



UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea Triennale in Informatica

**RETI NEURALI CONVOLUTIVE PER IL
RICONOSCIMENTO DI OCCLUSIONI NEI VOLTI**

Relatore:
Prof. Giuliano Grossi
Correlatore:
Dr. Sathya Bursic

Tesi di Laurea di:
Andrea Valota
Matricola: 908884

Anno Accademico 2019/2020

Ringraziamenti

Un sentito grazie a tutte le persone che mi hanno permesso di arrivare fin qui e di portare a termine questo lavoro di tesi. Vorrei ringraziare in particolare il mio relatore Giuliano Grossi e il mio correlatore Sathya Bursic per il supporto e la disponibilità mostrate durante questo percorso. Ringrazio mia madre per avermi dato la possibilità di intraprendere questo percorso di studi e per avermi sostenuto in questi anni. Ringrazio i miei amici che mi sono stati vicini in questi anni e un ringraziamento particolare va ad Alessia per avermi sostenuto e per l'appoggio che mi ha fornito durante questi anni.

Indice

Ringraziamenti	i
Indice	ii
Introduzione	1
1 Le CNN per la classificazione delle immagini	3
1.1 ResNet	9
1.2 RetinaFace	11
2 Dataset di volti con mascherine	12
2.1 MAFA	12
2.2 Medical Mask Dataset	14
2.3 WIDER FACE	14
2.4 FERET Color	16
2.5 Creazione del dataset per il training della rete	16
3 Classificazione di volti con mascherine	18
3.1 Detection con Retinaface	18
3.2 Finetuning di ResNet	18
3.3 Risultati	19
Conclusioni	27
Bibliografia	29

Introduzione

In questo elaborato viene descritto il lavoro di tirocinio che pone l'utilizzo di reti neurali convoluzionali alla base del riconoscimento del corretto impiego di mascherine per il contrasto di pandemie con patologie a carattere respiratorio, come quella prodotta dal covid-19. Impieghi tipici di riconoscitori siffatti possono essere il controllo di luoghi di accesso o impieghi di natura medicale ove si mira a salvaguardare la salute pubblica.

Si è scelto di affrontare questo problema di apprendimento supervisionato impiegando noti modelli di reti neurali profonde come RetinaFace e FaceNet a motivo della loro riconosciuta capacità di reperire oggetti (detection) e categorizzare (classification) immagini, attitudini quest'ultime ritenute molto "vicine" a quelle umane. Un esempio pionieristico di successo è rappresentato proprio dall'impiego di detti modelli per il riconoscimento facciale, un problema che presenta molte analogie con quello trattato in questa tesi.

Il lavoro nel suo complesso è stato sviluppato sotto due aspetti principali: la definizione di un dataset adeguato a rappresentare situazioni realistiche presenti in ambienti frequentati anche da molti individui e l'applicazione dei modelli di rete sopraccitati nel caso di poche (fino a quattro) classi utili a distinguere la contingenza di volti occlusi da mascherine correttamente indossate o meno. Per l'esattezza, la costruzione del nuovo dataset è stata compiuta a partire dai dataset pubblici MAsked FAces, Medical Mask Dataset, WIDER FACE, FERET a cui sono state aggiungente immagini ottenute da ricerche mirate.

Come modello di rete per la fase di classificazione è stato utilizzato un'implementazione di FaceNet basata sull'architettura InceptionResnet, precedentemente addestrato sul dataset VGGFace2 (un dataset costruito appositamente per riconoscimento facciale) e riaddestrato qui attraverso un processo semplificato di finetuning per le classi utili al nostro progetto.

Una preliminare valutazione dei risultati ottenuti, basate su metriche standard, suggerisce che la strategia introdotta è promettente e ottiene qualità elevata, fino al 98%, per le classi in cui si hanno un elevato numero di dati e un obiettivo "semplice" come determinare la presenza/assenza di mascherina. Nei casi in cui si chiede di classificare il posizionamento corretto o meno della mascherina sul volto, il ratio decade attorno all'86%. Tra gli sviluppi futuri che questo progetto dovrebbe considerare vi è sicuramente un incremento numerico e di varietà del dataset, che è premessa essenziale per elevare le capacità di modelli neurali come quelli qui utilizzati e eventualmente l'uso di classificatori più

potenti che impiegano le feature prodotte da FaceNet per apprendere sottospazi a più elevata capacità discriminatoria.

Capitolo 1

Le CNN per la classificazione delle immagini

Una rete neurale (in inglese neural network) è un modello matematico composto da neuroni artificiali di ispirazione alle reti neurali biologiche (quella umana o animale) e viene utilizzata per risolvere problemi ingegneristici di Intelligenza Artificiale legati a diversi ambiti tecnologici come l'informatica, l'elettronica o altre discipline. Le reti neurali fanno sì che i computer siano in grado di risolvere i problemi in modo indipendente.

Un neurone artificiale è la componente basilare di una rete neurale. Come già detto la sua struttura è basata su quella dei neuroni biologici in cui l'informazione arriva al neurone attraverso i dendriti, viene processata dal corpo cellulare e viene propagata attraverso l'assone. In un neurone artificiale, invece, l'informazione arriva nel corpo del neurone attraverso degli input pesati (ad ogni input corrisponde un peso); il corpo del neurone artificiale quindi somma gli input pesati e un bias e successivamente applica una funzione di trasferimento. Possiamo quindi riassumere la struttura di un neurone come:

$$y(k) = F\left(\sum_{i=0}^m w_i(k) \cdot x_i(k) + b\right)$$

Dove:

- $x_i(k)$ rappresenta il valore dell'input i al tempo k
- $w_i(k)$ rappresenta il peso al tempo k associato all'input i
- b rappresenta il bias
- F rappresenta la funzione di trasferimento

- $y(k)$ rappresenta l'output del neurone al tempo k

La scelta della funzione di trasferimento da utilizzare dipende dal problema che la rete deve essere in grado di risolvere. Solitamente la funzione scelta è una funzione non lineare, perché senza la non linearità ogni rete a più strati sarebbe equivalente a una a due strati in grado di separare solo regioni linearmente separabili dei dati in input. Se la funzione applicata è non lineare questa struttura prende il nome di *percettrone* [1].

Possiamo legare più percettroni tra di loro per creare reti complesse. In particolare un **Percettrone Multi-strato** o **MLP** è una rete neurale costituita da più percettroni connessi tra di loro in un grafo aciclico diretto. Solitamente i percettroni sono raggruppati in layer (o livelli): un layer di input, uno di output e vari layer nascosti. Il numero di livelli nascosti è arbitrario e influenza la capacità di apprendimento della rete. Nel MLP inoltre gli output dei percettroni di un layer diventano gli input di tutti i percettroni del layer successivo: in questo caso parliamo di layer completamente connesso [2].

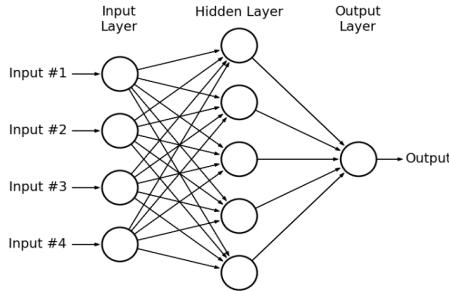


Figura 1.1: Struttura di un MLP

La funzione non lineare F è, in questo caso, applicata ad ogni output di ogni neurone e possiamo vedere l'output della rete come la composizione di varie funzioni. La non linearità delle funzioni applicate è, quindi, fondamentale perché la combinazione di funzioni lineari è sempre una funzione lineare mentre questo non vale per le funzioni non lineari e ciò permette alla rete di riconoscere elementi più complessi.

Le **Reti Neurali Convoluzionali** (Convolutional Neural Networks o CNNs) sono una tipologia di reti neurali che si è rivelata molto efficace in campi come la classificazione e il riconoscimento di immagini. Questo perché, come spiegato in [3], le reti convoluzionali hanno come principale vantaggio rispetto alle reti neurali standard quello di sfruttare l'invarianza traslazionale e l'utilizzo di pesi condivisi. Questo significa che, se un MLP dovesse imparare a riconoscere un oggetto all'interno di un'immagine, i pesi dei neuroni che imparano da una certa zona dell'immagine imparerebbero in modo indipendente dai neuroni che si rife-

riscono ad un'altra zona dell'immagine. Questo comporta quindi la necessità di avere dei dataset molto più ricchi e complessi per far fronte a molti più casi possibili in cui l'oggetto da riconoscere si trova in posizioni diverse dell'immagine. Al contrario, una rete convoluzionale aggiorna dei pesi condivisi che non sono relativi a determinate zone dell'immagine; questa differenza è dovuta all'operazione di convoluzione che verrà spiegata in seguito. Un'altra caratteristica delle CNN rispetto ad un MLP è anche il fatto che la CNN è in grado di effettuare un sottocampionamento su base spaziale, quindi ridurre il numero di elementi da considerare in relazione alla loro posizione nello spazio. Questo porta ad avere un addestramento più veloce. Una CNN è composta, come un MLP, da vari layer: un layer di input, uno di output e vari layer nascosti.

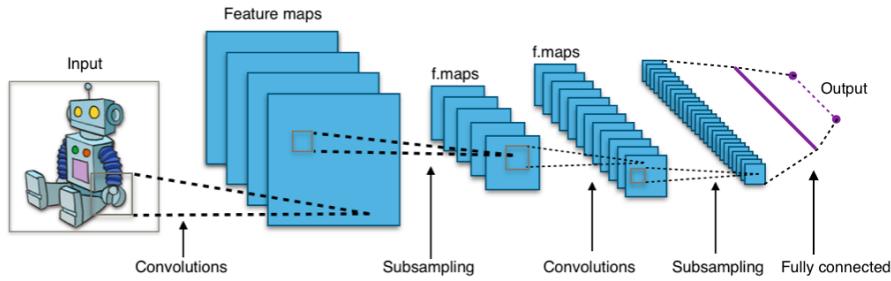


Figura 1.2: Struttura di una CNN

I layer intermedi più comuni sono:

- Convoluzione
- Attivazione
- Pooling

La convoluzione applica alle immagini in input una serie di filtri per estrarre delle feature. Con filtro intendiamo una matrice di pesi che viene fatta scorrere sull'input e per ogni posizione genera come output il prodotto scalare tra il filtro e la porzione coperta dell'input. In base al filtro applicato possono essere eseguite diverse operazioni come ad esempio l'estrazione dei contorni, la sfocatura dell'immagine o la messa a fuoco; utilizzando quindi diversi filtri possono essere estratte diverse feature utili alla rete. Inoltre possiamo applicare all'input del *padding*, ovvero aggiungere al contorno dell'immagine dei pixel di valore 0, per far sì che l'output abbia la stessa dimensione dell'input (che altrimenti risulta più piccolo a causa dell'operazione eseguita). Altro parametro utile per questa

fase è lo *stride* che indica di quanto spostare la matrice filtro ad ogni sua applicazione sull'input. In questo modo otteniamo delle matrici che contengono le feature estratte, dette *feature maps*.

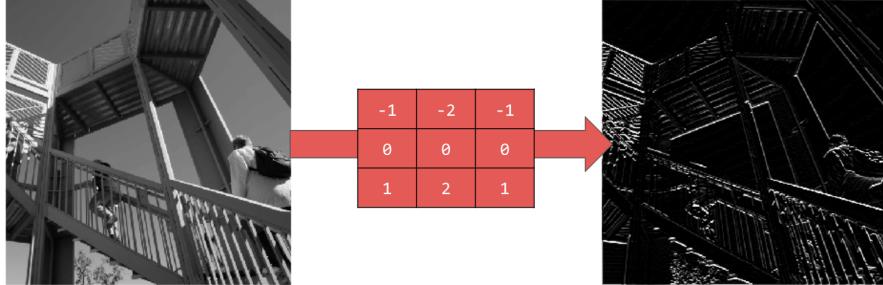


Figura 1.3: Esempio di filtri per estrarre feature

Nel layer di attivazione alla feature map viene applicata una funzione di attivazione. Questo passaggio è analogo all'applicazione della funzione di transizione in un percettrone. L'obiettivo di questo layer è quello di introdurre la non linearità nel sistema che fino ad ora sta calcolando solo operazioni lineari. Una delle funzioni più note e utilizzate è l'operazione *ReLU* (Rectified Linear Unit), che rimpiazza tutti i pixel con valori negativi in una feature map con il valore zero.

$$f(x) = \max(0, x)$$

Come illustrato in [4], questa funzione è particolarmente utilizzata poiché evita il problema dell'annullamento del gradiente, cosa che causa l'impossibilità da parte della rete di aggiornare i pesi nella fase di training (spiegata in seguito).

Il *Pooling* riduce la dimensionialità di ogni feature map mantenendo le informazioni principali. Un esempio di Pooling è il Max Pooling (come mostrato in Figura 1.4), che applica alla feature map un filtro che ricava solo i valori massimi tra quelli considerati dal filtro stesso. In questo modo si riducono le dimensioni delle feature maps, si rende la rete resistente rispetto a piccole trasformazioni o distorsioni e si rende meno costoso a livello computazionale il lavoro ai layer successivi.

Queste operazioni vengono ripetute su molti layer, ognuno dei quali impara ad identificare feature diverse. Dopo aver appreso le feature la CNN deve effettuare la fase di classificazione. Il penultimo layer è un layer completamente connesso che genera un vettore di K dimensioni, dove K è il numero di classi che la rete sarà in grado di prevedere. Si utilizza un layer completamente connesso per far sì che la rete possa imparare non solo dalle singole feature ma anche

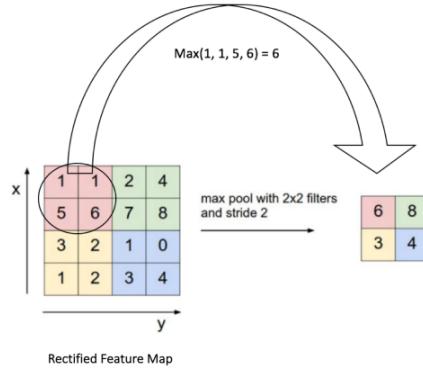


Figura 1.4: Esempio di Max Pooling

da combinazioni di esse. Il layer finale della CNN riceve il vettore creato nel layer precedente e ritorna come output un vettore che associa ad ogni classe possibile per la classificazione un valore. Questo valore viene ottenuto utilizzando la funzione SoftMax che effettua una normalizzazione dei valori presenti nel vettore. In questo modo otteniamo un valore di probabilità per ogni classe da cui si ottiene la classificazione predetta dalla rete.

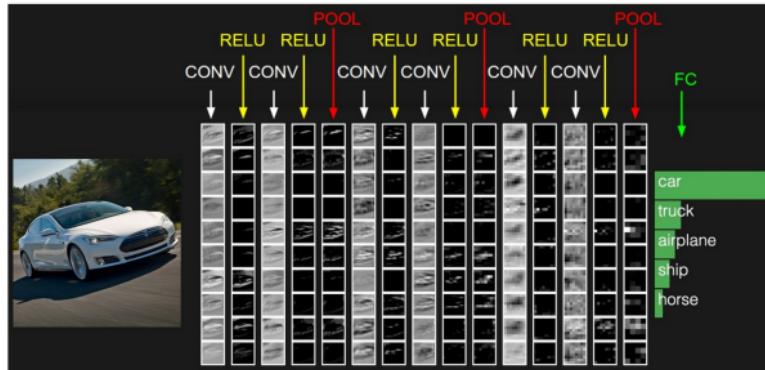


Figura 1.5: Esempio classificazione

Per far sì che la rete sia in grado di effettuare la classificazione deve essere prima effettuata una fase di training, solitamente utilizzando un algoritmo di *backpropagation* (approfondito in seguito). In questa fase alla rete vengono passate in input delle immagini che la rete utilizzerà per migliorare i propri parametri ed effettuare classificazioni più accurate. Inizialmente i parametri

possono essere inizializzati con valori casuali. Poi la rete riceve in input delle immagini che fa passare attraverso il processo di classificazione e in questo modo ottiene i valori di probabilità rispettive alle varie classi. Viene poi calcolata la *Loss Function*, ovvero una funzione che indica l'errore commesso dalla rete nella classificazione. Come mostrato in [5], esistono diverse loss function ed è possibile definirne infinite. Ogni loss function misura la discrepanza tra valori reali e valori predetti, ma il tipo di loss scelta influenza completamente la futura bontà del modello. Possiamo definire un modello generale di loss function per la i -esima istanza come una funzione che mette in relazione il valore predetto dalla rete con il valore che codifica la classe a cui l'immagine appartiene realmente. Un esempio di loss function è l'errore quadratico:

$$\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2$$

La cost function da minimizzare può essere quindi vista come la media sulla porzione di dataset utilizzata per la fase di training (*train set*):

$$\mathcal{E}(y, t) = \frac{1}{2N} \sum_{i=1}^N (y^{(i)} - t^{(i)})^2$$

Dove:

- i identifica l' i -esimo elemento del train set,
- y è l'insieme delle predizioni effettuate dalla rete sul train set,
- $y^{(i)}$ è la predizione effettuata dalla rete sull' i -esimo elemento,
- t è l'insieme degli indici corrispondenti alle classi reali degli elementi nel train set,
- $t^{(i)}$ è l'indice corrispondente alla classe reale a cui appartiene l' i -esimo elemento,
- N è la cardinalità dell'insieme di elementi considerati.

Successivamente, utilizzando il metodo di discesa del gradiente applicato alla cost function si calcola come ottimizzare i parametri. Questo metodo permette di diminuire il valore della loss function basandosi sul gradiente della funzione stessa; questo avviene poiché il gradiente ci permette di controllare il cambiamento del valore della funzione in relazione al cambiamento dei valori delle variabili considerate. In questo modo si può sapere come modificare il valore di ogni variabile per diminuire il valore della loss function. Se le variabili considerate sono i pesi della rete allora questo metodo definisce la variazione che il valore di ogni singolo peso deve subire per ridurre il valore della loss function e, di conseguenza, per effettuare una classificazione migliore.

Come spiegato in [6], la backpropagation consiste quindi nel passare alla rete un’immagine da classificare e calcolare il valore della loss function. Una volta calcolata la loss function il suo valore viene propagato in tutti i layer in direzione dei layer di input. Per ogni layer attraversato si incontrano tutti filtri e pesi che la rete ha utilizzato per eseguire la classificazione. Il processo inizia calcolando il gradiente per ogni unità di output. Successivamente, applicando la regola della catena delle derivate si può calcolare in maniera efficiente il gradiente del layer precedente e questo sistema può quindi essere usato per calcolare il gradiente rispetto a ogni variabile di quel layer. Eseguendo nuovamente il processo si continua a risalire verso i layer più vicini a quello di input e così facendo si calcola il gradiente per ogni peso nella rete. In questo modo a partire dall’output della rete e dal suo errore i pesi verranno modificati in modo da minimizzare l’errore, così che la classificazione successiva sia migliore rispetto alla precedente.

Un’alternativa ad effettuare il training da zero è quella di effettuare un *Fine-Tuning* della rete. In questo caso si parte da una rete pre-addestrata su un problema simile e si rimpiazza il livello di output con uno nuovo che rispetterà il numero di nuove classi che la rete dovrà riconoscere. In questo modo i valori iniziali dei pesi saranno quelli della rete pre-addestrata, tranne che per le connessioni tra il penultimo e l’ultimo layer, e il training complessivo della rete sarà molto più veloce. Questo è l’approccio che abbiamo utilizzato in questo progetto. In entrambi i casi specificati si parla di *Apprendimento Supervisionato* poiché è necessario per l’apprendimento che le immagini passate alla rete come training set abbiano indicato a quale classe appartengono realmente, in modo che la rete possa confrontare le sue predizioni con il risultato reale.

Esistono varie architetture di CNN che si sono susseguite negli anni:

- LeNet (1990s)
- AlexNet (2012)
- GoogLeNet (2014)
- VGGNet (2014)
- ResNet (2015)

Attualmente l’architettura ResNet è lo stato dell’arte per quanto riguarda i modelli di reti neurali convoluzionali ed è solitamente la scelta migliore per quanto riguarda gli utilizzi pratici.

1.1 ResNet

ResNet è un’architettura progettata per permettere di avere delle reti molto profonde (con un elevato numero di layer) e non incappare nei normali problemi che questo tipo di reti ha. Si è notato, infatti, che dopo una certa profondità

alcuni tipi di reti tendono a peggiorare le proprie performance invece che migliorarle. Questo avviene a causa del problema relativo all'annullamento del gradiente, un problema per il quale se la rete è troppo profonda i gradienti della loss function raggiungono velocemente il valore 0 e questo fa sì che i pesi della rete non si aggiornino. Inoltre si è anche notato che in reti profonde l'accuratezza della classificazione aumenta fino a saturarsi, dopodiché decresce rapidamente. Come spiegato in [7], per risolvere questo problema l'architettura ResNet non si pone come obiettivo quello di imparare solo la funzione obiettivo ma il suo residuo ottenuto sottraendo la funzione identità. Quindi se il mapping desiderato è $H(x)$ questa architettura fa in modo che i layer arrivino a mappare $F(x) = H(x) - x$. La funzione originale diventa quindi $F(x) + x$ e l'operazione $F(x) + x$ è effettuata da delle shortcut connections che effettuano un'addizione elemento per elemento tra l'input di un layer e il suo output.

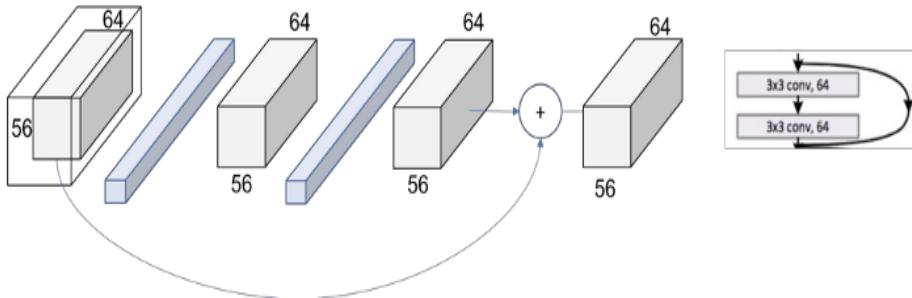


Figura 1.6: Esempio di operazioni in ResNet

Inoltre in questa architettura la riduzione del volume delle feature avviene aumentando lo stride invece che utilizzando un'operazione di pooling tranne che nella fase iniziale e finale. Per poter eseguire l'addizione elemento per elemento quando viene ridotta la dimensione delle feature lo shortcut effettua un'operazione di convoluzione per far sì che le dimensioni degli operatori da sommare siano uguali.

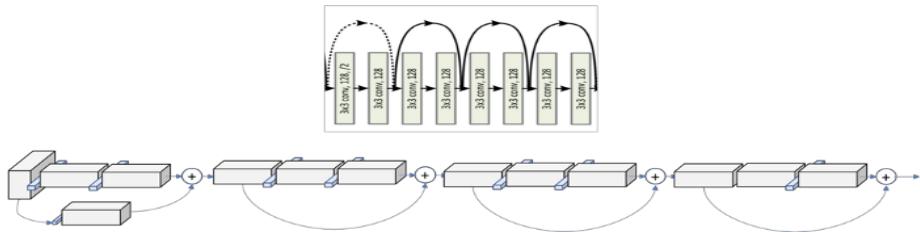


Figura 1.7: Esempio di un blocco di Resnet

Si è dimostrato che ottimizzare il residuo fa ottenere risultati migliori che ottimizzare la funzione stessa e che questo tipo di architettura risolve i problemi delle architetture profonde precedenti.

1.2 RetinaFace

RetinaFace è un single-stage face detector che abbiamo utilizzato in questo progetto. RetinaFace utilizza un architettura ResNet insieme ad una FPN (Fully Pyramidal Networks) per produrre delle rappresentazioni delle feature dell'immagine. In questo modo la rete è in grado di estrarre feature di vari livelli mentre la FPN permette al modello di riconoscere in maniera più accurata anche volti più piccoli. Come descritto in [8], RetinaFace utilizza dei livelli di una piramide di feature computata dalla ResNet per effettuare la detection. Successivamente, ai livelli della piramide di feature vengono applicati dei Context Modules per aumentare il campo ricettivo. Poi per le ancore negative viene calcolata la loss function di classificazione mentre su quelle positive viene calcolata una multi-task loss function che tiene conto di vari fattori differenti.

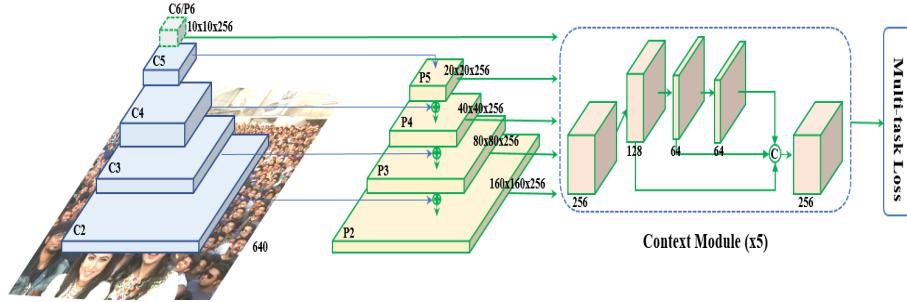


Figura 1.8: Struttura di RetinaFace

Vengono utilizzate delle ancore specifiche in base alla scala considerata sui livelli della piramide di feature: in questo modo le ancore cattureranno i volti. Durante la fase di training le ancore vengono confrontate con i bounding box reali dei volti e si considera di avere un match quando l'indice IoU (Intersection over Union) è maggiore di 0.5, mentre le ancore che non hanno un match vengono scartate. Una volta trainata, RetinaFace può essere utilizzata per effettuare la detection di volti su nuove immagini e la rete ritorna come output:

- bounding box del volto
- punti principali del volto (occhi, punta del naso e angoli della bocca)

Capitolo 2

Dataset di volti con mascherine

In questo lavoro i dataset considerati sono collezioni di immagini che la rete utilizza per effettuare la fase di training e quella di test. In particolare ci concentriamo sulle immagini che raffigurano volti e volti occlusi. La prima fase del progetto è stata proprio quella di recuperare delle immagini che potessero essere utilizzate dalla rete per distinguere i volti coperti da una mascherina e i volti che invece non sono occlusi o sono occlusi da oggetti di altro tipo. Per far sì che la rete esegua una classificazione accurata è necessario che il dataset che viene utilizzato come training set sia vasto e ricco di immagini che presentano caratteristiche diverse tra di loro. Altra caratteristica importante del dataset è anche la presenza di descrittori, che danno informazioni riguardo alle immagini contenute nel dataset stesso e permettono di automatizzare il processo di suddivisione delle immagini nelle varie classi.

L'obiettivo è fare in modo che la rete sia in grado di riconoscere le mascherine indossate correttamente, quelle indossate in modo errato (che non coprono naso e bocca) e i restanti casi come negativi. Non essendoci molti dataset già esistenti che rispecchiano le caratteristiche a noi necessarie la prima fase del lavoro è stata quella di cercare dei dataset utili al progetto e unirli in un unico nuovo dataset da utilizzare per automatizzare la fase di training della rete.

2.1 MAFA

Un dataset utile ai nostri fini è il dataset MAsked FAces (MAFA) [9]. Questo dataset contiene immagini recuperate da social network e da motori di ricerca come Google e Bing, cercate utilizzando come parole chiave “face mask”, “occlusion” e “cover”; le immagini contenenti solo volti non occlusi sono state rimosse manualmente. Le immagini presenti quindi rappresentano tipicamente alcuni individui, o piccoli gruppi di persone, con il volto occluso da indumenti,

oggetti o maschere di varia natura. In totale il dataset contiene 30,811 immagini, ognuna contenente almeno un volto e ad ogni immagine è associato un descrittore costituito da:

- *posizione del volto*: la posizione del volto viene fornita sotto forma di bounding box. I volti troppo piccoli o irriconoscibili sono annotati come “Ignore”;
- *posizione degli occhi*: vengono fornite due coppie di punti che identificano la posizione degli occhi;
- *posizione della maschera*: anche la posizione della mascherina viene fornita sotto forma di bounding box;
- *orientazione del volto*: vengono definite 5 orientazioni possibili in base a dove è rivolto il volto del soggetto: sinistra, sinistra-frontale, frontale, destra-frontale, destra;
- *grado di occlusione*: il grado di occlusione è un indice numerico che indica quante zone del volto sono occluse. Le zone del volto considerate sono: mento, bocca, naso e occhi;
- *tipo di occlusione*: nel dataset sono definiti 3 tipi di occlusioni: semplici (mascherine a tinta unita), complesse (mascherine di forma o con colori complessi), corpo umano (se il volto è coperto da mani, capelli, etc.).



Figura 2.1: Esempio di immagini in MAFA

Per recuperare delle immagini utili al nostro progetto sono stati utilizzati i descrittori forniti per recuperare le immagini di mascherine semplici che occludono almeno tre zone del volto. In queste immagini sono solitamente rappresentate delle mascherine chirurgiche o di stoffa indossate correttamente e queste sono una delle classi che la rete deve imparare a classificare. Un’altra classe utile è la classe delle mascherine indossate in modo errato, cioè le mascherine semplici che coprono meno di tre zone del volto. In questa classe sono presenti mascherine semplici ma indossate sotto il mento o in modo che non coprano il naso. Le occlusioni di tipo “corpo umano” sono state incluse nella classe dei negativi visto che sono un tipo di occlusioni che non vogliamo che la rete riconosca. Infine le mascherine che sono classificate dal MAFA come complesse possono essere

considerate come classe a parte o possono essere suddivise tra le mascherine indossate correttamente e le errate secondo la stessa divisione basata sul livello di occlusione applicata alle maschere semplici.

2.2 Medical Mask Dataset

Un altro dataset utilizzato in questo progetto è il dataset Medical Mask Dataset presentato da Humans in the Loop in [10]. Questo dataset contiene 6000 immagini raffiguranti persone che indossano mascherine. Ad ogni volto nelle immagini è associato un descrittore in cui sono contenuti bounding box e classe. Tra le varie classi sono presenti:

- volto con maschera,
- volto con maschera indossata in modo errato,
- volto senza maschera,
- volto coperto da un altro oggetto.



Figura 2.2: Esempio di immagini in Medical Mask Dataset

Le classi proposte si sono rivelate simili a quelle che vogliamo che la rete riconosca e quindi, ponendo i volti non mascherati e quelli occlusi da oggetti diversi da mascherine nella classe dei negativi, abbiamo potuto usare i descrittori per dividere le immagini nelle varie classi di training.

2.3 WIDER FACE

Un’ulteriore risorsa è il dataset WIDER FACE [11]. Questo dataset contiene 32,203 immagini e 393,703 descrittori associati a volti presenti nelle immagini. Le immagini presenti in questo dataset raffigurano scenari molto differenti, che variano da raffigurazioni di singole persone a rappresentazioni di eventi pubblici con un gran numero di partecipanti; per questo i volti nel dataset presentano grandi variazioni in scala, posa e occlusione. Nei descrittori di ogni volto sono annotati i seguenti attributi:

- *bounding box del volto*: la posizione del volto viene fornita sotto forma di bounding box,
- *blur*: un indice numerico con valori interi compresi tra 0 e 2 che indica il grado di sfocatura del volto,
- *espressione*: un indice booleano che indica se l'espressione del volto è normale o esagerata,
- *illuminazione*: un indice booleano che indica se il volto è illuminato normalmente o sovraesposto,
- *indice di validità*: un indice booleano che indica se il volto è da considerarsi valido oppure no (perché irriconoscibile),
- *occlusione*: un indice con valori interi compresi tra 0 e 2 che indica se il volto non è occluso, è parzialmente occluso o se è pesantemente occluso,
- *posa*: un incide booleano che indica se il volto si trova in posa o in una posa atipica.



Figura 2.3: Esempio di immagini in WIDERFACE

Un problema riscontrato con questo dataset è stato, però, proprio l'indice riguardante l'occlusione. L'indice permette di distinguere se un volto non presenta occlusioni, presenta delle occlusioni parziali o se presenta occlusioni pesanti. Non è però specificato il tipo di occlusione e quindi non c'è un modo semplice di sapere che tipo di oggetto sta occludendo il volto. Dato che il nostro progetto

punta a riconoscere solamente le mascherine, abbiamo accantonato questo dataset per quanto riguarda la raccolta di immagini per la fase di training visto che non abbiamo trovato un modo semplice per recuperare la sottoclasse di immagini che interessavano il progetto. Questo dataset però è stato utilizzato per effettuare dei test sul modello trainato. Dato che la maggior parte dei volti nel dataset non appartengono a scenari in cui l'occlusione può essere una mascherina (ad eccezione di uno scenario in particolare), abbiamo potuto considerare i volti occlusi come dei casi negativi proprio poiché le occlusioni erano di natura diversa da una mascherina e abbiamo verificato che la rete li classificasse come negativi. Inoltre, sempre a causa della natura delle immagini contenute (sono presenti un gran numero di volti occlusi e non), questo dataset può essere una fonte di una grande quantità di volti utili da inserire nella classe dei negativi.

2.4 FERET Color

Un ulteriore dataset utilizzato è FERET Color [12]. Il dataset presenta 11338 immagini raccolte in un ambiente controllato che rappresentano persone differenti in varie pose. In tutte le immagini le persone non indossano alcun tipo di oggetto o indumento che copra il volto in maniera analoga ad una mascherina (l'unica occlusione presente sono occhiali da vista). Questo dataset si è rivelato utile nella fase di test per verificare quanto il modello riconoscesse volti non occlusi mai visti nella fase di training come casi negativi.

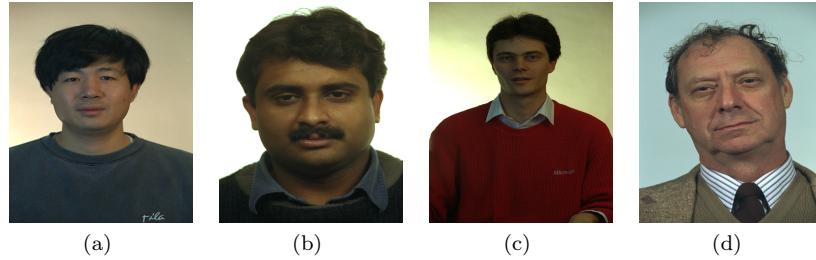


Figura 2.4: Esempio di immagini in FERET Color

2.5 Creazione del dataset per il training della rete

Utilizzando i dataset sopra elencati abbiamo quindi creato dei dataset con le classi adatte al progetto. Nella prima fase del progetto il dataset creato conteneva solo una classe di casi positivi e una di negativi. Le immagini utilizzate in questa fase sono i volti estratti da MAFA utilizzando principalmente i descrittori forniti per quanto riguarda i casi positivi, mentre per la classe dei negativi i volti sono stati estratti utilizzando RetinaFace. In questo dataset i casi positivi

sono i volti occlusi da maschere indossate correttamente mentre i volti occlusi da maschere complesse (vedi MAFA) e da oggetti di altro tipo sono etichettati come casi negativi. Questo primo dataset contiene:

- 12950 immagini nella classe dei positivi
- 16081 immagini nella classe dei negativi

Successivamente è stato necessario creare un altro dataset più dettagliato per aggiungere la classe di volti con mascherina indossata in modo errato. In questo caso i dati sono stati estratti dai dataset principalmente basandosi sui contenuti dei descrittori forniti, questo a causa della precisione necessaria per distinguere i casi errati da quelli positivi e sono state utilizzate immagini provenienti da MAFA e da Medical Mask Datsaset. Il dataset ottenuto è diviso in 3 classi e contiene rispettivamente:

- 23244 immagini di volti con mascherina indossata correttamente
- 1195 immagini di volti con mascherina indossata in modo errato
- 25914 immagini di casi negativi

I casi di mascherina indossata in modo errato sono presenti in numero minore rispetto alle altre poiché è risultato difficile trovare immagini di questi casi limitate e soprattutto perché per identificare questi casi servono annotazioni precise sui singoli volti in modo da dividere rigorosamente le classi.

Infine vista la situazione particolare delle maschere complesse del MAFA, che possono essere viste come casi positivi o negativi in base alle considerazioni fatte su quel tipo di mascherine, abbiamo deciso di distinguere dalle altre questa classe. Abbiamo quindi ottenuto un dataset con 4 classi contenente:

- 23244 immagini di volti con mascherina indossata correttamente
- 1195 immagini di volti con mascherina indossata in modo errato
- 15468 immagini di volti con mascherine complesse
- 10446 immagini di casi negativi

Capitolo 3

Classificazione di volti con mascherine

3.1 Detection con Retinaface

Nella fasi di creazione dei dataset utili al progetto è stato fondamentale l'utilizzo di RetinaFace [13] per l'estrazione dei volti dalle immagini. Grazie a questa rete siamo stati in grado di estrarre volti da immagini senza l'ausilio di descrittori che indicano la bounding box (coordinate del rettangolo) che include il volto. Le immagini recuperate in questo modo sono solitamente state utilizzate come casi negativi nei test dato che questa rete riconosce qualsiasi tipo di volto, non solo quelli occlusi da mascherine, e non c'è quindi un modo automatico per dividere i casi positivi da quelli negativi. La rete è stata utilizzata per isolare i volti nelle immagini appartenenti a WIDER FACE e a FERET Color (che da una prima analisi manuale sappiamo contenere al più casi negativi) e creare dei test set su cui confrontare il nostro modello. RetinaFace restituisce, come spiegato in precedenza, bounding box e punti principali del volto per ogni volto che identifica. Per utilizzare i volti ottenuti è bastato ritagliare le parti di immagine identificate dai bounding box ottenuti e poi normalizzare i dati in modo che la rete dedicata alla classificazione potesse utilizzarli (spiegato in seguito).

3.2 Finetuning di ResNet

Come rete per la fase di classificazione abbiamo utilizzato un'implementazione di Facenet basata sull'architettura InceptionResnet [14]. In questa fase abbiamo addestrato la rete attraverso il processo di finetuning spiegato in precedenza. Il modello di partenza utilizzato è stato un modello di InceptionResNetV1 addestrato sul dataset VGGFace2 [15], un dataset costruito appositamente per la *Face Recognition*. Il modello in questione era già addestrato per il riconoscimento di volti e quindi i parametri della rete erano già adatti a riconoscere una

tipologia di soggetti simile a ciò a cui punta il nostro progetto; per questo abbiamo deciso di utilizzare questo modello come punto di partenza e di effettuare il finetuning invece di effettuare il training della rete completamente nuovo. In particolare è stato necessario rimpiazzare l'ultimo layer lineare della rete, che nel modello addestrato su VGGFace2 ritornava classi relative a quel dataset, con un nuovo layer in grado di effettuare la classificazione sulle classi utili al nostro progetto. In questo modo la fase di training della rete è stata piuttosto rapida (proprio a causa dei pesi pre-esistenti in grado di riconoscere soggetti simili ai nostri obiettivi) e ci ha permesso di sfruttare ciò che la rete aveva imparato su VGGFace2 oltre che ciò che è stata in grado di imparare dal nostro dataset, ottenendo quindi risultati migliori rispetto al solo training sui nostri dataset.

Per effettuare il training della rete è stato necessario però fare sì che tutte le immagini avessero certe caratteristiche in modo che la rete considerasse solo il contenuto dell'immagine stessa. Per questo è stato necessario:

1. estrarre i volti dalle immagini del dataset,
2. ridimensionare le immagini ottenute a delle dimensioni standard,
3. mantenere solo i canali R, G e B dell'immagine.

L'estrazione dei volti è stata effettuata utilizzando i bounding box forniti con i descrittori dei dataset o quelli forniti da RetinaFace, mentre la dimensione standard utilizzata è 160x160 pixel. La necessità di rimuovere i canali non RGB dalle immagini si è presentata poiché alcuni dataset presentavano immagini in cui era presente il canale alfa e, per evitare problemi nella classificazione dovuti alla presenza di questo elemento, abbiamo deciso di eliminarlo.

Questa fase è stata ripetuta più volte a causa dei vari dataset che abbiamo considerato per questo progetto: il finetuning, infatti, è stato eseguito un numero di volte pari al numero di diversi dataset utilizzati per effettuarne il training. Così facendo siamo stati in grado di ottenere 3 modelli differenti per effettuare test e classificazioni diverse. Ogni modello è in grado di classificare le immagini nelle classi fornitegli dal dataset di training, di conseguenza il primo modello distinguerà casi positivi e negativi, il secondo modello identificherà volti con mascherina indossata correttamente, in modo errato e i casi negativi ed infine l'ultimo modello potrà anche dire se il volto considerato indossa una maschera complessa.

3.3 Risultati

Tutti i test effettuati in questo progetto prevedono l'utilizzo di un insieme di immagini divise in classi, definito come *test set*, che viene dato in input alla rete. La rete effettua la classificazione sulle immagini dopodiché vengono comparati i risultati predetti e le classi reali a cui appartengono le immagini. In base alle predizioni della rete si possono calcolare varie metriche che permettono

di valutare le performance della rete stessa. I principali valori che verranno utilizzati sono:

- accuratezza,
- matrice di confusione,
- curva ROC,
- l'AUC (o Area Under the ROC Curve),
- precisione,
- recall,
- F1 score.

L'accuratezza è un valore di probabilità che permette di valutare la bontà di un modello in base al numero di predizioni corrette che effettua e si calcola secondo la formula:

$$acc = \frac{\text{predizioni corrette}}{\text{elementi nel test set}}$$

Questo indice permette di valutare in maniera immediata le performance del classificatore e indica quanto questo sia in grado di classificare correttamente le immagini nelle rispettive classi.

La matrice di confusione è una matrice che permette di avere più informazioni sulla classificazione effettuata rispetto alla sola accuratezza. La matrice contiene una riga e una colonna per ogni classe considerata e contiene valori che indicano quanti elementi appartenenti ad una classe sono stati classificati come tali e quanti invece sono stati classificati erroneamente. Un esempio di matrice di confusione per due classi è:

		Predetti	
		NEGATIVI	POSITIVI
Reali	NEGATIVI	x	y
	POSITIVI	z	w

Tabella 3.1: Esempio di Matrice di confusione con 2 classi

Dove x, y, z e w sono rispettivamente:

- veri negativi,
- falsi positivi,
- falsi negativi,
- veri positivi.

Questa matrice permette di analizzare più in dettaglio quali sono gli errori commessi dalla rete poiché è facile, partendo dalla matrice, vedere dove il classificatore sbaglia e come sbaglia (in che classi pone gli errori). Utilizzando la matrice di confusione si può quindi capire come eventualmente modificare la fase di training della rete (ad esempio cambiando le proporzioni delle immagini nel training set) per fare in modo che gli errori vengano ridotti.

La curva ROC è un indicatore che permette di valutare la variazione della bontà della classificazione in base ad una certa soglia. Il classificatore ritorna per ogni immagine un valore di probabilità ed al variare di una soglia vengono valutati il *True Positive Ratio* e il *False Positive Ratio*. La curva ottenuta permette di effettuare delle valutazioni in base alla sua forma che ci permettono di confrontare tra di loro classificatori diversi e soprattutto di constatare se il classificatore prodotto è vicino ad un classificatore casuale o ad un classificatore che restituisce predizioni corrette. In particolare l'AUC ci permette di valutare la bontà del classificatore poiché, nel caso dell'area sotto la curva ROC, più questo valore è vicino a 1 più le predizioni della rete sono precise.

La precisione è un valore che indica la capacità del classificatore di non classificare come positivo un elemento negativo. Si calcola come il rapporto:

$$\frac{\text{veri positivi}}{\text{veri positivi} + \text{falsi positivi}}$$

La recall può essere interpretata come la capacità della rete di trovare tutti gli elementi positivi. Viene calcolata come:

$$\frac{\text{veri positivi}}{\text{veri positivi} + \text{falsi negativi}}$$

La metrica F1 score può essere interpretata come la somma armonica pesata di precisione e recall, in cui l'F1 score raggiunge il suo valore migliore a 1 e il peggiore a 0 [16].

Inizialmente gli esperimenti sono stati effettuati partendo dal dataset creato da noi composto solo da due classi, i casi positivi e i casi negativi. In questa fase il dataset è stato diviso in due parti: il 70% delle immagini è stato utilizzato come training set, mentre il restante 30% ha composto il test set. Nella fase di test sono stati ottenuti i seguenti risultati:

$$\text{accuratezza} = 0.9707$$

		Predetti	
		NEGATIVI	POSITIVI
Reali	NEGATIVI	4658	134
	POSITIVI	121	3797

Tabella 3.2: Matrice di confusione ottenuta nella prima fase di test

Per verificare la robustezza della classificazione il test è stato poi eseguito nuovamente su tutto il dataset ottenendo come risultati:

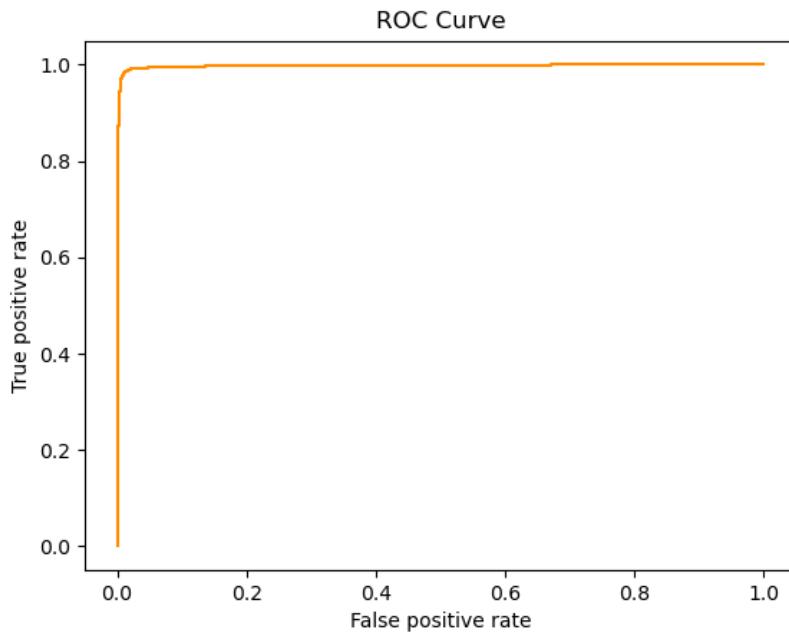


Figura 3.1: Curva ROC

$$\text{accuratezza} = 0.9856$$

$$AUROC = 0.992$$

Dai risultati ottenuti possiamo constatare che la rete costruita è in grado di distinguere i casi positivi da quelli negativi con certezza piuttosto alta. I valori nella matrice di confusione (Tabella 3.3) confermano la presenza di alcuni errori, in numero però ridotto rispetto al numero di elementi classificati correttamente. Anche precisione, recall e F1-score (Tabella 3.4) mostrano dei valori che indicano che la rete è in grado di riconoscere in modo consistente le due differenti

Predetti			
		NEGATIVI	POSITIVI
Reali	NEGATIVI	15907	174
	POSITIVI	244	12706

Tabella 3.3: Matrice di confusione relativa al test sul dataset completo

	PRECISION	RECALL	F1-SCORE
NEGATIVI	0.98	0.99	0.99
POSITIVI	0.99	0.98	0.98

Tabella 3.4: Precisione, recall e F1-score ottenute sul primo dataset

classi. Dalla forma della curva ROC e dal valore dell’AUC possiamo inoltre vedere come il classificatore si avvicini ad essere un classificatore perfetto per il dataset considerato; questo conferma ulteriormente il fatto che il nostro modello è in grado di effettuare delle predizioni corrette per il nostro dataset nella quasi totalità dei casi. I risultati ottenuti ci permettono quindi di dire che il finetuning della rete ci ha permesso di ottenere una rete in grado dividere in modo corretto i casi positivi da quelli negativi e ora bisogna verificare che il modello ottenuto sia robusto anche rispetto a delle immagini che non gli sono mai state presentate.

Per verificare che la rete non abbia imparato a distinguere solamente la tipologia di immagini del train set, abbiamo effettuato dei test con immagini di volti completamente nuovi, estrapolati da altri dataset grazie all’uso di RetinaFace. Abbiamo posto tutte le immagini estrapolate nella classe dei negativi dato che la maggior parte delle immagini conteneva casi negativi (volti non occlusi o occlusioni diverse da mascherine) e non c’è stato modo di isolare automaticamente gli sporadici casi positivi presenti. In particolare abbiamo effettuato la fase di test sui volti presenti in WIDER FACE e abbiamo ottenuto come risultati:

$$\text{accuracy} = 0.9950$$

Predetti			
		NEGATIVI	POSITIVI
Reali	NEGATIVI	124785	615
	POSITIVI	0	0

Tabella 3.5: Matrice di confusione relativa al test su WIDER FACE

In questo modo abbiamo potuto notare come la rete sia in grado di classificare come negativi anche volti nuovi e diversi tipi di occlusioni diverse da una mascherina anche se queste non sono mai state incontrate nel training.

Seguendo la stessa metodologia abbiamo testato il nostro modello anche su FERET Color. Anche qui i volti sono stati isolati utilizzando RetinaFace ma in questo caso sappiamo che tutti i casi presenti nel dataset sono immagini di persone con il volto non occluso quindi anche in questo caso tutte le immagini sono state poste nella classe dei negativi. I risultati ottenuti:

$$\text{accuratezza} = 0.9986$$

		Predetti	
		NEGATIVI	POSITIVI
Reali	NEGATIVI	3675	5
	POSITIVI	0	0

Tabella 3.6: Matrice di confusione relativa al test su FERET Color

Questo ci permette di dire che la rete riconosce correttamente come casi negativi la maggior parte dei volti non occlusi che gli vengono presentati.

I dati mostrano quindi come la differenza tra casi positivi e negativi sia stata imparata correttamente dalla rete e che la classificazione che questa restituisce approssima, con un grado di correttezza abbastanza alto, le classi obiettivo. Le due classi considerate fino ad ora però non racchiudono a pieno le differenze che vogliamo che la rete sia in grado di classificare. Il progetto, infatti, si pone come obiettivo quello di realizzare una rete in grado di distinguere anche se una mascherina, se presente, è indossata in maniera corretta. Per raffinare la rete in modo che sia in grado di effettuare questa ulteriore distinzione è però necessario suddividere ulteriormente il dataset in classi più fini.

Il passo successivo è stato quindi quello di aumentare il numero di classi che la rete deve essere in grado di riconoscere. Per questa fase è stato utilizzato il dataset composto dalle classi:

- volto con mascherina indossata correttamente,
- volto con mascherina indossata in modo errato,
- casi negativi.

La fase di training della rete ha seguito la stessa metodologia di quella effettuata con il dataset precedente quindi il 70% delle immagini sono state utilizzate per la fase di training e il 30% per la fase di test. I test effettuati hanno seguito la stessa metodologia dei precedenti, l'unica differenza è che in questa fase le classi considerate sono 3. Inoltre non è stato possibile effettuare test su dataset differenti poiché, a causa della mancanza di descrittori utili a differenziare la seconda classe dalle altre, non si sono trovati altri dataset che presentavano i casi voluti dalle nostre classi in modo ben identificato. La difficoltà principale

è quella di distinguere i casi di mascherina indossata in modo errato e quelle indossate in modo corretto in modo preciso visto che, in buona parte dei dataset considerati, questa differenza non viene segnalata. I risultati ottenuti:

$$accuracy = 0.8658$$

		Predetti		
		CORRETTE	ERRATE	NEGATIVI
Reali	CORRETTE	20364	82	2798
	ERRATE	233	541	421
	NEGATIVI	3115	108	22691

Tabella 3.7: Matrice di confusione relativa al test sul dataset con 3 classi

	PRECISION	RECALL	F1-SCORE
MASCHERINA CORRETTA	0.86	0.88	0.87
MASCHERINA ERRATA	0.74	0.45	0.56
NEGATIVI	0.88	0.88	0.88

Tabella 3.8: Precisione, recall e F1-score ottenute sul dataset con 3 classi

Come è possibile vedere dai dati, l'accuracy della rete si è ridotta dopo l'introduzione della terza classe. Possiamo attribuire l'aumento di errori, oltre che all'aumento di precisione richiesto alla rete, soprattutto allo sbilanciamento nei dati usati nel training. I casi di mascherina errata sono infatti presenti in numero ridotto rispetto alle altre classi ed è quindi più difficile per la rete imparare a riconoscere elementi di quella classe. Questo fatto è confermato in particolare dal valore recall (Tabella 3.8), che mostra chiaramente la difficoltà della rete nel riconoscere correttamente gli elementi presenti nella seconda classe. Un altro possibile fattore da considerare è la presenza di maschere complesse tra i negativi; questo potrebbe causare problemi alla rete che trova diversi tipi di maschere sia nei positivi che nei negativi e rende più imprecisa la classificazione.

Nell'ultima fase del progetto è stato quindi necessario creare l'ultimo dataset presentato in cui le maschere complesse sono state raccolte in una classe differente dalle altre per valutare se questo migliorasse le prestazioni della rete. Il training della rete anche in questo caso ha seguito la stessa metodologia e le stesse suddivisioni utilizzate nei casi precedenti. In questo modo è stato possibile vedere l'effetto che le maschere complesse hanno sul training e di conseguenza sui risultati della classificazione.

$$accuracy = 0.8611$$

		Predetti			
		COMPLESSE	CORRETTE	ERRATE	NEGATIVI
Reali	COMPLESSE	11446	3353	93	576
	CORRETTE	1074	21784	66	302
	ERRATE	167	262	664	102
	NEGATIVI	350	601	30	9465

Tabella 3.9: Matrice di confusione relativa al test sul dataset con 4 classi

	PRECISION	RECALL	F1-SCORE
MASCHERINA COMPLESSA	0.88	0.74	0.80
MASCHERINA CORRETTA	0.84	0.94	0.88
MASCHERINA ERRATA	0.78	0.56	0.65
NEGATIVI	0.90	0.91	0.91

Tabella 3.10: Precisione, recall e F1-score ottenute sul dataset con 4 classi

Come possiamo notare, il valore dell'accuratezza rimane molto simile al caso precedente e possiamo quindi ipotizzare che l'aumento degli errori della rete sia dovuto principalmente all'aggiunta della classe delle mascherine indossate in modo errato e al fatto che queste sono rappresentate da un numero relativamente piccolo di immagini rispetto alle altre classi. Anche in questo caso il valore di recall, mostrato nella Tabella 3.10, mostra che la rete ha problemi soprattutto con la classe di mascherine indossate in modo errato; questo conferma quindi l'ipotesi che la classe di mascherine indossate in modo errato è quella che impatta negativamente sulle performance della rete.

Conclusioni

Il fine del progetto è stato quello di creare una rete neurale convoluzionale in grado di riconoscere se un soggetto indossa correttamente o meno una mascherina (sanitaria) che protegga le vie respiratorie.

Una rete di questo tipo può rivelarsi utile in ambienti in cui è necessario verificare automaticamente che i soggetti indossino in modo corretto la mascherina, particolarmente in caso di pandemie come quella attuale. Il lavoro, in fase preliminare, ha richiesto la raccolta di numerose immagini, prese da varie fonti, che sono per effettuare l'addestramento di modelli di classificazione neurali. Il primo modello ottenuto si è rivelato molto efficace in caso di classificazione binaria, quello in cui si chiede di distinguere la presenza/assenza di mascherina sul volto. Nei casi in cui si chiede di classificare il posizionamento corretto o meno della mascherina sul volto, la prestazione decade decisamente.

Un possibile sviluppo per questo progetto può consistere nella ricerca di ulteriori immagini, specialmente di quelle contenenti i casi limite come le maschere indossate in modo errato, per aumentare la precisione della classificazione. Inoltre si può fare in modo che la rete riconosca un maggior numero di occlusioni diverse, rendendo la classificazione più fine, e si può creare un'implementazione che permetta alla rete di essere applicata direttamente anche a video invece che alle sole immagini. Un ulteriore sviluppo che questo progetto dovrebbe considerare è l'uso di classificatori più potenti rispetto alle reti neurali che impiegano le feature prodotte da FaceNet per apprendere sottospazi a più elevata capacità discriminatoria.

Bibliografia

- [1] A.Krenker, J.Bester, and A.Kos. Introduction to the artificial neural networks. In *ARTIFICIAL NEURAL NETWORKS - METHODOLOGICAL ADVANCES AND BIOMEDICAL APPLICATIONS*, pages 3–18. 2011.
- [2] M.Reidmiller. Advanced supervised learning in multi-layer perceptrons - from backpropagation to adaptive learning algorithms. In *Computer Standards and Interfaces*, pages 265–278. 1994.
- [3] S.Lawrence, C.Lee Giles, A.C.Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. In *IEEE TRANSACTIONS ON NEURAL NETWORKS*, pages 98–113. 1997.
- [4] H. Ide and T. Kurita. Improvement of learning for cnn with relu activation by sparse regularization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2684–2691, 2017.
- [5] Paolo Galeone. *Rete neurale convoluzionale per classificazione di immagini e localizzazione di oggetti*. Tesi di laurea, Università di Bologna, 2015/2016.
- [6] D.E.Rumelhart, G.G.Hinton, and R.J.Williams. Learning representations by back-propagating errors. *Nature*, 1986.
- [7] K.He, X.Zhang, S.Ren, and J.Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.
- [8] J.Deng, J.Guo, Y.Zhou, J.Yu, I.Kotsia, and S.Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. *arXiv:1905.00641v2*, 2019.
- [9] Shiming Ge, Jia Li, Qiting Ye, and Zhao Luo. Detecting masked faces in the wild with LLE-CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [10] Humans In The Loop. Medical mask dataset. humansintheloop.org/medical-mask-dataset.
- [11] S.Yang, P.Luo, C.C.Loy, and X.Tang. Wider face: A face detection benchmark. *arXiv:1511.06523*, 2015.

- [12] P.Philips, H.Moon, S.Rizvi, and P.Rauss. The feret evaluation methodology for face recognition algorithms. Technical report, University of Zurich, Department of Informatics, 01 1999.
- [13] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *CVPR*, 2020.
- [14] C.Szegedy, S.Ioffe, V.Vanhoucke, and A.Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261v2*, 2016.
- [15] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 67–74, 2018.
- [16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.