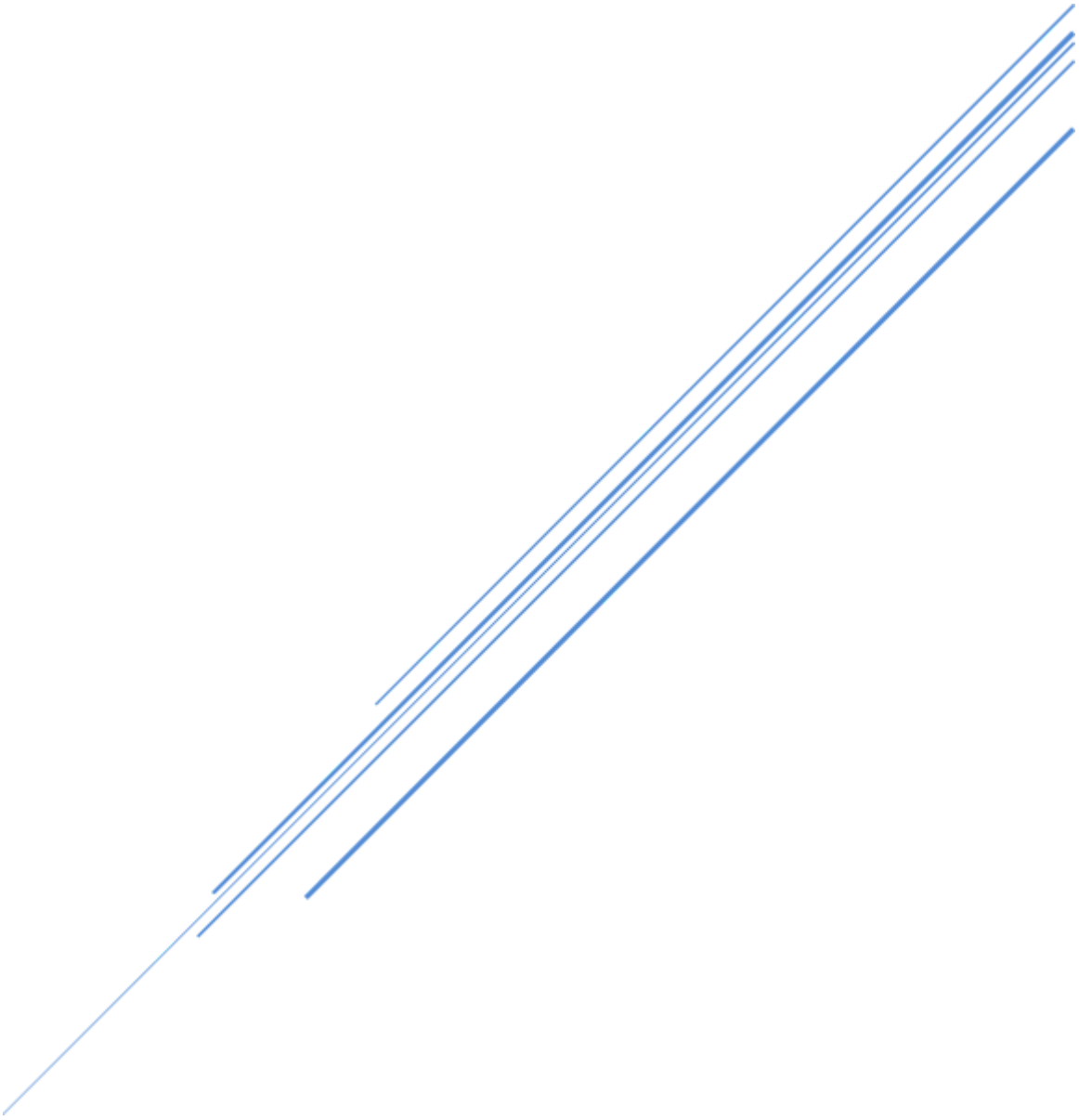
KINSHIP RECOGNITION:

Studio del tipo di relazione utilizzando reti siamesi



Università degli studi di Salerno
Fondamenti di visione artificiale e biometria

Abstract

L'indagine sulle immagini dei volti umani è onnipresente nell'ambito dell'image processing. Gli approcci tradizionali sono legati all'identificazione e alla verifica del volto ma stanno emergendo molte altre aree, come la stima dell'età e dell'espressione, l'analisi della somiglianza tra volti e il riconoscimento automatico della parentela.

Questo progetto ha l'obiettivo di costruire un modello, utilizzando il transfer learning, in grado di discriminare, in base alle immagini del loro viso, tra persone imparentate e persone non imparentate e distinguere il tipo di relazioni esistenti tra essi.

Inizialmente, è stato addestrato il modello VGG16 con pesi resnet50 sul dataset Families In the Wild (FIW) [1].

In seguito, utilizzando questo modello addestrato è stato creato ed addestrato un secondo modello utilizzando il dataset KinFaceW-I [2] che include immagini frontali di volti inespressivi e sorridenti di coppie di parenti: madre-figlia, madre-figlio, padre-figlio, padre-figlia.

Infine, sono stati effettuati esperimenti per verificare quale parte del volto, superiore o inferiore, contenga informazioni più significative per il riconoscimento della parentela.

1. Riconoscimento automatico delle parentele

Riconoscere la somiglianza tra due persone è un'abilità intrinseca di ogni essere umano.

Ciascuno di noi, trovandosi di fronte all'immagine di due fratelli, di una madre col proprio bambino o di un nonno col proprio nipote, è portato ad individuare quelle caratteristiche fisiche che mettono in evidenza il legame di parentela che li unisce.

Tuttavia, se il problema del riconoscimento delle parentele da immagini facciali è facilmente risolvibile per gli esseri umani, lo stesso non si può dire dal punto di vista della Computer Vision. La difficoltà principale è dovuta al fatto che non siamo in grado di giustificare il motivo per cui pensiamo che tra due persone esista un legame di parentela.

Infatti, è impossibile determinare un insieme di caratteristiche fisiche associabili ad un grado di parentela e, quindi, l'idea di trasferire direttamente la nostra conoscenza ad un algoritmo di riconoscimento risulta impraticabile.

Lo studio dell'esistenza di legami di parentela partendo da immagini facciali potrebbe essere considerato superfluo considerando l'incredibile accuratezza raggiunta con i test del DNA nel riconoscimento delle parentele: 99,99% in caso di parentela, 100% in caso di non parentela.

Tuttavia, l'utilizzo del test del DNA è molto limitato, per motivi di privacy, soldi, tempo e collaborazione. [3]

Siccome studi biologici hanno dimostrato che l'immagine contiene importanti informazioni sulla persona stessa (identità, sesso, età ed etnia) e importanti indizi per misurare la somiglianza genetica tra due individui, negli ultimi anni sempre più studiosi si sono avvicinati alla **Visual**

Kinship Verification, un'area di ricerca che si pone come obiettivo quello di verificare automaticamente l'eventuale esistenza di un legame di parentela tra due persone, partendo dalle immagini dei loro volti (Figura 1).

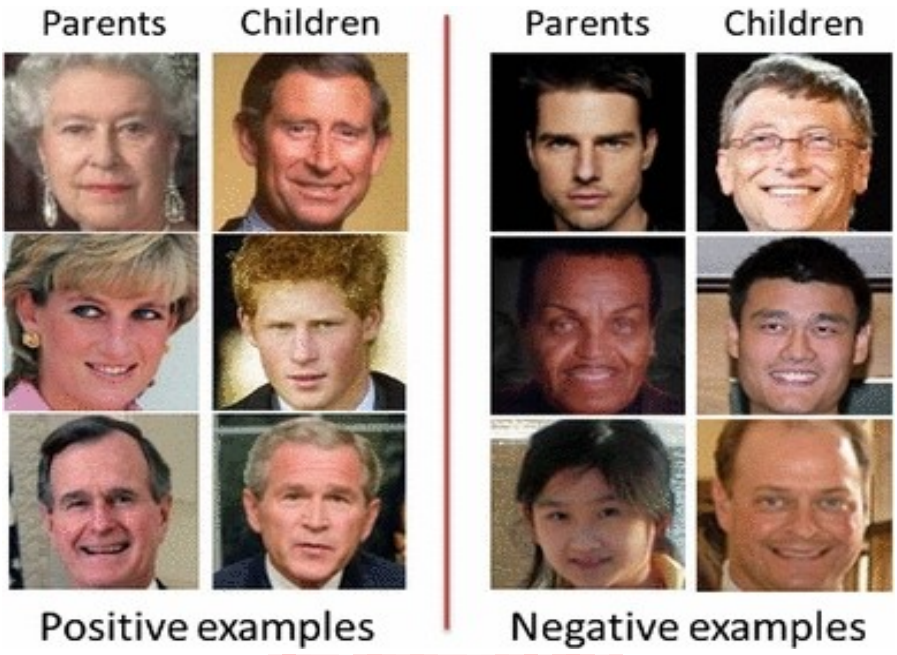


Figura 1 – Esempi di parentela e non parentela

La ricerca nel mondo del riconoscimento automatico delle parentele è ancora agli inizi e, di conseguenza, i risultati ottenuti fino ad ora non possono certamente essere considerati affidabili.

Le applicazioni pratiche che potrebbero trarre benefici da questi studi sono molteplici e toccano aree diverse: dall'analisi dei social media, alla gestione automatica degli album fotografici, fino alla ricerca dei bambini scomparsi e la gestione di problemi di sicurezza nazionale.

2. Dati utilizzati

Per l'addestramento della prima rete, su cui, in seguito, è stato applicato il transfer learning, sono stati utilizzati i volti del dataset Families In the Wild (FIW), il database di immagini più ampio e completo per il riconoscimento automatico della parentela.

Questo set di dati è ottenuto da immagini pubblicamente disponibili di celebrità per un totale di oltre tredicimila immagini.

Per l'addestramento della rete durante il transfer learning è stato utilizzato KinFaceW-I.

Il dataset è composto da un totale di duemila immagini suddivise come riportato nella tabella che segue.

Per ogni categoria ci sono duecentocinquanta coppie positive e duecentocinquanta coppie negative.

Padre - Figlio	Padre - Figlia	Madre - Figlia	Madre - Figlio
500 immagini	500 immagini	500 immagini	500 immagini

Per l'addestramento del nuovo modello, le immagini del dataset sono state suddivise come riportato nella tabella che segue.

Fase	Numero coppie
Addestramento	1200 coppie
Validazione	400 coppie
Testing	400 coppie

Inoltre, è stato effettuato il processo di image augmentation, una tecnica utilizzata per espandere artificialmente il set di dati poiché esso conteneva pochi campioni di dati per le dimensioni della rete da addestrare. In particolare, alla stessa immagine è stato applicato un insieme di trasformazioni permettendo un notevole ampliamento dei dati su cui effettuare l'addestramento.
Le trasformazioni utilizzate sono elencate di seguito e alcune di esse sono mostrate nella figura 2:

- Horizontal Shift
- Vertical Shift
- Brightness
- Zoom
- Channel Shift
- Horizontal Flip
- Vertical Flip
- Rotation
- Fill Mode
- Colorjitter
- Aggiunta di rumore
- Applicazione di filtri (Gaussiano, mediana, blur)

Infine, per testare l'accuratezza del modello in ambito reale, sono state utilizzate immagini di volti raccolte da puntate de "I soliti ignoti" utilizzando frontal face detector.

Figura 2 – Esempi di trasformazione (Originale, Horizontal shift, Vertical shift, Brightness, Zoom, Channel shift)

3. Tecnologie utilizzate

L'intero progetto è stato sviluppato con Python 3.9, utilizzando in particolar modo le librerie **Scikit-Learn**, **Keras**, **OpenCV** e **Pandas**.

Scikit-Learn è una libreria open-source di Python creata appositamente per il machine learning.

Racchiude diversi algoritmi per la classificazione, la regressione e il clustering, ed è stata progettata specificatamente per l'interoperabilità con le librerie numeriche e scientifiche **NumPy** e **SciPy**.

OpenCV (Open Source Computer Vision Library) è una libreria di software di visione artificiale e apprendimento automatico open source.

Essa contiene più di 2500 algoritmi ottimizzati che permettono di rilevare e riconoscere volti, identificare oggetti, classificare le azioni umane nei video, tracciare i movimenti della telecamera, tracciare oggetti in movimento e molto altro. [4]

Tutti i componenti necessari per l'implementazione di una rete (layers, funzioni di attivazione, ottimizzatori ecc...) sono espressi come moduli stand-alone che l'utente può combinare per la creazione e l'addestramento dei modelli e sono forniti da Keras. [5]

Oltre a Scikit-Learn, OpenCV e Keras, sono state di particolare rilevanza per l'implementazione del progetto le librerie Numpy, per la memorizzazione delle features in array e matrici multidimensionali, Pandas, per la lettura e la gestione dei file Excel e csv, e la libreria **Matplotlib**, per la realizzazione di grafici e la visualizzazione dei dati. [6][7]

La maggior parte degli esperimenti sono stati eseguiti sulla piattaforma **Colab**.

4. Scelta della rete

La prima decisione affrontata durante lo svolgimento del progetto è stata quella del tipo di rete neurale da utilizzare nel nostro sistema di riconoscimento delle parentele.

Le reti neurali sono dei modelli matematici complessi in grado di apprendere informazioni, sfruttando meccanismi simili a quelli dell'intelligenza umana.

In breve, è possibile vedere una rete neurale come una sorta di "black box", che riceve in ingresso uno o più input, li elabora e fornisce in output una o più etichette di classificazione (labels) a seconda del problema da risolvere. Il più grande punto di forza delle reti neurali sta nel fatto che queste non debbano essere esplicitamente programmate per svolgere un particolare compito, ma devono essere opportunamente addestrate mediante una serie di esempi della realtà da modellare.

Di particolare rilevanza nello sviluppo del progetto sono state le reti convoluzionali siamesi, una particolare tipologia di rete creata appositamente per gestire due input visuali come immagini e video.

L'architettura delle SNN's si ispira alla corteccia visiva animale; infatti, gli strati di convoluzione consentono di suddividere l'immagine in una serie di frammenti sovrapposti, di analizzarli e di individuarne le principali caratteristiche. In particolare, utilizza gli stessi pesi mentre lavora sui dati in input e successivamente restituisce il quadrato delle differenze come combinazione delle feature.

L’idea iniziale di costruire una rete ad hoc per riconoscere le eventuali parentele ed effettuarne il training da zero è stata immediatamente abbandonata.

E’ infatti molto raro che una rete implementata ex-novo superi le prestazioni di una delle tante architetture disponibili online, già allenate e testate su diversi database.

Inoltre, effettuare il training da zero di una rete neurale è un’operazione decisamente non banale per le enormi capacità computazionali.

Per tutti i motivi elencati in precedenza, abbiamo deciso di seguire una strada alternativa, ossia sfruttare la tecnica del transfer learning.

Questa pratica, molto diffusa nell’ambito del deep learning, consiste nel prendere una rete neurale preesistente, addestrata a svolgere una determinata attività, ed utilizzarla per risolvere un problema differente, ma correlato.

Il trasferimento della conoscenza può avvenire tramite due diverse strategie: la prima prevede l’utilizzo della rete di base come un estrattore di features; la seconda strada, invece, consiste nell’effettuare il fine-tuning della rete preesistente, proseguendo l’allenamento della stessa con il nuovo dataset.

Perché il transfer learning abbia successo, è necessario che le features estratte dalla rete originale siano quanto più generiche possibile.

Nel nostro caso, trattandosi di un problema di kinship verification, riguardante l’analisi facciale, si è scelto di utilizzare come base-network una rete pre-allenata su un database di face recognition.

In particolare, abbiamo deciso di testare le prestazioni delle due differenti architetture della rete VGG-Face (VGG-16 e ResNet-50), pre-allenate

sull'omonimo dataset, entrambe progettate e messe a disposizione dal Visual Geometry Group dell'università di Oxford.

4.1 VGG-16

VGG-16 è un modello di rete convoluzionale neurale proposto per la prima volta nel 2014 da K. Simonyan e A. Zisserman dell'Università di Oxford all'interno dell'articolo "Very Deep Convolutional Networks for Large-Scale Image Recognition"[8]. L'architettura di questa CNN come mostrato nella figura 3, è molto semplice, in quanto utilizza solamente una serie di strati convoluzionali (Figura 4) 3x3 impilati l'uno sull'altro, per un totale di 16 layers allenabili, intervallati da alcuni strati di max pooling, che gestiscono la riduzione dei volumi. Nello specifico, la topologia di VGG-16 è composta dai seguenti strati:

1. Strato convoluzionale (attivazione Relu) con 64 filtri (Conv1)
2. Strato convoluzionale con 64 filtri + max pooling
3. Strato convoluzionale con 128 filtri (Conv2)
4. Strato convoluzionale con 128 filtri + max pooling
5. Strato convoluzionale con 256 filtri
6. Strato convoluzionale con 256 filtri (Conv3)
7. Strato convoluzionale con 256 filtri + max pooling
8. Strato convoluzionale con 512 filtri
9. Strato convoluzionale con 512 filtri (Conv4)
10. Strato convoluzionale con 512 filtri + max pooling
11. Strato convoluzionale con 512 filtri
12. Strato convoluzionale con 512 filtri (Conv5)

13. Strato convoluzionale con 512 filtri + max pooling
14. Strato fully-connected con 4096 nodi (fc6)
15. Strato fully-connected con 4096 nodi (fc7)
16. Strato di output con attivazione softmax con 2622 nodi

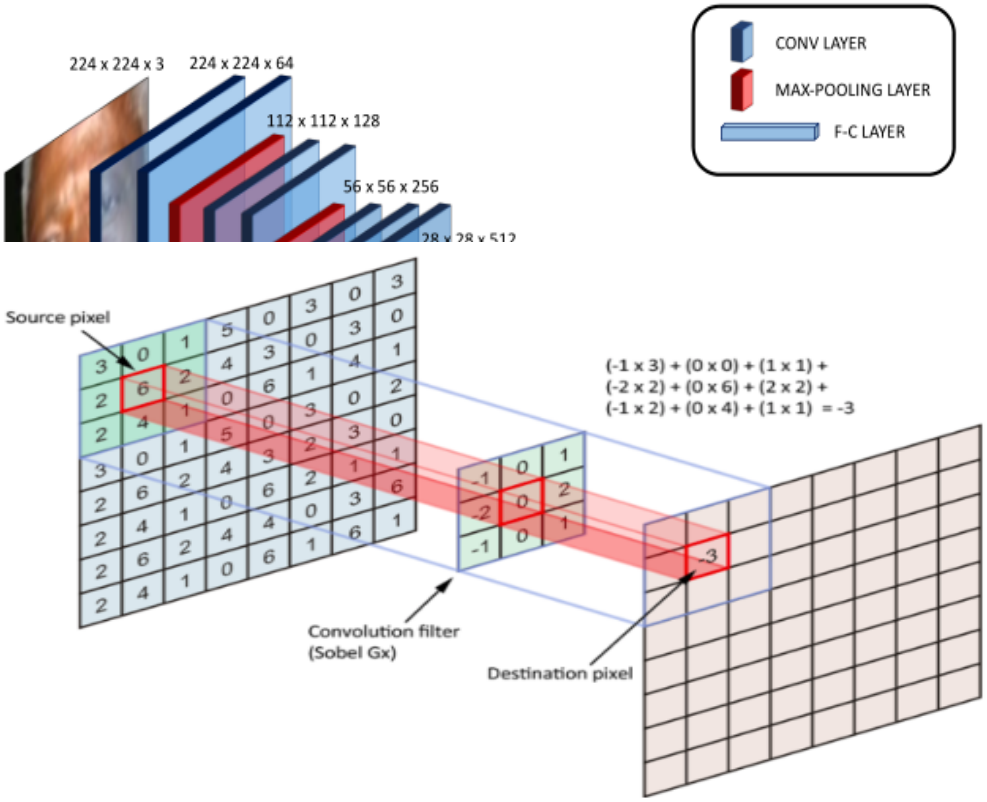


Figura 4 - Esempio di convoluzione

4.2 ResNet-50

ResNet-50 è una rete neurale convoluzionale profonda 50 strati strutturata nel seguente modo (Figura 5) [9]:

- Un layer di convoluzione con una dimensione del kernel di 7 * 7 e 64 kernel diversi, tutti con uno stride di dimensione 2 che ci danno 1 strato.
- Strato di max pooling con dimensione dello stride di 2.
- Nella successiva convoluzione c'è un kernel 1 * 1, seguito da un kernel 3 * 3 e infine un kernel 1 * 1. Questi tre strati vengono ripetuti in totale 3 volte, dando in totale 9 strati.
- Successivamente vi è un kernel 1 * 1, un kernel di 3 * 3, e alla fine un kernel di 1 * 1. Questo passaggio è ripetuto 4 volte creando in totale 12 strati.
- Un kernel 1 * 1, due kernel 3 * 3, e un ultimo kernel 1 * 1. Questo viene ripetuto 6 volte per un totale di 18 strati.
- Un kernel 1 * 1, con altri due kernel 3 * 3, e un kernel 1 * 1. Questo viene ripetuto 3 volte per un totale di 9 strati.
- Infine, vi è un avarage pool, uno strato pienamente connesso contenente 1000 nodi e una funzione softmax, tutto questo forma 1 strato.

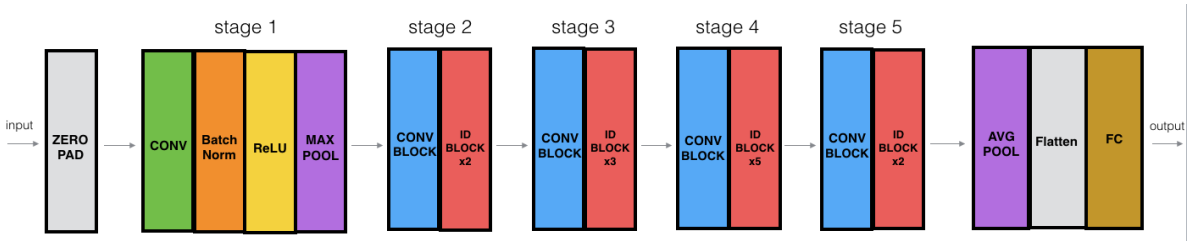


Figura 5 – Architettura ResNet50

5. Esperimenti

Il nostro lavoro progettuale per il conseguimento degli obiettivi assegnati è costituito in diversi passi: preparazione del modello di rete, addestramento del modello tramite Transfer Learning, Discussione dei risultati ottenuti.

5.1 Preparazione del modello di rete

Nella prima fase progettuale il problema da risolvere era quello del riconoscere l'esistenza della parentela tra due soggetti.

Per fare ciò avevamo a disposizione varie implementazioni di **reti siamesi**. Il nostro gruppo ha deciso di portare avanti due delle sopra citate, per fare in modo di avere un riscontro qualitativo su quale raggiungesse i migliori risultati.

Le selezionate sono: la **baseline** che utilizza una rete di tipo **ResNet50**, formata da molti strati convoluzionali e, di conseguenza, una maggiore pesantezza computazionale, e la **VGGFace** basata su una rete del tipo **VGG16** molto più piccola della precedente.

5.1.1 Configurazione delle reti utilizzate

Da questo punto abbiamo effettuato un gran numero di esperimenti alla ricerca dell'accuratezza ottima.

Una volta reso compatibile il codice con la nostra infrastruttura abbiamo cominciato con i primi addestramenti ottenendo un'accuratezza di circa il 52% per entrambe le reti.

Dopo varie modifiche a parametri come: epoche totali, step per epoca, validation step e accuratezza della funzione di ottimizzazione del modello, sono stati ottenuti i risultati riportati nella tabella che segue.

Parametri	**Baseline**	**VggFace**
Epoche	250	112
Step per epoca	120	222
Step di validazione	100	100
Ottimizzatore	Adam con precisione (0.00001)	Adam con precisione (0.00001)
Tempo addestramento	10 ore	4 ore e mezza
Accuratezza modello	82%	75%

Successivamente, i due modelli sono stati testati sul dataset **recognizing-faces-in-the-wild** e, una volta ottenuto il file csv con i

risultati della predizione ed averlo caricato su Kaggle, hanno ottenuto un'accuratezza di test del 78% per la VGGFace e del 52% per la Baseline.

Nonostante il modello Baseline abbia ottenuto risultati di gran lunga superiori durante l'addestramento, al momento del test l'accuratezza è peggiorata e, visionando i grafici, abbiamo scoperto che il problema era dovuto all'overfitting (Figura 6) causato, molto probabilmente, dalla grandezza della rete rispetto al numero di immagini date in input.

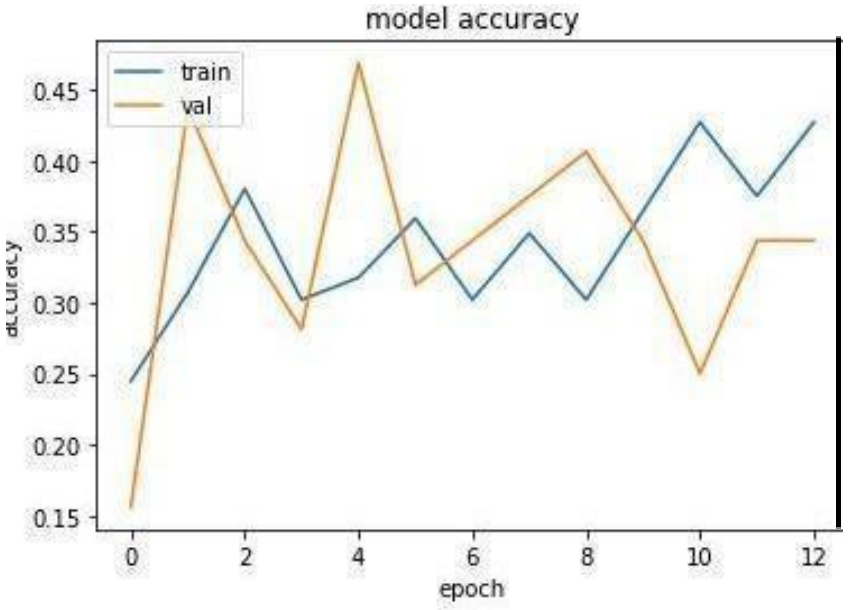


Figura 6 – Esempio di overfitting

5.2 Addestramento del modello tramite transfer learning

A questo punto abbiamo deciso di procedere alla seconda fase dell'addestramento utilizzando la VGGFace data la sua maggiore accuratezza.

La seconda fase progettuale consisteva nel risolvere oltre il problema sopracitato anche quello del riconoscimento del tipo di parentela tra le classi: Mamma-Figlia, Padre-Figlia, Mamma-Figlio, Padre-Figlio e non parentela.

Per fare ciò è stato utilizzato il Transfer Learning quindi, utilizzando il precedente modello, sono stati eliminati gli ultimi strati pienamente connessi e sono stati aggiunti nuovi strati e un nuovo strato di output per rispettare il numero di classi.

5.2.1 Addestramento del modello sul problema a 4 classi

Inizialmente, è stato risolto il problema su quattro classi escludendo, quindi, la classe di non parentela. In questo contesto è stato effettuato un **fine tuning** sugli ultimi strati convoluzionali ossia, veniva caricato il precedente modello con i rispettivi pesi e venivano addestrati solamente i nuovi strati pienamente connessi.

Dopo varie combinazioni dei parametri del modello abbiamo raggiunto un'accuratezza del 52%.

5.2.2 Addestramento del modello sul problema a 5 classi

In seguito, siamo passati alla risoluzione del problema a cinque classi che ci è stato assegnato. Su quest'ultimo è stato effettuato il fine tuning degli ultimi strati convoluzionali ottenendo un'accuratezza intorno al 50%. Successivamente, è stato diviso il numero di strati del modello a metà effettuando il **freeze** sulla prima metà e il **fine tuning** sulla seconda metà. Anche in questo caso non ci sono stati evidenti miglioramenti.

5.2.3 Problemi riscontrati

Sbilanciamento della batch

Dopo ulteriori analisi e prendendo visione dei grafici dei vari addestramenti, abbiamo notato che una batch in cui metà coppie avevano una relazione e la restante metà non l'avevano portava il modello a riconoscere solo le **non relazioni** provocando uno sbilanciamento dello stesso.

Per risolvere questo problema abbiamo posto il limite di coppie senza relazione a un 1/6 della dimensione della batch.

Dopo questa modifica, il modello era più bilanciato ma continuava ad avere accuratezza molto bassa e problemi di overfitting.

Overfitting

Siamo passati dunque alla risoluzione del problema dell'overfitting procedendo con tecniche di **image augmentation**. Inizialmente, sono state effettuate semplici trasformazioni casuali in openCV sulle immagini che venivano date in input al modello, ottenendo un lieve miglioramento di accuratezza ma senza risolvere completamente il problema.

In seguito, è stato aumentato il numero di immagini date in input al modello effettuando un totale di dodici trasformazioni per ogni singola immagine. Con questo accorgimento l'accuratezza del test ha raggiunto un picco del 56%. Come ulteriore risoluzione all'overfitting sono stati aggiunti dei nuovi strati **dropout** e regolarizzato i densi già aggiunti precedentemente con degli **L2 regularizer.**

Il loro compito è trovare un compromesso tra pesi piccoli minimizzando la funzione di costo, utilizzando un parametro λ chiamato **tasso di regolarizzazione** che serve per determinare il bilanciamento di questo compromesso: quando λ sarà piccolo, la funzione costo sarà minimizzata, mentre quando sarà grande, saranno utilizzati pesi più piccoli.

Accuratezza del modello bassa

Dopo ulteriori prove, l'accuratezza rimaneva bassa e abbiamo ipotizzato che la causa fosse la semplicità intrinseca del modello rispetto al problema che stavamo affrontando. Dunque, siamo passati a provare il transfer learning sul modello **Baseline**, precedentemente ri-addestrato con un'accuratezza del 70%, ottenendo un accuratezza sul transfer learning oscillante tra il 58 e il 62%.

5.2.4 Soluzione finale

Volendo migliorare ulteriormente l'accuratezza, come ultimo tentativo si è proceduto all'utilizzo del modello come semplice estrattore di feature, ovvero senza effettuare né freeze né fine tuning.

Essendo questa un'operazione molto onerosa, siamo tornati sulla **VGGFace.** Facendo delle piccole modifiche, rendendola più profonda con l'aggiunta di tre nuovi layer pienamente connessi. Inoltre, è stata utilizzata una batch che conteneva tutte le immagini di training contemporaneamente invece di dividerle in batch di dimensioni più piccole, ottenendo come risultato finale un'accuratezza del modello del 69 % e una accuratezza di test del 72% (Figura 7). Parametri architettura riassunti nella tabella sottostante.

Parametri	**Valori**
Epoche	30
Step per epoca	200
Step di validazione	100
Ottimizzatore	Adam con precisione 0.00001
Tempo di addestramento	3 ore
Accuratezza del modello	69%

Accuratezza test	72%

Grafici e matrice di confusione del modello finale

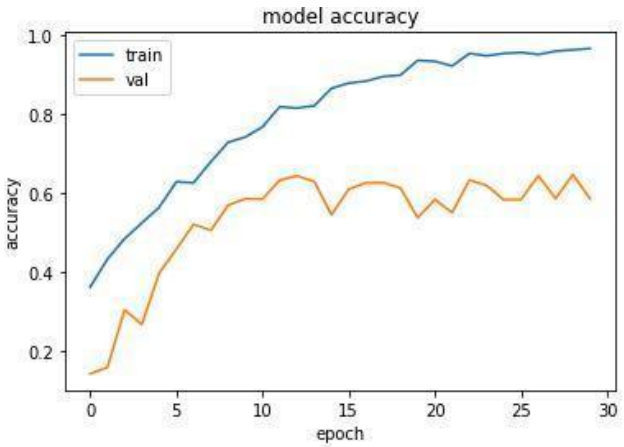


Figura 7 – Accuratezza del modello

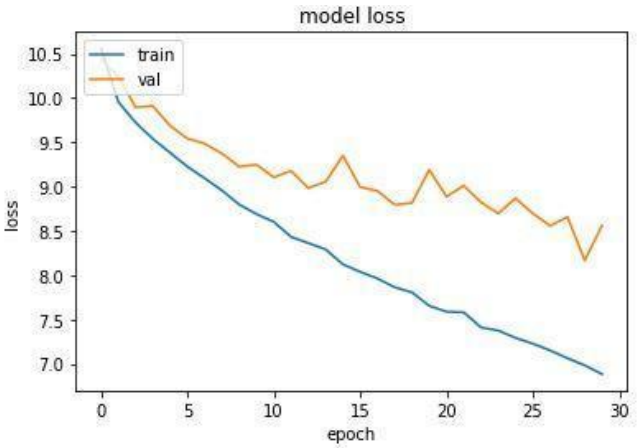


Figura 8 – Loss del modello

	FD	FS	MD	MS	NK
FD	26	11	5	1	7
FS	8	36	1	1	4
MD	0	1	38	3	8
MS	4	5	13	19	9
NK	13	6	14	8	158

Figura 9 – Matrice confusione finale

5.2.5 Test e discussione dei risultati ottenuti

Come ultima parte del progetto è stato testato il modello ottenuto con volti presi dal programma televisivo “I soliti ignoti”.

Messa in evidenza dei tratti somatici salienti

Inizialmente i risultati non sono stati soddisfacenti, siamo andati quindi a diminuire le porzioni dei volti dati input eliminando le parti meno significative (capelli e orecchie). Ottenendo, però, un notevole peggioramento dell'accuratezza del test perché le immagini su cui era addestrato precedentemente il modello erano ricche di informazioni.

Adattamento delle foto al dataset di partenza

Abbiamo poi adattato tutte le immagini per avvicinarci il più possibile a quelle del dataset di partenza, ottenendo un'accuratezza leggermente più alta. In questo caso la rete riusciva ad individuare la parentela tra il parente misterioso e parente effettivo, ma riconosceva anche persone non effettivamente imparentate. Nella tabella sottostante sono evidenziati i risultati ottenuti.

	Padre - figlia	Padre - figlio	Madre – figlia	Madre – figlio
Accuratezza test	38%	38%	75%	25%

	FD	FS	MD	MS	NK
FD	0	0	0	0	0
FS	0	0	0	0	0
MD	0	0	0	0	0
MS	0	0	1	0	0
NK	0	5	0	1	1

	FD	FS	MD	MS	NK
FD	0	0	0	0	0
FS	1	0	0	0	0
MD	0	0	0	0	0
MS	0	0	0	0	0
NK	2	0	3	0	2

Figura 12 - Matrice di confusione Padre Figlio

	FD	FS	MD	MS	NK
FD	0	0	0	0	0
FS	0	0	0	0	0
MD	0	0	1	0	0
MS	0	0	0	0	0
NK	0	0	2	0	5

	FD	FS	MD	MS	NK
FD	0	0	0	0	1
FS	0	0	0	0	0
MD	0	0	0	0	0
MS	0	0	0	0	0
NK	0	3	0	1	3

Figura 13 - Matrice di confusione Padre Figlia

5.2.6 Test e discussione dei risultati ottenuti

Dall'analisi dei risultati ottenuti si è riscontrata una maggiore capacità della rete, nel riconoscere persone appartenenti allo stesso genere. Molte volte, infatti, la parentela e la relazione corretta vengono individuate ma del sesso opposto. E così proprio come gli esseri umani ha più capacità nel discriminare l'esistenza della parentela piuttosto che il tipo di parentela.

5.3 Individuazione parte del volto più significativa per il riconoscimento della parentela

Come parte facoltativa del progetto è stato verificato quale parte del volto, superiore o inferiore, fosse più significativa per il riconoscimento della parentela.

Per fare ciò abbiamo utilizzato due metodi:

1. Divisione fissata delle immagini
2. Divisione tramite landmark facciali.

Per questa parte di progetto è stato utilizzato lo stesso modello usato per la prima parte del progetto e gli stessi parametri di addestramento. Unica variazione consisteva nell'input che, in questo caso, è solo una parte del volto costruite come descritto nei paragrafi successivi.

5.3.1 Divisione fissata delle immagini

Ogni immagine da 64×64 è stata divisa in due parti lungo la sua lunghezza creando due immagini da 32×64. In seguito, sono state ridimensionate per rispettare la dimensione dell'input del modello.

A causa del metodo di divisione utilizzato, che non teneva in considerazione la locazione precisa degli elementi del volto, della differenza della posa dei volti e delle differenze anatomiche dei diversi individui il risultato del taglio è variabile, ma, nella maggior parte dei casi, la parte superiore del volto conteneva elementi con maggiori informazioni per il riconoscimento della parentela: zona perioculare, parte superiore del naso, parte superiore delle orecchie e i capelli.

Mentre la parte inferiore del volto conteneva un numero inferiore di informazioni: la restante metà del naso e delle orecchie, bocca e mento.

Figura 14 – Esempio taglio fissato orizzontale

Utilizzando la parte inferiore del volto è stata ottenuta una precisione del 34%. Inoltre, analizzando i risultati dei test effettuati su di esso si nota uno sbilanciamento del modello verso la **non relazione**.

	FD	FS	MD	MS	NK
FD	2	0	0	18	30
FS	4	2	0	9	35
MD	4	1	1	8	36
MS	8	1	0	7	34
NK	11	7	0	40	141

Figura 15 - Matrice di confusione parte inferiore del volto

Utilizzando la parte superiore del volto è stata ottenuta una precisione del 40%. Inoltre, analizzando i risultati dei test effettuati su di esso si nota che la rete riconosce l'effettiva esistenza della parentela, non riuscendo però effettivamente a individuare la classe di appartenenza.

	FD	FS	MD	MS	NK
FD	1	24	4	0	21
FS	0	32	2	2	14
MD	0	28	10	0	12
MS	0	29	4	0	17
NK	3	86	21	1	88

Figura 16 - Matrice di confusione parte superiore del volto

5.3.2 Divisione tramite landmark facciali

Per ogni immagine è stato effettuato un ritaglio utilizzando delle maschere che in combinazione con l'utilizzo dei landmark facciali permettevano l'estrazione delle zone del volto interessate.

Come si può notare dalle immagini utilizzando questo tipo di ritaglio estraiamo solo le caratteristiche che compongono le singole parti del volto (Figura 17).

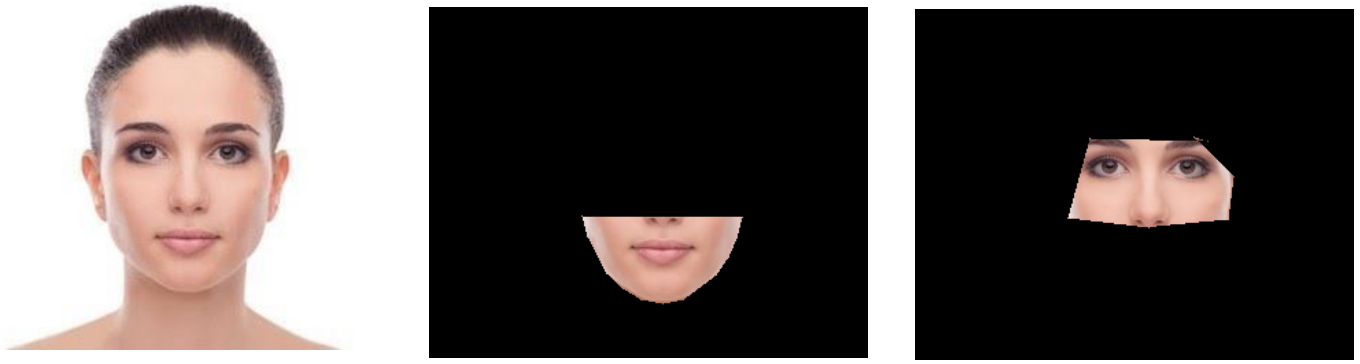


Figura 17 – Esempio taglio con landmark facciali

Il problema del modello addestrato con questo tipo di ritaglio è che, durante la fase di testing sui volti de "I soliti ignoti" in cui le immagini sono più dettagliate e di dimensioni differenti, si otteneva un'accuratezza minore dovuta alla peggiore qualità delle immagini con cui è stato addestrato il modello. Ciò rendeva meno distinguibili gli elementi discriminatori delle stesse. L'accuratezza ottenuta è stata del 30% per la parte superiore e 21% per l'inferiore. Sotto sono rappresentate le relative tabelle di covarianza ottenute.

	FD	FS	MD	MS	NK
FD	8	19	12	11	0
FS	12	11	19	7	1
MD	10	13	17	9	1
MS	13	7	17	11	2
NK	37	58	57	16	31

Figura 18 - Matrice di confusione parte superiore del volto

	FD	FS	MD	MS	NK
FD	3	0	24	1	22
FS	5	0	28	0	17
MD	6	0	25	1	18
MS	3	0	24	0	23
NK	17	2	85	3	92

Figura 19 - Matrice di confusione parte inferiore del volto

5.3.3 Discussione dei risultati ottenuti

In entrambi i casi l'accuratezza è stata minore rispetto ai modelli addestrati utilizzando il volto intero ma, elemento importante da constatare, è che la nostra rete, proprio come l'essere, umano discrimina e individua la parentela utilizzando per lo più le caratteristiche discriminanti della parte superiore del volto. Inoltre, le classi che sono state maggiormente riconosciute in entrambi i casi sono: Madre-Figlia e Padre-Figlio, proprio come il test sul volto intero.

6. Conclusioni

In questo progetto è stato affrontato il problema del riconoscimento automatico delle parentele dal punto di vista della computer vision.
Durante tutto l'arco della durata del progetto sono stati eseguiti oltre centoventidue differenti esperimenti, che ci hanno permesso di entrare in contatto con molte delle principali strategie riguardanti il mondo del riconoscimento facciale, delle reti neurali convoluzionali e, più in generale, del machine learning e del deep learning. Il contributo di questo progetto ha confermato la supremazia del fine-tuning, rispetto a tutte le altre strategie per la kinship verification e il riconoscimento della parentela.
I risultati raggiunti fino ad ora nell'ambito del riconoscimento automatico delle parentele da immagini facciali non hanno ancora consentito lo sviluppo di applicazioni reali di questo tipo, ma, allo stesso tempo, lasciano un ampio margine di miglioramento per i ricercatori. Crediamo che le challenge della face recognition possano portare a uno sviluppo maggiore per quest'area di ricerca, con la speranza di vederne presto sfruttate appieno tutte le sue potenzialità.

Tabella riassuntiva dei risultati ottenuti

Nome	Epoche	Step di validazione	Step per epoca	Accuratezza Modello	Accuratezza Test	Accuratezza Test ignoti
VggFace	**112**	**222**	**100**	**75%**	**78%**	**-**
TF VggFace	**30**	**100**	**200**	**68%**	**72%**	**45%**
TF Taglio orizzontale superiore	**30**	**100**	**200**	**33%**	**38%**	**30%**
TF Taglio orizzontale inferiore	**30**	**100**	**200**	**30%**	**34%**	**25%**
TF Taglio landmark superiore	**30**	**100**	**200**	**27%**	**30%**	**20%**
TF Taglio landmarki nferiore	**30**	**100**	**200**	**18%**	**21%**	**18%**

Bibliografia

[1] https://web.northeastern.edu/smilelab/fiw/
[2] https://www.kinfacew.com/datasets.html
[3] DDC
[4] OpenCV
[5] Keras
[6] Numpy
[7] Pandas
[8] VGG16
[9] ResNet50