UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Progettazione e sviluppo di un sistema per la creazione di dataset al fine di riconoscere il grado di stanchezza di un guidatore

Design and development of a system for the creation of datasets in order to recognize the degree of fatigue of a driver

Relatore:
Prof. Mancini Adriano

Laureando:
Gatti Giada

Anno Accademico 2020-2021

Indice

Capitolo 1

Introduzione

Il seguente lavoro di tesi è stato svolto in collaborazione con il collega Mattia Alesi. Si è deciso di suddividere la tesi in due parti: il collega Alesi nel suo elaborato presenta aspetti relativi all'addestramento della rete neurale, al fine di riconoscere l'apertura o chiusura di occhi, e l'implementazione di questo modello in *Google Coral*, mentre questo lavoro di tesi tratta la creazione e manipolazione del data-set e, in parallelo con il collega, l'addestramento della rete neurale.

La sonnolenza alla guida, pur essendo all'origine di molti incidenti, è ancora sottostimata come fattore determinante di rischio. Le statistiche indicano genericamente come prima causa di incidente stradale la "distrazione", frutto proprio della stanchezza e della sonnolenza del guidatore. Quest'ultima quasi mai viene presa in esame di per sé come "causa". Eppure basterebbe sapere che:

- l'eccessiva sonnolenza è associata approssimativamente (come causa diretta o concausa) ad un quinto degli incidenti stradali ed è una delle principali cause di incidenti mortali in autostrada;
- dormire meno di 5 ore per notte aumenta di 4,5 volte la probabilità di avere un incidente stradale;
- stare svegli per 24 ore induce errori alla guida simili a quelli commessi da chi ha livelli di alcool nel sangue uguali o superiori a 1,00 g/l;
- le persone affette dalla sindrome delle Apnee ostruttive nel sonno (OSAS) hanno un rischio di incidente stradale da 2 a 7 volte superiore a quello osservato nelle persone sane;
- gli incidenti causati dal "colpo di sonno" sono i più gravi, con un elevato rischio di mortalità dovuto alla totale inazione del guidatore, che addormentandosi non ha consapevolezza dell'imminente pericolo;
- i pericoli connessi alla sonnolenza aumentano con l'aumentare delle ore trascorse al volante senza pausa; particolarmente a rischio gli autisti pro-

fessionali e chi percorre lunghi tragitti in auto, soprattutto nelle prime ore del mattino o durante la notte.[1]

Colpo di sonno, distrazione, stanchezza: c'è chi all'interno della vettura controlla istante dopo istante le condizioni di autista e passeggeri e interviene in caso di bisogno. Sono le telecamere di monitoraggio(Fig. 1.1) [1] che, assistite dall'intelligenza artificiale, salveranno, secondo uno studio UE, nei prossimi 20 anni, oltre 25 mila vite. Il monitoraggio visivo di quanto accade all'interno dell'abitacolo potrebbe essere la soluzione di un problema fondamentale delle auto a guida autonoma: quando l'automobilista deve riprendere il controllo del veicolo dopo la guida autonoma in autostrada, per esempio, il sistema dell'auto deve verificare che il conducente non stia dormendo, leggendo il giornale o scrivendo email sullo smartphone. Una telecamera integrata nel volante rileva se il guidatore sta per addormentarsi, osservando i battiti delle palpebre, se si distrae e se gira la testa verso i passeggeri sul sedile posteriore. Grazie all'Intelligenza Artificiale (IA), il sistema trae le dovute conclusioni da queste informazioni: avverte gli automobilisti distratti, consiglia di fermarsi se sono stanchi e può persino ridurre la velocità, a seconda della configurazione scelta dalla casa costruttrice del veicolo e in base alle norme di legge, basandosi sulla registrazione della posizione delle palpebre e sulla frequenza dei battiti di ciglia desume il grado di stanchezza del guidatore. Questo gli permette di emettere un avviso adeguato alla situazione e di intervenire con i sistemi di assistenza alla guida.[2]

Figura 1.1: Esempio rilevatore di stanchezza conducente

[1]Immagine tratta da: https://www.elettronicain.it/blog/2019/12/26/le-telecamere-che-salvano-la-vita-a-conducente-e-passeggeri/

I vantaggi che se ne traggono da questi veicolo sono molteplici. Primo su tutti la riduzione degli incidenti stradali e delle vittime. Migliaia di persone perdono ogni anno la vita in incidenti sulle strade europee, il 95% dei quali è dovuto all'errore umano. Le auto e i camion senza conducente possono, dunque, ridurre drasticamente questi numeri e migliorare la sicurezza stradale. Tuttavia, le auto a guida autonoma possono comportare diversi rischi legati alla sicurezza informatica. L'utilizzo dell'intelligenza artificiale, e in particolare su tecniche di machine learning per raccogliere, analizzare e trasferire dati al fine di prendere decisioni, rende i veicoli autonomi altamente vulnerabili a un'ampia gamma di attacchi informatici che potrebbero compromettere il loro corretto funzionamento e costituire una grave minaccia per la sicurezza di passeggeri, pedoni e altri veicoli.

Ma andiamo a vedere più nel dettaglio come funzionano questi rilevatori di stanchezza. In pratica, è un segnale acustico abbinato ad un pittogramma (o a una scritta) che compare sul display del cruscotto (Fig. 1.2)[2], invitando a fermarsi per una pausa. Questa scritta che comparirà e' soltanto un suggerimento e la vettura non condizionerà il suo comportamento nel caso in cui l'autista decidesse di prosegure la sua marcia, ignorandola.

Figura 1.2: Esempio scritta sul display

Ce ne sono di due tipi, uno dal funzionamento molto semplice, mentre l'altro richiede maggiore tecnologia. Il primo si può definire rilevatore di stanchezza "a tempo", e funziona in base alla durata del viaggio. Il computer di bordo inizia a misurarlo nel momento in cui si accende il motore calcolandolo fintanto che questo non viene spento.In realtà non viene misurata la stanchezza effettiva, ma si basa su una supposizione: il sistema ipotizza che un pilota dopo due ore continuative di viaggio potrebbe essere stanco e perdere concentrazione. Tanto è vero che se si ferma l'auto e si spegne il motore, il tempo del rilevatore si azzera e ricalcola l'intervallo prefissato a partire dalla ripartenza. Ben più complesso risulta il ri-

[2]Immagine tratta da: https://www.avvenire.it/economia/pagine/rilevatore-di-stanchezza-in-auto-cos-e-e-come-funziona

levatore di stanchezza "reale": spesso, per funzionare al meglio, il sistema viene integrato all'interno di un pacchetto di optional che prevede anche il segnalatore di cambio improvviso di carreggiata. Tutto ciò avviene tramite delle telecamere che monitorano costantemente il volto del guidatore, analizzando il battito delle palpebre, le eventuali smorfie del viso interpretandole come possibili sbadigli, i cambi improvvisi nella direzione senza che venga inserita la segnalazione luminosa. Sulla base di questi segnali, che sono in realtà dei sintomi, il sistema capta lo stato della persona, interpretando i segnali ricevuti e valutandone il grado di stanchezza. Questo tipo di analisi appare più dinamica, maggiormente veritiera rispetto alla prima, poiché in questo caso viene misurato l'effettivo affaticamento e dunque il consiglio della sosta ha più senso. Le vetture più evolute, oltre ai sensori, ora contemplano anche la presenza di una telecamera interna che rileva i movimenti della testa del guidatore e delle pupille, interpretando sia la direzione dello sguardo sia l'inclinazione del capo. Se gli occhi puntano lateralmente oppure se l'inclinazione della testa è innaturale, ecco che il sistema emette impulsi e segnali. In ogni caso, si parla di un ausilio tecnico che lascia come sistema di sicurezza fondamentale il caro vecchio buon senso del guidatore. [3]

1.1 Obiettivi

Nel presente lavoro di tesi utilizziamo immagini, ricavate con una ricerca tramite *SerpApi* che è un servizio attraverso il quale è possibile in modo programmatico effettuare ricerche di immagini su Google tramite delle parole chiavi e una *Neural Network* per rilevare il livello di attenzione del guidatore in base a dei movimenti tipici dell'essere umano che si focalizzano sugli occhi.

Per poter fare ciò ci serviamo del pacchetto Mediapipe, utilizzando *Facemesh* e la libreria *OpenCV* per lavorare e manipolare le immagini [4]. Dopodiché andiamo ad addestrare una *Neural Network* per il riconoscimento del pattern, che ci serve ad identificare oggetti e segnali nei sistemi di controllo, facendo uso della libreria *Keras*.

Una *Neural Network* combina diversi livelli di elaborazione, utilizzando semplici elementi che operano in parallelo e sono ispirati ai sistemi nervosi biologici.

1.2 Struttura della Tesi

La parte essenziale del progetto di tesi è quella di sviluppare un sistema che riconosca, attraverso il movimento degli occhi, la stanchezza del guidatore in tempo reale. Il riconoscimento è realizzato tramite una *Neural Network*, appositamente addestrata, su un dataset di 10k immagini .

Utilizzando *Facemesh*, una soluzione per la geometria del viso che stima 468 punti di riferimento 3D del viso in tempo reale anche su dispositivi mobili, si

può creare una maschera che definisce i punti chiave del volto identificati da "landmarks". Andremo poi a testarlo con le immagini valutando principalmente l'accuratezza. Metteremo poi il modello in esecuzione su un dispositivo pensato per fare inferenza tramite reti neurali quale Google Coral.

Il presente lavoro di tesi è strutturato nei seguenti capitoli:

- introduzione;
- strumenti e metodi;
- analisi e sviluppo del modulo *software*;
- conclusioni e sviluppi futuri;
- appendice.

Capitolo 2

L'intelligenza Artificiale

Nonostante un ampio dibattito in campo scientifico, ancora oggi non esiste una definizione univoca di Intelligenza Artificiale. Quando si pensa a Intelligenza Artificiale vengono in mente scenari fantascientifici o futuristici, nei quali le macchine diventeranno "intelligenti" come l'uomo e saranno dunque in grado di svolgere i suoi stessi compiti fino a sostituirlo in parte o del tutto. Robot che si confrontano con gli esseri umani, li aiutano, prevedono o prevengono i loro comportamenti e addirittura mostrano empatia nei loro riguardi. Rifacendoci alle teorie di Turing, si può dire che l'IA è "la scienza di far fare ai computer cose che richiedono intelligenza quando vengono fatte dagli esseri umani". John McCarthy, docente del Computer Science Department della Harvard University, la definisce "la scienza di creare ed ingegnerizzare macchine intelligenti e in particolar modo programmi informatici intelligenti". Per Techopedia, è un'area delle scienze informatiche incentrata sulla creazione di macchine intelligenti in grado di lavorare e reagire come esseri umani. Tuttavia molti concordano sul fatto che lo scopo dell'Intelligenza Artificiale non è replicare l'intelligenza umana, bensì riprodurne o emularne solo alcune funzioni. In senso colloquiale si parla di IA quando una macchina imita funzioni cognitive che gli uomini associano alla mente umana come l'apprendimento e la risoluzione dei problemi. [5]

2.1 Cos'è realmente l'intelligenza artificiale

L'Intelligenza Artificiale è un ramo dell'informatica che permette la programmazione e progettazione di sistemi sia hardware che software che permettono di dotare le macchine di determinate caratteristiche che vengono considerate tipicamente umane quali, ad esempio, le percezioni visive, spazio-temporali e decisionali. Nasce nel 1956 con l'avvento dei computer. Gli anni successivi alla nascita dell'Intelligenza Artificiale furono anni di grande fermento intellettuale e sperimentale: università e aziende informatiche, tra cui in particolare l'IBM, puntarono alla ricerca e allo sviluppo di nuovi programmi e software in grado

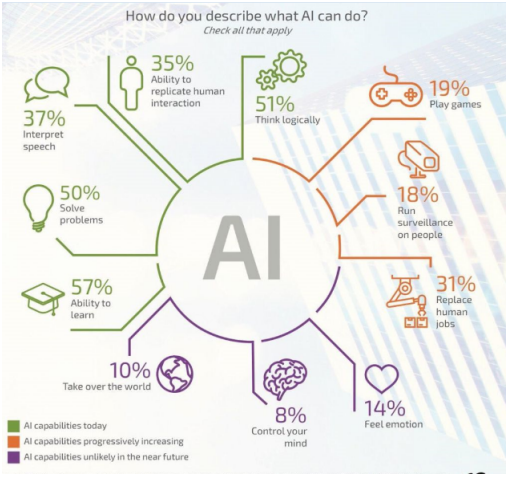


Figura 2.1: Intelligenza Artificiale.

di pensare e agire come gli esseri umani almeno in determinati campi e settori. Nacquero così programmi in grado di dimostrare teoremi sempre più complessi e, soprattutto, nacque il Lisp, ossia il primo linguaggio di programmazione che per oltre trent'anni fu alla base dei software di Intelligenza Artificiale(Fig. 2.1) [1].

Alla base delle problematiche legate allo sviluppo di sistemi e programmi di Intelligenza Artificiale vi sono tre parametri che rappresentano i cardini del comportamento umano, ossia una conoscenza non sterile, una coscienza che permetta di prendere decisioni non solo secondo la logica e l'abilità di risolvere problemi in maniera differente anche a seconda dei contesti nei quali ci si trova.

L'uso delle reti neurali e di algoritmi in grado di riprodurre ragionamenti tipici degli esseri umani nelle differenti situazioni, hanno permesso ai sistemi intelligenti di migliorare sempre di più le diverse capacità di comportamento. Per poter realizzare ciò, la ricerca si è concentrata non solo sullo sviluppo di algoritmi sempre nuovi, ma soprattutto su algoritmi sempre più numerosi

Al fine del funzionamento dell'Intelligenza Artificiale è determinante la scelta dell'algoritmo che sta alla base della stessa. È sempre necessario un algoritmo per far funzionare i dati che vengono inseriti nel programma. Un algoritmo richiede sempre una serie di passaggi per risolvere un problema che viene scomposto in calcoli elementari. Bisogna porsi delle domande prima di iniziare. Si deve comprendere cosa si vuole creare, le motivazioni alla base, il come si deve fare, l'arco temporale e quali sono i costi stimati.

La definizione del problema è quindi il primo passo fondamentale da fare e deve essere svolto dall'uomo che avrà poi il supporto dell'Intelligenza Artificiale. Questa fase prevede l'individuazione dei dati in ingresso, la scelta degli obiettivi da raggiungere e la relazione esistente tra i dati e i risultati. Bisogna sempre verificare se il problema è ben definito perché un'errata definizione compromette-

[1]Immagine tratta da: https://botsify.com/blog/ai-marketing-tools-online-business/

rebbe i risultati successivi. Se i passaggi preliminari sono corretti si può passare alla scelta dei dati da inserire.

Successivamente si va a scegliere la tipologia di agente che si intende utilizzare. Si può avere un approccio orientato alla ricerca, ottimale nel caso di applicazioni matematiche, orientato all'apprendimento, ottimale nel caso dell'utilizzo di esperienze passate, orientato alla pianificazione, ottimale con una sequenzialità delle operazioni, e orientato al ragionamento automatico. In base all'approccio adottato si deve definire il paradigma.

Quando si parla di conoscenza dell'uomo e di trasferimento di tale conoscenza alla macchina, infatti, non si parla solo di conoscenza sterile, ossia di nozioni apprese dai libri o da altri strumenti di studio ma si parla piuttosto di esperienza e di possibilità di comprendere nuove informazioni tramite quelle già presenti nel sistema di partenza. Tali informazioni vengono fornite alla macchina tramite diverse modalità, le più importanti delle quali sono quelle che si basano sulla Teoria dei Linguaggi Formali e sulla Teoria delle Decisioni.

Quando si utilizza la Teoria dei Linguaggi Formali, si sceglie di utilizzare diversi approcci che si rifanno alle teorie delle Stringhe e ai loro utilizzi che rappresentano dei veri e propri linguaggi formali le cui proprietà variano proprio a seconda dell'approccio utilizzato.

La Teoria delle Decisioni, invece, si basa su un albero di decisione(Fig. 2.2)[2], che permette di valutare per ogni azione/decisione le possibili conseguenze prendendo quindi poi la decisione più conveniente.Basta sapere che un albero di decisione si basa su modelli predittivi a partire da una serie di informazioni iniziali e dati di partenza. Tali dati possono poi essere suddivisi in maniera tale da definire sia la struttura, ossia il tipo di previsioni possibili, sia l'accuratezza delle stesse. Proprio l'accuratezza dei dati permette di ottenere dei sistemi intelligenti che si differenziano tra di loro per le risposte in grado di dare a seconda non tanto del numero di dati sul quale si basano le decisioni, ma a seconda della precisione degli stessi.

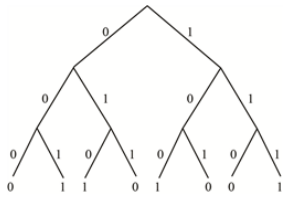


Figura 2.2: Albero teoria delle decisioni.

L'intelligenza Artificiale viene abbondantemente utilizzata anche nel quotidiano. Ad esempio, i vari strumenti di riconoscimento vocale che vengono regolarmente utilizzati, dagli smartphone ai sistemi di sicurezza, si basano su algoritmi tipici dell'Intelligenza Artificiale, in particolare quelli relativi all'apprendimento

[2]Immagine tratta da: https://www.intelligenzaartificiale.it/

automatico. Molto noto, nel panorama dell'apprendimento automatico e dell'Intelligenza Artificiale, è l'utilizzo che si fa di questo strumento nel settore automobilistico. I veicoli in grado di muoversi nel traffico anche senza pilota sono oggi qualcosa che va oltre la sperimentazione, anche se il loro utilizzo è limitato solo a determinati settori e situazioni.

Ulteriori settori in cui l'Intelligenza Artificiale viene utilizzata in maniera regolare sono il mercato azionario, la medicina e la robotica. Infine, anche molti moderni smartphone e dispositivi mobili presentano piattaforme basate su sistemi di Intelligenza Artificiale, che permettono una vera e propria interazione tra il telefono e il suo proprietario, fondamentale per diverse funzioni. Alcuni moderni telefoni, ad esempio, presentano dei sensori in grado di rendersi conto se il proprietario del telefono si sta muovendo a piedi o in veicolo: in questo caso automaticamente potrà impostarsi sulla modalità di guida per garantire la massima sicurezza nell'uso. Ancora, alcuni telefoni accenderanno automaticamente la torcia incorporata quando si renderanno conto che il proprietario si sta muovendo al buio. Le funzioni sono differenti e molto varie a seconda dei telefoni, ma tutte volte a migliorare il comfort e la sicurezza di quanti ne fanno uso.

Al fine del funzionamento dell'Intelligenza Artificiale è determinante la scelta dell'algoritmo che sta alla base della stessa. È sempre necessario un algoritmo per far funzionare i dati che vengono inseriti nel programma. Un algoritmo richiede sempre una serie di passaggi per risolvere un problema che viene scomposto in calcoli elementari. Bisogna porsi delle domande prima di iniziare. Si deve comprendere cosa si vuole creare, le motivazioni alla base, il come si deve fare, l'arco temporale e quali sono i costi stimati. L'informatica permette di risolvere problemi con il computer date regole e risorse necessarie ad operare. Risolvere un problema significa, dati degli input, arrivare ad un output soddisfacendo un criterio di verifica. È molto opportuno comprendere ciò che si vuole realizzare e questo lo si può fare solo definendo nel dettaglio il problema da risolvere. Durante lo svolgimento del processo è comunque necessaria una supervisione attiva perché il sopraggiungimento di dati non previsti potrebbe interferire con il raggiungimento dell'output.

Ci si chiede spesso quale possa essere il futuro dell'Intelligenza Artificiale. Da un lato l'entusiasmo per l'evoluzione tecnologica è sicuramente molto evidente in diversi settori, dall'altro la paura che a breve le macchine potrebbero sostituire del tutto l'uomo in molti luoghi di lavoro si è insinuata in maniera sempre più insistente nelle menti di molti. Con l'uso massivo dell'Intelligenza Artificiale sarà possibile perdere ulteriori posti di lavoro ma è anche vero che si apriranno sempre più strade per la realizzazione di nuove tipologie di figure professionali. Ma il contrasto tra uomo e macchina è un settore molto più ampio che non è solo relativo all'evoluzione dell'Intelligenza Artificiale(Fig. 2.3) [3] e dei sistemi intelligenti, ma

[3]Immagine tratta da: https://www.thenewsteller.com/the-impact-of-artificial-intelligence-on-peoples-lives/

anche e soprattutto relativo alla morale e all'etica lavorativa e al corretto utilizzo delle macchine nel rispetto dell'uomo. Probabilmente la direzione che si prenderà non è ancora ben delineata, ma potrà portare a una nuova rivoluzione culturale e industriale. [6]

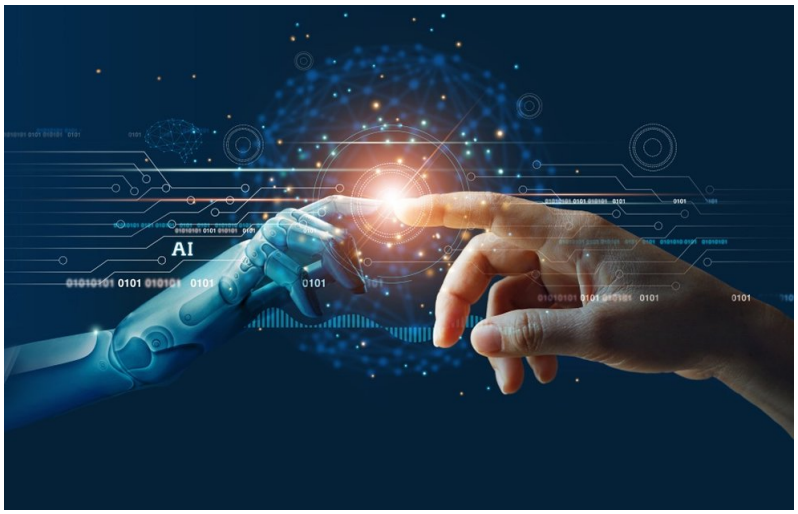


Figura 2.3: Artificial Intelligence

2.2 Machine Learning e Deep Learning

All'interno dell'Intelligenza Artificiale troviamo il Machine Learning, e come sottoinsieme di quest'ultimo il Deep Learning (Fig. 2.4) [4], che possono essere descritti come un insieme di algoritmi e di metodi di programmazione. In Machine Learning, all'algoritmo deve essere detto come eseguire una stima accurata utilizzando più informazioni, ad esempio eseguendo l'estrazione delle funzionalità. Nel deep learning, l'algoritmo può imparare a eseguire una stima accurata tramite la propria elaborazione dei dati, grazie alla struttura di rete neurale artificiale.

2.2.1 Il *Machine Learning*

Il Machine Learning è una sottospecie di sottogruppo dell'Intelligenza Artificiale, i cui una macchina è capace di ricevere una serie di dati, modificando gli algoritmi man mano che ricevono informazioni su ciò che stanno elaborando. Esso quindi, in sintesi, è inteso come la capacità di una macchina di apprendere senza essere programmata esplicitamente e ciò implica un enorme utilizzo di dati e un efficiente

[4]Immagine tratta da: https://vitolavecchia.altervista.org/caratteristiche-e-differenza-tra-machine-learning-e-deep-learning/

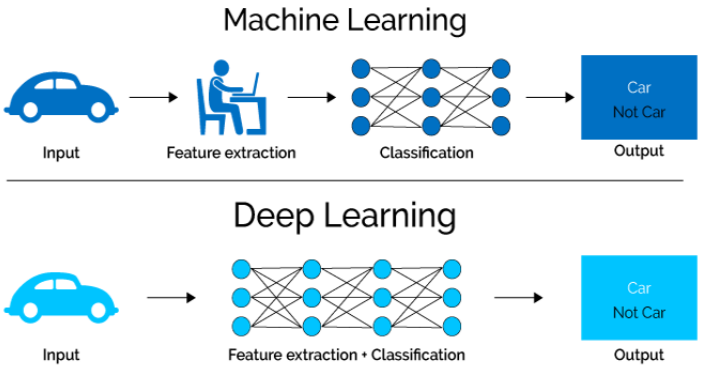


Figura 2.4: Machine Learning e Deep Learning

algoritmo al fine di addatarsi. Il Machine Learning usa metodi di rete neurale, modelli statistici e ricerche operative per trovare informazioni nascoste nei dati

Un esempio classico di machine learning è rappresentato dai sistemi di visione artificiale, ovvero la capacità di un sistema computazionale di riconoscere oggetti acquisiti digitalmente da sensori di immagine. L'algoritmo impiegato in questi casi dovrà riconoscere determinati oggetti, distinguendoli tra animali, cose e persone, e nello stesso tempo imparando dalle situazioni, ovvero avere memoria di ciò che si è fatto per impiegarlo efficacemente nelle prossime acquisizioni di visione artificiale impiegate soprattutto nei sistemi Automotive.

Per far sì che un algoritmo di Machine Learning possa funzionare si deve partire con la preparazione e l'inserimento dei dati. È una fase molto delicata perché si devono andare a selezionare dati coerenti con l'obiettivo prefissato in modo tale da rendere l'algoritmo il più efficiente possibile. Successivamente si passa alla fase di addestramento che va a fornire, all'algoritmo, tutti gli esempi degli input a disposizione e i risultati che si desidera ottenere dagli stessi. Passando alla fase di training, l'algoritmo va ad apprendere le caratteristiche degli input per raggiungere un determinato output. Questa fase prevede un sistema di feedback che permette all'algoritmo di migliorarsi. Il processo per far funzionare un algoritmo di Machine Learning può richiedere molto tempo per diventare operativo e può richiedere la supervisione del programmatore, il quale deve verificare che le varie fasi proseguano in maniera ottimale e deve andare ad intervenire laddove si presentino problemi. Non sempre si riesce a generare un algoritmo di Machine Learning funzionante, quindi è necessario focalizzarsi e comprendere tutte le fasi che lo compongono.

2.2.2 Il *Deep Learning*

Il Deep Learning è uno degli approcci all'apprendimento automatico che ha preso spunto dalla struttura del cervello, ovvero l'interconnessione dei vari neuroni. L'apprendimento approfondito utilizza enormi modelli di reti neurali con varie

unità di elaborazione; sfrutta i progressi computazionali e tecniche di allenamento per apprendere modelli complessi attraverso una enorme quantità di dati. [7]

Il *Machine Learning* utilizza le reti neurali per analizzare grandi quantità di dati, per poter apprendere dalle decisioni sbagliate e correggere i propri errori. Il *Deep Learning* (o Apprendimento Profondo) addestra le reti neurali e le rende capaci di imparare a gestire gli input, a fare previsioni man mano più accurate, e a risolvere problemi con maggiore efficienza .[8]

Il *Deep Learning* è il ramo più avanzato del *Machine Learning*: è una tecnica di apprendimento in cui si espongono Reti Neurali artificiali a grandi quantità di dati, in modo che possano imparare a svolgere compiti.

2.3 Rete Neurale

Le reti neurali, che si basano sull'imitazione del cervello umano, sono composte da neuroni artificiali che vengono organizzati in una struttura interconnessa che permette il collegamento degli input e degli output dei vari neuroni. Questo tipo di struttura permette ai neuroni di ricevere sia dati iniziali sia dati elaborati da altri neuroni, a seconda del livello del neurone stesso. L'architettura della rete neurale artificiale prevede che i neuroni siano disposti su diversi livelli, rendendo necessaria la disposizione del numero dei livelli (layer) e del numero di neuroni per ogni livello.

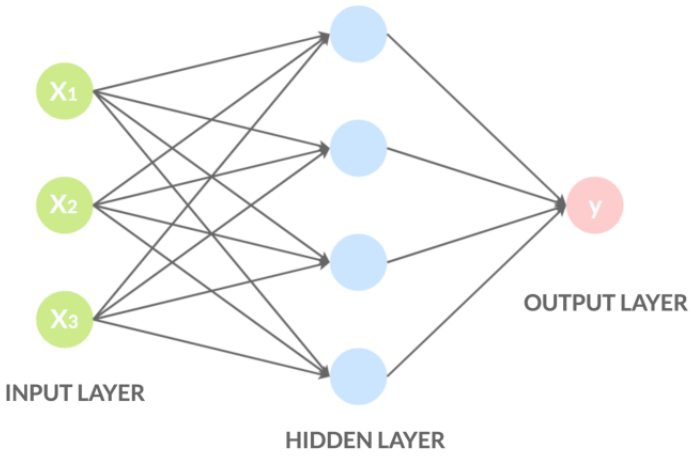


Figura 2.5: Schema Elementare di una Rete Neurale.

Le Reti Neurali sono composte da livelli di nodi che contengono un livello di input, uno o più livelli nascosti di input e di output (Fig. 2.5) [5] . Ciascun nodo si connette ad un altro in cui gli si associano un peso e una soglia. Se l'output di qualsiasi singolo nodo è al di sopra della soglia specificata tale nodo viene attivato

[5]Immagine tratta da: https://it.quora.com/

ed invia i dati al successivo livello di rete. In caso contrario non si passa alcun dato al livello successivo di rete.

Da un punto di vista matematico alla base delle Reti Neurali possiamo esprimere una funzione $f(x)$ come composizione di altre funzioni $g(x)$ che a loro volta possono essere espresse in funzioni più semplici. Dunque possiamo pensare ad una Rete Neurale come un insieme interconnesso di funzioni elementari in cui gli output sono gli input delle successive funzioni.

In generale, le Reti Neurali si affidano ai dati di addestramento per imparare a migliorare la loro esattezza. Una volta ottimizzati questi algoritmi di apprendimento sono dei potenti strumenti nella *computer science* e nell'Intelligenza Artificiale.

Alla base delle Reti Neurali c'è il Percettrone, in completa analogia con il Neurone in una rete neurale biologica. In Figura 2.5 ogni pallino è un nodo che rappresenta un Percettrone. Questi sono funzioni che prendono n elementi in input e restituiscono solamente una singola uscita, che viene inviata come input per i Percettroni successivi. Per tal motivo parliamo di Reti Neurali Stratificate.

2.4 Funzionamento rete neurale

Una rete neurale si presenta come un sistema "adattivo" in grado di modificare la sua struttura basandosi sia su dati esterni sia su informazioni interne che si connettono e passano attraverso la rete neurale durante la fase di apprendimento e ragionamento.Come? Una rete neurale biologica riceve dati e segnali esterni; questi vengono elaborati in informazioni attraverso un imponente numero di neuroni (che rappresentano la capacità di calcolo) interconnessi tra loro in una struttura non-lineare e variabile in risposta a quei dati e stimoli esterni stessi. È in quest'ottica che si parla dunque di "modello" matematico-informatico. Allo stesso modo, le reti neurali artificiali sono strutture non-lineari di dati statistici organizzate come strumenti di modellazione: ricevono segnali esterni su uno strato di nodi (che rappresenta l'unità di elaborazione, il processore); ognuno di questi "nodi d'ingresso" è collegato a svariati nodi interni della rete che, tipicamente, sono organizzati a più livelli in modo che ogni singolo nodo possa elaborare i segnali ricevuti trasmettendo ai livelli successivi il risultato delle sue elaborazioni (quindi delle informazioni più evolute, dettagliate).

In linea di massima, le reti neurali sono formate da tre strati:

- lo strato degli ingressi: è quello che ha il compito di ricevere ed elaborare i segnali in ingresso adattandoli alle richieste dei neuroni della rete;
- lo strato H – hidden: è quello che ha in carica il processo di elaborazione vero e proprio;

- lo strato di uscita: qui vengono raccolti i risultati dell'elaborazione dello strato H e vengono adattati alle richieste del successivo livello-blocco della rete neurale.

[9]

2.5 Algoritmi di apprendimento

Nelle reti artificiali ovviamente il processo di apprendimento automatico è semplificato rispetto a quello delle reti biologiche. Non esistono analoghi dei neurotrasmettitori, ma lo schema di funzionamento è simile. I nodi ricevono dati in input, li processano e sono in grado di inviare le informazioni ad altri neuroni. Attraverso cicli più o meno numerosi di input-elaborazione-output, in cui gli input presentano variabili differenti, diventano in grado di generalizzare e fornire output corretti associati ad input non facenti parte del training set. Gli algoritmi di apprendimento utilizzati per istruire le reti neurali sono divisi in tre categorie. La scelta di quale usare dipende dal campo di applicazione per cui la rete è progettata e dalla sua tipologia (feed-forward o feedback). Gli algoritmi sono:

- supervisionato;
- non supervisionato;
- di rinforzo;

2.5.1 Algoritmo supervisionato

Nell'apprendimento supervisionato(Fig. 2.6) [6] si fornisce alla rete un insieme di input ai quali corrispondono output noti (training set). Analizzandoli, la rete apprende il nesso che li unisce. In tal modo impara a generalizzare, ossia a calcolare nuove associazioni corrette input-output processando input esterni al training set. Man mano che la macchina elabora output, si procede a correggerla per migliorarne le risposte variando i pesi. Ovviamente, aumentano i pesi che determinano gli output corretti e diminuiscono quelli che generano valori non validi. Il meccanismo di apprendimento supervisionato impiega quindi l'Error Back-Propagation,algoritmo per l'allenamento delle reti neurali artificiali usato in combinazione con un metodo di ottimizzazione, ma è molto importante l'esperienza dell'operatore che istruisce la rete. Il motivo risiede nel non facile compito di trovare un rapporto adeguato fra le dimensioni del training set, quelle della rete e l'abilità a generalizzare che si tenta di ottenere. Un numero eccessivo di parametri in ingresso e una troppo potente capacità di elaborazione, paradossalmente, rendono difficile alla rete neurale imparare a generalizzare, perché gli

[6]Immagine tratta da: https://www.slideserve.com/brygid/reti-neurali-2

input esterni al training set vengono valutati dalla rete come troppo dissimili ai sofisticati e dettagliati modelli che conosce. D'altro canto, un training set con variabili scarse porta per la via opposta alla stessa conclusione: la rete, in questo caso, non ha sufficienti parametri per apprendere a generalizzare. Il giusto compromesso, insomma, è un compito che necessita di molta preparazione ed esperienza.

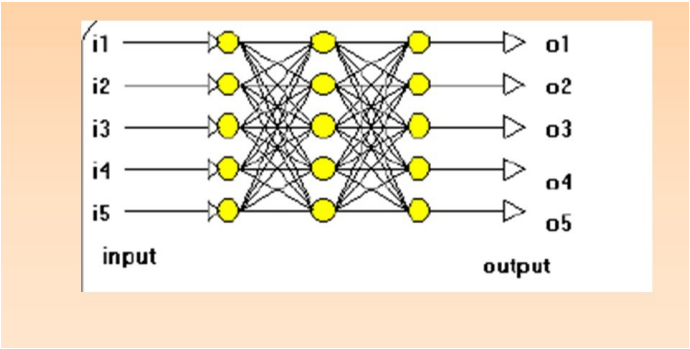


Figura 2.6: Apprendimento supervisionato

2.5.2 Apprendimento non supervisionato

In una rete neurale ad apprendimento non supervisionato(Fig. 2.7) [7], essa riceve solo un insieme di variabili di input. Analizzandole, la rete deve creare dei cluster rappresentativi per categorizzarle. Anche in questo caso i valori dei pesi è dinamico, ma sono i nodi stessi a modificarli. Esempi di reti ad apprendimento non supervisionato sono SOM e la rete di Hopfield.

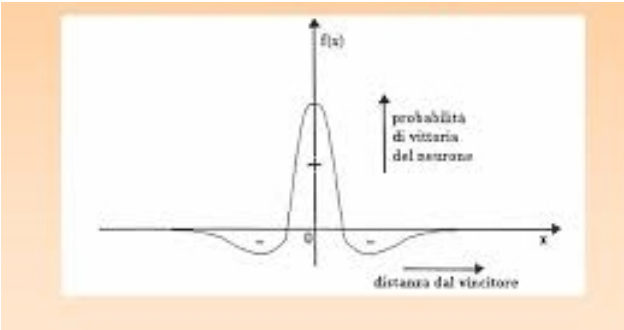


Figura 2.7: Apprendimento non supervisionato

[7]Immagine tratta da: https://www.slideserve.com/brygid/reti-neurali-2

2.5.3 Apprendimento per rinforzo

Nelle reti neurali che apprendono mediante l'algoritmo di rinforzo(Fig. 2.8) [8], non esistono né associazioni input-output di esempi, né un aggiustamento esplicito degli output da ottimizzare. I circuiti neurali imparano esclusivamente dall'interazione con l'ambiente. Su di esso, eseguono una serie di azioni. Dato un risultato da ottenere, è considerato rinforzo l'azione che avvicina al risultato; viceversa, la rete apprende a eliminare le azioni negative, ossia foriere di errore. Detto in altri termini, un algoritmo di apprendimento per rinforzo mira a indirizzare la rete neurale verso il risultato sperato con una politica di incentivi (azioni positive) e disincentivi (azioni negative).

Usando tale algoritmo, una macchina impara a trovare soluzioni che non è esagerato definire creative. Una rete neurale così implementata, per esempio, è stata utilizzata per giocare ad Arcade Breakout. Risultato: dopo sole 4 ore di continuo miglioramento, i circuiti hanno individuato una strategia di gioco mai ideata da un essere umano in questo videogame.

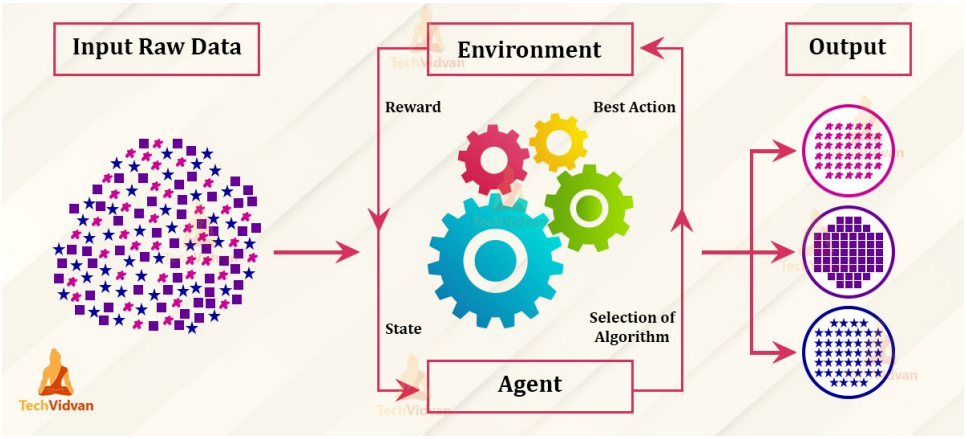


Figura 2.8: Apprendimento per rinforzo

2.6 Vantaggi e svantaggi rete neurale

Possiamo elencare importanti vantaggi che hanno le reti neurali come:

- elevato parallelismo, per cui possono processare in tempi abbastanza rapidi grandi quantità di dati;
- tolleranza ai guasti;
- tolleranza al rumore;

[8] Immagine tratta da: https://datapeaker.com/it/big-data/introduzione-all'27apprendimento-per-rinforzo-per-principianti/

- evoluzione adattiva, si aggiorna in presenza di modifiche ambientali;

Oltre a questi numerosi vantaggi, però, le reti neurali presentano anche degli svantaggi:

- funzionamento al black-box, sono in grado di fornire output corretti ma non si riesce ad esaminare i singoli stadi che li determinano;
- gli output forniti spesso non rappresentano la soluzione perfetta;
- il periodo di learning è più o meno lungo, poichè dipende da molti fattori come il numero e la complessità delle variabili di input, algoritmo utilizzato...
- non sono idonee a risolvere determinate categorie di problemi,

[10]

Capitolo 3

Strumenti Utilizzati

In questo capitolo andremo ad illustrare gli strumenti utilizzati quali *MediaPipe Face Mesh*, un dataset ottenuto dalla ricerca tramite Api di Google, Serpapi, la *MobileNetV2* e le Librerie, tra le quali OpenCV e Keras.

3.1 MediaPipe Face Mesh

MediaPipe Face Mesh è una soluzione per la geometria del viso che stima 468 punti di riferimento 3D del viso. Utilizza l'apprendimento automatico per dedurre la geometria della superficie facciale 3D approssimativa da un'immagine o da un flusso video, richiedendo solo un singolo ingresso della telecamera senza la necessità di un sensore di profondità.Può individuare caratteristiche come occhi, naso e labbra all'interno del viso, inclusi dettagli come i contorni delle labbra e la silhouette del viso. [11]

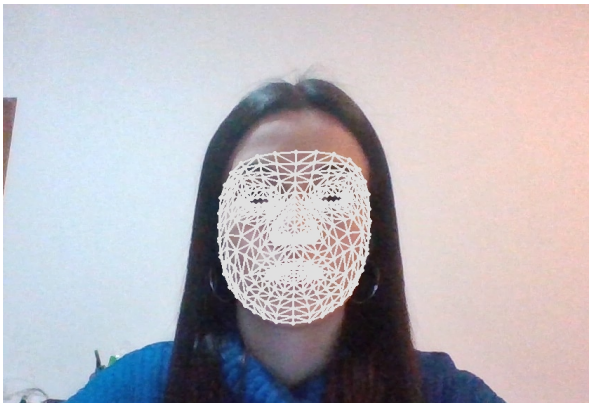


Figura 3.1: Esempio di maschera facciale.

Questa soluzione offre prestazioni in tempo reale fondamentali per le esperienze online.
Cose che possiamo eseguire con l'aiuto di *MediaPipe*:

- rilevamento facciale e mesh facciale;
- posa e rilevamento olistico;
- rilevamento e tracciamento di oggetti;
- stima della Posa;

Possiamo trovare diversi modelli di riferimento tra cui il Modello di riferimento facciale. [12]

3.2 Dataset

Con il termine *Dataset* andiamo ad indicare l'insieme dei dati con cui il modello verrà addestrato. In questo elaborato abbiamo utilizzato un *Dataset* per l'implementazione della *Neural Network* o rete neurale. Inizialmente, per il rilevamento degli occhi e delle loro parti, siamo andati a ricercare delle immagini tramite il motore Google e suddividerlo poi in base alle nostre occorrenze. Questo *Dataset* è un set di dati su larga scala delle immagini dell'occhio umano. La particolarità di questo set di dati è che le immagini sono suddivise in diverse categorie, il che le rende più adatte per l'addestramento.

Figura 3.2: Esempio del Dataset

3.3 MobileNetV2

MobileNetV2 è un'architettura di rete neurale convoluzionale.[1] Si basa su una struttura residua invertita in cui le connessioni residue sono tra gli strati del collo di bottiglia. Lo strato di espansione intermedio utilizza leggere circonvoluzioni in profondità per filtrare le caratteristiche come fonte di non linearità. Nel complesso, l'architettura di *MobileNetV2* contiene il livello iniziale di convoluzione completa con 32 filtri, seguito da 19 livelli residui di collo di bottiglia. [13] (Fig. 3.3)

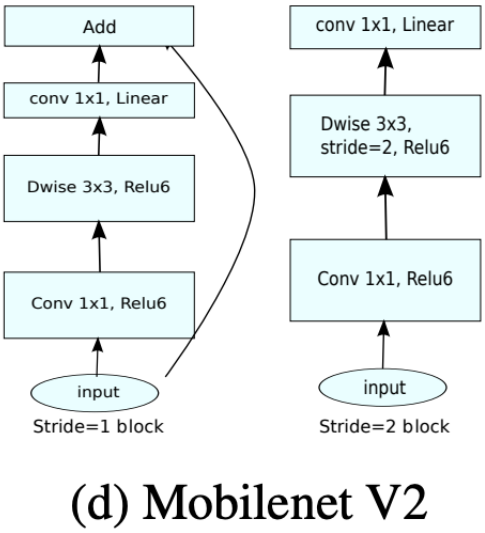


Figura 3.3: Struttura della MobileNetV2.

3.4 Librerie

3.4.1 OpenCV

OpenSource Computer Vision Library o *OpenCV* è una libreria *software open-source* che opera nell'ambito della *Computer Vision* e del *Machine Learning*. Per lo sviluppo del progetto di cui ci siamo occupati, vengono utilizzate principalmente librerie per la visualizzazione delle immagini.

[1] servono per estrarre le caratteristiche delle immagini o video, utilizzando come operatore principale le convoluzioni

3.4.2 Keras

Keras è una libreria *opensource* che serve per l'apprendimento automatico e per l'addestramento delle Reti Neurali, usando il linguaggio *Python*.

Lo scopo di tale libreria è quello di permettere la configurazione di reti neurali; *Keras* non funge da *Framework* ma da interfaccia semplice (API) per l'accesso e programmazione a diversi *Framework* di apprendimento automatico.

Tra questi *Framework* supporta le librerie *TensorFlow*, *Microsoft Cognitive Toolkit* (in precedenza CNTK) e *Theano*.[2]

Questa libreria mette a disposizione dei componenti fondamentali su cui si va poi a sviluppare modelli complessi di apprendimento automatico.

In questo elaborato si fa uso della libreria *Keras* [14] per andare ad addestrare la *Neural Network*.

3.5 Linguaggi Utlizzati

Il principale linguaggio di programmazione utilizzato in questo lavoro è stato *Python*.

Python [15] e' un linguaggio di programmazione ad alto livello multi paradigma, open-source. Una caratteristica di questo linguaggio è che tutti i tipi di dato sono oggetti quindi, esso mette a disposizione funzionalità per la loro manipolazione. In esso è possibile definire dei moduli, file contenenti attributi e metodi che possono essere riutilizzati e inseriti in diversi programmi, senza ambiguità fra le variabili del programma principale e dei moduli.

[2]Libreria per la computazione numerica per sviluppare codice *Python*. La sintassi del *Theano* è molto simile a *Numpy*

3.6 Editor di Sviluppo

Per sviluppare il nostro progetto abbiamo utilizzato l'editor chiamato *Google Colab* con cui siamo andati a sviluppare un codice in *Python* e con il quale abbiamo addestrato la stessa *Neural Network*

3.7 Google Colab o Colaboratory

Google Colab è un ambiente che consente di utilizzare una scheda Gpu ad altissime prestazioni. L'approccio è molto semplice, poiché, una volta che si apre il notebook ci si connette per allocare una macchina standard per noi. La cosa interessante è che si può collegare Drive, quindi ci consente di selezionare l'account Google e di montare il nostro Drive di Google. Google Colab è una piattaforma che ci permette di eseguire codice direttamente sul Cloud, sfruttando la potenza di calcolo fornita da Google. Per scrivere un codice con *Google Colab*, si utilizza il *Jupyter Notebook* [16], ovvero documenti interattivi in cui possiamo andar a scrivere un codice; più precisamente questi documenti permettono di suddividere il codice scritto in celle per eseguire e spiegare il codice stesso. [17] Il Linguaggio più utilizzato su Colab è Python.

Analizziamo in dettaglio cosa ci possiamo fare con questa piattaforma:

- scrivere ed eseguire codice in Python,
- documenta il tuo codice che supporta equazioni matematiche,
- Integra *PyTorch, TensorFlow, Keras, OpencCV*,
- uso della GPU.

In conclusione possiamo dire che l'introduzione di *Colaboratory* (Fig. 3.4) ha semplificato l'apprendimento e lo sviluppo di molte applicazione di *Machine Learning*.

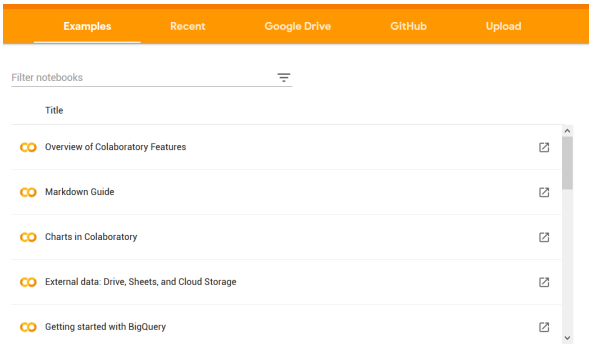


Figura 3.4: Interfaccia Iniziale Google Colab.

3.7.1 TensorFlow

TensorFlow [18] è una libreria software che permette di velocizzare significativamente i calcoli necessari ad un algoritmo di Machine Learning, contiene una serie di funzioni che sono in grado di sfruttare i Tensori . Tensorflow molto utilizzata in Machine Learning per creare modelli di CNN con poche righe di codice. Fa uso dio API come Keras che possiede, inoltre, le funzioni necessarie a creare e salvare modelli di CNN per poter essere utilizzati successivamente in altre applicazioni *"Cosa significa TensorFlor"*? Il nome è composto da due termini:

- *Tensor* (o Tensore in algebra lineare) é un array multidimensionale ossia una matrice di tre o più dimensioni.
- *Flow* è un flusso di operazioni.

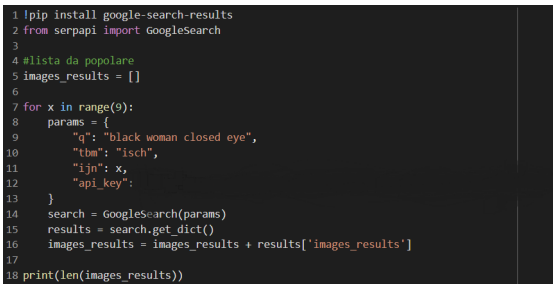
Capitolo 4

Sviluppo del Progetto

Grazie agli strumenti illustrati nei precedenti capitoli abbiamo una base solida per poter sviluppare il nostro progetto. L'obiettivo di questo elaborato è quello di implementare e addestrare una Rete Neurale in grado di rilevare l'apertura/-chiusura degli occhi. In modo particolare focalizziamo la nostra nostra attenzione sull'apertura e la chiusura degli occhi. Questo lavoro è stato diviso in due parti: la prima parte relativa all'utilizzo di Facemesh per l'estrazione degli occhi e della bocca, mentre la seconda riguarda l'addestramento della Rete Neurale tramite un set di immagini molto ampio. In questo elaborato andremo a trattare la parte relativa al data-set, mentre il mio collega Alesi Mattia si addentrerà nell'addestramento della rete e i risultati finali.

4.1 Creazione data-set

Come punto di partenza, ci siamo addentrati nella ricerca delle immagini di occhi aperti e chiusi che ci consentiranno poi di addestrare la rete neurale. Il nostro scopo è quello di aumentare il data-set della tesi precedente in modo da avere dei risultati migliori. Il dataset è stato creato utilizzando le Api messe a disposizione dal servizio SerpApi. Questa applicazione consente di utilizzare il motore di ricerca Google in modo che fornisca i risultati in formato Json. In questo modo, ogni elemento conterrà tra i suoi metadati l'URL univoco dell'immagine che si intende estrapolare, salvata tramite istruzione da riga di comando. Ripetendo il tutto per ogni elemento del risultato della query si ottiene il dataset di immagini.



```
!pip install google-search-results
from serpapi import GoogleSearch

#lista da popolare
images_results = []

for x in range(9):
    params = {
        "q": "black woman closed eye",
        "tbm": "isch",
        "ijn": x,
        "api_key":
    }
    search = GoogleSearch(params)
    results = search.get_dict()
    images_results = images_results + results['images_results']

print(len(images_results))
```

Figura 4.1: Codice per ricercare immagini tramite Serpapi

Come si può vedere dalla fig. 4.2, per andare a ricercare, siamo andati ad inserire dei parametri come ad esempio:

- *q*: parametro che definisce la query in qui eseguire la ricerca;
- *tbm*: è opzionale e definisce il tipo di ricerca che si vuole fare. In questo caso 'isch' indica l'API di Google Immagini;
- *ijn*: è opzionale e definisce il numero di pagina per Google Immagini(100 immagini per ogni pagina);
- *api key*: è il parametro che definisce la chiave privata SerpApi da utilizzare

In particolare, possiamo andare ad estrapolare gli URL delle immagini. Una volta fatta questa ricerca, andiamo a salvare il Json che ci restituisce e carichiamo uno script nel nostro editor, Google Colab, che ci consente di scaricare le immagini, salvarle per poi scremarle.

Per poter avere una suddivisione chiara, abbiamo creato delle cartelle:

- Open-eye: contiene immagini di volti con occhi aperti;
- Closed-eye: contiene immagini di volti con occhi chiusi;

Queste cartelle rappresentano le categorie di oggetti di cui si intende effettuare il riconoscimento tramite la rete neurale. Successivamente, ogni immagine sarà ritagliata in un quadro di 128x128 per poter isolare la sola parte degli occhi.

In aggiunta è presente anche una cartella lips (immagini con la bocca) che verrà poi utilizzata per sviluppi futuri.

Per poter rendere la base di dati condivisa, abbiamo memorizzato il tutto su Google Drive, accessibile da Google Colab, tramite il codice riportato in figura.

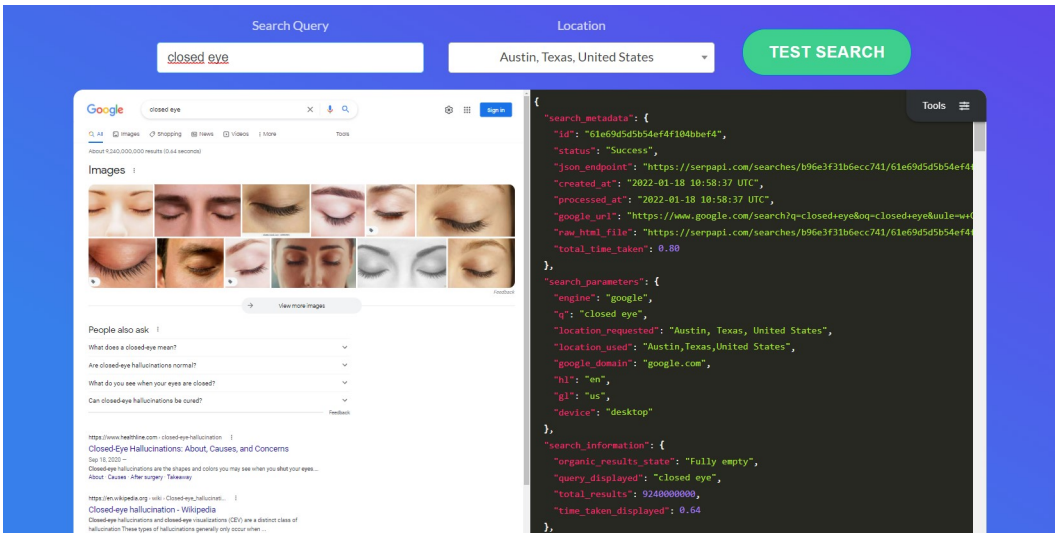


Figura 4.2: Ricerca immagine tramite SerpApi

```
1 import os, numpy as np
2 from google.colab import drive
3
4 drive.mount('/content/drive/')
5 main_dir = os.path.dirname("/content/drive/MyDrive/colab_img/")
6
7 img_count = 0
8 unrecognized = []
```

Figura 4.3: Importazione di Google Drive in Google Colab

4.2 Manipolazione immagini con FaceMesh

Il prossimo passo è utilizzare FaceMesh per poter estrarre il bounding box, o rettangolo, degli occhi e salvarli. Quindi, dopo aver salvato su Google Drive le immagini, è necessario elaborarle per convertirle in una forma che può essere usata come input dalla rete neurale MobileNetV2(Keras). Ma cosa fa nello specifico FaceMesh? E' una libreria che, a partire da una rete neurale pre-addestrata, è in grado di riconoscere i tratti somatici del volto. In particolare, è necessario convertire le immagini, tramite la libreria OpenCv, in oggetti che possano essere letti da FaceMesh , che a questo punto, effettua il riconoscimento e restituisce la posizione dei "landmarks" del volto.

Il ritaglio del solo elemento interessato (occhio aperto/chiuso) è svolto nativamente dal linguaggio Python

Ad ogni punto della mesh corrisponde un id, quindi Facemesh proietta questa mesh per avvolgere la faccia dell'immagine, poi, conoscendo le coordinate di

questi punti e gli id dell'occhio, ad esempio, si può tirare fuori solo l'occhio dell'immagine. Da qui, dovremmo estrarre 2 bounding box che contengono i due occhi, che saranno poi processati dalla rete neurale.

```
from drive.MyDrive.colab_img import face_mesh_connections

left_eye_idxs = set( [ i[0] for i in list(face_mesh_connections.FACEMESH_LEFT_EYE) ])
right_eye_idxs = set( [ i[0] for i in list(face_mesh_connections.FACEMESH_RIGHT_EYE) ])
lips_idxs = set( [ i[0] for i in list(face_mesh_connections.FACEMESH_LIPS) ])

crop_pxs_offset = (5,5)
crop_size = (128, 128)
```

Figura 4.4: Importazione id occhi Facemesh

In FaceMesh abbiamo una corrispondenza fra pixel e id dei punti che definiscono il viso. Questi id, per quanto riguarda gli occhi, sono fissi. Una volta che li conosciamo, prima trasformo le coordinate in pixel e poi faccio un controllo, ossia, se questo punto appartiene al set dell'occhio sinistro, lo considero come occhio sinistro, altrettanto per l'occhio destro. Dopodiché, tramite la libreria OpenCv mi ritaglio l'immagine in base a queste coordinate e, per ognuno, ci vado a creare un bounding box, definito da due coppie di pixel(in alto a a sinistra e in basso a destra) e tramite *cvcrop* andiamo ad estrarre solo quella porzione dell'immagine totale. Possiamo notare il risultato di FaceMesh nella variabile "results-multifacelandamarks". Tramite degli algoritmi che già esistono, come MediaPipe, decritto precedentemente, si può estrarre già il bounding box, corrispondente al rettangolo che include l'occhio sinistro e l'occhio destro. Questo rettangolo, poi, verrà passato alla rete neurale per addestrarla che quando troverà un'immagine poi sarà in grado di riconoscere se quell'occhio è aperto o chiuso. A questo punto, metteremo in esecuzione il modello su un dispositivo ad alte prestazioni, pensato per fare inferenza tramite reti neurali, Google Coral. Essa consente di effettuare calcoli ad alta velocità di tensori o matrici multidimensionali. Quindi l'obiettivo finale di questa prima fase è quello di crearsi un data-set con alcune immagini di occhi aperti e altre di occhi chiusi.

4.3 Addestramento rete neurale

L'addestramento di una rete neurale è la fase in cui, questa rete, impara a svolgere il suo compito. In questa fase, si vanno a tarare dei parametri in modo tale che l'output della rete, a fronte di un certo input, si avvicini a quello desiderato. In Google Colab tutto questo è possibile, grazie all'accelerazione della GPU. Abbiamo diviso il data set in due parti: una parte che ci servirà per per l'addestramento e una parte per la validazione. Nella parte vera e propria di addestramento, vengono analizzate le immagini, i risultati vengono confrontati con delle etichette e si aggiornano i pesi della rete. Mentre, la parte di validazione, viene fatta al termine di ogni epoca.

```python
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory as img_dataset_from_dir
import matplotlib.pyplot as plt

!mkdir /content/trim_img_dataset
!unzip /content/drive/MyDrive/colab_img/TRIM/trim_images.zip -d /content/trim_img_dataset/ #local filesystem

IMG_SIZE = (128, 128) #crop_size
BATCH_SIZE = 32

train_dir = "/content/trim_img_dataset"

train_dataset = img_dataset_from_dir(   train_dir,
                                        shuffle = True,
                                        batch_size = BATCH_SIZE,
                                        image_size = IMG_SIZE,
                                        validation_split = 0.2,
                                        subset = "training",
                                        seed = 123
                                        )

validation_dataset = img_dataset_from_dir( train_dir,
                                        shuffle = True,
                                        batch_size = BATCH_SIZE,
                                        image_size = IMG_SIZE,
                                        validation_split = 0.2,
                                        subset = "validation",
                                        seed = 123
                                        )

```

Come possiamo vedere dallo script, siamo andati a definire le dimensioni delle immagini tramite la variabile *'IMG SIZE'*, settata a (128,128) e il numero dei campioni tramite la variabile *'BATCH SIZE'*, settata a 32. A questo punto, possiamo andare a stampare le prime 9 immagini con le relative etichette che differenziano l'apertura e la chiusura degli occhi, che la rete dovrà riconoscere.

Figura 4.5: Dataset con le relative etichette

4.3.1 Data augmentation

Letteralmente, *data augmentation* significa *aumento dei dati* , ossia un insieme di tecniche che ampliano il dataset senza andare a raccogliere nuovi elementi, ma andando ad applicare ai dati già esistenti dei cambiamenti casuali. Nel nostro caso, facendo riferimento allo script, siamo andati ad effettuare, tramite Keras, un capovolgimento orizzontale e una rotazione in senso antiorario(poiché il valore è positivo).

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
    tf.keras.layers.experimental.preprocessing.RandomRotation(0.05),
])
```

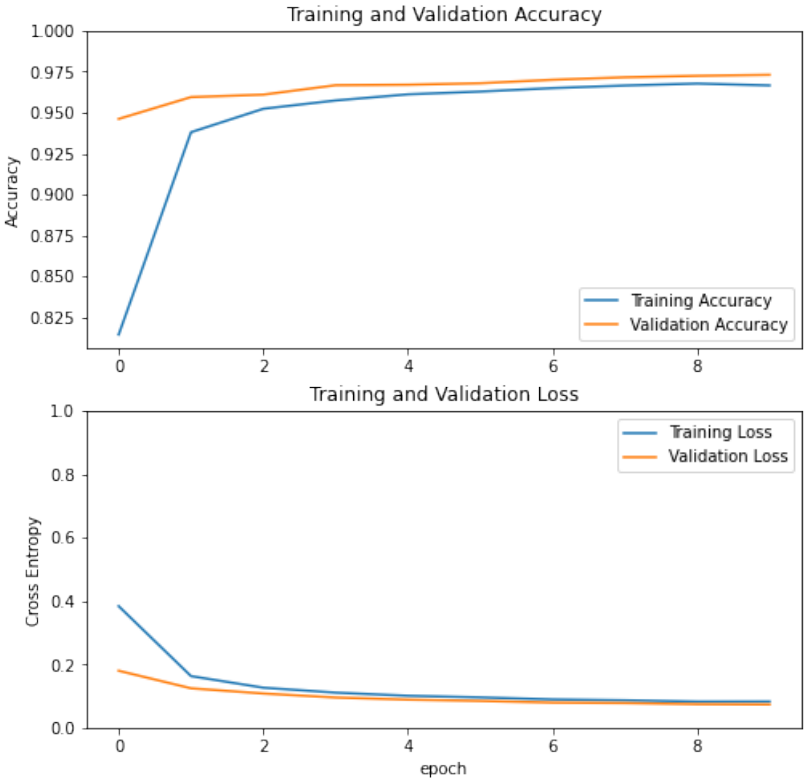
4.3.2 Feature Extraction

In questa fase, la prima cosa importante da fare, è quella di andare a congelare la base convoluzionale creata precedentemente per estrarre poi le caratteristiche. Per andare a vedere quali valori otteniamo per la *loss* e per l'*accuracy*, alleniamo le prime 10 epoche.

```
initial_epochs = 10
history = model.fit(train_dataset,
                    epochs = initial_epochs,
                    validation_data = validation_dataset)
```

```
Epoch 1/10
518/518 [==============================] - 224s 425ms/step - loss: 0.3838 - accuracy: 0.8144 - val_loss: 0.1800 - val_accuracy: 0.9461
Epoch 2/10
518/518 [==============================] - 217s 419ms/step - loss: 0.1631 - accuracy: 0.9380 - val_loss: 0.1245 - val_accuracy: 0.9595
Epoch 3/10
518/518 [==============================] - 216s 417ms/step - loss: 0.1265 - accuracy: 0.9523 - val_loss: 0.1081 - val_accuracy: 0.9610
Epoch 4/10
518/518 [==============================] - 216s 417ms/step - loss: 0.1109 - accuracy: 0.9574 - val_loss: 0.0955 - val_accuracy: 0.9667
Epoch 5/10
518/518 [==============================] - 215s 416ms/step - loss: 0.1010 - accuracy: 0.9612 - val_loss: 0.0887 - val_accuracy: 0.9670
Epoch 6/10
518/518 [==============================] - 222s 429ms/step - loss: 0.0959 - accuracy: 0.9628 - val_loss: 0.0847 - val_accuracy: 0.9679
Epoch 7/10
518/518 [==============================] - 219s 422ms/step - loss: 0.0894 - accuracy: 0.9650 - val_loss: 0.0794 - val_accuracy: 0.9700
Epoch 8/10
518/518 [==============================] - 217s 419ms/step - loss: 0.0867 - accuracy: 0.9665 - val_loss: 0.0780 - val_accuracy: 0.9716
Epoch 9/10
518/518 [==============================] - 218s 420ms/step - loss: 0.0826 - accuracy: 0.9677 - val_loss: 0.0745 - val_accuracy: 0.9725
Epoch 10/10
518/518 [==============================] - 212s 408ms/step - loss: 0.0828 - accuracy: 0.9666 - val_loss: 0.0735 - val_accuracy: 0.9731
```

Da queste, possiamo notare come i valori tendono ad aumentare man mano che si va avanti con le epoche. Possiamo anche avere una visualizzazione migliore attraverso i grafici che rappresentato, appunto, le curve di apprendimento di accuratezza perdita .



Notiamo che il parametro *accuracy* si assesta intorno al valore di 96.6 %, mentre il parametro *loss* è 8.2 %

4.3.3 Fine Tuning

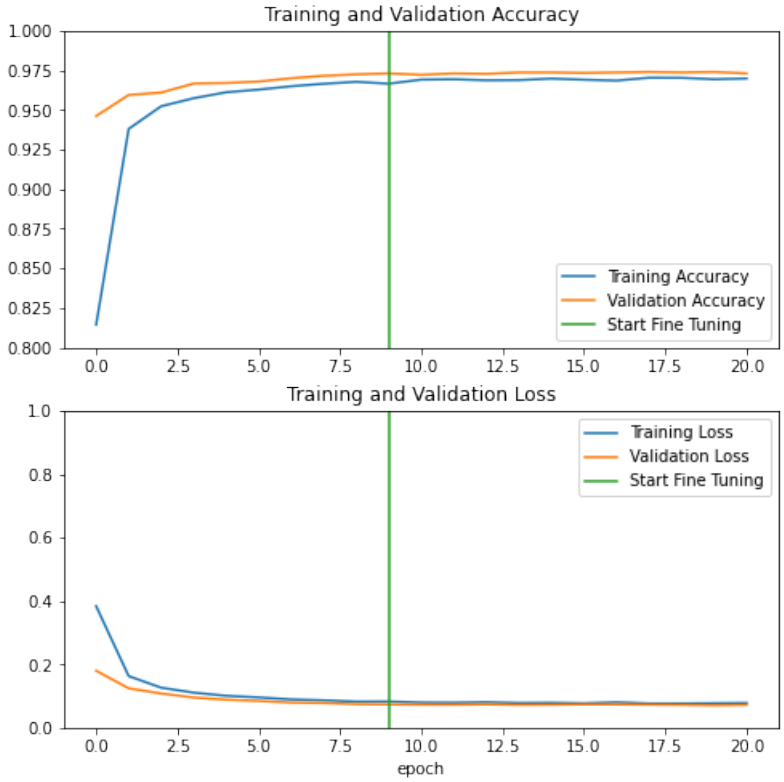
Questa fase ha l'obiettivo di migliorare le prestazioni del modello addestrato. Quindi, quello che ora dovremmo fare, è andare a scongelare il modello base e impostare come non addestrabili gli strati inferiori. Andiamo quindi ad osservare le ulteriori dieci epoche e i loro grafici relativi all'accuratezza e alla perdita.

```
fine_tune_epochs = 10
total_epochs =  initial_epochs + fine_tune_epochs

history_fine = model.fit(train_dataset,
                         epochs = total_epochs,
                         initial_epoch = history.epoch[-1],
                         validation_data = validation_dataset)
```

```
Epoch 10/20
518/518 [==============================] - 220s 418ms/step - loss: 0.0797 - accuracy: 0.9691 - val_loss: 0.0726 - val_accuracy: 0.9722
Epoch 11/20
518/518 [==============================] - 213s 411ms/step - loss: 0.0792 - accuracy: 0.9694 - val_loss: 0.0726 - val_accuracy: 0.9731
Epoch 12/20
518/518 [==============================] - 215s 415ms/step - loss: 0.0806 - accuracy: 0.9687 - val_loss: 0.0735 - val_accuracy: 0.9728
Epoch 13/20
518/518 [==============================] - 215s 415ms/step - loss: 0.0782 - accuracy: 0.9688 - val_loss: 0.0723 - val_accuracy: 0.9737
Epoch 14/20
518/518 [==============================] - 214s 414ms/step - loss: 0.0787 - accuracy: 0.9698 - val_loss: 0.0726 - val_accuracy: 0.9737
Epoch 15/20
518/518 [==============================] - 216s 416ms/step - loss: 0.0772 - accuracy: 0.9691 - val_loss: 0.0737 - val_accuracy: 0.9734
Epoch 16/20
518/518 [==============================] - 212s 409ms/step - loss: 0.0803 - accuracy: 0.9685 - val_loss: 0.0735 - val_accuracy: 0.9737
Epoch 17/20
518/518 [==============================] - 213s 411ms/step - loss: 0.0766 - accuracy: 0.9703 - val_loss: 0.0729 - val_accuracy: 0.9740
Epoch 18/20
518/518 [==============================] - 216s 417ms/step - loss: 0.0768 - accuracy: 0.9703 - val_loss: 0.0722 - val_accuracy: 0.9737
Epoch 19/20
518/518 [==============================] - 212s 409ms/step - loss: 0.0774 - accuracy: 0.9694 - val_loss: 0.0709 - val_accuracy: 0.9740
Epoch 20/20
518/518 [==============================] - 212s 409ms/step - loss: 0.0781 - accuracy: 0.9699 - val_loss: 0.0723 - val_accuracy: 0.9731
```

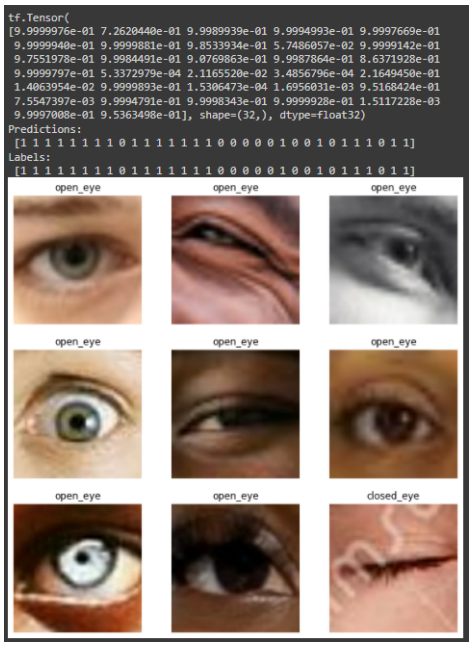
Dalla figura precedente possiamo notare il comportamento del sistema durante le ulteriore dieci epoche.



Analizzandole graficamente, notiamo che l'andamento delle curve segue il modello precedente, ma con un miglioramento delle prestazioni. In questo caso *l'accuracy* è al 97% e la perdita scende a 0.78%

4.3.4 Test

Ora, possiamo mettere alla prova il modello di rete neurale creato per valutare, praticamente, se è in grado di riconoscere il tipo di immagine che gli viene data in ingresso(occhio aperto o occhio chiuso).



Si evidenzia come il modello riesca ad effettuare il riconoscimento in modo ottimale, anche in condizioni "critiche" in cui l'immagine è leggermente artefatta o l'occhio non perfettamente centrale. Non si può escludere di migliorare ulteriormente la precisione del modello andando ad aumentare il numero di epoche di addestramento, il numero di batch o anche la risoluzione delle stesse immagini del data set.

4.4 Lettura di un video e predizione

Il passo successivo è stato quello di andare a creare dei video in cui, io ed il mio collega, andavamo ad aprire/chiudere gli occhi, anche con la presenza di occhiali. Successivamente, siamo andati a processare questi video.

```
success,image = video.read()
index = 1
ALL_FRAMES = []

while success:
    try:
        success,image = video.read()
        #cv2_imshow(image)
        ALL_FRAMES.append( image )
        print('Read frame ',index, " ",success)

        index += 1
    except:
        print("Error reading frame")
        continue
else: print("End of video")
```

Nel test che andiamo a fare, indichiamo con **0** :*occhio chiuso* e con **1** :*occhio aperto* e andiamo a fare la predizione di ogni bounding box (occhio sinistro e

occhio destro), convertendo le immagini in scala di grigi. A questo punto si ha una predizione e possiamo verificare se la rete neurale legge correttamente lo stato degli occhi e della bocca. Per fare ciò, vediamo lo stato degli occhi durante l'esecuzione del video se corrisponde al suo stato effettivo.

```
for index, frame in enumerate(ALL_FRAMES):
    print("\n\nVideo frame ", index)
    face_items = face_recognition(frame)
    if face_items:
        for item in face_items:
            img = crop_to_square(frame, item)
            eye_prediction(img)
    else:
        print("Can't detect face")
```

Ora, valutiamo i risultati attraverso la matrice di confusione (Fig. 4.6) , tabella riepilogativa utilizzata per valutare le prestazioni di un modello di classificazione.

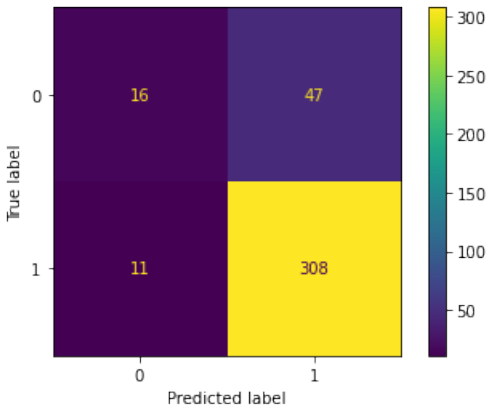


Figura 4.6: Matrice di confusione

In questa matrice, con *Predicted label* si indicano le classi previste dal modello, mentre con *True label* le classi corrette del dataset. Nella diagonale, si indicano le previsioni corrette del modello, mentre le altre celle le previsioni sbagliate. Il video di test è composto da 191 fotogrammi in cui sono sempre mostrati gli occhi; si hanno quindi 392 elementi (occhi) su cui effettuare il riconoscimento. Nel nostro caso è stato possibile riconoscere correttamente 324 immagini e le restanti 58 sono previsioni errate.

Capitolo 5

Conclusioni

Alla fine di questa tesi possiamo dire che questo progetto contribuirà allo sviluppo di un sistema per il riconoscimento del grado di stanchezza di un guidatore. La stanchezza alla guida, come descritto nei capitoli precedenti, è una tra le cause di incidenti stradali e, grazie a queste soluzioni, si garantisce un minor rischio sia per il conducente del veicolo ma anche per gli altri che, purtroppo, ne risulteranno coinvolti. Grazie a questo progetto ho anche appreso un nuovo linguaggio, Python, mai studiato prima di ora, di cui ho trovato interessante alcune implementazioni e di come esso, sia molto versatile. Tutto questo messo in pratica in un ambiente di sviluppo come Google Colab, il quale ci fornisce una GPU gratuita e un ambiente di lavoro che è possibile condividere tramite il proprio account di Google. Infine, l'argomento della tesi è stato per me molto interessante, poiché, ad oggi, la sicurezza stradale è un tema da non sottovalutare e, poter trovare soluzioni come quella che abbiamo trattato è stato per me di particolare interesse.

Bibliografia

[1] https://www.aci.it/laci/sicurezza-stradale/guida-e-sonnolenza/il-sonno-al-volante.html.

[2] https://www.elettronicain.it/blog/2019/12/26/le-telecamere-che-salvano-la-vita-a-conducente-e-passeggeri/.

[3] https://www.avvenire.it/economia/pagine/rilevatore-di-stanchezza-in-auto-cos-e-e-come-funziona.

[4] https://it.mathworks.com/discovery/neural-network.html.

[5] https://www.economyup.it/startup/intelligenza-artificiale-che-cos-e-e-perche-trasformera-le-aziende/.

[6] https://www.intelligenzaartificiale.it/.

[7] https://www.innovationpost.it/2018/02/14/intelligenza-artificiale-deep-learning-e-machine-learning-quali-sono-le-differ

[8] %http://www.umbertosantucci.it/atlante/deep-learning/.

[9] https://www.ai4business.it/intelligenza-artificiale/deep-learning/reti-neurali/.

[10] https://www.intelligenzaartificiale.it/reti-neurali/.

[11] https://blog.tensorflow.org/2020/03/face-and-hand-tracking-in-browser-with-mediapipe-and-tensorflowjs.html.

[12] https://google.github.io/mediapipe/solutions/face_mesh.html.

[13] https://paperswithcode.com/method/mobilenetv2.

[14] https://www.ionos.it/digitalguide/online-marketing/marketing-sui-motori-di-ricerca/cose-keras/.

[15] https://www.python.it/.

[16] https://jupyter.org/.

[17] https://isolution.pro/it/t/google-colab/what-is-google-colab/google-colab-che-cos-e-google-colab.

[18] https://www.tensorflow.org/.

[19] https://www.andreaminini.com/ai/tensorflow/.

Elenco delle figure

Ringraziamenti

Al termine di questo percorso, desidero ringraziare alcune persone che mi hanno supportato per poterlo concludere al meglio.

Ringrazio la mia famiglia per tutto l'appoggio e l'aiuto che mi hanno dato, nonostante le mille difficoltà, soprattutto durante il primo anno ed è proprio grazie a loro che sono riuscita a superare questi momenti e continuare questo percorso nel migliore dei modi.

Ringrazio i miei colleghi di Università, con i quali ho condiviso momenti di studio. Ringrazio le mie amiche che mi hanno sempre incoraggiato con i loro consigli e ascoltato nei momenti piu' bui, standomi sempre vicine.

Infine, ringrazio il mio relatore Adriano Mancini per la possibilità che mi ha dato e per i preziosi consigli e suggerimenti.