# AmbrogioAI

## Introduzione

AmbrogioAI è un adattamento della CNN ResNet-50, nello specifico come si può vedere anche nel seguente file , AmbrogioAI è un'implementazione della rete ResNet-50 che permette di classificare l'abbigliamento di un generico soggetto presente in una foto in una tra le seguenti classi:

- Elegante
- Casual
- Sportivo.

## Prerequisiti

- Node js installato per aver accesso al comando npm (esclusiva front).
- python 11.x , versione attualmente supportata (esclusiva backend).
  **(!)WARNING: le successive versioni del linguaggio non sono state testate quindi possono presentare errore inaspettati**

## Avvio del progetto

### Server in Windows

Eseguire il seguente comando in un terminale dentro alla root del progetto scaricato da github

```
.\StartWebServer.bat
```

Lo script in questione chiamerà prima lo script per installare le dipendenze chiamato installDependencies.bat il quale installerà tutte le dipendenze necessarie per il server, in seguito verrà avviato il server flask.

### Server in linux/macOS

**(!)WARNING: Servizio non testato sono quindi possibili errori dovuti alle dipendenze.**
Eseguire il seguente comando in un terminale dentro alla root del progetto scaricato da github:

```
.\StartWebServer.sh
```

Lo script in questione chiamerà prima lo script per installare le dipendenze chiamato installDependencies.sh il quale installerà tutte le dipendenze necessarie per il server, in seguito verrà avviato il server flask.

### Front in Windows

Eseguire il seguente comando in un terminale dentro alla root del progetto scaricato da github:

```
.\StartFront.bat
```

Lo script in questione eseguirà prima "npm i" per installare tutte le dipendenze del front e poi "npm run dev" il quale avvierà il front accessibile nella route "localhost:5173" accessibile da qualsiasi browser.

### Front in Linux/macOS

Eseguire il seguente comando in un terminale dentro alla root del progetto scaricato da github:

```
.\StartFront.sh
```

Lo script in questione eseguirà prima "npm i" per installare tutte le dipendenze del front e poi "npm run dev" il quale avvierà il front accessibile nella route "localhost:5173" accessibile da qualsiasi browser.

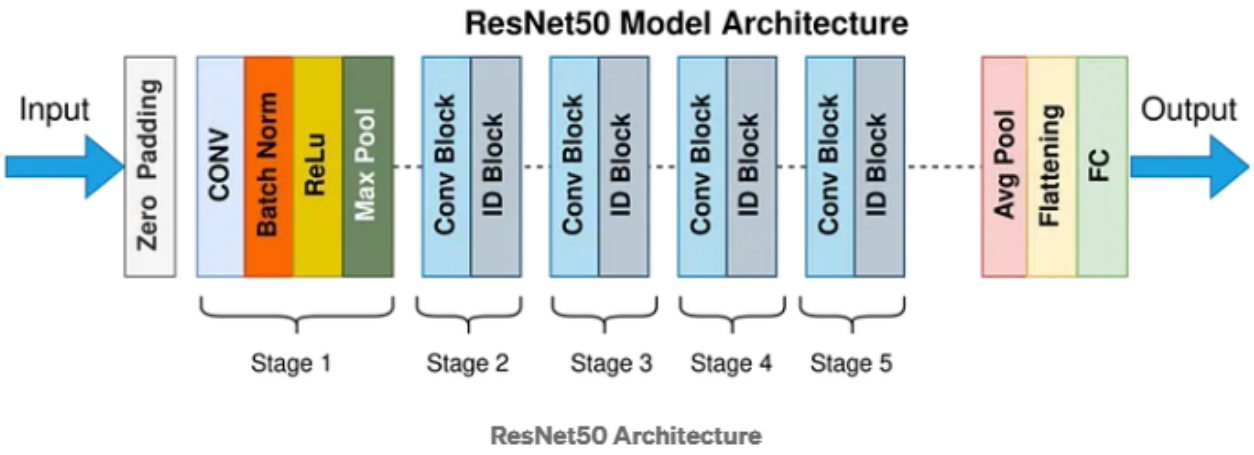
---

## Modelli disponibili

All'interno del progetto sono presenti due possibili versioni dell'intelligenza artificiale:

- AmbrogioResNet50:
  Implementazione della CNN fornita dalla libreria python torch che supporta anche diversi metodi di ottimizzazione come la discesa del gradiente stocastico, RMSProp e Adam.
  Ci si aspetta che sia la versione più accurata ed è, inoltre, customizzabile tramite i classici parametri come learning rate e momentum.
- ResNet50FromScratch:
  Soprannominata anche "AmbrogioSimple" questa CNN è sempre una ResNet-50 ma è senza alcun meccanismo di ottimizzazione, realizzata osservando l'architettura del

Classificatore.




## Struttura dell'applicazione

- Server Python con flask:
  presente nella cartella "backend" è presente il file main.py il quale crea l'applicazione flask e rende disponibili delle API per permettere al client di:

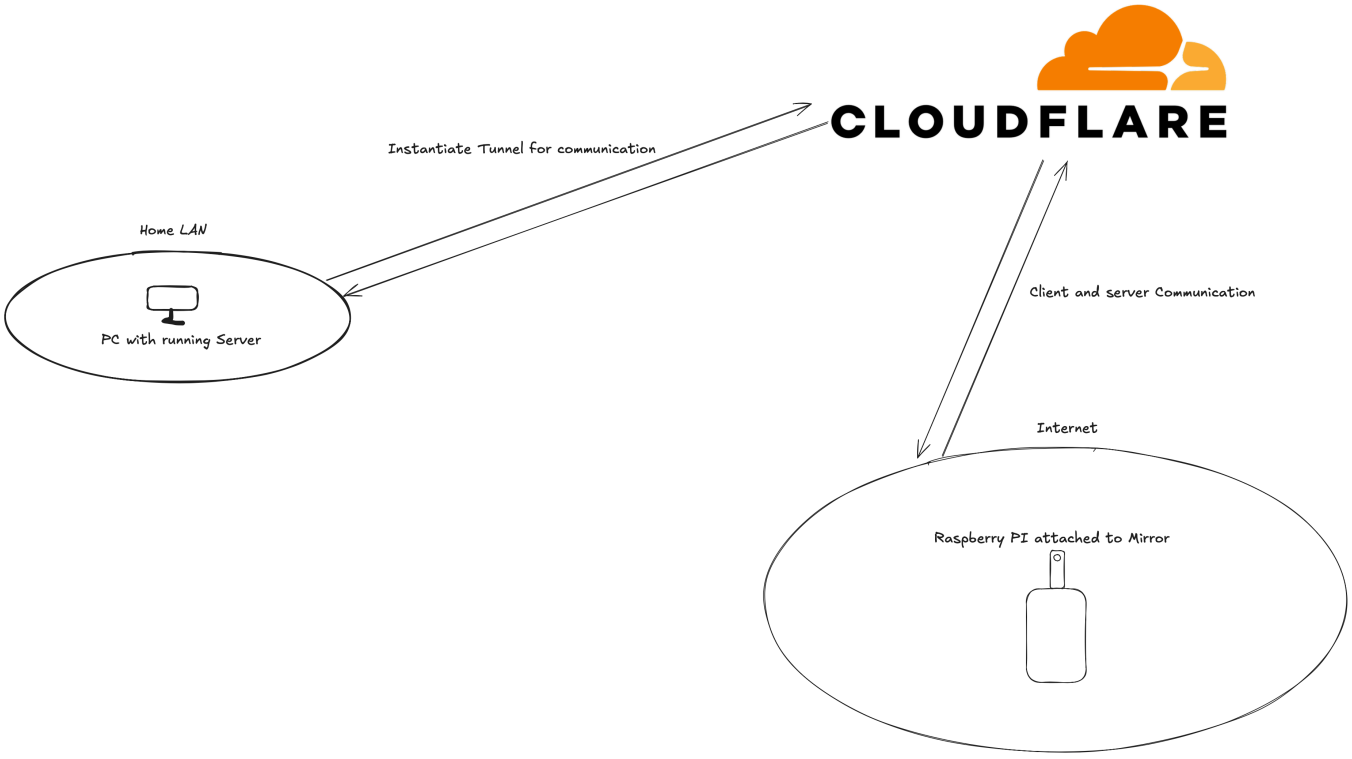
| Rotta API | Metodo | Dati da passare | Risposta | Descrizione |
|---|---|---|---|---|
| /test | GET | - | "Hello World" | Route utile per testare la raggiungibilità del server, usata nella parte front per visualizzare e calcolare la latenza |
| /chooseModel | POST | {"model": modelIndex as integer} | {"modelChosen": modelName as String} | Permette di scegliere quale modello impostare e successivamente quale da utilizzare per eseguire le predizioni in base ad il modelIndex passato.<br>0 => AmbrogioAI<br>1 => resNet50FromScratch |
| /removeModel | POST | - | - | Imposta a None il modello scelto.<br>Se l'applicativo front provasse a gettare il currentModel e ricevesse None come risposta reindirizzerebbe l'utente |

| Rotta API | Metodo | Dati da passare | Risposta | Descrizione |
|---|---|---|---|---|
| | | | | alla pagina di scelta del modello. |
| /currentModel | GET | - | Se model != None {"model": modelName as String}. Altrimenti risponde con errore 403 | Ottieni il nome del modello corrente(attualmente scelto per predirre). |
| /predict | POST | Foto scattata passata come file con il tag "image" | { "prediction": array di lunghezza 3 con le percentuale di appartenenza alle classi } | Passata una immagine al server, esso restituisce la predizione fatta dal currentModel |

- Client Javascript con framework ReactJS:
  presente nella cartella "front" permette di chiamare tutte le API di cui sopra e verificare tutte le risposte ricevute.

## Mappa Delle Comunicazioni tra Client e Server Prototype

Di seguito mostro una mappa rappresentate il giro che le richieste fanno dal client al server e viceversa.




## Sviluppi Futuri

- Sintetizzatore vocale per permettere la comunicazione con il server senza passare dal Front
- Ampliare il dataset utilizzando le foto passate dal front così da permettere ad ambrogio di personalizzarsi e di migliorare in base all'utilizzatore.
    - Non tutte le immagini verrebbero salvate, ma solo se l'utilizzatore lo desidera, inoltre viene chiesto all'utente se è soddisfatto della predizione eseguita da ambrogio, se no viene chiesto all'utente a quale classe apparterrebbe la foto appena scattata così da avere anche la label della nuova immagine.