UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Magistrale in INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Sviluppo di un applicativo per l'identificazione dei tumori cutanei compatibile con i principali sistemi operativi per la diagnosi precoce

Development of an application for skin cancer detection compatible with main operating systems for early diagnosis

Relatore: Chiar.mo

Prof. **Aldo Franco Dragoni**

A cura di:

Francesco Baioni

A.A. 2022/2023

"*L'immaginazione è più importante della conoscenza. La conoscenza è limitata, l'immaginazione abbraccia il mondo*".

Sommario

L'avvento dell'era digitale e delle tecnologie in ambito informatico ha aperto nuovi orizzonti in tutti i campi della scienza. In particolare, con l'ausilio dell'Intelligenza Artificiale (IA) si è rivoluzionato non soltanto l'approccio mentale allo svolgimento di determinate azioni, ripetitive o meno che siano, ma se ne sono anche migliorate significativamente le prestazioni. Tante, ad oggi, sono le applicazioni dell'IA in ambito medico come strumento di supporto diagnostico, tra cui quella in ambito dermatologico. La diagnosi precoce di neoplasie alla pelle è, infatti, cruciale per aumentare le probabilità di guarigione e migliorare la qualità di vita dei pazienti. In risposta a questa esigenza, la presente tesi ha l'obiettivo di proporre un applicativo multi-piattaforma rivolto all'identificazione dei tumori della pelle, allo scopo di fornire a chi ne fa uso uno strumento capace di effettuare un'analisi preliminare di una o più lesioni cutanee. La metodologia utilizzata vede l'applicazione di reti neurali convoluzionali presenti in letteratura per la detection, segmentazione e classificazione di suddette neoplasie. E' stato, quindi, ottenuto un applicativo capace di girare sui principali sistemi operativi, sia desktop sia mobile, previa corretta conversione del codice utilizzato, fornendo così un supporto concreto circa la necessità o meno di un'investigazione più approfondita. Da sottolineare è come questa soluzione non debba essere intesa come un completo sostituto del medico specialista, bensì come uno strumento di supporto da integrare all'iter diagnostico.

Indice

Elenco delle figure

Introduzione

La pelle è l'organo più grande del corpo umano ed è esposta a una vasta gamma di agenti ambientali che possono contribuire allo sviluppo di tumori cutanei. L'incidenza di queste neoplasie è in costante crescita, con una stima di milioni di nuovi casi diagnosticati ogni anno in tutto il mondo. Tra i diversi tipi di tumori cutanei, giusto per citarne un esempio maggiormente conosciuto, il melanoma è uno dei più aggressivi, con potenziali metastasi e risultati fatali se non diagnosticato e trattato preventivamente. Nonostante rappresenti solo il 4% di tutti i tumori della pelle, è responsabile del 75% di tutte le morti per cancro cutaneo[5]. Anche i tumori diversi dal melanoma, come il carcinoma basocellulare e il carcinoma spinocellulare, rappresentano un carico significativo per il sistema sanitario, con impatti sulla salute dei pazienti e sui costi associati alle cure.
La diagnosi precoce dei tumori della pelle è pertanto strettamente correlata all'aspetto del paziente. Tuttavia, il processo di rilevamento può essere complesso e richiede competenze specializzate dettate da anni di esperienza e approfondito studio della materia. Questo ha portato alla necessità di sviluppare soluzioni innovative che possano supportare sia i professionisti della salute che il pubblico generale nel riconoscimento delle lesioni cutanee sospette.
Pertanto, alla luce di ciò, è evidente come interessante, formativo e soprattutto proficuo possa apparire la creazione di un sistema che, mediante l'ausilio dell'IA, riesca in maniera automatizzata a determinare l'entità di una lesione della pelle. In tutto questo, la fa da padrone la già citata IA, componente senza la quale l'intero progetto non sarebbe possibile. Infatti, il punto cardine nell'utilizzo di tale tecnologia e che, per certi versi, ne rappresenta il motivo di cotanta diffusione ad oggi, nella buona e nella cattiva sorte, risiede proprio nel fatto che mediante il suo utilizzo si riescono a normalizzare determinate attività altrimenti complicate e che richiederebbero l'intervento di

esperti. Il tutto arricchito dal fatto che, con una minima dose di conoscenza in ambito informatico ed un hardware non eccessivamente performante, si è in grado di far apprendere alla macchina, con un sistema di addestramento, i task più semplici come il riconoscimento di oggetti all'interno di un'immagine con ottimi risultati. Con questo ovviamente, ed è importante sottolinearlo, non significa che il lavoro e l'esperienza dei professionisti verranno completamente sostituiti da questi meccanismi, piuttosto deve essere pensato come un supporto agli stessi e non.
Detto questo, il tema centrale che vede lo sviluppo di questa tesi riguarda proprio la progettazione di un'applicazione multi-piattaforma, atta a facilitare la cattura e l'analisi delle immagini delle lesioni cutanee, utilizzando algoritmi di deep learning per l'identificazione di potenziali aree di interesse dannose, fornendo un risultato finale dell'analisi fatta in toto circa l'immagine acquisita.
Nello specifico, verrà descritto nel primo capitolo da dove ha origine l'idea, in quanto proposta di un esame tenuto ormai quasi due anni fa e soprattutto lo sviluppo di un primo prototipo, che seppur diverso nel formato, portava ad un risultato molto simile. Dopodiché, si darà uno sguardo approfondito a quella che è la letteratura e lo stato dell'arte attuale relativamente a progetti simili o che condividono in minima parte l'obiettivo di questo lavoro, mettendo proprio a confronto pro e contro di entrambe le parti. Menzione importante la ottengono il terzo ed il quarto capitolo invece, dove si andrà ad illustrare prima la metodologia utilizzata per lo sviluppo dell'app, quindi pipeline di esecuzione, task, tecniche e ragionamenti, per poi passare direttamente all'implementazione pratica, quindi ambiente di sviluppo, linguaggi di programmazione e così via.
Il tutto per concludere ovviamente con i risultati che verranno approfonditi nel quinto ed ultimo capitolo, assieme alle valutazioni complessive del progetto per poi terminare con le vere e proprie conclusioni.

Capitolo 1

Origini

1.1 Idea

Seppur poco utile ai fini dei risultati finali, è bene parlare anche delle origini che questo progetto ha avuto. Infatti, l'idea dell'applicativo nasce ben due anni prima in occasione dello svolgimento di un esame tenuto dallo stesso relatore di questo lavoro. L'esame, dal nome Sistemi Operativi Dedicati, esplodeva la schedulazione dei processi in contesti "real-time" e l'elaborazione concorrente in ambienti distribuiti privi di memoria condivisa. Inoltre, il corso approfondisce anche la programmazione a livello kernel nel Sistema Operativo Linux, sistema open source ampiamente diffuso con licenza GPL. In questo contesto, un po' per curiosità ed un po' per sfida personale, il sottoscritto propose l'idea di costruire interamente da zero un dispositivo basato su una qualche board del tipo Arduino/Raspberry, con annesso modulo camera, capace di effettuare delle foto da dare in pasto a reti neurali in grado a loro volta di riconoscerle più dettagliatamente di quanto non facesse l'occhio umano. E perché no, anche l'occhio di un esperto.
Il tutto, ovviamente, trovava le radici nell'ambito medico e, più precisamente, in quello del riconoscimento delle neoplasie, che come già anticipato risulta essere un problema assai diffuso. Questo infatti portava ad uno dei primi e principali punti concernenti questo lavoro, allo stadio quasi primordiale: ovvero, la capacità di fornire ad un qualsiasi utente finale la possibilità di utilizzare un dispositivo, ad un prezzo più che ragionevole (dove il costo era dato esclusivamente dall'acquisto delle poche componenti che lo costituivano, facenti parte dell'elettronica di largo consumo) per la scansione delle proprie

lesioni e la diagnosi preventiva. Va da sè che, come già detto, l'elemento che costituiva il valore aggiunto all'intero progetto era proprio la parte di intelligenza artificiale, la quale, attraverso del comunissimo hardware e qualche ora di addestramento, rendeva possibile il riconoscimento di ben otto diverse categorie di tumori[6] con elevate percentuali di veri positivi. Queste erano:

- Actinic keratoses and intraepithelial carcinoma / Bowen's disease
- Basal cell carcinoma
- Benign keratosis-like lesions or solar lentigines
- Melanoma
- Melanocytic nevi or commonly moles
- Dermatofibroma
- Vascular lesion
- Squamous cell carcinoma

Inoltre, il dispositivo costruito si poteva adoperare in tutto e per tutto come dispositivo a sè stante: ovvero erano stati implementati task che facevano sì che lo stesso potesse continuare a scattare e riconoscere nuove foto, a discrezione dell'utente, oppure spegnere il dispositivo attraverso l'apposito pulsante posto lateralmente. Nel capitolo successivo poi ne vengono approfonditi gli aspetti implementataivi, più tecnici.
C'è da sottolineare che, nonostante fosse un primo becero tentativo, racchiudeva già al proprio interno quasi tutte le caratteristiche ritenute necessarie per il corretto utilizzo di uno strumento del genere: possibilità di scatare foto, elevato rate di riconoscimento di queste e loop dell'intero processo fino a spegnimento del dispositivo. Va da sé che non mancavano punti deboli all'interno dell'intera struttura di sviluppo, primo fra tutti l'innegabile vincolo dato dall'alimentazione del dispositivo (direttamente conneso alla rete elettrica casalinga poiché basato su scheda Raspberry che ha il proprio alimentatore), giusto per citarne uno. Altra pecca risultava essere la poca fluidità del sistema: infatti soventi erano dei piccoli lag o blocchi del sistema, niente che andava ad impattare così drasticamente il suo corretto utilizzo ma che nell'esperienza utente vanno comunque ad incidere.

Infatti, proprio in fase di chiusura esame, sono state proposte anche dei follow up, futuri upgrade da poter applicare al sistema nel suo complesso, da cui l'idea di creare direttamente l'applicativo a livello software, multi-piattaforma, oggetto di questa tesi.

1.2 Primo prototipo

Come già anticipato, il primo prototipo di Skin Detector è stato sviluppato in occasione di un esame e consisteva in una grezza ma allo stesso tempo abbastanza completa e complessa versione dell'attuale lavoro. Infatti, si possono distinguere due "modelli" del sistema descritto:

- Dispositivo fisico costituito da board Raspberry Pi 3[7], con tanto di modulo camera, display LCD e tasti per selezione features
- Applicazione software multi-piattaforma per smartphones o pc, oggetto di questo lavoro

Da notare come, già all'epoca, si era pensato proprio ad entrambe le varianti al fine di poter fornire un prodotto specifico per ogni tipo di utilizzo. Detto questo, nel primo caso si parla di un vero e proprio apparecchio fisico, il quale è andato in contro non solo ad uno studio dal punto di vista software (linguaggio di programmazione, codici, reti neurali etc) ma anche dal punto di vista hardware e di progettazione di quest'ultimo. Infatti, dal momento che tutte le componenti fisiche del sistema dovevano coesistere all'interno dello stesso spazio, preferibilmente di minime dimensioni al fine di favorirne l'utilizzo, è stata necessaria una stratificazione delle parti e conseguente isolamento elettrico tra di loro. Di seguito alcuni esempi della struttura esterna.

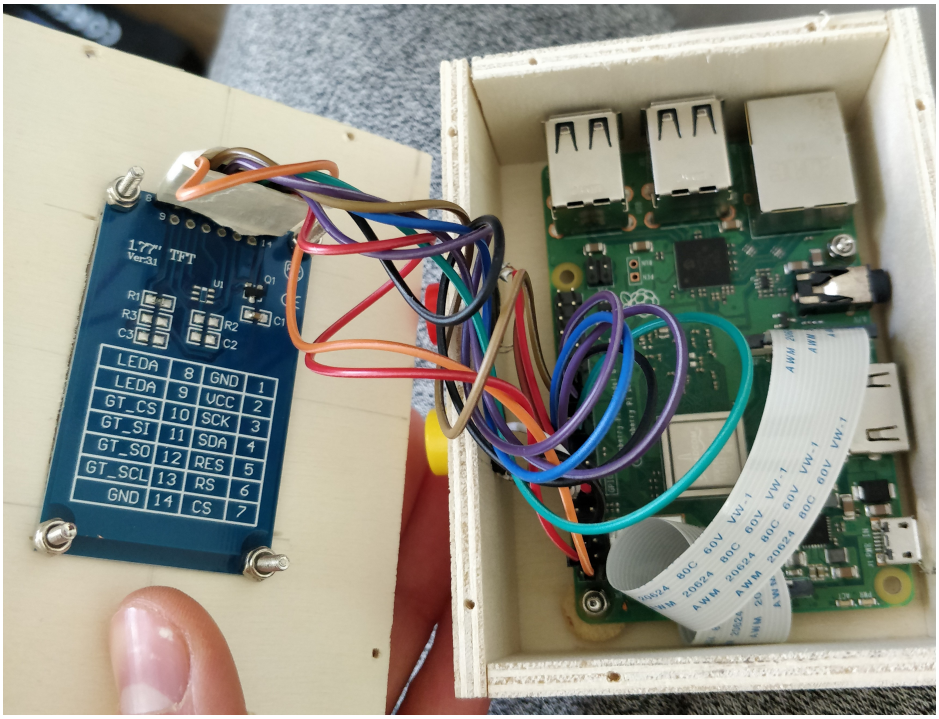

Figura 1.1: Architettura - primo prototipo

Come si può notare, il manufatto risulta essere chiaramente un prototipo ed, in quanto tale, grezzo, in primis nella sua forma estestica, ed in secundis nella composizione che si trova all'interno. Purtroppo l'oggetto è stato completamente costruito da zero dalle inesperte mani del sottoscritto.
In ogni caso, nel complesso lo strumento era utilizzabile ed adempieva ai suoi task. Tutto ciò veniva permesso dalla scheda principale di lavoro, la quale, come già citato, risultava essere una Raspberry Pi 3, pressoché un mini computer, il cuore centrale e operativo del sistema. A questa erano collegati:

- Due pulsanti, giallo e rosso, rispettivamente per scattare le foto e spegnere il dispositivo
- Modulo camera Picamera da 5 Megapixel[8], con annessa lente macro per ottenere foto più nitide di oggetti più piccoli, quali la maggior parte dei nei
- Un SPI TFT Display da 1.77 pollici e 128x160 Pixel

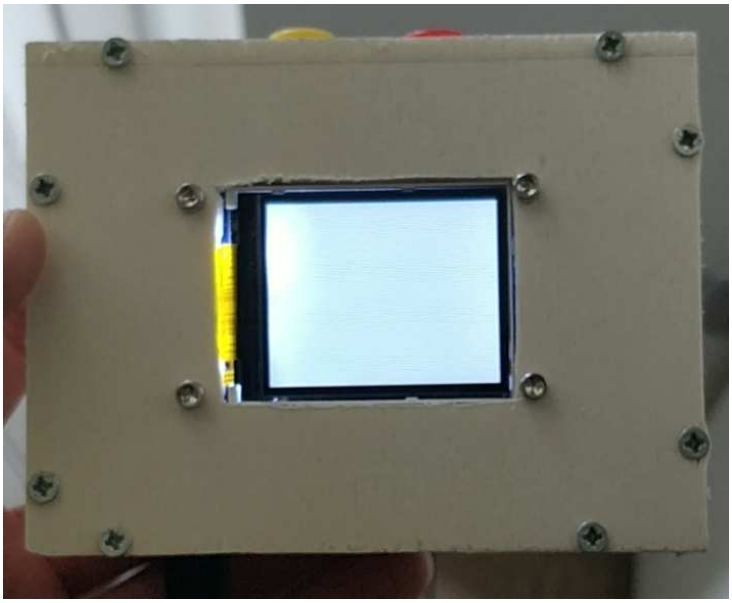

Figura 1.2: Illustrazione - primo prototipo

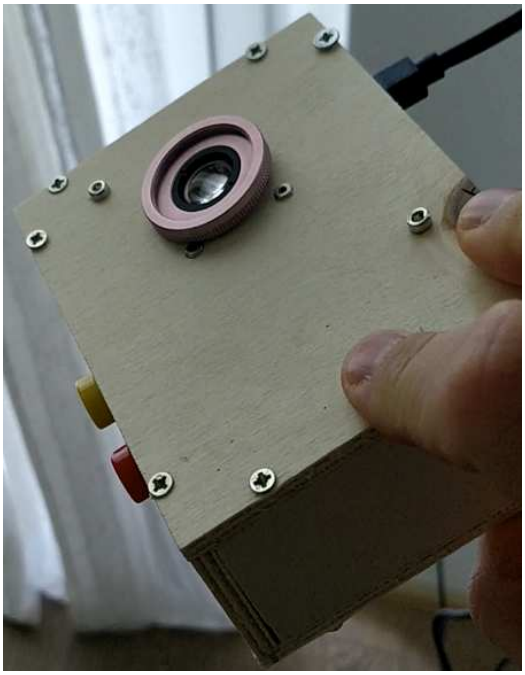

Figura 1.3: Illustrazione - primo prototipo

Tutto quanto appena descritto rappresentava la base hardware, sulla quale occorreva, ovviamente, far girare il software. Di fatti, il sistema operativo utilizzato è quello standard per le schede Raspberry, ovvero il Raspberry Pi OS[9], distro Unix leggerissima pensata appositamente per questi hardware. Quest'ultima è stata poi ottimizzata in termini di task da svolgere, i quali venivano caricati non appena il sistema veniva attaccato all'alimentazione e quindi avviato. In particolare, veniva fatto eseguire uno script in Python[10] che gestiva quasi completamente l'intero processo. Lo script, infatti, una volta importate le corrette librerie, tra cui quelle necessarie per la prediction della rete neurale, caricava il modulo camera avviando così la fotocamera che permetteva, attraverso la pressione del pulsante giallo, di scattare una foto. Quest'ultima poi veniva analizzata dalla rete neurale, un modello di MobileNetV2[11], precedentemente caricata, la quale forniva una predizione sull'appartenenza della stessa foto alle categorie di neoplasie su cui era stata fatta addestrare (le otto classi descritte nella sezione precedente).
Tutto ciò ha portato ad avere un'accuratezza massima di circa l'80%, non male per un prototipo. Basti pensare che la percentuale dei veri positivi dei dermatologi si attesta intorno all'86% e l'accuratezza all'80%[12].
Questo racchiude un po' tutte le motivazione e lo sviluppo del progetto che hanno portato poi alla stesura di questo lavoro, di cui nei seguenti capitoli si entrerà più nello specifico.

Capitolo 2

Tecniche e tecnologie previste

Le tecniche e tecnologie implementate in questo lavoro sulle quali varrebbe la pena soffermarsi sono fondamentalmente due: reti neurali convoluzionali e metodi python per la gestione delle immagini. Questi ultimi derivano per la quasi totalità da un unico pacchetto chiamato opencv-python e sono discretamente semplici da applicare, avendo chiaro in mente il risultato che si vuole ottenere. Per esempio: per effettuare una rotazione di x gradi si può utilizzare il metodo cv2.rotate(); per disegnare un cerchio cv2.circle() e così via. Nei prossimi capitoli verranno analizzati più a fondo.
In questo capitolo, invece, si vuole porre maggiormente l'attenzione sulle reti neurali, riassumendo la loro architettura, il funzionamento e le modalità di utilizzo previste per questa tesi.
Le CNN, o Convolutional Neural Networks, o ancora ConvNets, sono un tipo di rete neurale artificiale specializzate nell'elaborazione di dati in forma di griglia, come immagini e video. Per questo, vengono ampiamente utilizzate nell'ambito dell'elaborazione delle immagini grazie alla loro capacità di apprendere pattern e feature da queste ultime.
Ne esistono di tanti diversi tipi e diversi sono di conseguenza gli scopi verso cui vengono applicate, sebbene vi possano essere modelli diversi anche per uno stesso scopo. Infatti, tra un modello e l'altro spesso ciò che cambia è l'architettura. Questa è un aspetto cruciale per il successo della rete nel proprio task. Fondamentalmente, ogni rete è costituita da strati che possono essere di vario tipo:

- Strati Convoluzionali, costituiscono il cuore della rete, contenendo al proprio interno i cosiddetti neuroni, kernel di convoluzione in grado di

scansionare l'immagine e rilevarne dei pattern

- Strati di pooling, di solito posizionati dopo quelli convoluzionali, possono essere composti da più neuroni collegati a tutti i neuroni dell'output precedente e svolgono il ruolo di classificare le feature estratte nelle fasi precedenti in classi specifiche

- Strati Completamente Connessi, solitamente situati alla fine della rete, hanno la funzione di classificare le feature estratte nelle fasi precedenti in classi

- Strati di Attivazione, aggiungono la capacità di apprendere da dati non lineari attraverso le funzioni di attivazione

- Input e Output Layer, rispettivamente strato di inizio e fine rete corrispondenti ad un'immagine che deve essere analizzata e la classificazione della stessa in base a classi specifiche

Questi gli strati principali adottati in ogni rete che si rispetti. Ovviamente, ne sono presenti altrettanti ma l'obiettivo è quello di fornire una breve panoramica sulla struttura.
Procedendo invece verso i task realizzabili attraverso questi sistemi, primo tra tutti è la classificazione. La classificazione è il processo di assegnazione di un'etichetta o una classe a un'intera immagine. Le fasi principali sono:

- Estrazione delle Feature

- Classificazione sulla base del punto precedente. Spesso, viene utilizzato uno strato denominato softmax per ottenere le probabilità delle diverse classi

- Addestramento Supervisionato su un ampio dataset di immagini etichettate con le rispettive classi

Di esempi ne esistono tanti che vanno dalla più utilizzata ResNet, citata anche nel proseguo di questo lavoro, alla AlexNet passando per InceptionNet. Dopodiché, un altro task assai diffuso è quello della segmentazione. La segmentazione è il processo di suddivisione di un'immagine in regioni o pixel che condividono caratteristiche simili. Anche qui le fasi principali sono:

- Rete encoder, che comporta l'apprendimento delle feature dall'immagine, proprio come nelle CNN di classificazione
- Rete decoder, che utilizza le feature apprese per creare una mappa di segmentazione, assegnando un'etichetta a ciascun pixel
- Up-sampling, spesso utilizzata per aumentare la risoluzione della mappa di segmentazione

Anche qui gli esempi più conosciuti possono essere la U-Net, la SegNet o la Deeplab.
Infine, un ulteriore task largamente sfruttato ed implementato in questa tesi, come i due precedenti appena descritti, è quello della detection. Questo rappresenta il processo di individuazione e classificazione degli oggetti all'interno di un'immagine, per il quale le principali fasi sono:

- Estrazione delle feature attraverso una serie di strati convoluzionali e di pooling che estraggono automaticamente feature rilevanti dall'immagine
- Detection, la quale utilizza le feature estratte per individuare oggetti nell'immagine. Spesso, sono utilizzati box di ancoraggio (o bounding boxes) per circondare gli oggetti individuati
- Classificazione in cui vengono classificati gli oggetti individuati in base alla loro categoria o classe

Esempi più blasonati sono sicuramente la YOLO (You Only Look Once), la Faster R-CNN (utilizzata in questo lavoro) e la SSD (Single Shot MultiBox Detector).

Capitolo 3

Revisone della letteratura

Ad oggi, sono note e diverse le applicazioni che fanno detection di neoplasie della pelle, sviluppate sia da prestigiose università ma anche da privati o semplici appassionati del settore informatico e medico. In questa sezione verranno illustrati i metodi e le tecniche allo stato dell'arte relative allo sviluppo di un progetto di questo tipo, portando anche esempi di altri applicativi simili, ponendo l'attenzione sulle differenze tra questi ed il presente lavoro. La query di ricerca utilizzata, fondamentalmente, verifica la presenza o meno delle parole "skin", "cancer" e "detection" all'interno del titolo dei lavori. Una volta aperti poi, si procede alla lettura prima dell'abstract e, se ritenuti interessanti, si prosegue con lo studio e l'analisi della pipeline di esecuzione, confrontandola con quella precedentemente prevista per questo lavoro. Una volta fatto questo, sono stati scelti dieci lavori concernenti la creazione di un sistema per la skin cancer detection ed i tre più rilevanti sono stati riportati di seguito.

3.1 Identificazione melanoma

Il primo lavoro preso in esame, S. Jain et al.[1], è uno studio proposto dall'istituto MAEER's MIT, Pune, India.
La metodologia proposta per la rilevazione del melanoma prevede vari passaggi. Inizialmente, l'immagine di una lesione cutanea viene pre-processata per migliorarne la qualità (date le numerose variabili in gioco quali condizioni di luce, qualità sensore fotocamera etc), ad esempio mediante la regolazione del contrasto e della luminosità. Successivamente, viene applicata la segmen-

tazione delle immagini e l'individuazione dei bordi attraverso il thresholding automatico. L'immagine segmentata viene poi sottoposta all'estrazione delle caratteristiche, tra cui caratteristiche geometriche e le regole ABCD[13] della dermoscopia. Infine, le caratteristiche estratte vengono utilizzate per classificare le lesioni come melanoma, pelle normale o neoformazione.

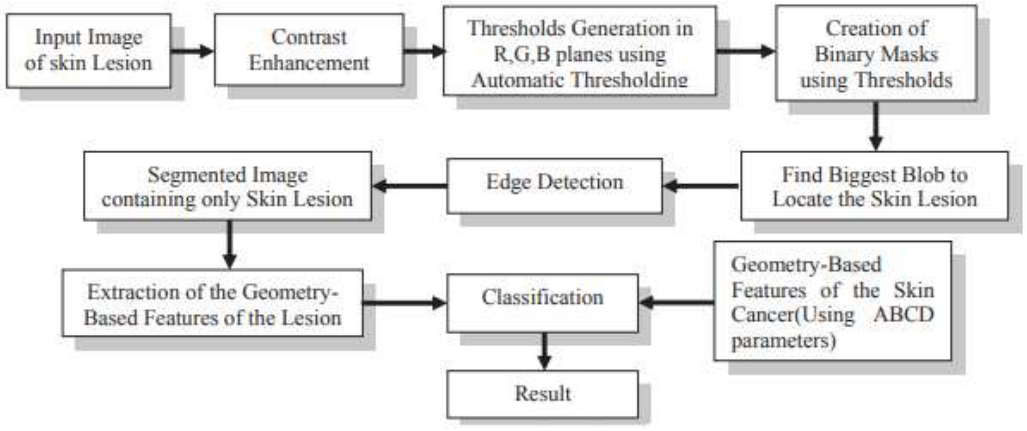

Figura 3.1: S. Jain et al.[1] - Processo

Nello specifico, per ogni macro fase:

- Image Pre-processing: ridimensionamento dell'immagine e regolazione del contrasto e della luminosità. Questo viene fatto per compensare l'illuminazione non uniforme nell'immagine e migliorarne la qualità generale, utilizzando tecniche come la gamma correction.
- Image Segmentation: realizzato utilizzando prima l'automatic thresholding di Otsu[14] ai piani R, G e B, poi mescolando le maschere di questi per ottenere la maschera finale della lesione. Dopodiché, vengono applicati ulteriori tecniche di edge detection.
- Feature Extraction: estrazione di features geometriche come area, perimetro, diametro e irregolarità della lesione.
- Classification: basata sulle features precedentemente estratte.

I risultati dimostrano che il sistema proposto è efficace nella segmentazione delle immagini delle lesioni cutanee, consentendo una diagnosi accurata.

Purtroppo non vengono forniti dati e metriche a supporto della tesi seppur il risultato finale sia l'identificazione del tumore o della normale pelle.
In ogni caso, molto interessante risulta essere l'intero processo, da cui poi sono state tratte inforamzioni utili per la realizzione di questo lavoro.

3.2 Identificazione multiclasse

Il secondo lavoro analizzato, M. K. Monika et al.[2], è stato realizzato da una collaborazione tra Gokaraju Rangaraju Institute of Engineering and Technology, Aditya Institute of Technology and Management e Vignan's Institute of Information Technology, tutti istituti che hanno sede in India.
Anche questo lavoro presenta una pipeline di task simile alla precedente, di seguito nella foto:

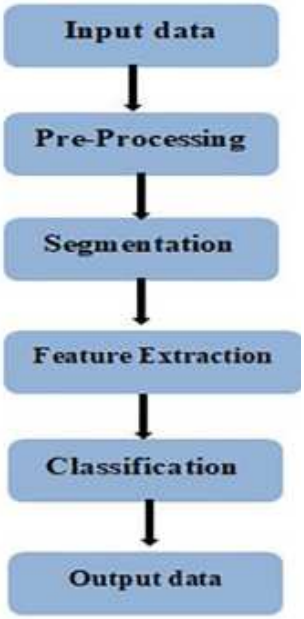

Figura 3.2: M. K. Monika et al.[2] - Processo

Di fatti, analizzando i diversi studi proposti concernenti questi sistemi che presentano un fine comune, ovvero quello di fare detection di neoplasie, non potrebbe essere altrimenti. In ogni caso, ognuno di essi presenta comunque più o meno piccole differenze che li caratterizzano. Nello specifico,

questo lavoro apporta alcune milgiorie a livello di image pre-processing ed implementa tecniche come le seguenti:

- Input data: immagini di training derivanti dal dataset della challenge proposta da ISIC nel 2019.
- Pre-processing: basato prima su grayscale conversion e poi image enhancement e noise removal attraverso filtro gaussiano e mediano. Dopodiché, applicazione del Dull Razor[15] al fine di rimuovere i peli all'interno della foto.
- Segmentation: mediante color-based K-means.
- Feature Extraction: applicazione della regola ABCD e GLCM[16] al fine di estrarre caratteristiche della lesione come simmetria, bordi, colore e diametro.
- Classification: attraverso il Multiclass Support Vector Machine[17], variante multiclasse del più conosciuto SVM (Support Vector Machine) che lavora sul concetto dei piani decisionali classificando gli oggetti.

Risultato di questo processo è un'accuratezza e precisione che si attestano rispettivamente attorno al 96,25% e 96,32%, su un classificazione finale basata su otto classi (per altro le stesse proposte nel primo prototipo dal sottoscritto).

3.3 Inferenza su dispositivo mobile

Distaccandosi un po' dai processi convenzionali di classificazione delle neoplasie, questo lavoro, X. Dai et al.[3], proposto dalle Università di North Carolina, Cardiff, New York, Washington e Tokyo nell'ambito medico ed informatico, vuole trasferire il carico di inferenza direttamente su un dispositivo mobile cercando di favorirne gestione e facilità d'uso.
Infatti l'idea alla base che è stata portata avanti nello sviluppo del lavoro è quella di effettuare la parte di training delle reti neurali su di un hardware prestante, come quello di un pc di alta gamma, e la parte di inferenza, tendenzialmente più leggera, su un dispositivo mobile in modo tale da ottenere in tempi stretti il risultato della predizione e sempre in locale su quest'ultimo dipositivo. Questo, tra l'altro, rappresenta lo stesso ragionamento seguito nel

lavoro di questa tesi.
Di seguito lo schema riassuntivo del processo ideato:

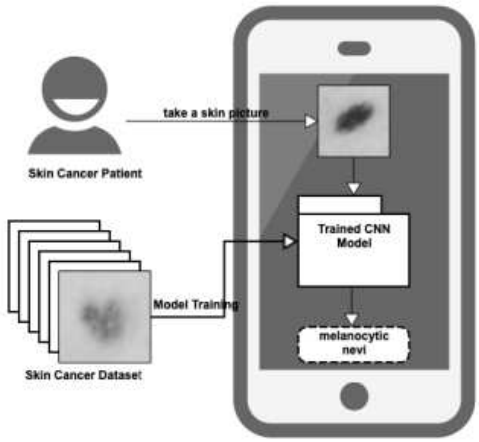

Figura 3.3: X. Dai et al.[3] - Processo

Fondalmente, il lavoro proposto termina qui. Non sono state previste particolare funzioni di image pre-processing o ricerca delle features, se non quelle già effettuate in automatico dalla rete neurale. Quest'ultima, nello specifico, risulta essere un CNN, ovvero Convolutional Neural Network, ancora più nello specifico una AlexNet[18], addestrata su un set di dati costituito da sette diverse categorie di tumori, proveniente dalla stessa fonte della ISIC Challenge menzionata precedentemente. Il tutto con risultati pari ad un Loss di 0.71 ed una Accuracy del 75.2%.

Capitolo 4

Metodologia

Ottimo, ora che è stato descritto il contesto in cui è stata sviluppata l'idea di questa tesi e ne sono state descritte anche le fonti con cui quest'ultima si deve confrontare, è bene entrare nel vivo dell'implementazione, passando prima per le metodologie e tecniche utilizzate.

Figura 4.1: Processo di sviluppo di questa tesi

Di seguito, una dettagliata descrizione del processo in ogni sua fase.

4.1 Input

Per adempiere all'obiettivo finale del lavoro, ovvero quello di classificare l'entità della lesione della pelle, si ha bisogno, ovviamente, di una foto dalla quale partire, anche a fronte della rete neurale che si è voluto adoperare.
In questo caso, perciò, si parte proprio dall'immagine acquisita. Essendo l'applicativo sviluppato pensato per operare su più piattaforme, previa conversione del codice nel formato di applicazione supportato dal sistema operativo dell'utente finale, occorre porsi come da buona norma nel cosiddetto worst case. Infatti bisogna tenere in conto i diversi tipi di camere esistenti sul mercato, dalle webcam per i pc a quelle dei telefoni, più o meno precise. Inoltre, occorre tenere in mente anche le condizioni attraverso cui vengono

effettuate le foto: luminosità, contrasto, offuscamento, ombre e chi più ne ha più ne metta. Non solo. Importante è anche la dimensione della stessa foto, e quindi anche della sua risoluzione. Non tutti i sensori delle camere scattano le foto alla stessa maniera: ci sono sensori che fanno foto in 4k, chi 2k, chi 1080p, chi sceglie un formato da 1:1, chi 3:4 e, purtroppo anche qui, chi più ne ha più ne metta.
Bene, ora che è chiaro come l'input dell'intero processo può rappresentare il primo complesso ostacolo allo sviluppo dello stesso, occorre trovare una soluzione universale per tutti i tipi di camera. Spoiler: purtroppo non esiste. O meglio...si possono trovare dei meccanismi per ovviare a questo problema, come poi è stato fatto in questo lavoro.
Infatti, una soluzione banale quanto immediata è stata quella di settare la dimensione dell'immagine che si sta acquisendo a 640x480, formato che è anche uno dei più grandi accolti dal pacchetto python responsabile della gestione della foto. Perché sì, altro spoiler: l'intero processo è stato realizzato grazie ad un pacchetto di gestione per le applicazioni, denominato Kivy[19], il quale gira mediante sulla base del linguaggio di programmazione ad oggi più diffuso, ovvero Python. Ma questa parte più tecnica viene discussa in maniera dettagliata nel seguente capitolo di implementazione.
Dicevamo, si è posto un limite alla dimensione di acquisizione dell'immagine, visibile ovviamente nell'istante in cui si sta per premere il tasto "Capture" per scattare la foto, in modo tale da evitare di fare conversioni strane prima di passare alla fase successiva di detection e favorire la classificazione proposta a fine pipeline da parte della rete neurale. In realtà, a voler essere precisi, una volta acquisita l'immagine, questa viene ridimensionata a 600x450, esatto formato accettato dalla rete terminale.
In ultima battuta, per favorire un'esperienza utente migliore, è stato implementato alla fine dell'intero processo, quindi una volta ottenuto il risultato dell'entità della lesione, un bottone che permette il riavvio dell'intera pipeline di esecuzione, tornando così di fatto alla situazione iniziale in cui si presenta la possibilità di scattare una foto. Questo per far sì che l'utente possa poi effettuare una foto migliore rispetto alla precedente, o semplicemente ottenere una doppia verifica. Inoltre, anche un sistema di regolazione luminosità, contrasto e blur è stato implementato nella fase di Image Enhancement al fine di migliorare la qualità dell'immagine acquisita, senza doverne per forza scattare un'altra.

4.2 Identificazione

Una volta ottenuta l'immagine in input, si è deciso di procedere con una fase di detection. Qui, infatti, si torna un po' alla questione precedentemente analizzata, ovvero quella in cui si deve sempre pensare al worst case. Per questo, si è deciso di prevedere all'interno del flow di esecuzione anche un meccanismo di detection che possa identificare al meglio il presunto tumore e ritagliare l'immagine attorno ad esso, al fine di escludere parti superflue e potenzialmente non informative per la classificazione finale.
Detto questo, com'è stata possibile la realizzazione di questa fase? Beh, un task di detection, per essere applicato nella maniera corretta, cosa che presuppone ovviamente anche l'introduzione di un modello di rete neurale, ha bisogno di un dataset su cui addestrarsi. Questo è stato prelevato dalla International Skin Imaging Collaboration (ISIC) challenge del 2018[6] all'interno della quale si possono trovare due cartelle ed un file csv:

- Una prima cartella riguarda il dataset di immagini originali, ovvero quelle effettivamente ottenute da una fotocamera o al microscopio, per un totale di 10.015 files
- Una seconda cartella invece include le esatte maschere delle immagini sopra citate (per ovvie ragioni, anche queste pari a 10.015 files)
- Il file CSV descrive la classe di appartenenza di ciascuna immagine ad una delle seguenti categorie di neoplasie:
  - Melanoma (mel)
  - Melanocytic nevus (nv)
  - Basal cell carcinoma (bcc)
  - Actinic keratosis (akiec)
  - Benign keratosis (bkl)
  - Dermatofibroma (df)
  - Vascular lesion (vasc)

Per questa fase, è stata necessaria solo la prima cartella dato che le maschere ancora non erano necessarie (verranno poi utilizzate per un test sull'implementazione di un task di segmentation, descritto sia nella fase di

Feature Extraction che dei risultati finali). Perciò, delle oltre dieci mila immagini a disposizione, ne sono state prelevate soltanto duecento da includere all'interno dell'addestramento.
A questo punto, è stato necessario utilizzare un framework di assegnazione label per le immagini al fine di ottenere un dataset, composto da train e test come ogni addestramento prevede, in cui venivano descritte le coordinate dei bounding boxes identificanti le diverse lesioni. Quest'operazione è stata effettuata a mano dal momento che non si disponeva di un dataset con questi dati già presenti ed attraverso l'ausilio del framework MakeSense.AI[20] si è riusciti ad assegnare alle duecento immagini una cornice contenente al suo interno l'intera lesione. Quest'ultima poi, viene trascritta su di un file del formato XML attraverso le sue coordinate, per ogni file.
Una volta ottenuto, quindi, il dataset di train e di test con le relative assegnazioni dei bounding boxes, è stato possibile aviare il vero e proprio addestramento, facendo fine tuning su di un modello di Faster R-CNN ResNet-50 FPN[21], con i seguenti risultati:

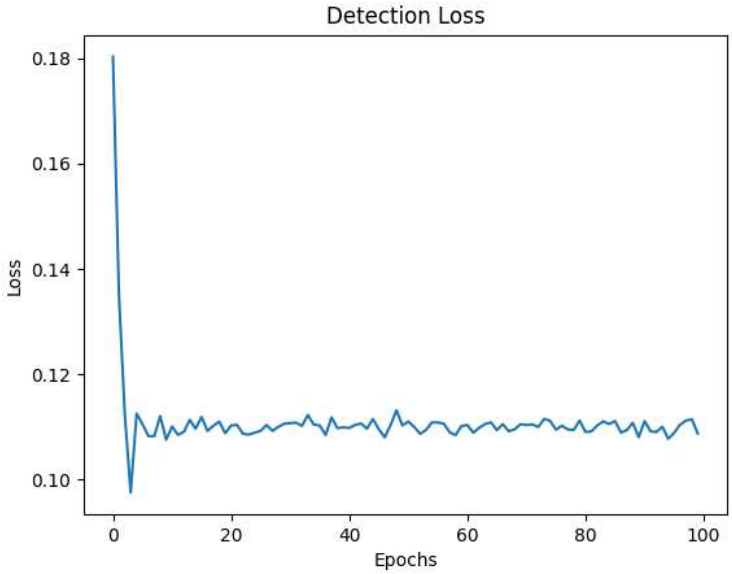

Figura 4.2: Loss task Detection

Come si può notare dal grafico, l'addestramento ha portato ad una loss che si attesta intorno allo 0.11, che comunque risulta essere un buon risultato,

considerando sia il tipo di dateset (purtroppo l'oggetto da indentificare non è dei migliori considerando la sovente somiglianza tra lesione e resto della pelle) e la sua dimensione (per ottenere risultati ancora migliori occorrono dataset più grandi).
Di fatti, nelle immagini di test mostrate nei capitoli successivi, appare comunque chiaro come la rete abbia appreso in modo abbastanza efficiente l'dentificaione di questi oggetti, nonostante il rumore presente all'interno di ogni immagine dato da peli e altri segni artificiali.
Menzione particolare va fatta per il modello di rete neurale. Infatti, quando si parla di Faster R-CNN ResNet-50 FPN, si sta descrivendo un modello di rilevamento degli oggetti che utilizza l'architettura Faster R-CNN, versione più rapida ed efficace delle sorelle maggiori e semplici CNN, basata sulla rete ResNet-50, altra famosa rete largamente utilizzata e potenziata con l'approccio FPN (Feature Pyramid Network) per la rilevazione efficace di oggetti in immagini di diverse dimensioni. Questi modelli sono ampiamente utilizzati in applicazioni di visione artificiale per compiti come il rilevamento degli oggetti in tempo reale, il tracciamento degli oggetti e molto altro.
Infine, come anticipato, una volta data in pasto alla rete neurale, si riottiene l'immagine di partenza ritagliata al bounding box identificato, perciò, tendenzialmente, un'immagine sottodimensionata. Quest'ultima poi, durante la fase di Feature Extraction, verrà nuovamente ridimensionata alla grandezza originaria per affrontare l'ultimo step di classificazione.

4.3 Regolazione immagine

A questo punto, ci si trova in una fase in cui si ha a disposizione un'immagine, più o meno grande, che vede come soggetto in primo piano la lesione sospetta. Infatti, grazie alla precedente fase di detection in cui si è riuscito a identificare l'intera porzione di tumore, centrarla e rimuovere il rumore attorno, si ha la capacità di lavorare direttamente su quest'ultima. Come? Beh, andando a migliorarne la qualità al fine di renderla il più simile alla realtà. Questo perché, come anticipato nelle precedenti fasi, ogni foto può essere ottenuta a discrezione delle capacità dell'utente finale e ciò comporta magari elevati contrasti, zone in ombra, scarsa luminosità etc.
Per ovviare a questo problema, quindi, si passa per una fase di enhancement dell'immagine, la quale prevede tre regolazioni:

- Deoffuscamento

- Equalizzazione dell'istogramma dei colori
- Regolazione del contrasto

Infatti, mediante gli appositi slider visibili a schermo, è possibile modificare i parametri dei metodi in python che vanno ad apportare le modifiche all'immagine e vedere su quest'ultima il risultato aggiornato non appena si rilascia lo slider.
In realtà, a livello implementativo esistono diversi modi per ottenere lo stesso risultato. In questo caso, per questioni sia di tempo che pesantezza del codice, si è deciso di adottare quelli più leggeri e che permettessero in maniera più rapida l'ottenimento del risultato a schermo.
Per questo, è stato utilizzato nella quasi totalità dei casi, metodi predefiniti dalla libreria python opencv[22], largamente utilizzata per effettuare operazioni circa le immagini.
Purtroppo, non è stato possibile applicare tecniche di machine learning al fine di ottenere un risultato autonomo e di buon livello dal momento che molte delle soluzioni testate non rientravano nelle tempistiche considerate accettabili per l'esecuzione di questo task. In ogni caso, l'implementazione è molto intuitiva e l'utente viene guidato nella regolazione di questi parametri. La scelta e l'implementazione degli stessi verrà descritta più ampiamente nel prossimo capitolo.

4.4 Estrazione delle features

Uno dei principali metodi con cui vengono identificate le neoplasie è la cosiddetta regola ABCD. Questa prevede l'analisi di determinate caratteristiche della lesione, le quali vengono poi confrontate con dei valori nominali per determinarne la natura malevola o meno. Per cui, ogni lettera si associa ad una caratteristica, nell'ordine:

- A -> Asimmetria
- B -> Bordi
- C -> Colore
- D -> Diametro

Infatti, tendenzialmente, si approfondisce l'analisi di un tumore sospetto della pelle, con magari un esame istologico, nel momento in cui una di queste proprietà risulta particolarmente fuori dai limiti ammessi. Per esempio: nel caso in cui una lesione abbia diametro superiore ai 6mm; oppure quest'ultima presenti dei bordi frastagliati; oppure, ancora, sia evidentemente asimmetrica; ed infine, se presenta una colorazione disomogenea. Ovviamente non devono essere tutte verificate contemporaneamente ma, piuttosto, è sufficiente uno dei casi sopra descritti per far scattare l'"allarme".

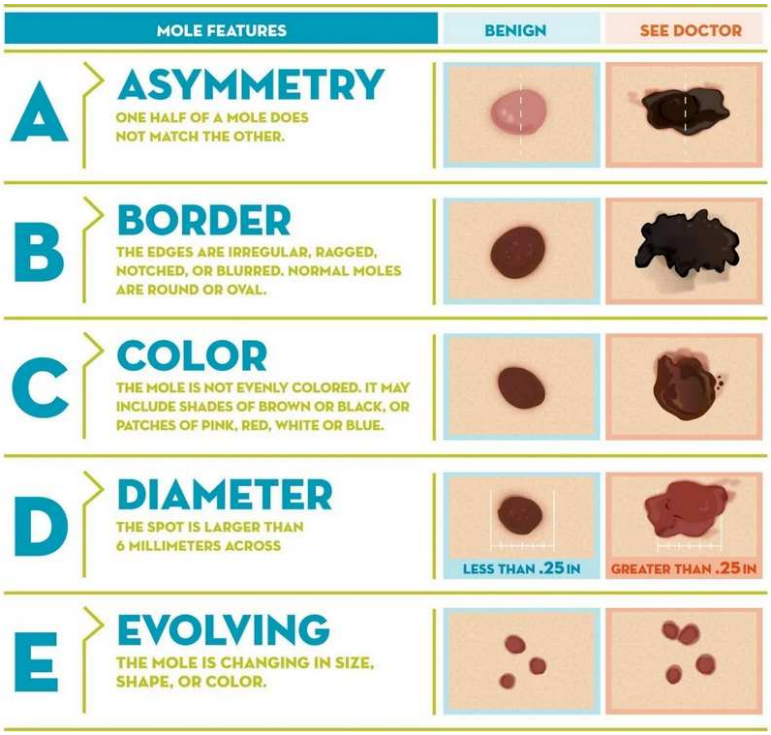

Figura 4.3: Regola ABCD[4]

In realtà, alla lista andrebbe aggiunta anche un'ultima lettera, la E. infatti, la regola vera e propria utilizzata nei metodi di medicina allo stato dell'arte, comprende anche la parte evolutiva (E appunto) della lesione: un neo che in poco tempo raddoppia le proprie dimensioni molto probabilmente non è un semplice neo. Purtroppo, però, in questa sede non è stato possibile implementare un sistema efficiente in grado di immagazzinare foto relative alla stessa lesione per poi analizzarle nel tempo dal momento che avrebbe

richiesto un ulteriore sforzo anche in termini di permessi all'utente. Inoltre, si sarebbe complicata notevolmente l'intera pipeline di esecuzione, dovendo prevedere anche ulteriori meccanismi di interazione con l'utente finale, che in questi casi è preferibile ridurre al minimo. Per ultimo, ma non per importanza, l'implementazione di un sistema del genere avrebbe comportato la quasi totale sostituzione della figura del professionista, ovvero il dermatologo, in quanto in questa maniera non vi sarebbe più necessità di una visita. E questo non è l'obiettivo di tale lavoro.
In ogni caso, è importante sottolineare come anche qui, attraverso l'ausilio di python, sia stato possibile implementare tecniche di machine learning capaci di analizzare tali caratteristiche direttamente sulla foto acquisita. Per prima cosa, è stato necessario segmentare l'immagine, attraverso un modello di Deeplabv3_Resnet50[23] pretrainato, ancora una volta CNN basata sulla già discussa Resnet50, discretamente utilizzata per tasks di segmentazione semantica. Come già anticipato, il dataset complessivo constava anche di una cartella costituita dalle maschere, quindi immagini binarie, in bianco e nero, delle immagini delle neoplasie originali. Utilizzando questo dataset e dandolo in pasto alla rete sopra indicata, che lato suo vanta come caratteristiche principali una risoluzione elevata e l'utilizzo delle Dilated Convolutions per espandere il campo visivo senza dover ridurre la risoluzione dell'immagine, si è stati in grado di ottenere una loss con dei minimi locali di poco superiori allo 0,1 ed una accuratezza superiore al 90%, risultato assai soddisfacente.

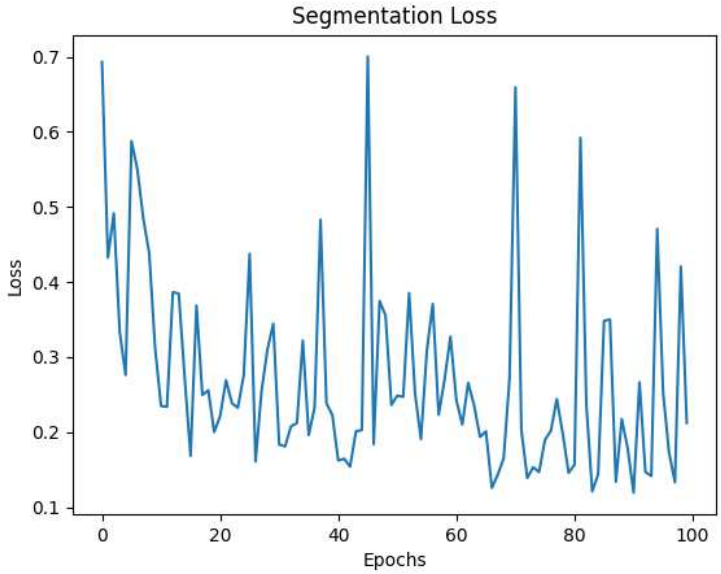

Figura 4.4: Loss task Segmentation

A questo punto, ottenuta l'immagine segmentata dell'immagina di input, si è passato al calcolo vero e proprio delle proprietà ABCD, fondamentalmente attraverso il già citato pacchetto opencv-python. Grazie a quest'ultimo, infatti, è stato possibile calcolare l'asimmetria della lesione semplicemente circoscrivendo il più grande cerchio su di essa e calcolando la percentuale di scostamento dal raggio del cerchio. Se quest'ultima è inferiore al 10%, allora scatta l'"allarme". Stessa cosa vale per i bordi, dal momento che più la lesione risulta avere dei bordi omogenei e più, tendenzialmente, è simmetrica (l'esempio più immediato è un classico neo benigno rotondeggiante). Per quanto riguarda il colore, invece, sono state calcolate le principali tinte di colore presenti all'intero dell'immagine (che una volta segmentata contiene al proprio interno esclusivamente la lesione) e ne sono stati identificati quelli più tendenti al marrone. Se i principali colori estratti dall'analisi risultano essere vicini a questi, non ci sono problemi, altrimenti scatta di nuovo l'"allarme". Infine, per il diametro è stato sufficiente raddoppiare il raggio del cerchio precedentemente circoscritto alla lesione ed ottenere il suo valore. Se quest'ultimo risultasse essere maggiore di 6mm, indovinate... bene, scatterà l'"allarme".
Per concludere, il controllo di queste proprietà è stato messo in AND con la

classificazione definita a fine pipeline e, tra di loro, in OR in modo tale da avere una maggiore sicurezza sull'effettiva entità del tumore facendo sì che la classificazione debba restituire una delle classi diverse da quella del neo E almeno una delle proprietà ABCD facciano scattare l'allarme.

4.5 Classificazione

In ultima battuta, l'intera pipeline di esecuzione si conclude con la fase di classificazione. Questa però non è stata pensata in ottica di predizione esatta della classe di appartenenza (una delle sette del dataset precedentemente analizzato) o, meglio, il task è esattamente questo ma il risultato finale cambia.
Infatti, al fine di fornire solamente un supporto all'utente finale senza innescare in quest'ultimo ansie e angosce preventive, si è deciso di non mostrare a schermo direttamente la classe predetta, bensì solamente un messaggio con la scritta:

- "No problem" in verde oppure
- "Check" in rosso, a simboleggiare l'effettiva necessità di un visita dal dermatologo

Nel primo caso, non sono state rilevate anomalie a livello di proprietà ACDB, come visto nel capitolo precedente, o la lesione è stata classificata semplicemente come neo benigno (tutto questo ovviamente all'oscuro dell'utente finale). Nel secondo caso, invece, una delle proprietà ABCD è stata per così dire violata e la classe di appartenenza della lesione risulta essere diversa da quella del neo.
Anche in questo caso, per adempiere al corretto svolgimento del task di classificazione, è stato necessario implementare una rete neurale. Nello specifico, è stata utilizzata una Resnet pretrainata, questa volta Resnet18[24], CNN che, rispetto a quelle viste nei precedenti capitoli, presenta 18 strati di profondità anziché 50, mantenendo comunque le ottime prestazioni e caratteristiche architetturali. Necessaria è stata una fase di fine tuning del modello sul dataset a disposizione. I risultati migliori sono stati ottenuti con un'accuracy dell'85% e loss intorno allo 0.001.

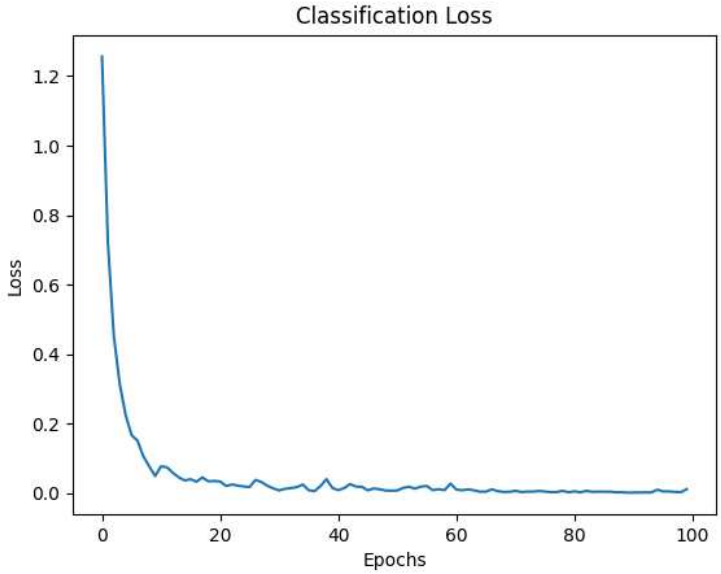

Figura 4.5: Loss task Classification

Infine, a conclusione dell'intero processo e, per certi versi, come punto di riavvio, è stato inserito un bottone che prevede il ripristino della schermata iniziale, permettendo così all'utente di scansionare un ulteriore lesione sospetta.

Capitolo 5

Implementazione

Bene, ora che è stata descritta in maniera dettagliata la metodologia seguita per la realizzazione di questo lavoro, è bene passare alla parte implementativa, quindi dataset, codice, linguaggio di programmazione, IDE, strumenti e chi più ne ha più ne metta.

5.1 Dataset

Innanzitutto, occorre partire da ciò che ha permesso la realizzazione di tale progetto, ovvero il dataset. Purtroppo, non esistono molti dataset a disposizione online per lo svolgimento di un task di questo genere, probabilmente una collaborazione tra l'università e qualche ente medico che lavora in ambito dermatologico avrebbe giovato ad una più granulare implementazione. In ogni caso, causa tempistiche serrate, si è deciso proseguire con ciò di più idoneo presente in internet.
Di fatti, ad un prima ricerca balza subito all'occhio la presenza di diversi dataset che descrivono diverse tipologie di lesioni della pelle, per la maggior parte derivanti dall'ISIC Challenge (International Skin Imaging Collaboration Challenge), ovvero, come già anticipato, una competizione e un'organizzazione che si concentra sulla promozione della ricerca nell'ambito della diagnosi e della classificazione delle lesioni cutanee, in particolare dei melanomi e dei tumori della pelle. L'obiettivo principale, infatti, di tale challenge è sviluppare algoritmi di intelligenza artificiale in grado di analizzare immagini dermoscopiche e fotografie delle lesioni cutanee al fine di identificare segni di cancro della pelle e fornire una valutazione precisa del rischio.

Se si naviga un po' all'interno del loro sito, si può notare come diverse siano le edizioni di queste sfide e, di conseguenza, diversi gli obiettivi ed i dataset. Alcuni prevedono la "semplice" discriminazione tra neo e melanoma, con magari un dataset costituito da migliaia di foto di una e l'altra categoria (spesso in realtà queste sono nettamente sbilanciate per cui si ricorre a tecniche come la data augmentation), mentre altri la discriminazione tra più categorie (sette o otto, come si è potuto già constatare), altre ancora la sola segmentazione fornendo immagini originali e maschere. Il tutto accompagnato sovente da un csv che associa nomi delle immagini alla classe di appartenenza.
In questo caso, si è optato per il dataset derivante dalla challenge del 2018 che prevedeva la discriminazione tra sette categorie di neoplasie:

- mel
- nv
- bcc
- akiec
- bkl
- df
- vasc

Come già descritto nel capitolo della detection, questo constava di due cartelle ed un file csv:

- Cartella con le immagini originali, 10.015 per l'esattezza
- Cartella con le maschere, in numero pari a quello delle immagini originali
- Csv di associazione immagine-categoria

Di seguito, degli esempi da entrambe le cartelle.

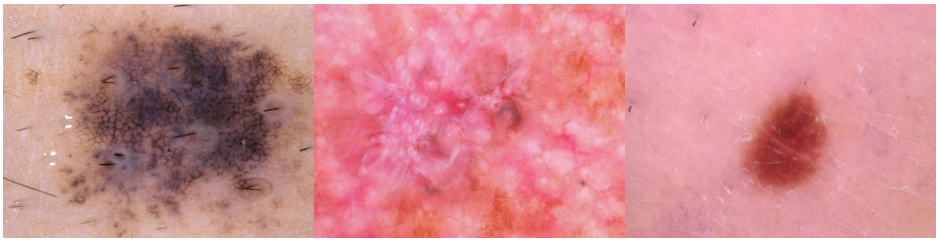

Figura 5.1: Esempio delle immagini originali

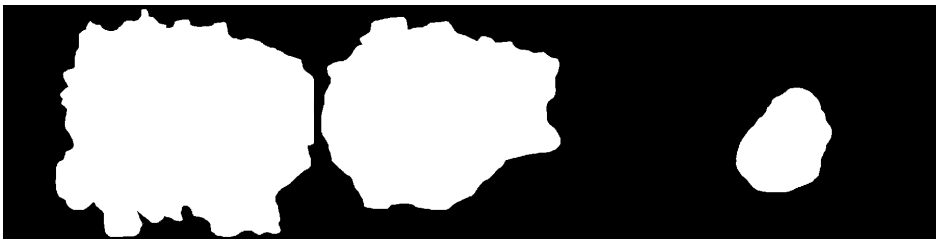

Figura 5.2: Esempio delle maschere

Questi elementi sono stati largamente studiati e analizzati prima dell'avvio di ogni fase, ponendo particolare attenzione, ovviamente, a quelle in cui veniva utilizzati dei modelli di reti neurali. Qui, infatti, sono state presi, modificati ed elaborati i file al fine di ottenere il miglior risultato in fase di addestramento rete. Per esempio: nella fase di detection, si è deciso di non sfruttare tutte e 10mila le immagini, bensì un suo sottoinsieme, circa 200, al fine di ottenere buoni risultati in output senza incorrere in problemi come l'overfitting; nella fase di classificazione, sono state suddivise tutte le immagini, sfruttando il file csv fornito, nelle proprie classi di appartenenza, equiparato il numero degli esempi all'interno di ognuno (aumentato quelle meno dense come la Vascular Lesione e diminuito i Melanocytic Nevus per evitare overfitting) e, poi, effettuato il training vero e proprio; infine, nella fase di segmentazione sono state utilizzate tutte e 10.015 le immagini al fine di ottenere risultati più accurati e precisi, data la maggiore necessità da parte della rete di soffermarsi su ogni pixel per apprendere.
Va da sé, che in letteratura esisteranno infiniti altri tumori della pelle ma è vero anche che riuscire ad ottenere un dataset dettagliato e preciso per ognuno di questi non è semplice, motivo per cui si è proseguito con ciò che era disponibile in internet al tempo della definizione del progetto.

5.2 Strumenti

5.2.1 IDE e linguaggio di programmazione

Diversamente da quanto accaduto per il primo prototipo, in questo lavoro si sono utilizzati degli strumenti per così dire standard. Laddove, nel primo caso, si necessitava sia dell'hardware descritto che dell'accesso diretto a quest'ultimo per la scrittura di codice, il testing, collegamenti e lo sviluppo in toto dell'apparecchio, per il seguente lavoro è stato sufficiente un IDE ed il linguaggio di programmazione.
Infatti, nello specifico, si è utilizzato in primis Visual Studio Code[25] per lo sviluppo dell'applicativo in sé. Quest'ultimo risulta essere un popolare ambiente di sviluppo integrato (IDE) gratuito ed open source sviluppato da Microsoft. Tra le principali caratteristiche troviamo:

- Multi-Piattaforma, fondamentale nel passaggio da un OS all'altro per il testing di determinate funzioni dell'applicativo (es: si vogliono testare delle features prima in Windows e poi in una qualsiasi distro Linux)
- Estensibilità, aggiungendo funzionalità per linguaggi di programmazione specifici, strumenti di sviluppo e molto altro (git, powershell etc etc)
- Editor semplice, intuitivo e reattivo

Mediante il suo ausilio, è stato possibile disporre di particolare fluidità nella gestione dell'intero progetto, potendo passare da uno script all'altro, da una immagine all'altra o attraverso qualsiasi altro file presente all'interno delle cartelle. Insomma, chi ha frequentato questo corso non può assolutamente non conoscerlo.
Dopodiché, come già anticipato, su di esso è stato fatto girare completamente del codice python. Anche qui, chi ha frequentato questo corso non può assolutamente non conoscerlo o addirittura non averlo utilizzato nemmeno una volta. Giusto per dare un po' di contesto a chi meno esperto, Python è un linguaggio di programmazione interpretato, ad alto livello e versatile che vanta la prima posizione indiscussa all'interno della classifica dei linguaggi maggiormente utilizzata al mondo. Questo perché presenta, tra le tante altre caratteristiche, una sintassi chiara e leggibile, un'ampia libreria standard, multi-piattaforma ed è fondamentalmente open-source.

Con questi soli due elementi ed un performante pc a supporto del sottoscritto, in grado di concludere addestramenti in relativamente brevi tempi, è stato realizzato l'intero progetto, passando ovviamente per tanti diversi script, di prova e non, addestramenti per le reti neurali, tests e lo script finale, ovvero l'applicazione in sé, somma di tutti questi addendi.
Menzione particolare va anche al framework MAKESENSE.AI, già menzionato nel capitolo relativo alla detection, il quale è stato fondamentale per lo sviluppo di quest'ultima dal momento che fornisce uno strumento chiaro e di semplice utilizzo al fine di definire i buonding boxes per i propri dataset.

5.2.2 Script

Bene, ora passiamo alla vera e propria fase implementativa andando ad analizzare il codice sorgente dell'applicativo. Ovviamente, non verranno descritte nello specifico tutte e 400 le linee di codice dal momento che non deve essere un tutorial, piuttosto verranno descritte le componenti principali e peculiari del sistema.
Non si potrebbe non partire dal pacchetto che in questo caso la fa da padrone e che ha reso possibile l'implementazione su diverse piattaforme dell'applicativo, ovvero Kivy. Quest'ultimo è un framework open source per lo sviluppo di applicazioni multi-piattaforma in Python di cui si possono riassumere le caratteristiche nella seguente lista:

- Cross-Platform, infatti è stato progettato per consentire lo sviluppo di applicazioni per diverse piattaforme, compresi dispositivi desktop (Windows, macOS, Linux), dispositivi mobili (Android, iOS) e persino Raspberry Pi, previa corretta conversione del codice
- Open Source
- Scritto interamente in Python e progettato per essere "pythonic", ovvero utilizza una sintassi e uno stile di programmazione Python familiare e leggibile
- Interfacce utente intuitive e programmabili a piacimento mediante l'utilizzo di bottoni, label, slider, photo display etc
- Supporta la grafica 2D e offre funzionalità per il rendering grafico avanzato, oltre che la gestione di eventi

- Licenza MIT, che consente una grande flessibilità e la possibilità di sviluppare anche applicazioni commerciali, con a supporto una documentazione discretamente completa
- Compatibilità con touchscreen

Oltre a questo, fondamentale è stato l'utilizzo delle librerie concernenti l'addestramento delle reti neurali e più in generale la computer vision, quali Pytorch in primis e Detecto in secundis. Purtroppo, proprio quest'ultimo non ha permesso la verifica delle funzionalità direttamente sullo smartphone dal momento che ancora non è stato implementato da parte del supporto di Kivy nella fase di conversione dal mero codice python all'applicativo apk. Ovviamente si tratta solo di questione di tempistiche per le quali nel breve periodo verranno fixati anche questi piccoli bug. Ne sono state comunque verificate le effettive funzionalità dell'applicativo prive delle reti, cosa che quindi è servita esclusivamente come constatazione dell'avviamento e della corretta esecuzione della pipeline. Per concludere, sono stati aggiunti anche pacchetti di contorno come opencv-python, sklearn, numpy e altri per gestire operazioni su immagini, calcoli e strutture dati.
Detto questo, l'applicativo si presenta nella seguente forma:

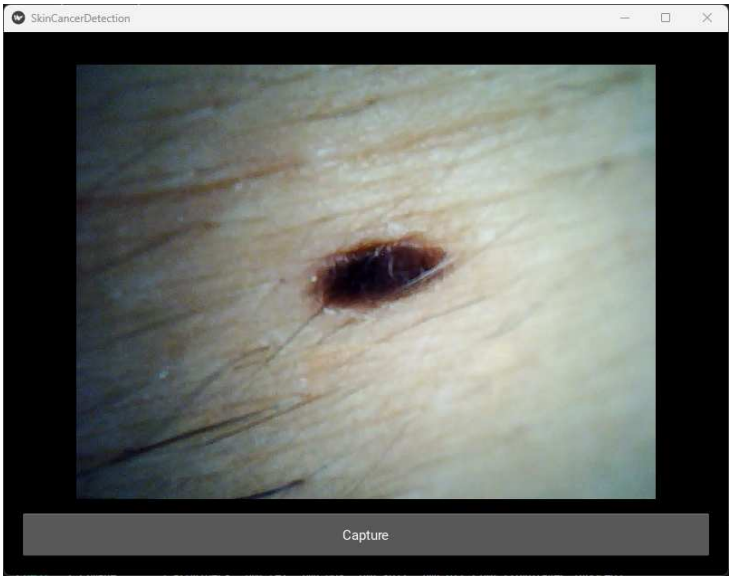

Figura 5.3: Schermata iniziale

Come si può vedere, sono presenti due widget, proprietari di kivy, all'interno della schermata iniziale: camera-immagine e bottone "cattura". Il funzionamento è intuitivo per cui alla pressione del bottone si associa una funzione, o per meglio dire in python, un metodo, il quale salverà l'istantanea della camera, nelle dimensioni prestabilite di 600x450 e nel formato di immagine png.

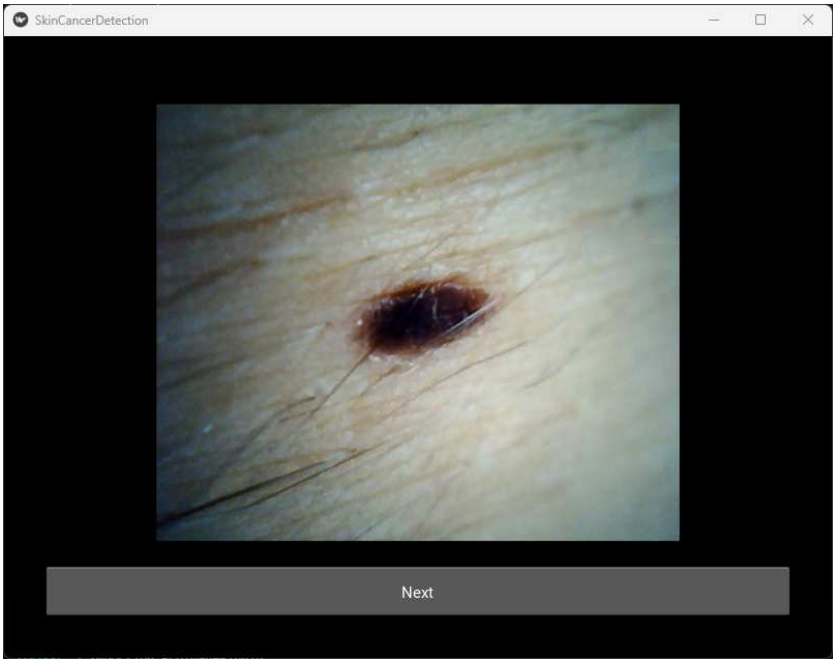

Figura 5.4: Immagine catturata

Quest'ultima azione in realtà innesca simultaneamente, attraverso un diverso metodo, anche il caricamento della schermata successiva, simile nella struttura alla precedente. Infatti, anche qui saranno presenti un'immagine ed un bottone per procedere alla fase successiva. Nel backend, però, accade ciò che è più importante: ovvero, viene scaricato dal Google Drive del sottoscritto, all'interno di una cartella pubblica, il modello di rete neurale dedicato alla detection, al quale verrà data in pasto la foto scattata. Una volta effettuata la detection e ritagliato l'immagine, questa verrà mostrata a schermo, come di seguito.

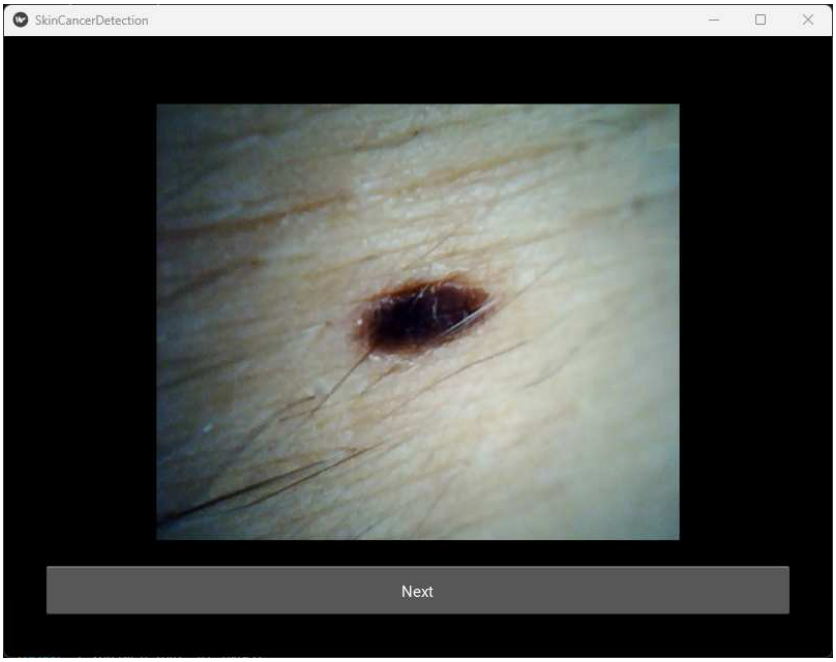

Figura 5.5: Detection

Dopodiché, alla pressione del tasto “Next”, verrà eliminata la CNN e caricata la schermata relativa alla fase di Image Enhancement. Quest'ultima prevede l'esposizione di ben quattro slider, sempre widget proprietari kivy, attraverso i quali si può andare a settare i parametri relativi al deoffuscamento, illuminazione, contrasto e equalizzazione. Nello specifico si va ad agire su:

- Parametri α e β per la regolazione del contrasto
- Parametro α per la regolazione dell'equalizzazione
- Valore del blur

Il tutto è stato pensato in modo tale che ogniqualvolta si rilascia uno slider, la nuova immagine modificata viene caricata nell'apposito widget presente in testa alla schermata e sopra gli slider, al fine di dare un responso immediato all'utente.

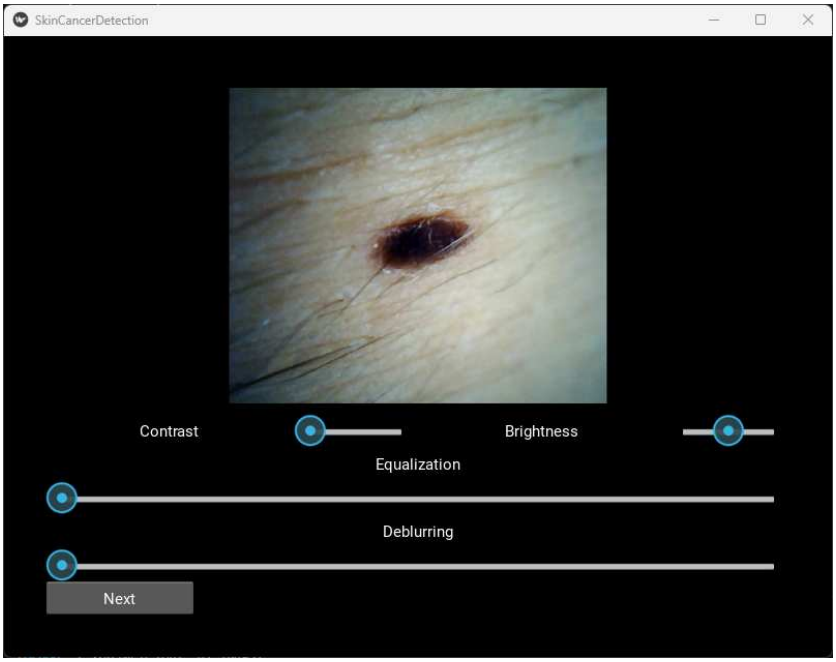

Figura 5.6: Image Enhancement

Alla pressione del tasto “Next”, anche qui, si procede alla fase successiva della Feature Extraction che in realtà a livello grafico non comporta nessuna modifica se non del bottone al quale, in fase di svolgimento dei calcoli, viene modificata la label in “Next (wait..)” per poi tornare alla versione già utilizzata “Next”. Nel backend, si vanno poi ad effettuare tutti i calcoli relativi alle già citate proprietà ABCD, caricando anche qui il modello di rete dedicato alla segmentation dalla stessa cartella precedentemente menzionata.

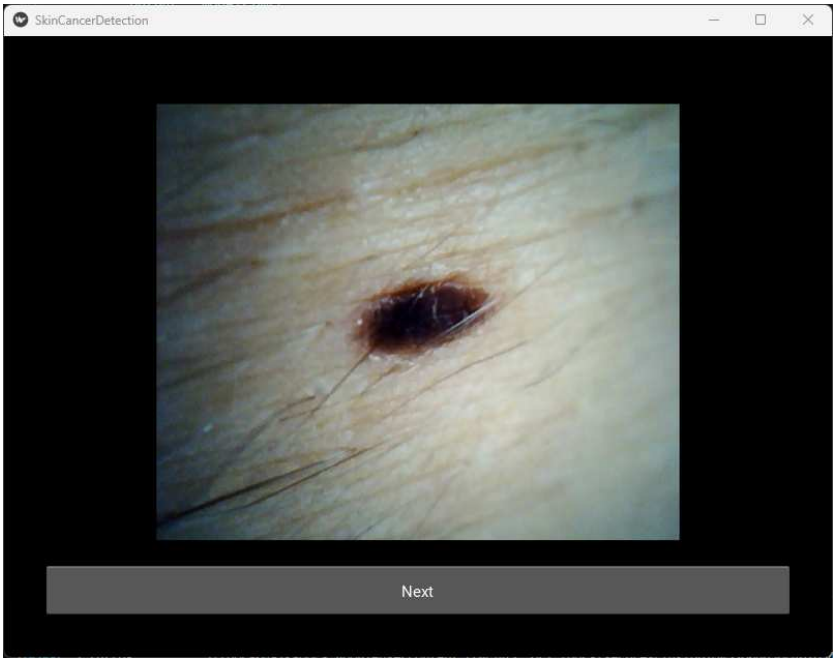

Figura 5.7: Feature Extraction

Una volta terminate le operazioni e clickato sul bottone “Next” si arriva alla schermata finale con il risultato. Come già discusso, non viene menzionata la natura della lesione in modo preciso, ma viene fornito solamente un messaggio con scritto “Check” in rosso in caso di sospetta neoplasia e “No problem!” in verde in caso di nessun problema rilevato. Anche qui, nella fase di esecuzione dei calcoli, viene scaricato il modello di rete adibito alla classificazione che poi viene cancellato in fase terminale del processo.

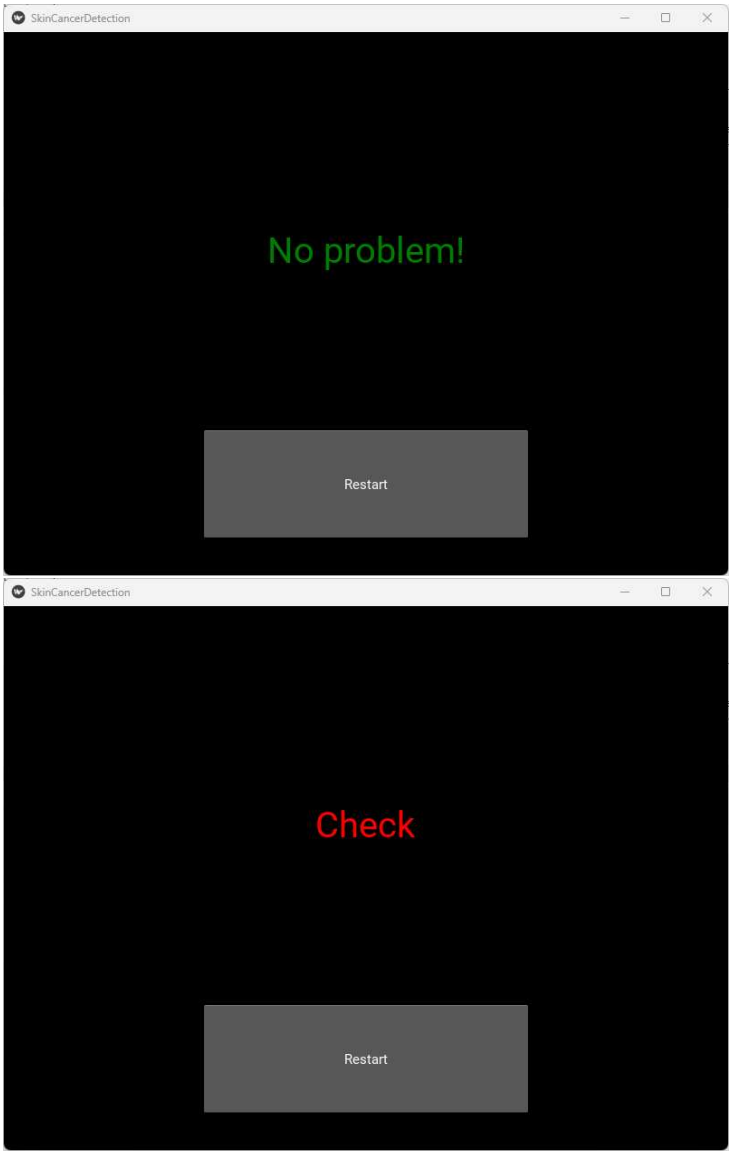

Figura 5.8: Risultati

Infine, in quest'ultima schermata è presente anche un bottone che permette il riavvio dell'applicazione nel caso in cui l'utente voglia ispezionare altre lesioni, previa rimozione di tutti i file lasciati in sospeso e dell'ultima foto elaborata.

Capitolo 6

Risultati e discussione

Tendenzialmente, quando si ha a che fare con dei modelli di rete neurale, al fine di verificare le effettive performances si ricorre alla matrice di confusione, all'interno della quale vengono messi in relazione falsi/veri positivi/negativi, e al calcolo di metriche come l'accuracy, la precision, f1 e così via. Ma quando i modelli sono diversi e su task diversi, diventa difficile fornire un valore globale di performance.
Partendo dal principio, quindi da quello che era l'obiettivo del progetto, si può affermare che questo è stato raggiunto dal momento che attraverso l'applicativo si ha la possibilità di monitorare lesioni sospette. Sarebbe corretto affermare anche che l'accuratezza complessiva è pari a quella finale del task di classificazione dal momento che è il principale addendo nell'equazione che contribuisce a definire l'entità della neoplasia. In realtà, bisognerebbe includere anche il task di segmentazione, componente fondamentale nel calcolo della proprietà ABCD nella fase di Feature Extraction. C'è anche da aggiungere come non è stato possibile effettuare un vero e proprio test dell'applicativo dal momento che, come si può immaginare, non sono in molti a volersi sottoporre come cavia, per cui gli unici test effettuati sono stati quelli in cui si fotografava una stampa di uno dei nei all'interno del dataset di test e se ne verificava l'entità confrontando il risultato fornito con quello presente sul csv.
Perciò, per quanto potuto testare, si può affermare che il prodotto sembri rispettare tutti i requisiti di progetto.
Per quanto riguarda le performances delle reti, si sono ottenuti dei buoni risultati considerando natura e dimensioni del dataset. Detection e segmentation si possono riassumere nei rispettivi grafici di loss in fase di training e

Mean Average Precision (MAP) in fase di test, che descrive quanto l'output predetto dalla rete si allontana in termini di pixel o area di pixel dal ground truth:

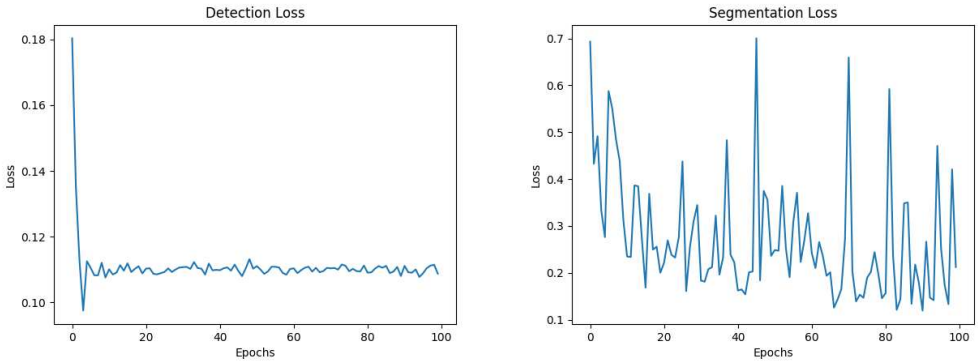

Figura 6.1: Losses

Task	MAP
Detection	89,7%
Segmentation	89,1%

Mentre per la classificazione, sono state analizzate loss in fase di training, matrice di confusione e le principali metriche come accuracy, precision, recall e f1 per la fase di test. Di seguito i risultati.

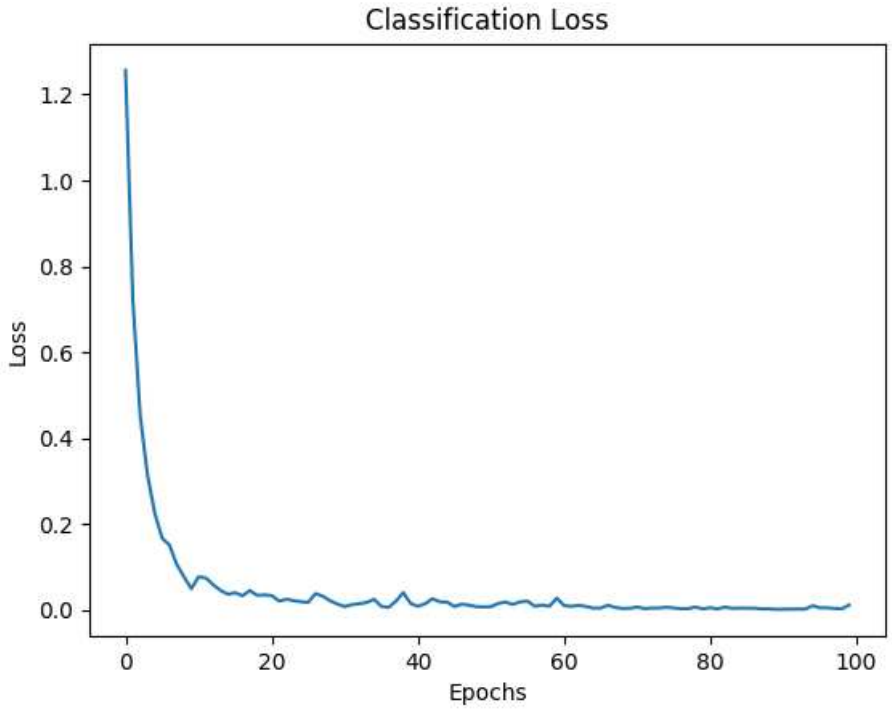

Figura 6.2: Loss

Accuracy	Precision	Recall	F1
84,88%	85,34%	84,88%	84,91%

Esempi diretti a sostegno di questi risultati possono essere le seguenti immagini.

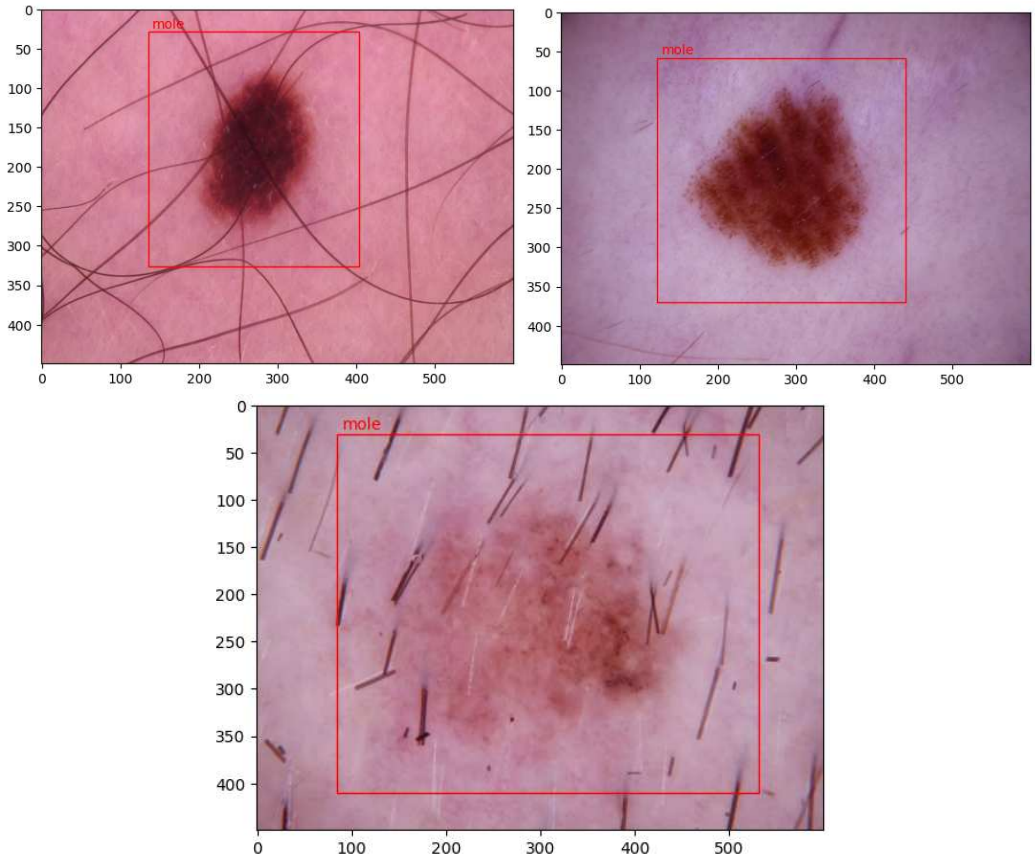

Figura 6.3: Risultati della fase di detection

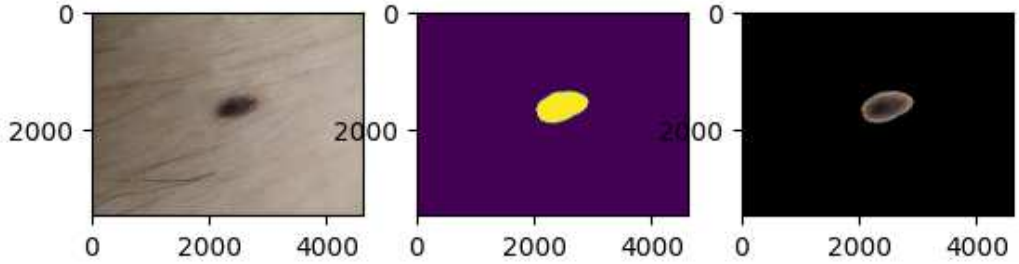

Figura 6.4: Risultato segmentation

Come si può vedere da figura 6.1, i risultati della fase di detection sembrano essere decisamente buoni, nonostante le considerazioni fatte in precedenza

sulle difficoltà intrinseche del dataset (sottile differenza tra neoplasia e pelle). Dalla figura 6.2, invece, si può constatare come anche la fase di segmentazione produca in output un'immagine che descrive quasi puntualmente la lesione interessata. Ciò era immaginabile anche a fronte dell'accuracy e delle altre metriche ottenute in fase di addestramento.
In aggiunta, a tal proposito, va fatta una menzione particolare ad un test effettuato sulla segmentazione nel calcolo dell'accuracy del modello di classificazione. Questo perché, in uno stadio embrionale del progetto, nella fase di definizione della pipeline di esecuzione, era stato pensato di fornire in pasto alla rete terminale di classificazione non le immagini originali, quindi neoplasie più pelle, bensì solo ed esclusivamente la lesione. Questo ovviamente era possibile dando a sua volta in pasto l'immagine al modello di segmentazione prima di approdare alla fase di classificazione. Purtroppo, però, nonostante le rosee aspettative, si sono ottenuti dei risultati leggermente peggiori rispetto al caso dove venivano fornite le immagini originali, motivo per cui si è pensato di abbandonare l'idea. Il riscontro pratico di questo risultato, inoltre, è stato possibile analizzando la rete di classificazione con un pacchetto Pytorch[26] chiamato "Captum"[27], il quale fornisce strumenti avanzati per l'interpretazione dei modelli di deep learning. Attraverso quest'ultimo si è percepito come il modello di classificazione si concentri maggiormente sia sull'interno del neo che sul suo contorno, quindi i bordi. Laddove questi non sono presenti, causa segmentazione della lesione, va da sé che l'accuratezza diminuisce.
Altro test da sottolineare è stato quello per il quale si è analizzato il comportamento della rete di classificazione prima e dopo l'image enhancement. Infatti, attraverso questo insieme di tecniche è stato possibile constatare come, al ricevimento di un'immagine più pulita e definita, la rete tenda a classificare meglio le categorie di tumori, con un incremento di alcuni punti percentuale sulle principali metriche.
Infine, parlando proprio di metriche, rispetto ai casi di studio presi in esame nel secondo capitolo di questo lavoro, si può affermare che l'intero progetto risulta in linea con quelli analizzati a livello di risultati e performances ottenute, seppur rimanendo ovviamente leggermente indietro.

Conclusioni

Con questo lavoro si è voluto affrontare l'importante tematica dell'applicazione dell'apprendimento automatico in campi quali la medicina, e, nello specifico, nell'ambito della diagnosi dei tumori della pelle. Avere a disposizione strumenti che riducono il tasso di errore umano è fondamentale per svolgere pratiche in ambienti in cui un semplice sbaglio può costare la vita di una persona.
Come si è potuto constatare, l'obiettivo inziale del progetto, ovvero la creazione di un sistema multi-piattaforma capace di supportare l'utente finale nel riconoscere la natura nociva o meno delle proprie lesioni cutanee, è stato raggiunto. Ciò è stato possibile grazie alle tecnologie presenti oggigiorno, tra le quali emergono le CNN, fondamentali per questo scopo.
Sicuramente nello svolgimento di questo progetto le sfide presentatesi sono state tante e la risoluzione di queste nella quasi totalità dei casi è stata ben lontano dall'essere semplice.
Primo tra tutti, la ricerca del dataset. Infatti, prerogativa necessaria per un corretto sviluppo del lavoro è avere a disposizione un dataset delle dimensioni adeguate. Purtroppo, come si è potuto constatare, non sempre questa condizione è rispettata e a volte occorre rimediare con quello che si trova in internet. Magari una qualche tipo di collaborazione tra Università e sito medico appropriato e verificato avrebbe giovato ad entrambe le parti, ma purtroppo non vi è stato modo di approfondire la questione. Avere un dataset ben strutturato e di notevoli dimensioni è utile per tante ragioni:

- Aiuta a studiare meglio la materia e comprendere maggiormente il problema
- Migliora le performances di addestramento dei modelli

- Evita, potenzialmente, l'applicazione di tecniche come, per esempio, la data augmentation per livellare le diverse classi, per esempio, in un problema di classificazione

Altro punto fondamentale su cui vale la pena riflettere sono gli stessi modelli. Di fatti, qui sono stati testati complessivamente pochi modelli di reti neurale rispetto alla vasta pletora presente online. Va da sé che, un po' per l'obiettivo che, ricordiamo, non era quello di ottenere il miglior classificatore in assoluto presente sul mercato ma piuttosto adoperare nuove tecniche e verificare l'effettiva realizzazione del progetto in ambito "locale", un po' per il contesto universitario, e non lavorativo, in cui comunque piace mettersi in gioco, e soprattutto date le tempistiche strette, si è deciso di ricorrere a qualcosa che fosse già più o meno noto senza ricercare soluzioni potenzialmente migliori ma ancora in via di sviluppo. In maniera del tutto simile, con la possibilità di applicare algoritmi e far girare codici su macchine "corazzate" riservate a questi scopi, sicuramente si sarebbero potute ottenere prestazioni più elevate a parità di dataset e modelli utilizzati; ancora meglio se con dataset più grande e modelli anche più complessi e pesanti.
In ogni caso, la soluzione appare discretamente completa, sia per quanto riguarda la possibilità di avere un applicativo che in pochi semplici passi può essere utilizzato su di un pc con camera dedicata che su di uno smartphone. In quest'ultimo caso è importante sottolineare come l'idea alla base è veramente interessante. Infatti, nel caso dello smartphone, la peculiarità di questo sviluppo consiste proprio nel fatto che si va a trasferire tutta la complessità computazionale data dall'addestramento delle reti neurali, che in genere implica ore se non giorni per essere portati a termine, direttamente sul pc, lasciando la sola parte di inferenza, decisamente più leggera, nel dispositivo mobile. Questo comporta avere a portata di mano uno strumento comunque efficace ed efficiente anche in termini di tempo di esecuzione, equiparabile al computer. Inoltre, importante è anche la gestione della privacy, infatti per come è stato progettato il sistema, alla fine di ogni run si procede alla rimozione di tutti i file utilizzati, eliminando ogni traccia di foto potenzialmente private.
Detto questo, rilevanti sono anche i possibili sviluppi futuri. Non a caso in apertura a questo lavoro ho deciso di inserire una frase del fenomenale Albert Einstein, la quale cita: "L'immaginazione è più importante della conoscenza. La conoscenza è limitata, l'immaginazione abbraccia il mondo". In questo caso è molto simbolica e rappresentativa. Questo perché una volta ottenuta

la base di sviluppo, quale potrebbe essere il presente lavoro, si aprono infinte opzioni da seguire a discrezione dello sviluppatore. Prima tra tutte la possibilità di implementare la "E" mancante nella regola ABCD. Infatti, si potrebbe pensare di sviluppare un sistema che raggruppi lesioni simili e ne confronti lo stato ad intervalli regolari, monitorando le foto ricavate. Qui ovviamente si aprirebbe tutto un discorso di permessi e privacy a cui bisognerebbe dedicare gran parte del lavoro. Un'altra opzione potrebbe essere quella di sviluppare un intero apparecchio a sé stante, un po' come fatto nel prototipo proposto qui, molto più performante e completo sotto ogni punto di vista. Altra idea, seppur banale, potrebbe essere quella di scrivere direttamente da scratch una rete neurale innovativa e dedicata esclusivamente al task di classificazione su questi dataset. Questo ragionamento poi può ovviamente valere per tutti gli altri task analizzati.
Certamente le soluzioni possibili sono tante, più o meno complesse, ma non per questo meno interessanti. Con il presente lavoro si è dimostrato come l'implementazione di modelli di apprendimento automatico su dispositivi di diverso genere, mobile e pc, possa migliorare la rapidità, la privacy e l'efficienza delle applicazioni in ambito medico, offrendo un notevole potenziale per migliorare la fornitura di servizi sanitari e la diagnosi medica. L'idea fondante però rimane la stessa: ovvero, questi sistemi non devono e non dovranno essere pensati come completi sostituti degli attuali professionisti. Questo è fondamentale. In un mondo che si sviluppa ad alta velocità è di rilievo mantenere i rapporti e le relazioni tra persone e animali, e ciò non deve essere tralasciato.

Bibliografia

[1] S. Jain, N. Pise, *et al.*, "Computer aided melanoma skin cancer detection using image processing," *Procedia Computer Science*, vol. 48, pp. 735–740, 2015.

[2] M. K. Monika, N. A. Vignesh, C. U. Kumari, M. Kumar, and E. L. Lydia, "Skin cancer detection and classification using machine learning," *Materials Today: Proceedings*, vol. 33, pp. 4266–4270, 2020.

[3] X. Dai, I. Spasić, B. Meyer, S. Chapman, and F. Andres, "Machine learning on mobile: An on-device inference app for skin cancer detection," in *2019 fourth international conference on fog and mobile edge computing (FMEC)*, pp. 301–305, IEEE, 2019.

[4] Family caregivers online, "What is melanoma? learn the abcde of moles." `https://familycaregiversonline.net/what-is-melanoma-learn-the-abcde-of-moles/`, September 16, 2021.

[5] American Society of Clinical Oncology (ASCO), "Melanoma: Statistics," 03/2023.

[6] Tschandl, P. Rosendahl, C. Kittler, "The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," 14 August 2018.

[7] Raspberry Pi, "Raspberry pi 3." `https://www.raspberrypi.com/products/raspberry-pi-3-model-b/`.

[8] Raspberry Pi, "Getting started with the camera module." `https://projects.raspberrypi.org/en/projects/getting-started-with-picamera`.

[9] Raspberry Pi, "Raspberry pi os." https://www.raspberrypi.com/software/.

[10] Van Rossum, Guido and Drake Jr, Fred L, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018. cite arxiv:1801.04381.

[12] C. A. Morton and R. M. Mackie, "Clinical accuracy of the diagnosis of cutaneous malignant melanoma," *British Journal of Dermatology*, vol. 138, no. 2, pp. 283–287, 1998.

[13] A. F. Duarte, B. Sousa-Pinto, L. F. Azevedo, A. M. Barros, S. Puig, J. Malvehy, E. Haneke, and O. Correia, "Clinical abcde rule for early melanoma detection," *European Journal of Dermatology*, vol. 31, no. 6, pp. 771–778, 2021.

[14] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[15] M. Sharafudeen and V. Chandra, "Detecting skin lesions fusing handcrafted features in image network ensembles," *Multimedia Tools and Applications*, vol. 82, pp. 1–21, 06 2022.

[16] Wikipedia, "Matrice di co-occorrenza." https://it.wikipedia.org/wiki/Matrice_di_co-occorrenza, 25 feb 2021.

[17] J. Weston and C. Watkins, "Multi-class support vector machines," tech. rep., Citeseer, 1998.

[18] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[19] Kivy, "Kivy:the open source python app development framework." https://kivy.org/.

[20] Makesense.ai, "Makesense.ai." https://www.makesense.ai/.

[21] Pytorch, "fasterrcnn_resnet50_fpn." https://pytorch.org/vision/main/models/generate/torchvision.models.detection.fasterrcnn_resnet50_fpn.html.

[22] J. Howse, *OpenCV computer vision with python*, vol. 27. Packt Publishing Birmingham, 2013.

[23] Y. Heryadi, E. Irwansyah, E. Miranda, H. Soeparno, K. Hashimoto, *et al.*, "The effect of resnet model as feature extractor network to performance of deeplabv3 model for semantic satellite image segmentation," in *2020 IEEE Asia-Pacific Conference on Geoscience, Electronics and Remote Sensing Technology (AGERS)*, pp. 74–77, IEEE, 2020.

[24] M. Odusami, R. Maskeliūnas, R. Damaševičius, and T. Krilavičius, "Analysis of features of alzheimer's disease: Detection of early stage from functional brain changes in magnetic resonance images using a finetuned resnet18 network," *Diagnostics*, vol. 11, no. 6, p. 1071, 2021.

[25] Microsoft, "Documentation for visual studio code." https://code.visualstudio.com/docs.

[26] Pytorch, Automatic Differentiation In, "Pytorch," 2018.

[27] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, *et al.*, "Captum: A unified and generic model interpretability library for pytorch," *arXiv preprint arXiv:2009.07896*, 2020.

Ringraziamenti

A conclusione di tutto ciò, desidero dedicare un piccolo spazio a chi, con dedizione e pazienza, ha contribuito alla realizzazione di questo elaborato e mi è stato accanto in questo lungo percorso.
In primis, un ringraziamento speciale al mio relatore il Prof. Aldo Franco Dragoni e il suo prestigioso team dell'AIRTLab per la pazienza e la disponibilità offertami, nonostante gli innumerevoli impegni. Rimanendo nell'ambito universitario, un ringraziamento va anche ai Dott. Paolo Sernani e Dott.ssa Selene Tomassini per i preziosi consigli e per avermi suggerito puntualmente le giuste modifiche da apportare alle diverse fasi del progetto.
Ringrazio infinitamente la mia famiglia ed i miei parenti che mi hanno sempre sostenuto, appoggiando ogni mia decisione, fin dalla scelta del mio percorso di studi e che, nel momento del bisogno si sono dimostrati sempre disponibili e pronti a darmi manforte. Senza il loro aiuto, emotivo ed economico soprattutto, non avrei mai potuto intraprendere questo percorso di studi e non sarei qui ora a scrivere queste righe. Un grazie speciale va ai miei amici e colleghi, con i quali purtroppo non passo mai abbastanza tempo insieme, fonti di divertimento e condivisione, sempre pronti a strapparti un sorriso quando più ne hai bisogno. Infine, ma non per importanza, ringrazio la mia cara Giovanna per essermi stata sempre accanto, per aver sopportato le mie infinite lamentele, per la sua pazienza e il suo senso del dovere, che mi hanno portato ad essere la persona che sono oggi.
Per ultimo, credo sia importante dedicare questa tesi anche a me stesso, a ricordo dei sacrifici e della tenacia che mi hanno permesso di arrivare fin qui. Fermarsi e rammentarsi degli importanti obiettivi raggiunti nella vita ogni tanto fa bene.

Sperando che ne arrivino molti altri...