precisione le lettere dell'alfabeto ASL, ovvero American Sign Language, realizzate tramite l'utilizzo delle mani dell'utente.

Il sistema avrà l'obiettivo di sfruttare due modelli di rete neurale addestrati in modo tale da essere in grado di riconoscere in un primo momento la mano di uno o più utenti e solo in un secondo momento di riconoscere anche la simbologia della mano individuata, sfruttando due dataset differenti:

- uno contenente un totale di circa 15 mila immagini raffiguranti una o più mani di persone
- l'altro contenente un totale di circa 38 mila immagini, da suddividere per le 25 lettere utilizzate (la Z è stata esclusa dal dataset perché nel linguaggio dei segni viene realizzata effettuando un preciso movimento)

L'applicativo permetterà il riconoscimento dei segni in tempo reale, sfruttando sia la camera frontale che la camera posteriore dello smartphone.

Oltre ciò, all'interfaccia principale sono stati aggiunti alcuni pulsanti che permettono all'utente di interagire in real-time con le rilevazioni.

Questo caso di studio è un evidente esempio di applicazione di Machine Learning, dove il software potrà valersi di una rete neurale convoluzionale (CNN). La scelta ricade sul fatto che una rete neurale risulta pratica e comoda per la risoluzione di problemi di "object detection", come in questo caso.

Inoltre, l'applicativo è stato pensato per essere realizzato per una piattaforma Android, ponendo maggiore anche verso la calibrazione delle risorse, poiché dovrà svolgere i propri compiti su sistemi embedded, come gli smartphone.

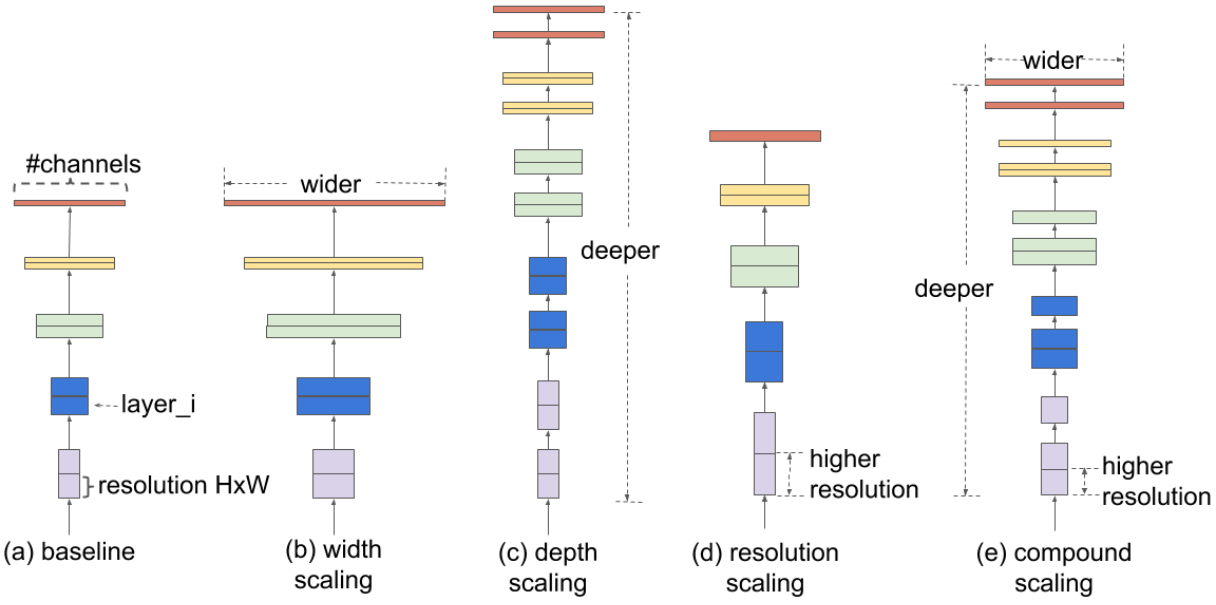
Capitolo 2

2. Rete Neurale

L'obiettivo primario di questo progetto è quello di realizzare un modello di rete neurale addestrata in grado di riconoscere con un elevata dimestichezza le mani di una o più persone, per poi interpretarne il loro significato all'interno dell'alfabeto ASL.

A tale scopo, si è utilizzato è deciso di utilizzare una particolare architettura di rete neurale convoluzionale, chiamato **EfficientNet**, che si basa su una metodologia di ridimensionamento uniforme per tutte le dimensioni di profondità, larghezza e risoluzione, utilizzando un coefficiente composto. A differenza della partica convenzionale che ridimensione arbitrariamente questi fattori, questa architettura, invece, sfrutta un set di coefficienti fissi. Tale metodo è giustificato dal fatto che se l'immagine di input è più grande, questa richiede più livelli per aumentare il campo ricettivo e più canali per catturare modelli a grana fine.

In particolare, si è utilizzata la rete EfficientNetB0 che si basa sui blocchi residui dell' *'inverted bottleneck method'*, utilizzato dalla rete MobileNetV2, oltre ai blocchi di compressione e stimolazione.



2.1. Hand Detection

La prima delle reti neurali si occupa del riconoscimento delle mani di uno o più utenti, la quale risulterà poi fondamentale anche per il corretto riconoscimento del linguaggio dei segni.

Infatti, durante l'implementazione realizzata lato Android, l'output di questa prima rete neurale viene utilizzato come discriminante per l'altra rete neurale, in quanto bisogna raggiungere una soglia minima di riconoscimento per attivare l'altro processo, che si occupa del riconoscimento e della traduzione dell'alfabeto dei segni.

2.1.1. Dataset

Per il training di questo modello si è deciso di utilizzare un dataset pubblico rilasciato da Kaggle contenente oltre 15 mila immagini, dove in ogni immagine erano presenti una o più mani, raffigurate sia per palmo che per dorso.

2.1.2. Training

L'addestramento della rete è stato effettuato tramite Python, utilizzando le varie librerie di TensorFlow e le API di Keras, due librerie software open source per il Machine Learning.

Dopo aver validato la corretta processazione dei dati e il corretto ridimensionamento, si è proceduto alla creazione del modello vero e proprio, sfruttando la grande funzionalità dei livelli e degli ottimizzatori di Keras.
Inoltre, si è deciso di sfruttare anche due callbacks, ovvero degli oggetti che permettono si eseguire delle azioni durante l'addestramento stesso.

In questo caso, sono stati utilizzati per salvare solo il migliore dei checkpoint, mantenendo così solo il risultato migliore, quindi non necessariamente l'ultimo registrato, e per ridurre il learning rate in caso di addestramento non funzionale.

Una volta terminato il processo di addestramento del modello, è stato finalmente caricato il migliore dei checkpoint registrati, in modo da ottenere un grado di riconoscimento ancora maggiore.

2.1.3. TfLite

Una volta accertato che questa prima rete neurale funzionasse in modo equo e che avesse dei livelli di precisione significativamente alti e dei livelli di perdita di precisione decisamente bassi, il modello addestrato è stato convertito in un formato adatto ai sistemi embedded, ovvero quello di TensorFlow Lite, necessario per essere importato all'interno dell'applicazione Android come file di Machine Learning.

2.2. Sign Language Detection

La seconda delle reti neurali si occupa, invece, dell'individuazione e del riconoscimento del significato dei vari segni realizzati con le mani da parte di uno o più utenti.

Come anticipato in precedenza, questa rete neurale dipende strettamente dalla precedente, poiché nello sviluppo della *MainActivity* lato Android si è deciso di adottare una soglia minima di riconoscimento, valutata in base all'output ottenuto dalla prima rete neurale, per permettere all'applicazione di sfruttare il modello alla base della seconda, così da cercare di ottenere il migliore dei risultati fra quelli stimati.

2.2.1. Dataset

Per il training di questo modello si è deciso di utilizzare un dataset pubblico rilasciato da Kaggle contenente oltre 70 mila immagini, poi ridimensionato alla sua metà, all'interno del quale in ogni immagine erano presenti delle mani poste in una determinata posizione, a simboleggiare una particolare lettera dell'alfabeto ASL, ad esclusione della lettera Z.

Dunque, il dataset conteneva all'incirca 1500 immagini per lettera, dove le mani sono disposte in diverse posizioni spaziali.

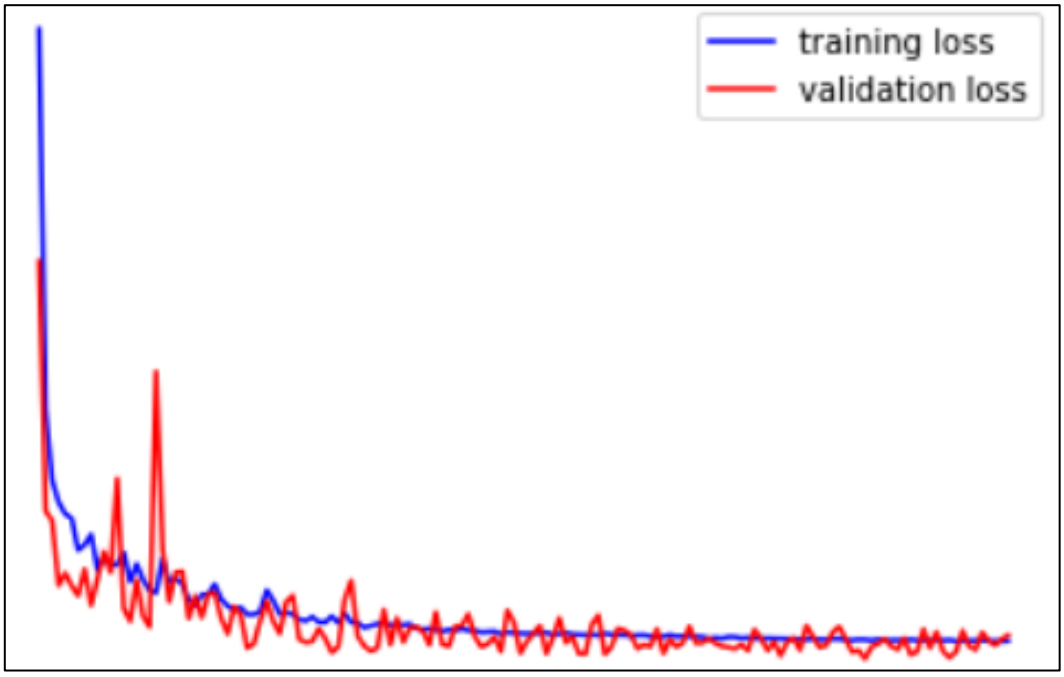
2.2.2. Training

L'addestramento di questa seconda rete è stato effettuato sulle orme del precedente tramite Python, utilizzando le varie librerie di TensorFlow e le API di Keras.

Dopo aver validato la corretta processazione dei dati e il corretto ridimensionamento, si è proceduto alla creazione del modello vero e proprio, sfruttando, come in precedenza, la grande funzionalità dei livelli e degli ottimizzatori di Keras e due callbacks per cercare di garantire anche in questo caso una soglia minima di riconoscimento dei segni all'interno delle immagini.

2.2.3. Graphing

Durante l’addestramento di questa rete, a differenza della precedente, prima di procedere con il caricamento del migliore dei checkpoint registrati, si è deciso di valutare i dati ottenuti tramite Matplotlib, ovvero una libreria per la creazione di grafici per il linguaggio Python.



Dal grafico in sovrimpressione, infatti, si può chiaramente evincere come ci sia un chiaro miglioramento dei dati riguardanti i livelli di perdita, sia in fase di allenamento che di validazione, nel corso di tutto l’addestramento e di come questo sia costante nel tempo, tranne in soli due casi eccezionali.

Una volta terminato e validato il processo di addestramento del modello, è stato finalmente caricato il migliore dei checkpoint registrati, in modo da cercare l’ottenimento un grado di una maggiore precisione nel riconoscimento dei segni linguistici.

2.2.4. TfLite

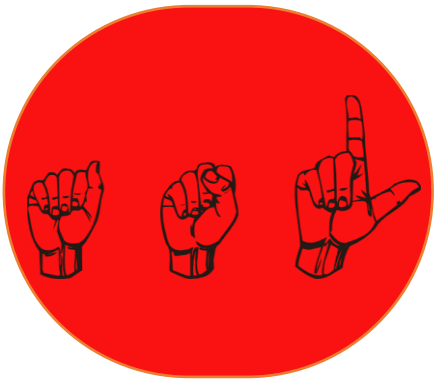
Una volta accertato che la nuova rete neurale funzionasse in modo equo e che avesse dei livelli di perdita di precisione decisamente bassi, sia in fase di allenamento che di validazione, tali da garantire un riconoscimento soddisfacente, il modello addestrato è stato convertito come il precedente nel TensorFlow Lite e importato all'interno dell'applicazione Android come file di Machine Learning.

Capitolo 3

3. Applicazione Android

Per una presentazione e dimostrazione anche pratica di ciò di cui si è discusso in precedenza si è scelto di realizzare e implementare un applicativo software per dispositivi mobili.

In particolare, si è optato di sfruttare le conoscenze pregresse per sviluppare un'applicazione su di una piattaforma Android, così da sfruttare al meglio TensorFlow, la quale ha nativamente grande interoperabilità con l'ambiente in questione.



3.1. Interfaccia

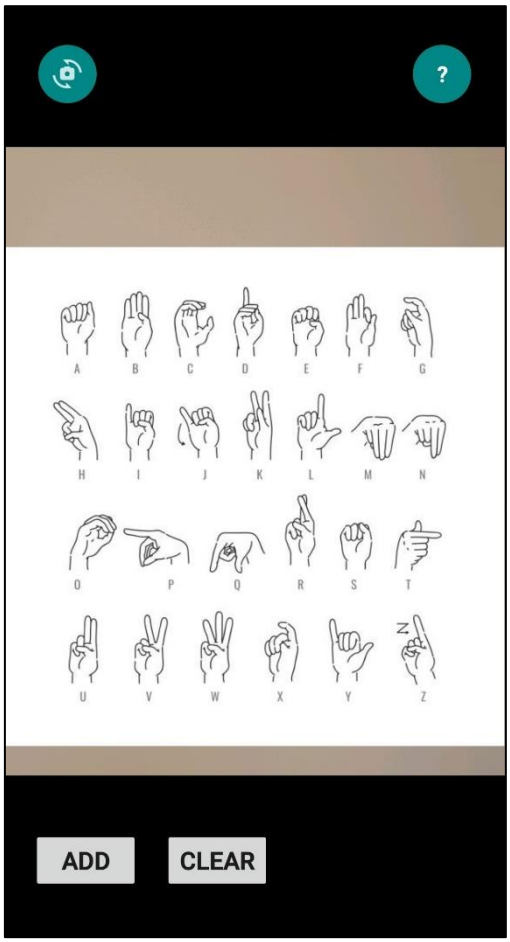
L'applicazione presenta un'unica schermata principale che permette tutte le operazioni necessarie.

Trattandosi di un'applicazione che ha come caposaldo la reattività in real-time, è stato deciso di mantenere un'unica schermata, che permettesse ugualmente di svolgere più azioni, mantenendo una grande semplicità di utilizzo per l'utente.

Questa schermata principale contiene quattro pulsanti che permettono all'utente di creare una propria consapevolezza della potenza del linguaggio dei segni.

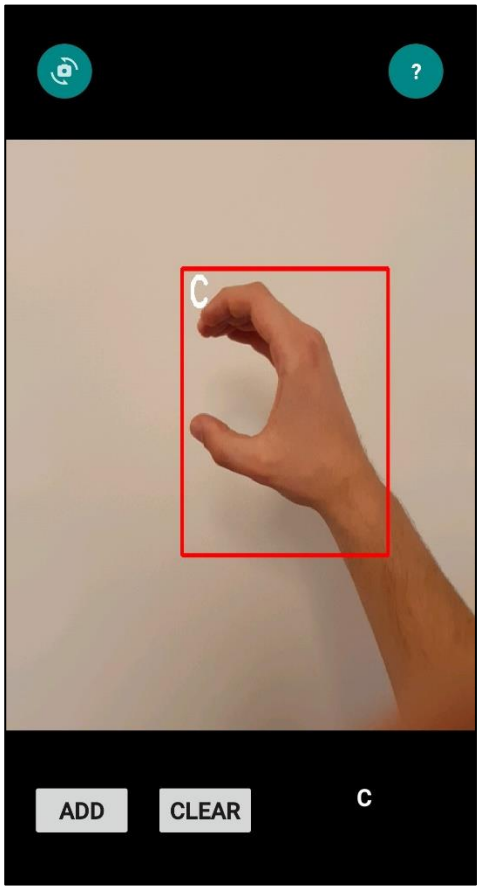
3.1.1. Dictionary Button

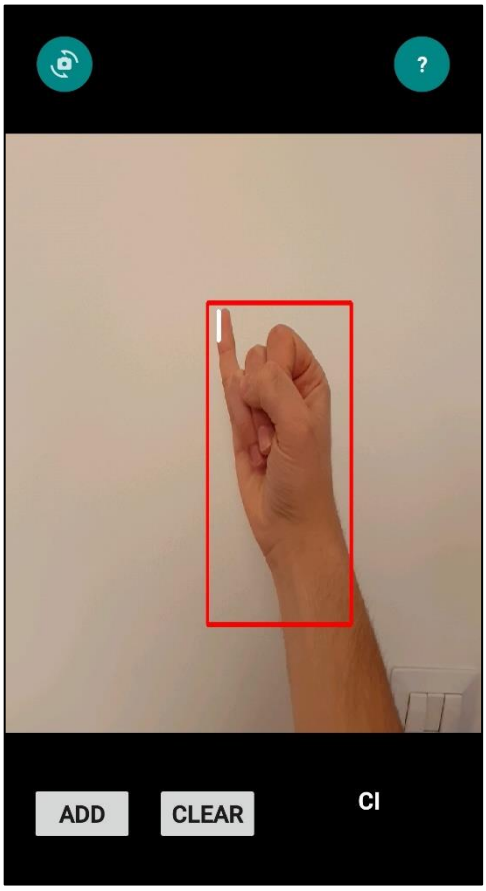
Il pulsante “Dictionary” semplifica ancor di più l’utilizzo dell’applicazione, mostrando all’utente contenente l’intero alfabeto del linguaggio dei segni.
Alla pressione del bottone si sovrappone all’immagine live un pop-up temporaneo.

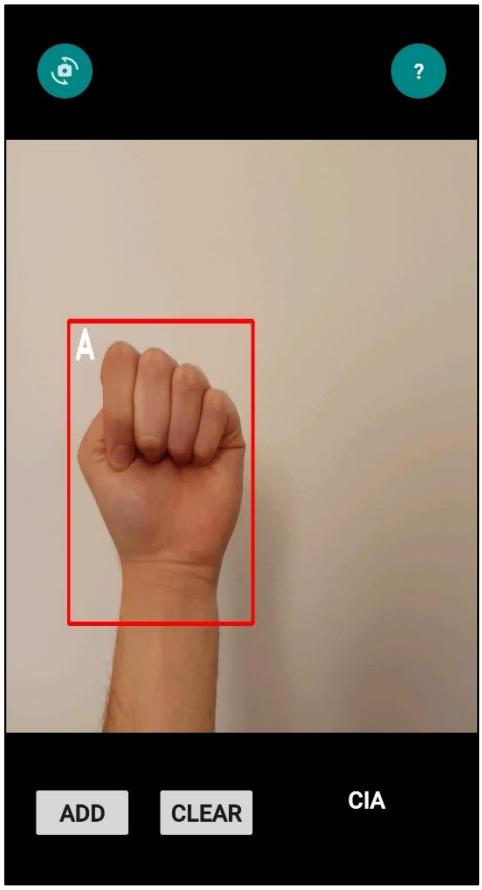


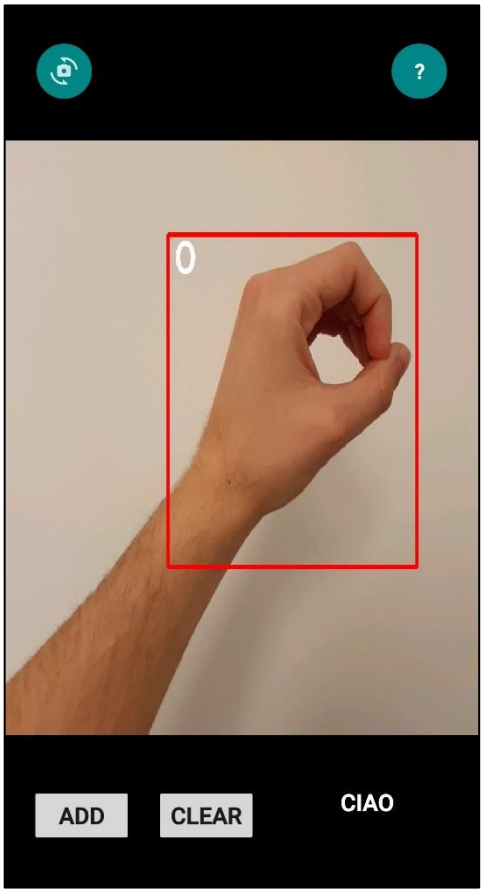
3.1.2. Add Button

Il pulsante “Add” permette all’utente di ‘giocare’ con il riconoscimento dei segni, ovvero permette di aggiungere all’interno di una text-box la lettera mostrata e tradotta in tempo reale.
Alla pressione del pulsante, la lettera che si trova in alto a sinistra del rettangolo all’interno del quale viene riconosciuto il segno della mano, viene copiato nella text-box in basso a destra.


C
ADD
CLEAR
C

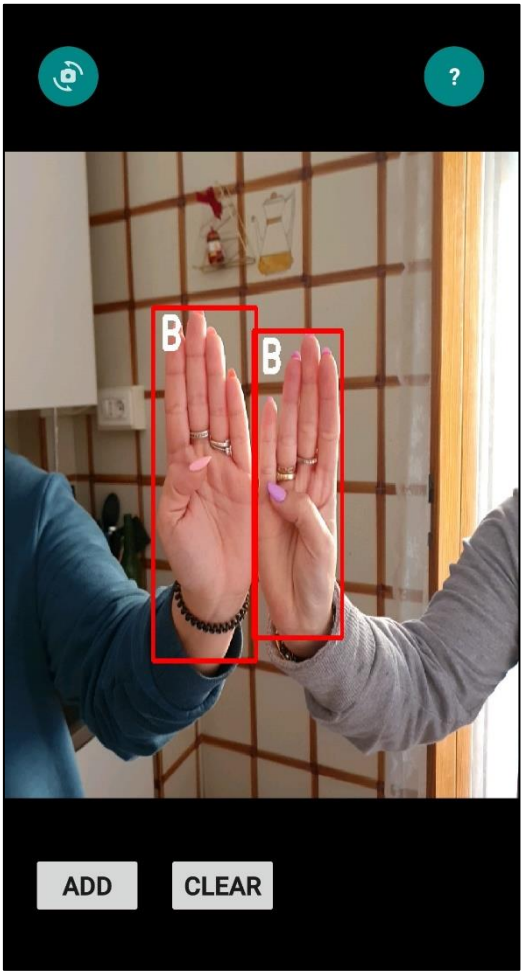

I
ADD
CLEAR
CI


A
ADD
CLEAR
CIA


O
ADD
CLEAR
CIAO

3.1.3. Clear Button

Il pulsante “Clear”, come il precedente, permette all’utente di ‘giocare’ con il riconoscimento dei segni.
Alla pressione del pulsante, la parola rappresentata all’interno della text-box in basso a destra viene cancellata definitivamente.



3.1.4. Switch Camera Button

Il pulsante “Switch Camera” permette di scegliere quale camera utilizzare per il riconoscimento: quella interna, rivolta verso l’utente, oppure quella esterna, rivolta dall’altro lato (di default l’applicazione avvia quella esterna, dopo aver chiesto il permesso di accesso alla fotocamera).
Alla pressione del pulsante, la camera in utilizzo viene disattivata e l’altra viene attivata e inizializzata per l’analisi delle immagini.

Capitolo 4

Conclusioni

L’obiettivo principale del progetto, ovvero il riconoscimento di tutte le lettere dell’alfabeto ASL (American Sign Language), con il corrispondente tracking in real time, è stato raggiunto con risultati piuttosto soddisfacenti.

L’applicativo Android rispetta le aspettative e svolge i compiti, principali e secondari, in maniera adeguata, senza un eccessivo impiego della CPU.

Il modello addestrato restituisce un corretto valore in uscita con una precisione molto elevata:

- precisione del **96%** per quanto riguarda l’hand tracking
- precisione del **89%** per quanto riguarda il sign language tracking

Chiaramente i risultati fanno fede alla valutazione effettuata prima del passaggio del modello nell’applicativo Android, che, tuttavia, mantiene la sua correttezza notevolmente elevata. Inoltre, considerando che il numero di campioni utilizzati, soprattutto per l’addestramento della seconda rete neurale, non è molto elevato se si considera la singola lettera, i risultati ottenuti si possono definire piuttosto soddisfacenti.