POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Riconoscimento automatico delle parentele

Tecniche di Machine Learning e Deep Learning per affrontare la sfida della Visual Kinship Verification.

Relatore
prof. Andrea Bottino

Candidato
Matteo Vaccari

Anno accademico 2017-2018

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

1.1 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, in alcuni casi, facilmente risolvibile per gli esseri umani, lo stesso non si può dire dal punto di vista della Computer Vision. La difficoltà principale deriva dal fatto che, molto spesso, non siamo in grado di giustificare il motivo per cui pensiamo che due persone abbiano un legame sanguigno. È impossibile determinare un insieme di connotati fisici strettamente associabili ad uno specifico grado di parentela e, di conseguenza, l'idea di trasferire direttamente la nostra conoscenza ad un algoritmo di riconoscimento risulta impraticabile.

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), lo studio di questo stesso argomento partendo da immagini facciali potrebbe quasi essere considerato superfluo. Nella pratica, tuttavia, l'utilizzo del test del DNA è molto limitato, per motivi di privacy, soldi, tempo e collaborazione. Tutto ciò ha spinto i ricercatori a trovare delle strade alternative, che consentissero di raggiungere risultati altrettanto soddisfacenti. Studi biologici [1] hanno dimostrato che l'immagine del volto di una persona non contiene soltanto importanti informazioni sulla persona stessa (come identità, sesso, età ed etnia), ma anche alcuni indizi interessanti per misurare la somiglianza genetica tra due individui. Motivati da queste scoperte, negli ultimi anni sempre più studiosi si sono avvicinati alla Visual Kinship Verification, un'area di ricerca relativamente nuova, 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.

L'analisi facciale è da sempre un'importante materia di studio nell'ambito della Computer Vision. In particolare, face detection e face recognition sono i due campi di ricerca che, a partire dai primi anni '70, hanno attirato maggiormente l'attenzione degli studiosi. I motivi di questa popolarità sono da ricercare soprattutto negli innumerevoli risvolti commerciali di queste tecnologie, sia nell'ambito della sicurezza e della sorveglianza, sia in settori quali fotografia, telefonia e social media. Più recentemente, l'attenzione si è spostata anche su nuove tematiche correlate all'analisi del volto umano come gender classification, age estimation, facial expression recognition ed ethnicity classification. La Visual Kinship Verification è quindi solamente l'ultima di una serie di sfide, che sfruttano le informazioni facciali con lo scopo di ricavare informazioni sulla persona (fig 1.1).

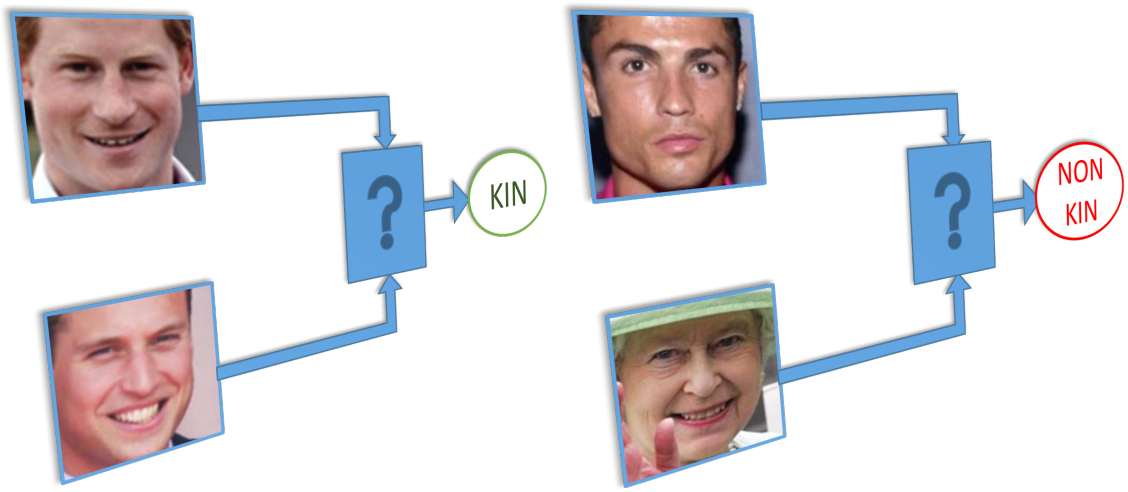

Figura 1.1. Riconoscimento automatico delle parentele.

Il recente sviluppo delle tecniche di deep learning e, in particolare, l'introduzione delle reti convoluzionali neurali (CNNs) hanno permesso di raggiungere risultati incredibilmente accurati nella maggior parte dei problemi citati in precedenza. Tuttavia, le informazioni genetiche sulle parentele sono, nel dominio visivo, molto meno discriminative rispetto a quelle dei problemi più convenzionali. La grande variabilità dei fattori che influenzano le apparenze facciali tra i membri di diverse famiglie rende

questa sfida più complicata delle precedenti, ma allo stesso tempo più stimolante. La ricerca nel mondo del riconoscimento automatico delle parentele è ancora alle prime armi e, di conseguenza, i risultati ottenuti fino ad ora non possono certamente essere considerati affidabili: non esiste infatti, ad oggi, alcuna applicazione reale che utilizzi un sistema di automatic visual kinship verification. 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. Un esempio significativo è rappresentato dall'attentato alla maratona di Boston del 2103, quando due fratelli Ceceni (fig 1.2) fecero esplodere due bombe durante lo svolgimento della gara, causando la morte di 3 persone, oltre a più di 260 feriti. In quel caso, un sistema di riconoscimento automatico delle parentele avrebbe permesso di identificare molto più velocemente i due attentatori, riducendo notevolmente il tempo delle indagini. Un'altra applicazione pratica potrebbe essere l'installazione di algoritmi di kinship verification all'interno delle telecamere che si occupano della sorveglianza dei confini: un sistema di questo tipo permetterebbe di verificare la parentela tra un bambino e il proprio accompagnatore adulto, complicando in tal modo il traffico illegale di minori. Inoltre, in molti Stati, tra cui gli U.S.A, un parametro importante per poter ottenere lo status di rifugiato è proprio quello di avere almeno un parente nella Nazione ospitante. Attualmente vengono utilizzati test del DNA per verificare o confutare tali richieste; un sistema di verifica automatica della parentela basato sulle immagini permetterebbe di ridurre notevolmente i costi e i tempi di tale procedura.

Figura 1.2. L'immagine dei due attentatori ripresa da una telecamera di sicurezza.

Dal 2017 la Northeastern University di Boston organizza ogni anno la Recognizing Families in the Wild (RFIW) Challenge [2], competizione di kinship verification e family classification nata con l'obiettivo di dare un'ulteriore spinta alla ricerca in questo settore. L'introduzione del database Families In the Wild (FIW) [3], il più grande e completo nell'ambito del riconoscimento delle parentele, ha infatti portato questo problema ad una dimensione decisamente superiore rispetto al passato (da 2000 coppie di immagini a 644.000 per la kinship verification, da 101 famiglie a 1000 per la family classification), permettendo così ai ricercatori di sfruttare appieno le nuove tecniche di apprendimento data-driven. Pur non avendo partecipato ufficialmente alla competizione, per una questione di tempistiche, abbiamo utilizzato i dati forniti dalla RFIW Challenge 2018, con lo scopo di fornire una nostra soluzione per per questo problema. In questa tesi, in particolare, vengono proposti due differenti approcci: il primo basato sull'estrazione di vettori di caratteristiche (features) da reti convoluzionali neurali pre-allenate, mentre il secondo incentrato sul fine-tuning delle stesse CNNs sul dataset FIW, tramite un'implementazione Siamese delle reti.

Capitolo 2

Background

In questo capitolo viene fornita una breve panoramica su tutto ciò che riguarda il mondo del riconoscimento automatico delle parentele. Nella prima parte vengono descritti, in modo sintetico, i principali approcci testati dai ricercatori in questi anni e i rispettivi database di riferimento. In seguito, vengono illustrati i diversi task affrontati dalla competizione *Recognizing Families in The Wild*, da cui nasce l'idea di questa tesi, con particolare risalto al database *FIW* e agli esperimenti condotti dagli organizzatori.

2.1 Lavori Correlati

La *literature* riguardante la Visual Kinship Verification ha una storia molto recente, tuttavia in questi anni sempre più ricercatori hanno affrontato questa sfida, con metodi molto differenti tra loro. In particolare, è possibile suddividere gli approcci in due categorie: basati sulle features e basati sul modello. I metodi del primo tipo consistono nell'estrazione di features discriminative che rappresentano il volto originale; su queste features vengono quindi utilizzate tecniche di machine learning, supervisionato o no, per determinare l'eventuale parentela. I metodi model-based, invece, imparano modelli discriminativi per la kinship verification direttamente dalle immagini in ingresso; qui di seguito vengono brevemente descritti gli approcci più rilevanti.

Il primo tentativo di risolvere il problema del riconoscimento delle parentele da immagini facciali risale al 2010, quando *Fang et al.* [4] proposero un metodo basato sull'estrazione e la selezione di features di basso livello dalle immagini dei due volti; le features più discriminanti venivano quindi date in pasto ad un K-Nearest

Neighbors (KNN) Classifier con metrica Euclidea per determinare l'eventuale parentela. Nell'ambito di questa ricerca gli autori hanno anche valutato le performance umane su questo problema. Un anno più tardi *Zhou et al.* [5] proposero un nuovo descrittore spaziale a piramide (SPLE) per rappresentare le immagini facciali; su questo descrittore applicarono una support vector machine (SVM) per verificare la parentela. Nel 2012 gli stessi autori [6] presentarono un nuovo descrittore piramidale basato sui filtri di Gabor denominato GGOP (Gabor-based Gradient Orientation Pyramid); anche in questo caso utilizzarono un classificatore SVM per la classificazione. DAISY è un altro descrittore rappresentante le features salienti di un'immagine, sviluppato da *Guo et al.* [7] ed adattato al problema della kinship verification. Sempre tra gli approcci feature-based si annovera l'algoritmo proposto da *Kohli et al.* [8], che utilizza la descrizione locale dell'immagine Weber Face pre-processata come rappresentazione di auto-similarità. *Yan et al.* [9] svilupparono il metodo PDFL (prototype-based discriminative feature learning), che impara multipli set di features di livello intermedio, in grado di caratterizzare al meglio la relazione di parentela tra due immagini facciali. CBFD (compact binary face descriptor) è un altro metodo per l'apprendimento delle features proposto da *Lu et al.* [10]; questo descrittore, e la sua versione accoppiata (Coupled CBFD), si basano sull'estrazione e la proiezione in uno spazio binario dei vettori delle differenze tra un pixel e quelli adiacenti (PDVs). Gli stessi autori [11] hanno successivamente implementato un metodo di analisi (DMMA) a partire dalle features discriminanti estratte da diverse patch dell'immagine. Infine, *Dibeklioglu et al.* [12] hanno proposto un metodo che utilizza features rappresentanti la dinamica facciale e l'aspetto spazio-temporale durante l'espressione di un sorriso, con lo scopo di riconoscere le parentele tramite la somiglianza tra espressioni facciali.

Tra i principali approcci model-based viene citato NRML (neighborhood repulsed metric learning), il metodo proposto da *Lu et al.* [13] per imparare uno spazio metrico in cui i campioni intra-classe (con relazioni di parentela) vengano avvicinati il più possibile e i campioni inter-classe, che giacciono nelle vicinanze, vengano respinti il più lontano possibile, simultaneamente. Gli stessi autori hanno anche proposto una versione 'multiview' (MNRML), per trovare una metrica comune tra diversi descrittori. *Yan et al.* [14] proposero un nuovo metodo di apprendimento multimetrico discriminante (DMML) per riconoscere le parentele da immagini facciali; questo approccio prevede l'estrazione di più features con molteplici descrittori facciali per apprendere contemporaneamente metriche diverse. Un approccio simile è stato proposto da *Hu et al.* [15], con lo scopo di apprendere diverse metriche sotto le quali la distanza tra ogni coppia positiva sia minore di una soglia inferiore, mentre quella tra coppie negative sia maggiore di un limite superiore; questo metodo è noto

con il nome di 'large margin multi-metric learning' (LM^3L). Infine, i due metodi proposti da *Xia et al.* [16] [17] utilizzano algoritmi basati sul transfer learning, con l'obiettivo di mitigare la differenza tra le distribuzioni delle immagini facciali di figlio e genitore.

Nella tabella 2.1 vengono schematizzati i lavori citati in questo paragrafo.

2.2 Database

La scelta del database è centrale per tutti i problemi di Computer Vision. I risultati ottenuti con gli approcci citati nel paragrafo precedente sono strettamente correlati al proprio dataset di riferimento. Dal 2010 in poi, ogni database per il riconoscimento automatico delle parentele che sia stato reso pubblico è subito diventato un caso di studio per molti ricercatori, tanto da poter considerare ognuno di questi come una vera e propria pietra miliare nell'ambito della Kinship Verification(fig 2.1).

In ordine cronologico, CornellKin [18] è il primo dataset pubblico per il riconoscimento automatico delle parentele. Creato ad hoc nell'ambito del progetto "Towards Computational Models of Kinship Verification" di *Fang et al.* [4], questo database consiste di 143 coppie di immagini (100x100 pixels) genitore-figlio di personaggi pubblici e celebrità (7 delle iniziali 150 coppie sono state rimosse dal dataset originale per motivi di privacy).

Un anno più tardi venne rilasciato l'UB KinFace Database [19], che focalizza la sua attenzione sull'impatto dell'età nel campo della Kinship Recognition. Il dataset contiene 600 immagini (64x64 pixels) di 400 persone suddivise in 200 gruppi; ognuno di questi gruppi contiene tre immagini: una del figlio e due del genitore, da giovane e da anziano. UB KinFace può essere ulteriormente suddiviso in base all'etnia in due categorie, Asiatici e non, entrambe contenenti 200 gruppi.

TSKinFace [20] (Tri-subject Kinship Face) è il primo database creato per riconoscere le parentele da un punto di vista one-to-two, ossia verificare se esiste un legame tra un figlio ed una coppia di genitori. Questo rappresenta il primo tentativo di coinvolgere più di due persone nel problema della Kinship Verification. In particolare vengono prese in esame le relazioni padre-madre-figlio (FM-S) e padre-madre-figlia (FM-D) con, rispettivamente, 513 e 502 gruppi tri-subject ciascuna. Le immagini, raccolte da fotografie di personaggi pubblici e social network (*flickr*), sono state tagliate e ridimensionate a 64x64 pixels in base alla posizione degli occhi.

Input	Authors (Year)	Kinship Verification Algorithm	Database
Image	Fang et al. [1] (2010)	Gabor-based gradient orientation pyramid	CornellKin
	Siyu et al. [2] (2011)	Transfer learning	UB KinFace
	Zhou et al. [3] (2011)	Spatial pyramid based learning	Private database
	Shao et al. [4] (2011)	Gabor filters with metric learning	UB KinFace
	Zhou et al. [5] (2012)	Gabor-based gradient orientation pyramid	Private database
	Xia et al. [6] (2012)	Transfer subspace learning based algorithm	UB KinFace Ver2.0 and FamilyFace
	Kohli et al. [7] (2012)	Self-similarity representation of Weber faces	UB KinFace and IIITD Kinship
	Fang et al. [8] (2013)	Reconstruction using parts from a set of families	Family101
	Lu et al. [9] (2014)	Multi-view NRML	KinFaceW-I and KinFaceW-II
	Hu et al. [10] (2014)	Large margin multi metric learning	KinFaceW-I and KinFaceW-II
	Yan et al.[11] (2014)	Discriminative multimetric learning	KinFaceW-I, KinFaceW-II, CornellKin, and UB KinFace
	Dehghan et al. [12] (2014)	Discrimination via gated autoencoders	KinFaceW-I and KinFaceW-II
	Guo et al. [13] (2014)	Graph-based approach	Sibling-Face and Group-Face
	Yan et al. [14] (2015)	Prototype discriminative feature learning	KinFaceW-I, KinFaceW-II, CornellKin, and UB KinFace
	Liu et al. [15] (2015)	Inheritable Fisher vector feature based kinship	KinFaceW-I and KinFaceW-II
	Alirezazadeh et al. [16] (2015)	Genetic algorithm for feature selection	KinFaceW-I and KinFaceW-II
	Qin et al. [17] (2015)	Relative symmetric bilinear model and spatially voted feature selection method	TSKinFace, Family101, KinFaceW-I, and KinFaceW-II
	Zhou et al. [18] (2016)	Ensemble similarity learning	KinFaceW-I and KinFaceW-II
	Robinson et al. [19] (2016)	Fine-tuning VGG network	Families in the Wild
	Xu and Shang [20] (2016)	Joint learning of multiple bilinear similarity models	KinFaceW-I and KinFaceW-II
	Wu et al. [21] (2016)	Utilized color-texture features	TSKinFace, KinFaceW-I, and KinFaceW-II
	Yan [22] (2016)	Neighborhood repulsed correlation metric learning	TSKinFace, KinFaceW-I, and KinFaceW-II
	Lopez et al. [23] (2016)	Chromaticity and color features	KinFaceW-I and KinFaceW-II
	Xu and Shang [20] (2016)	Used structured similarity fusion	KinFaceW-I and KinFaceW-II
	Li et al. [24] (2016)	Siamese convolutional neural net	KinFaceW-I and KinFaceW-II
	Wang et al. [25] (2017)	Denoising auto-encoder based metric learning	Families In the Wild (FIW)
	Lu et al. [26] (2017)	Discriminative deep multi-metric learning	KinFaceW-I and KinFaceW-II
	Liu et al. [27] (2017)	Status-aware projection learning	KinFaceW-I and KinFaceW-II
	Kohli et al. [28] (2017)	Kinship verification via representation learning	WVU, CornellKin, UB KinFace, KinFaceW-I, and KinFaceW-II
	Mahpod et al. [29] (2018)	Multi-view hybrid distance learning	CornellKin, KinFaceW-I and KinFaceW-II
video	Dibeklioglu et al. [30] (2013)	Spatio-temporal features utilizing facial dynamics	UvA-NEMO Smile
	Dibeklioglu [31] (2017)	Visual transformation aided contrastive learning	UvA-NEMO, KinFaceW-I, and KinFaceW-II

Tabella 2.1. Riassunto dei lavori citati nel paragrafo 2.1.

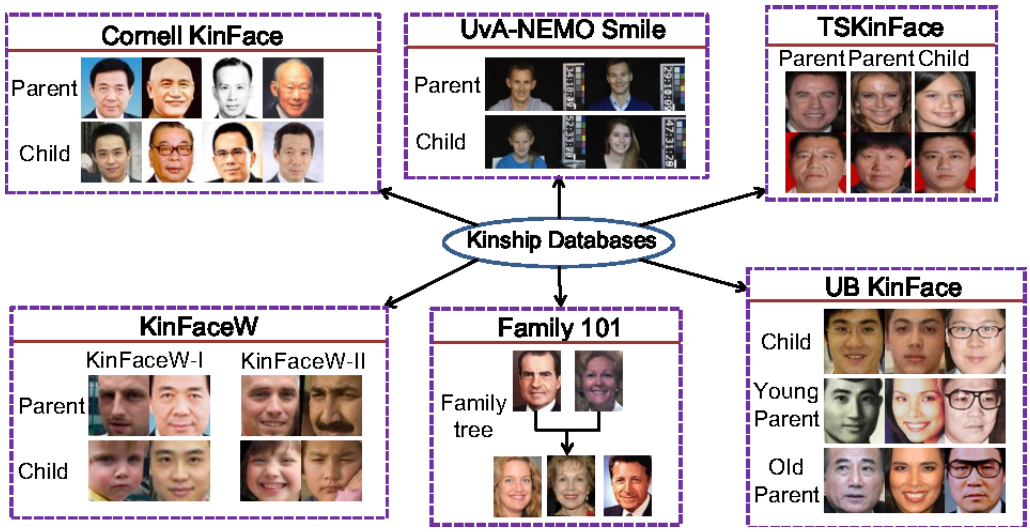

Figura 2.1. Principali databases per il riconoscimento delle parentele fino al 2016.

Il database Uva-Nemo Smile [21], introdotto nel 2012 da *Dibeklioglu et al.*, permette di affrontare il tema della kinship verification da un punto di vista 'video-based'. Il dataset contiene 1240 video di sorrisi, spontanei e non, di 400 persone, con età compresa tra gli 8 e i 76 anni. La dinamica del sorriso è stata inizialmente studiata per risolvere problemi di age estimation e solo un anno più tardi, nell'ambito del progetto "Like Father, Like Son: Facial Expression Dynamics for Kinship Verification" [12], è stata considerata come parametro per il riconoscimento delle parentele.

Nel 2013 alcuni ricercatori della Cornell University raccolsero all'interno del database Family101 [22], circa 15000 immagini di 607 individui appartenenti, appunto, a 101 famiglie differenti (206 nuclei familiari). Questo dataset ha da subito attirato l'attenzione su una nuova sfida, la Family Classification. Si tratta di individuare, data l'immagine di un individuo, la famiglia alla quale questo individuo appartenga. Family101 è il primo database a garantire una discreta distribuzione di quelle caratteristiche facciali che dipendono da etnia, sesso ed età.

I database più recenti, prima dell'introduzione del FIW database, che verrà descritto successivamente, sono KinFaceW-I e KinFaceW-II [23]. Entrambi contengono immagini 64x64 pixel di persone in un ambiente non controllato, senza restrizioni in termini di posa, illuminazione, sfondo ed espressioni. Come nella maggior parte dei database precedenti, vengono presi in considerazione quattro tipi di parentela: padre-figlio (F-S), padre-figlia (F-D), madre-figlio (M-S) e madre-figlia (M-D). KinFaceW-II contiene 250 coppie per ogni tipo di relazione, mentre KinFaceW-I ne contiene rispettivamente 156, 134, 116 e 127. La differenza principale tra i due dataset sta nel fatto che le immagini dei familiari nel primo provengono da fotografie

differenti, mentre nel secondo, nella maggior parte dei casi, dalla stessa fotografia.

Dataset	No. Family	No. People	No. Faces	Age Varies	Family Trees
CornellKin[8]	×	300	300	×	×
UBKinFace[20, 25]	×	400	600	✓	×
KFW-I[16]	×	1,066	1,066	×	×
KFW-II[16]	×	2,000	2,000	×	×
TSKinFace[18]	×	2,589	×	✓	✓
Family101[7]	101	607	14,816	✓	✓
FIW	**1,000**	**10,676**	**30,725**	✓	✓

Tabella 2.2. Confronto tra FIW e i database correlati. (fonte: Families in The Wild. A large-scale kinship recognition image database.)

2.3 RFIW Challenge

[3] La RFIW Challenge è una competizione organizzata dal Department of Electrical and Computer Engineering della Northeastern University di Boston, in collaborazione con la FG 2018 [27], il primo forum internazionale per la ricerca nell'ambito del riconoscimento di volti, gesti e movimenti del corpo da immagini e video.

La sfida si compone di due task distinti:

Track 1- Kinship Verification: Questo protocollo affronta il problema del riconoscimento delle parentele da un punto di vista *one-to-one*. L'obiettivo è quello di verificare se esiste, o meno, un legame sanguigno tra due coppie di immagini facciali. È un problema classico di tipo booleano, dove il sistema risponde *kin* (true) in caso di parentela, e *non-kin* (false) altrimenti. La ricerca, sia in campo psicologico, che in quello della computer vision, ha dimostrato che a differenti gradi di parentela, corrispondo differenti caratteristiche fisiche, per questo motivo è necessario processare ogni tipologia di parentela indipendentemente dalle altre.

Track 2- Family Classification: Questo task si concentra su un problema leggermente diverso: data l'immagine del volto di una persona, riconoscere a quale famiglia questa persona appartenga. Essenzialmente, si tratta di un problema di riconoscimento *one-to-many*, la cui difficoltà cresce all'aumentare del

numero di famiglie. La RFIW Challenge mette a disposizione un database contenente membri appartenenti a 1000 famiglie differenti, permettendo una buona approssimazione della reale distribuzione delle famiglie nel mondo.

Inizialmente era stato proposto, per l'edizione 2018, anche un terzo protocollo di tipo *one-to-two* (**Tri-Subject Verification**), con lo scopo di individuare se un figlio appartenesse o meno ad una determinata coppia di genitori. I dati per affrontare questo terzo task non sono ancora stati resi pubblici e saranno probabilmente disponibili solamente per la prossima edizione della RFIW Challenge.

In questa tesi è stato affrontato soltanto il primo task, tuttavia, con alcuni accorgimenti, molte delle tecniche utilizzate potrebbero essere applicate anche al problema della Family Classification.

2.3.1 FIW Database

La RFIW Challenge nasce con l'ambizione di portare lo studio della Automatic Kinship Verification da un livello di ricerca ad uno di sviluppo di tecnologie per applicazioni reali. Gli organizzatori hanno individuato due cause principali che hanno finora impedito questo passaggio: la prima risiede nei database esistenti, troppo piccoli e limitati per poter riflettere la reale distribuzione delle famiglie del mondo, mentre la seconda sta nella grande variabilità dei fattori che influenzano le apparenze facciali tra i diversi membri delle famiglie. Il Families In the Wild (FIW) database è stato rilasciato, nell'ambito della sfida, per ovviare a queste due problematiche.

Figura 2.2. Esempi di fotografie di 27 delle 1000 famiglie nel database Families in The Wild.

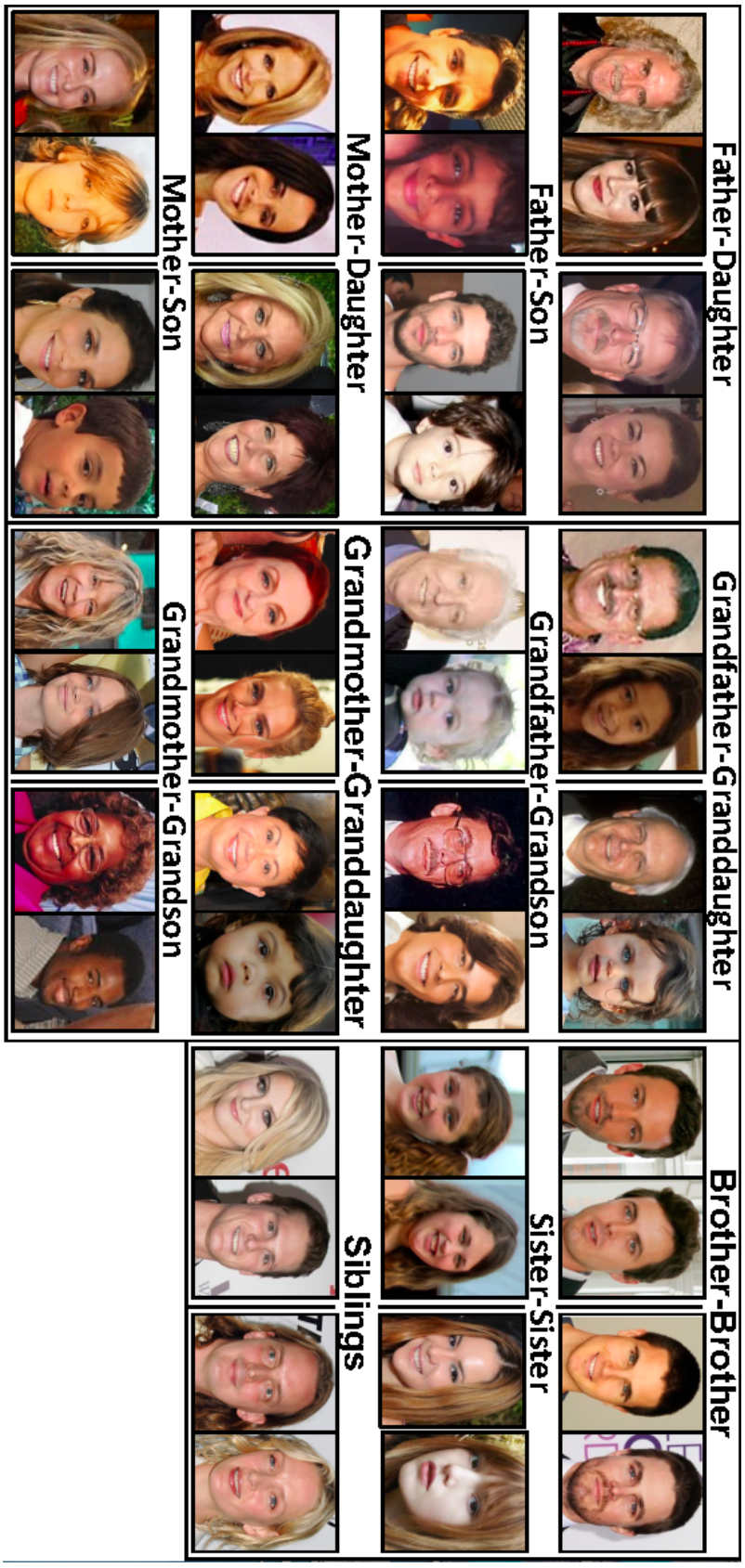

Figura 2.3. Esempi di coppie per le 11 relazioni presenti all'interno del FIW Dataset

FIW è il più grande e completo dataset disponibile per la verifica automatica delle parentele. Il database è composto da oltre 30000 immagini 224x224 di più di 10000 persone, appartenenti a 1000 famiglie differenti, 10 volte di più rispetto a Family101. Al suo interno sono contenute coppie di immagini per undici tipologie di parentela differenti (fig 2.3): fratello-fratello (B-B), sorella-sorella (S-S), fratello-sorella (SIBS), padre-figlio (F-S), padre-figlia (F-D), madre-figlio (M-S), madre-figlia (M-D) e, per la prima volta, le quattro relazioni tra nonno/a e nipote (GF-GD, GF-GS, GM-GD, GM-GS). Le tabelle 2.2 e 2.3 danno un'idea di come FIW superi di gran lunga i dataset pre-esistenti in termini di quantità, qualità ed intenti.

	siblings			parent-child				grandparent-grandchild				Total
	B-B	S-S	SIBS	F-D	F-S	M-D	M-S	GF-GD	GF-GS	GM-GD	GM-GS	
KinWild I	0	0	0	134	156	127	116	0	0	0	0	533
KinWild II	0	0	0	250	250	250	250	0	0	0	0	1,000
Sibling Face	232	211	277	0	0	0	0	0	0	0	0	720
Group Face	40	32	53	69	69	62	70	0	0	0	0	395
FIW	**103,724**	**39,978**	**73,506**	**92,088**	**129,846**	**82,160**	**112,618**	**7,078**	**4,830**	**6,512**	**4,614**	**656,954**

Tabella 2.3. Numero di coppie per ogni tipo di parentela in FIW e nei database correlati. (fonte: Families in The Wild. A large-scale kinship recognition image database.)

Il processo di creazione del database è stato suddiviso in tre step. Il primo riguarda la raccolta delle immagini (Data Collection): su diversi motori di ricerca (es. Google, Yahoo, Bing) e social network (es. Pinterest), sono state selezionate 1000 famiglie di personaggi pubblici con almeno 3 membri e 8 foto di famiglia. Le oltre 13000 fotografie di famiglia sono state scelte combinando la ricerca per nazionalità ed occupazione (es. attori cinesi, politici statunitensi, calciatori brasiliani...), in modo da garantire una buona eterogeneità. Il secondo step è il processo di annotazione dei dati (Data Annotation), attraverso un apposito tool di etichettatura. Un face detector individua i volti all'interno delle immagini, scartando quelli non rilevati per scarsa risoluzione, e richiede di inserire, per ogni nuovo membro, il nome, il sesso e il tipo di relazione con i membri già inseriti. Ad ogni membro della famiglia viene quindi associato un MemberID (MID) univoco e ad ogni famiglia un Family-ID (FID). L'ultimo passo riguarda l'accoppiamento delle immagini (Data Parsing). Utilizzando la matrice delle relazioni, creata durante lo step 2, viene generata una lista di coppie di immagini, positive e negative, per ognuno degli 11 tipi di paretela presenti all'interno del database.

In entrambe le sfide ogni lista di coppie è stata ulteriormente suddivisa in Train, Validation e Test set. I primi due insiemi sono stati resi disponibili con le rispettive etichette per allenare e validare il proprio modello, mentre il test set è "cieco" e può essere processato soltanto per generare sottomissioni.

2.3.2 Benchmarks

Gli organizzatori della sfida hanno condotto una serie di esperimenti di prova (benchmarks) sul database FIW, in modo da trovare dei risultati di riferimento sia per il protocollo di *kinship verification*, sia per quanto riguarda la *family classification*. Per ogni esperimento, e per ogni tipo di parentela, l'intero insieme delle coppie (train, validation e test set) è stato suddiviso in cinque sottogruppi ulteriori. È stata valutata l'accuratezza dei risultati su ognuno di questi sottogruppi, utilizzando gli altri quattro per l'addestramento, e la media delle cinque è stata presa come valore di riferimento.

In particolare, sono stati testati i principali approcci basati su feature extraction, apprendimento metrico e finetuning delle reti. Per quanto riguarda i metodi del primo tipo, sono stati presi in considerazione sia descrittori tradizionali (SIFT [28] e LBP [29]), sia descrittori profondi, in particolare quelli estratti dalle reti preallenate VGGFace [30] e Resnet-22 [31]. Per comparare i descrittori della coppia è stata calcolata la distanza coseno tra i due e, in base al fatto che questa fosse minore o maggiore di una certa soglia, è stata effettuata la classificazione tra parenti e non parenti. Sono stati quindi applicate ai descrittori estratti da VggFace quattro tra le principali tecniche di metric learning, Information theoretic metric learning (ITML) [32], Discriminative Low-rank Metric Learning (DLML) [33], Locality Preserving Projections (LPP) [34] e Large Margin Nearest Neighbor (LMNN) [35], e due approcci basati sugli autoencoders (AE): graph regularized marginalized Stacked AE (GmDAE) [36] e marginalized denoising metric learning (mDML) [37]. Infine, gli organizzatori hanno effettuato il finetuning sul database FIW delle reti Resnet-22, con *CenterFace* (CF) [38] come loss function, e della nuova rete SphereFace [39], che ha recentemente raggiunto lo stato dell'arte nell'ambito del riconoscimento facciale.

I risultati di questi esperimenti sono riportati nella tabella 2.4. Come si può notare, i risultati migliori sono stati raggiunti tramite il finetuning della rete Sphereface, con un'accuratezza media del 69,18%. In generale, la tecnica del finetuning ha portato ad un netto miglioramento rispetto all'utilizzo delle semplici reti preallenate. Tra queste, VGG-Face ha ottenuto risultati del 3.55% più alti rispetto a ResNet-22, ed entrambe superano le prestazioni dei descrittori SIFT ed LBP. Le tecniche di apprendimento metrico e gli autoencoders, invece, non hanno portato a miglioramenti significativi e, in alcuni casi, hanno anche peggiorato i risultati della rete di base. Per quanto riguarda la tipologia di parentela, i risultati più accurati sono stati ottenuti nei dataset dei fratelli, seguiti da quelli delle coppie genitore-figlio e, per finire, da quelle nonno-nipote: questo significa che più è ampia la distanza

di età tra i due membri della coppia, più diventa complicato il riconoscimento della parentela.

	siblings			parent-child				grandparent-grandchild				
Method	B-B	S-S	SIBS	F-D	F-S	M-D	M-S	GF-GD	GF-GS	GM-GD	GM-GS	Acc. ± Std.
LBP	55.52	57.49	55.39	55.05	53.77	55.69	54.65	55.79	55.92	54.00	55.36	55.33 ± 1.01
SIFT	57.86	59.34	56.91	56.37	56.24	55.05	56.45	57.25	55.35	57.29	56.74	56.80 ± 1.17
ResNet-22	65.57	69.65	60.12	59.45	60.27	61.45	59.37	55.37	58.15	59.74	59.70	61.34 ± 3.81
VGG-Face	69.67	75.35	66.52	64.25	63.85	66.43	62.80	62.06	63.79	57.40	61.64	64.89 ± 4.68
+ITML	57.15	61.61	56.98	58.07	54.73	57.26	59.09	62.52	59.60	62.08	59.92	59.00 ± 2.44
+LPP	67.61	66.22	71.01	62.54	61.39	65.04	63.54	63.50	59.96	60.00	63.53	64.03 ± 3.32
+LMNN	67.11	68.33	66.88	65.66	67.08	68.07	66.16	61.90	60.44	63.68	60.15	65.04 ± 3.00
+GmDAE	68.05	68.55	67.33	66.53	68.30	68.15	66.71	62.10	63.93	63.84	63.10	66.05 ± 2.36
+DLML	68.03	68.87	67.97	65.96	68.00	68.51	67.21	62.90	63.96	63.11	63.55	66.19 ± 2.36
+mDML	69.10	70.15	68.11	67.90	66.24	70.39	67.40	65.20	**66.78**	63.11	63.45	67.07 ± 2.44
ResNet+CF	69.88	69. 54	69.54	68.15	67.73	71.09	68.63	**66.37**	66.45	**64.81**	64.39	67.87 ± 2.15
SphereFace	**71.94**	**77.30**	**70.23**	**69.25**	**68.50**	**71.81**	**69.49**	66.07	66.36	64.58	**65.40**	**69.18** ± 3.68

Tabella 2.4. Accuratezza di benchmark per gli esperimenti svolti dagli organizzatori della RFIW Challenge. (fonte: Visual Kinship Recognition of Families in the Wild)

2.3.3 Performance umane

Gli organizzatori della competizione hanno anche valutato l'abilità umana nell'ambito del riconoscimento delle parentele, in modo da permettere un confronto tra le prestazioni di uomo e macchina. Oltre 150 volontari sono stati sottoposti a due esperimenti distinti: nel primo è stata comunicata loro la tipologia di parentela da riconoscere, mentre nel secondo esperimento (caso booleano) questa informazione è stata omessa, per cercare di capire quanto la conoscenza del tipo di parentela influisse sulla risposta. I partecipanti hanno dovuto valutare tra 406 coppie di immagini facciali, nello specifico 50 per ogni relazione tra fratelli (BB, SS, SIBS), 36 per quelle genitore-figlio (FD, FS, MD, MS) e 28 per le parentele nonno-nipote (GFGD, GFGS, GMGD, GMGS).

In entrambi gli esperimenti, i partecipanti hanno raggiunto un'accuratezza media che oscilla tra il 57% e il 58%, con un risultato leggermente inferiore nel caso booleano. Per quanto riguarda le singole tipologie di parentela, l'accuratezza media non è mai scesa sotto al 50% e, allo stesso tempo, non ha mai superato il 67,5% in nessuno dei diversi test set. Inoltre, analizzando le coppie che sono state classificate più frequentemente in modo errato, gli organizzatori hanno potuto trarre alcune

interessanti conclusioni. In primo luogo hanno notato che nel primo esperimento, soprattutto per le parentele genitore-figlio e nonno-nipote, le coppie in cui vi è una contraddizione tra l'età mostrata nelle foto e quella indicata dalla direzione della relazione (es. genitore a 30 anni e figlio a 50) sono state quelle marcate più spesso in modo scorretto. Questo problema, invece, non sembra peggiorare le prestazioni nel caso booleano. In secondo luogo, hanno constatato come i partecipanti abbiano spesso sbagliato a riconoscere le relazioni tra i volti di persone con un'etnia diversa dalla loro. Infine, un dato interessante è che, per tutte le tipologie di parentela, la persona che ha ottenuto l'accuratezza maggiore è sempre stata di sesso femminile. Le donne, inoltre, si sono spesso rivelate meno soggette ai tipici errori di età ed etnia commessi dagli uomini. Queste osservazioni hanno dimostrato come sia la natura che il background di ogni individuo influisca sulla sua abilità nel riconoscere le parentele. Nella figura 2.4 viene riportato un diagramma a scatola e baffi dei risultati dei due esperimenti.

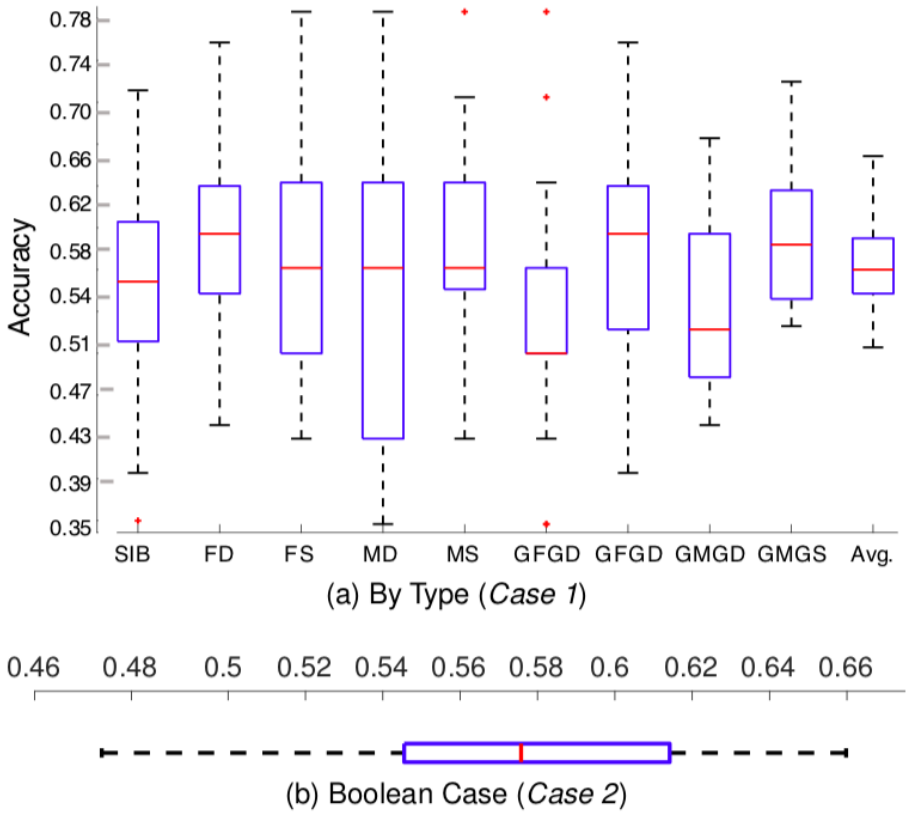

Figura 2.4. Box plot delle prestazioni umane nei due esperimenti di kinship verification (fonte: Visual Kinship Recognition of Families in the Wild).

2.3.4 Stato dell'arte (vincitore RFIW 2017)

In questo paragrafo vengono riportati i risultati (tab 2.5) ottenuti dai partecipanti alla RFIW Challenge 2017 [40] nella Track-1 (Kinship Verification). Viene quindi brevemente descritto l'approccio proposto dai ricercatori dell'Institute of Computing Technology in Cina (*mysee1989*), che hanno raggiunto il primo posto in questa competizione.

REF	USERNAME	F-D	F-S	M-D	M-S	SIBS	B-B	S-S	Avg.
1.A	BASELINE	68.53	69.02	73.11	71.54	68.37	69.97	75.09	70.81
1.B	mysee1989	70.79	71.46	77.87	78.62	79.91	74.77	70.57	74.86
1.C	ella	66.89	65.284	72.31	71.68	71.09	64.67	70.11	68.86
1.D	DQYCQU	65.196	63.38	70.86	65.22	72.10	64.00	66.51	66.58
1.E	viking	61.31	64.22	63.81	63.296	63.58	62.70	64.57	63.22
1.F	eranda	60.77	62.15	65.03	63.13	64.59	61.70	64.31	63.097
1.G	faud	61.31	61.98	62.30	57.69	63.83	60.60	61.62	61.33
1.H	Ji.L	59.95	59.72	61.12	55.69	59.40	57.68	58.30	58.83
1.I	BIU-team	56.16	60.594	60.00	54.33	61.205	59.70	58.02	58.21
1.J	oualid.laiadi	54.92	55.93	54.89	54.33	57.96	53.38	52.25	54.81

Tabella 2.5. Accuratezza (%) dei risultati nella Track-1 della RFIW 2017. I 3 migliori risultati sono evidenziati rispettivamente in blu (1st), rosso (2nd) e arancione (3rd). (fonte: RFIW 2017 [40])

I vincitori della sfida hanno introdotto KinNet [41] (fig 2.5), un framework per il riconoscimento automatico delle parentele, basato sul trasferimento della conoscenza da un problema di face recognition ad uno di kinship verification. Come architettura di base per KinNet, gli autori hanno testato diverse variazioni di reti neurali residuali (Resnet), pre-allenandole per riconoscere i volti all'interno di un immagine su un sottoinsieme del dataset MS-Celeb-1M [24]. Successivamente hanno effettuato il fine-tuning della rete, sostituendo l'ultimo fully-connected layer (di dimensione 41.856) con un fc-layer da 1024 neuroni, seguito da un L2-normalization layer per normalizzare le features 1024-dimensionali ad una lunghezza unitaria. Infine gli autori hanno utilizzato un classificatore soft triplet-loss per trovare uno spazio metrico in cui le coppie di parenti fossero il più vicino possibile tra loro e il più lontano possibile rispetto alle coppie negative. Durante la fase di fine-tuning i layers

convoluzionali 7x7 di livello più basso sono stati resi non-allenabili, mentre i pesi dei layers di livello più alto sono stati modificati, per adattarsi al problema di kinship verification. Per migliorare le prestazioni, gli autori hanno inoltre proposto una strategia per l'aumento del numero di immagini, in modo che ognuno degli individui possedesse lo stesso numero di figure.

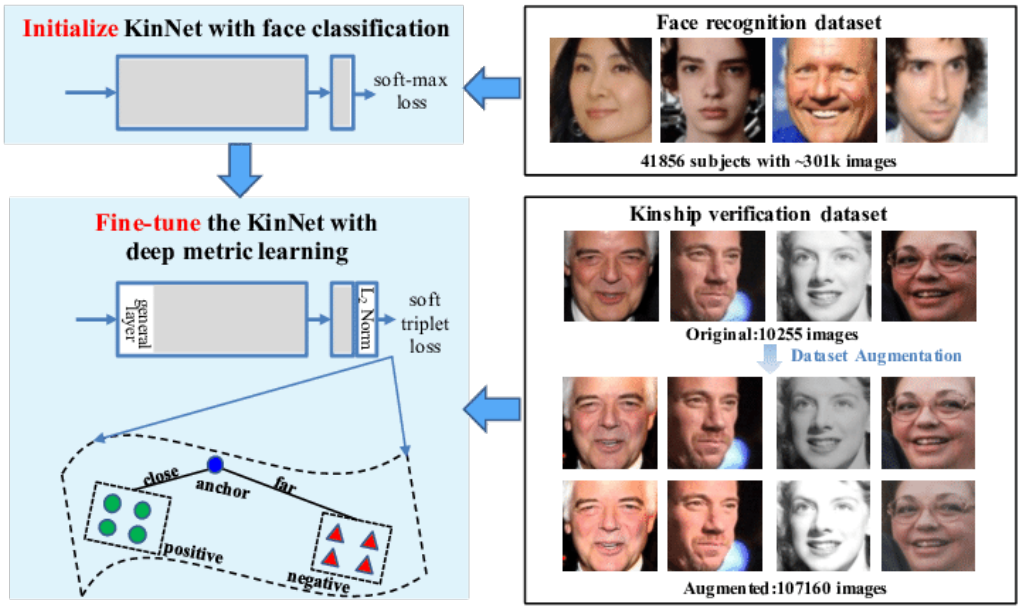

Figura 2.5. KinNet proposta dall'Institute of Computing Technology in Cina.

2.3.5 RFIW 2018

Sebbene il database FIW sia stato usato sia nell'edizione 2017 sia nella 2018 della RFIW Challenge, la suddivisione delle coppie di immagini è stata modificata tra la prima e la seconda edizione. Come si evince dalla tabella 2.6, nell'ultima edizione sono state prese in esame circa 15.000 coppie in più per ogni tipo di parentela, per un totale di 654.304 coppie contro le 538.519 del 2017. Inoltre, è possibile notare come nella RFIW 2018 siano stati inclusi i nuovi 4 tipi di parentela nonno/a-nipote. Questo rappresenta il primo tentativo di estendere la sfida della kinship verification ad un livello multi-generazionale.

Gli esperimenti svolti in questa tesi utilizzano la suddivisione fornita nell'ambito della RFIW Challenge 2018 e, per questo motivo, non è stato possibile effettuare un paragone equo con i risultati dell'edizione precedente. I nostri risultati sono stati quindi confrontati con quelli ottenuti dagli altri partecipanti alla sfida (2.7), sebbene i metodi proposti da questi ultimi non siano ancora stati resi disponibili, in modo da valutare la nostra ipotetica posizione in classifica. In seguito è stato effettuato anche

RFIW 2017

Tipo	Train	Validation	Test	Tot
B-B	52.483	17.342	19.946	89.771
S-S	19.286	6.218	6.524	32.028
SIBS	40.846	7.434	15.076	63.356
F-D	42.458	11.460	23.506	77.424
F-S	53.974	13.696	45.988	113.658
M-D	34.828	10.698	20.674	66.200
M-S	38.312	9.816	47.954	96.082
Tot	282.187	76.664	179.668	538.519

RFIW 2018

Tipo	Train	Validation	Test	Tot
B-B	29.812	55.546	18.196	103.554
S-S	19.778	35.024	4.796	59.598
SIBS	28.428	15.422	9.716	53.566
F-D	41.604	35.238	15.040	91.882
F-S	64.826	44.870	18.166	127.862
M-D	39.110	29.012	14.394	82.516
M-S	66.464	31.094	14.806	112.364
GF-GD	1.478	4.846	838	7.162
GF-GS	1.388	1.926	1.588	4.902
GM-GD	1.580	3.768	952	6.300
GM-GS	1.284	1.844	1.470	4.598
Tot	295.752	258.590	99.962	654.304

Tabella 2.6. Confronto della suddivisione dei dati tra l'edizione 2017 e 2018 della RFIW Challenge.

un paragone tra i nostri risultati e quelli conseguiti dagli organizzatori nei diversi esperimenti di benchmark, oltre che con le performance umane nei due esperimenti descritti nel paragrafo 2.3.3.

#	User	Entries	Date of Last Entry	Results										
				By Pair Type										Average ▲
				MD ▲	MS ▲	SS ▲	BB ▲	SIBS ▲	GMGD ▲	GMGS ▲	FS ▲	GFGS ▲	FD ▲	GFGD ▲	
1	eranda	24	02/12/18	0.737946 (1)	0.708564 (1)	0.772519 (1)	0.691471 (1)	0.686497 (1)	0.638655 (1)	0.632653 (1)	0.688979 (1)	0.636650 (1)	0.689229 (1)	0.619332 (1)	0.682045 (1)
2	sizhangyu	8	05/03/18	0.661873 (3)	0.675604 (2)	0.711218 (3)	0.638492 (4)	0.652841 (2)	0.594538 (2)	0.526531 (5)	0.669492 (2)	0.525819 (6)	0.655585 (2)	0.591885 (3)	0.627625 (2)
3	mariaivanovska	5	02/01/18	0.666250 (2)	0.598406 (5)	0.728524 (2)	0.662563 (3)	0.629683 (3)	0.575630 (3)	0.610884 (3)	0.615270 (3)	0.569270 (2)	0.633045 (3)	0.579952 (4)	0.624498 (3)
4	mngyyu	2	01/06/18	0.658886 (4)	0.604687 (4)	0.702043 (4)	0.684436 (2)	0.616200 (4)	0.569328 (4)	0.627891 (2)	0.612243 (4)	0.563602 (3)	0.630718 (4)	0.594272 (2)	0.624028 (4)
5	szl281	6	12/27/17	0.595526 (6)	0.578144 (6)	0.646789 (6)	0.572214 (6)	0.574002 (5)	0.553571 (5)	0.576871 (4)	0.574205 (6)	0.556675 (4)	0.593551 (5)	0.559666 (6)	0.580110 (6)
6	Bekhouche	7	02/09/18	0.615534 (5)	0.604822 (3)	0.680150 (5)	0.627006 (5)	0.590161 (5)	0.545168 (6)	0.521769 (6)	0.581526 (5)	0.532746 (5)	0.582114 (6)	0.578759 (5)	0.587250 (5)
7	azzasama	26	02/11/18	0.567042 (7)	0.540862 (7)	0.601960 (7)	0.513190 (7)	0.553932 (7)	0.524160 (7)	0.502041 (7)	0.541231 (7)	0.524559 (7)	0.559574 (7)	0.551313 (7)	0.543624 (7)

Tabella 2.7. Accuratezza (%) dei risultati dei partecipanti alla Track-1 della RFIW 2018. (fonte: RFIW 2018 [2])

Capitolo 3

Progettazione

In questo capitolo vengono descritti i diversi passi della pipeline utilizzata in questa tesi, per l'implementazione di un sistema di riconoscimento automatico delle parentele. in particolare, abbiamo seguito due strade alternative, che ci consentissero di sfruttare una rete pre-allenata per il riconoscimento facciale, nella nostra soluzione di kinship verification (*transfer learning*). Nel primo caso abbiamo estratto i vettori di caratteristiche direttamente dalla rete originale, senza un ulteriore addestramento, e li abbiamo analizzati per individuare gli eventuali legami; la seconda strada è stata quella di effettuare un'operazione di finetuning della rete sul database FIW, in modo da modificare i pesi di alcuni strati della rete, per adattarli al nostro specifico problema.

Per ognuno degli step seguiti, abbiamo testato strategie differenti, con lo scopo di raggiungere la più alta accuratezza di risultati per tutti i diversi dataset. È bene ricordare, infatti, che l'intera pipeline è stata eseguita, in tutte le sue varianti, undici volte, come il numero di gradi di parentela disponibili all'interno del nostro database di riferimento. Inoltre, tutti i passaggi sono stati effettuati, in una prima fase, sui set di training e validazione, per l'addestramento del sistema, quindi sono stati replicati sui diversi test set per la classificazione.

3.1 Tecnologie utilizzate

L'intero progetto è stato sviluppato con Python 3.6, utilizzando in particolar modo le librerie *Scikit-Learn* e *Keras*, molto utili, rispettivamente, per gli algoritmi di apprendimento automatico e per le reti neurali:

Scikit-Learn [42]**:** si tratta di una libreria open-source di Python creata appositamente per il machine learning. Racchiude diversi algoritmi per la classificazione, la regressione e il custering, ed è stata progettata specificatamente per l'interoperabilità con le librerie numeriche e scientifiche *NumPy* e *SciPy*. In questa tesi abbiamo utilizzato i metodi di Scikit-Learn nelle fasi di normalizzazione, PCA e classificazione con support vector machine.

Keras [43]**:** questa API di alto livello scritta in Python, che si appoggia su librerie di deep learning di livello più basso, come *Theano* o *Tensorflow* (nel nostro caso), permettendo all'utente di interfacciarsi in maniera rapida e comprensibile al mondo delle reti neurali. Keras (fig 3.1) è una libreria modulare e facilmente estensibile, sviluppata con particolare attenzione all'esperienza umana (user-friendly). 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, in modo intuitivo, per la creazione e l'addestramento dei modelli. È proprio questa caratteristica che ci ha spinto a scegliere di utilizzare Keras, rispetto ad altre librerie di deep learning, per l'implementazione di questo progetto.

Oltre a *Scikit-Learn* 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, *Scipy*, per il calcolo delle distanze tra descrittori, *csv*,

Figura 3.1. Logo Keras + Tensorflow.

per la lettura e la gestione dei file Excel, e la libreria *Matplotlib*, per la realizzazione di grafici e la visualizzazione dei dati.

La maggior parte degli esperimenti sono stati eseguiti sulla piattaforma Cloud per il deep learning *Floydhub* [44], utilizzando nello specifico una GPU Tesla K80 da 12 GB di memoria.

3.2 Operazioni preliminari

Prima di procedere con l'implementazione della soluzione vera e propria, è stato necessario effettuare alcune operazioni preliminari. In primo luogo, abbiamo dovuto iscriverci alla sfida *RFIW 2018*, tramite il sito della Northeastern University, in modo da ricevere l'autorizzazione per poter scaricare il database *Families in the Wild*. Le immagini facciali sono organizzate all'interno di una serie di sottocartelle, suddivise per famiglia e membri della famiglia, secondo il formato *FId/Mid/PId_FaceId.jpg*: FId e MId rappresentano, rispettivamente, numero della famiglia e numero del membro all'interno della famiglia, mentre con PId e FaceId vengono indicati l'id della fotografia e il numero del volto all'interno della fotografia stessa.

Oltre alle immagini, abbiamo effettuato anche il download dei 33 file Excel corrispondenti a training, validation e test set per ognuno degli undici tipi di parentela. Ogni riga dei file di training e validation contiene i path della prima e della seconda immagine della coppia oltre all'etichetta, 0 o 1, che indica l'eventuale parentela; nei file di test quest'ultima colonna non è presente, in modo tale da non permettere ai partecipanti alla sfida di utilizzare queste immagini per l'addestramento del modello, ma solamente nella fase di classificazione.

Infine, le righe dei file .csv sono state mescolate tra di loro, in modo da alternare le coppie positive e negative all'interno dei set di training e validazione. L'operazione di shuffling dei dati è estremamente importante quando si vuole addestrare una rete neurale o un classificatore, soprattutto nel caso in cui il training venga effettuato per batch (un gruppo di campioni per volta). In questo caso, avere batch estremamente sbilanciate potrebbe alterare drasticamente l'abilità di apprendimento del classificatore.

3.3 Scelta della rete

La prima decisione affrontata durante lo svolgimento della tesi, è stata quella del tipo di rete neurale da utilizzare nel nostro sistema di riconoscimento automatico

delle parentele.

Le reti neurali artificiali 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 (fig 3.2). 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, con algoritmi di apprendimento automatico, mediante una serie di esempi della realtà da modellare.

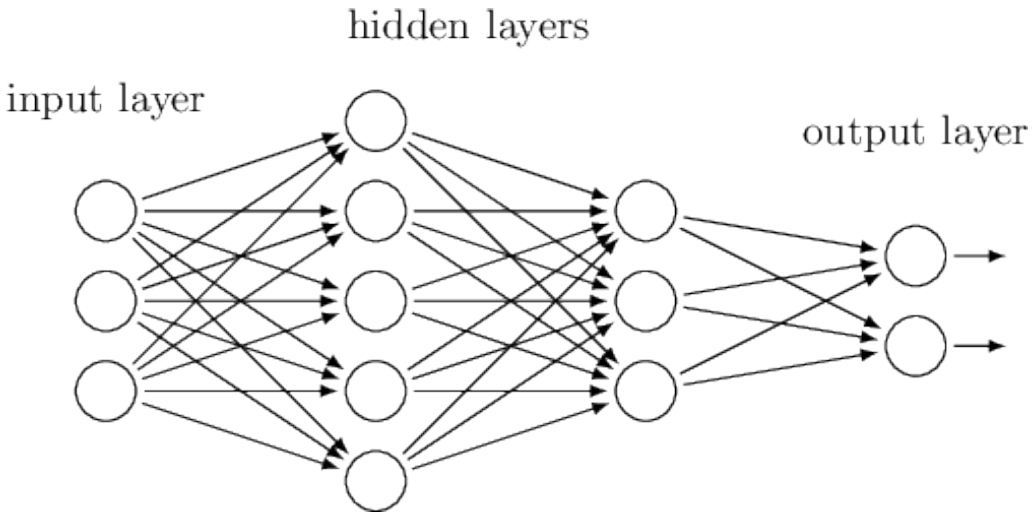

Figura 3.2. Esempio di topologia di una rete neurale.

Di particolare rilevanza nello sviluppo del progetto sono state le reti convoluzionali neurali, dette anche ConvNets o CNN's, una particolare tipologia di rete creata appositamente per gestire input visuali come immagini e video. L'architettura delle CNN's si ispira alla corteccia visiva animale; in particolare, gli strati di convoluzione consentono di suddividere l'immagine in una serie di frammenti sovrapposti, di analizzarli e di individuarne le principali caratteristiche.

L'idea iniziale di costruire una rete ad hoc per riconoscere le eventuali parentele ed effettuarne il training da zero (*from scratch*) è stata quasi immediatamente abbandonata. È 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 from scratch di una rete neurale è un'operazione decisamente non banale; oltre alle enormi capacità computazionali richieste, di gran lunga superiori rispetto a quelle raggiungibili da un classico PC,

per allenare una ConvNet da zero è indispensabile disporre di un database sufficientemente grande. Nel nostro caso specifico sarebbe stato necessario effettuare il training di 11 reti differenti, una per ogni grado di parentela, attraverso altrettanti sottoinsiemi del database Families In the Wild. Le dimensioni di questi sottoinsiemi, in particolare quelli riguardanti le relazioni nonno-nipote, non ci avrebbero permesso di raggiungere risultati abbastanza accurati, rendendo questa strada impraticabile con le risorse a nostra disposizione.

Per tutti i motivi elencati in precedenza, abbiamo deciso di seguire una strada alternativa, ossia implementare il nostro sistema di riconoscimento automatico sfruttando la tecnica del *transfer learning*. Questa pratica, molto diffusa nell'ambito dell'apprendimento profondo, consiste nel prendere una rete neurale preesistente, addestrata a svolgere una determinata attività, ed utilizzarla per risolvere un problema differente, ma in qualche modo correlato. Il trasferimento della conoscenza può avvenire tramite due diverse strategie: la prima prevede l'utilizzo della rete di base come un estrattore fisso di vettori di caratteristiche ed il successivo training di un classificatore lineare mediante tali vettori; la seconda strada, invece, consiste nell'effettuare il fine-tuning della rete preesistente, proseguendo l'allenamento della stessa con il nuovo dataset. Tutti e due i metodi sono stati testati in questa tesi ed entrambi verranno descritti nel dettaglio nei paragrafi successivi.

Perché il transfer learning abbia successo, è necessario che le features estratte dalla rete originale siano quanto più generiche possibile. Per esempio, per implementare una rete in grado di classificare le varie tipologie di un oggetto (es. tra i vari modelli di un automobile), si è soliti partire da una rete addestrata sul database ImageNet, che comprende oltre un milione di immagini appartenenti a 1000 diverse categorie, per poi effettuare il fine-tuning sul nuovo dataset specifico. 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.

I modelli delle reti VGG-16 e Resnet-50, con i rispettivi pesi, sono stati scaricati dalla repository github `https://github.com/rcmalli/keras-vggface` e rappresentano le implementazioni Keras dei modelli sviluppati dal Visual Geometry Group.

3.3.1 VGG-16

VGG-16 (fig 3.3) 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" [46].

L'architettura di questa CNN è molto semplice, in quanto utilizza solamente una serie di strati convoluzionali 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

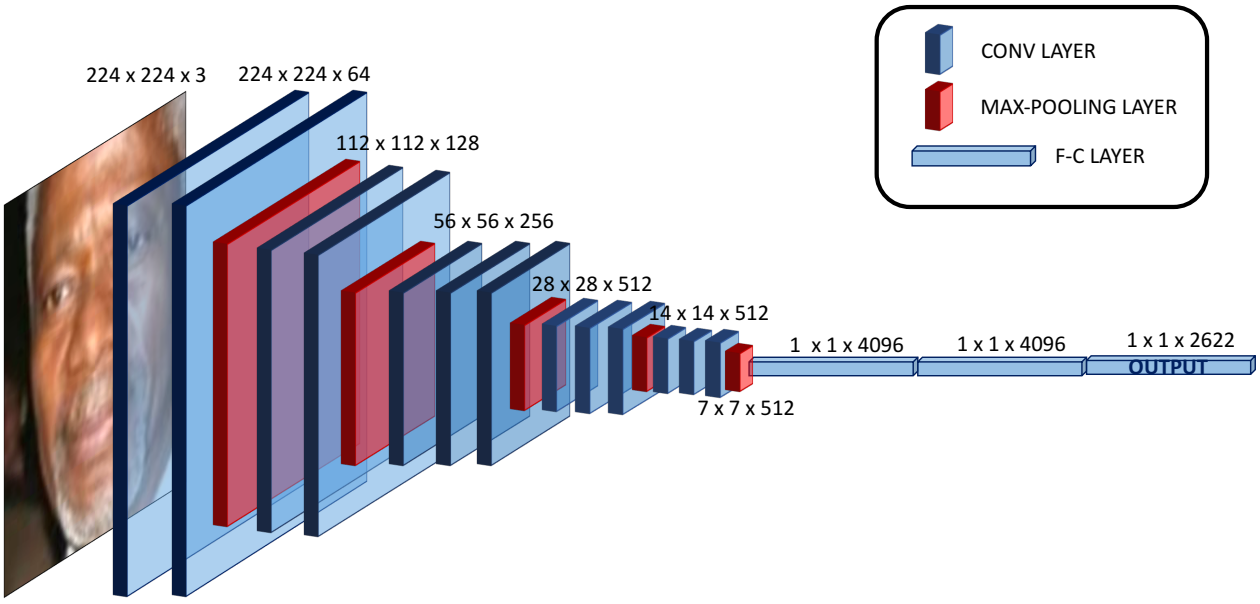

Figura 3.3. Architettura della rete VGG-16.

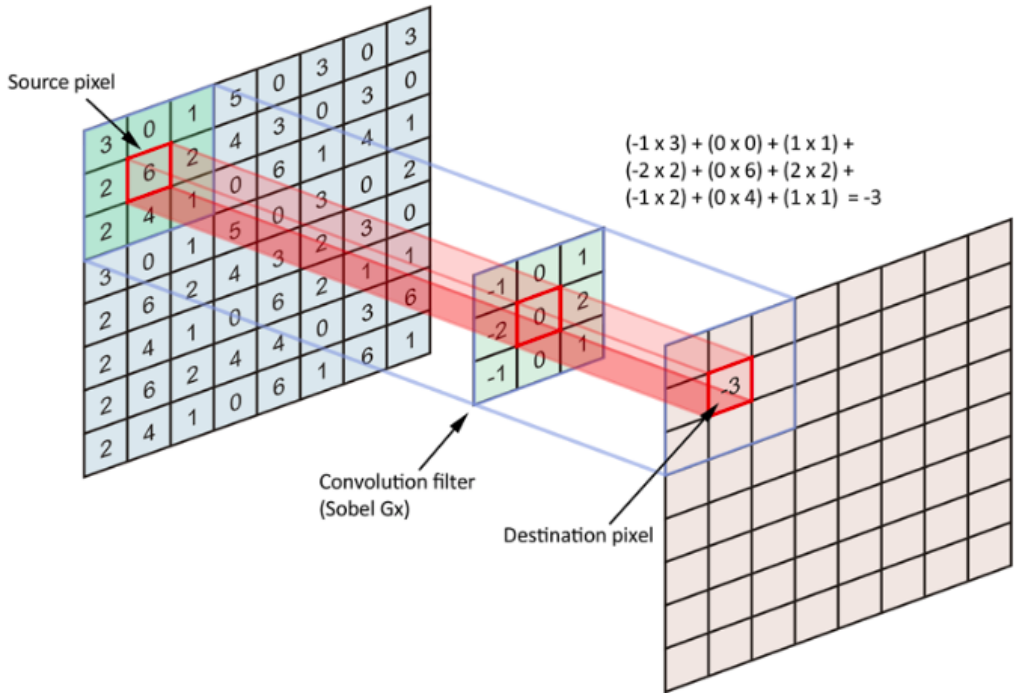

Figura 3.4. Esempio di convoluzione (fonte: Medium).

Strati Convoluzionali Questi layers rappresentano il fulcro delle ConvNets (fig 3.4). Sono costituiti da blocchi tridimensionali di neuroni artificiali, connessi localmente ad una regione limitata dell'immagine in input e del filtro successivo. Ogni neurone calcola il prodotto scalare tra i propri pesi e la regione a cui è connesso tramite la formula: $Y = \sum(weight * input) + bias$; il risultato di quest'operazione è un output di dimensione [w, h, n], dove w e h rappresentano altezza e larghezza dell'input, mentre n equivale al numero di filtri dello strato convoluzionale. Il fatto che i neuroni della rete non siano totalmente interconnessi tra loro, come accade invece nelle reti neurali classiche, permette di gestire immagini molto grandi in input senza memorizzare un eccessivo numero di parametri.

Strati di Attivazione Ogni layer di convoluzione è necessariamente seguito da uno strato di attivazione, tanto che molto spesso questi ultimi vengono considerati parte integrante degli strati convoluzionali. Lo scopo delle funzioni di attivazione è quello di fornire alla rete un comportamento non-lineare, in modo da permettere la risoluzione di problemi complicati, utilizzando un numero relativamente ridotto di nodi. Senza la presenza di una funzione di attivazione non lineare, una rete neurale si comporterà sempre come una rete ad un solo strato, a prescindere dal numero effettivo di hidden-layers presenti. È possibile utilizzare diverse funzioni di attivazione, nel caso di VGG-16 è stata scelta la

funzione *ReLU*, dall'inglese Rectified Linear Unit (fig 3.5):

$$ReLU(x) = max(0, x); \tag{3.1}$$

che porta a zero la risposta a tutti i valori negativi, mentre lascia tutto invariato per valori uguali o superiori a zero.

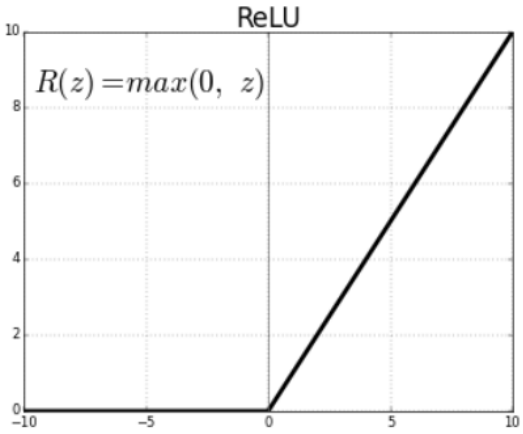

Figura 3.5. Funzione di attivazione ReLU.

Strati di max-pooling I layer di Pooling consentono di effettuare un'operazione di sotto-campionamento spaziale dei volumi in ingresso. Nel caso di un'immagine, il processo consiste nel sostituire i valori di una regione di pixel, di dimensione pari alla grandezza del filtro di pooling, con un unico pixel dal valore pari al massimo tra i pixel iniziali , nel caso di *max-pooling* (fig 3.6), o alla loro media, nel caso di *average-pooling*. Nella rete Vgg-16, gli strati di max-pooling utilizzano filtri di dimensione 2x2, con passo 2 sia in orizzontale che in verticale: l'output di questi strati sarà quindi un volume di dimensione $[w_{out}, h_{out},$ n$]$, dove $w_{out} = w_{in}/2$ e $h_{out} = h_{in}/2$, mentre n rimane invariato. Il fatto che vengano ridotte le dimensioni spaziali della rappresentazione consente di ridurre il numero di parametro e, di conseguenza, anche i tempi computazionali e la probabilità di sovradattamento ai dati (overfitting).

Strati Fully-Connected Dopo la serie di strati di convoluzione e di max-pooling, la rete VGG-16 comprende due strati fully-connected (fc-layers) da 4096 nodi. Questi layer equivalgono a quelli delle classiche reti neurali artificiali, il loro comportamento è identico a quello degli starti di convoluzione, con l'unica differenza nel fatto che i nodi sono interamente connessi a tutti i neuroni degli

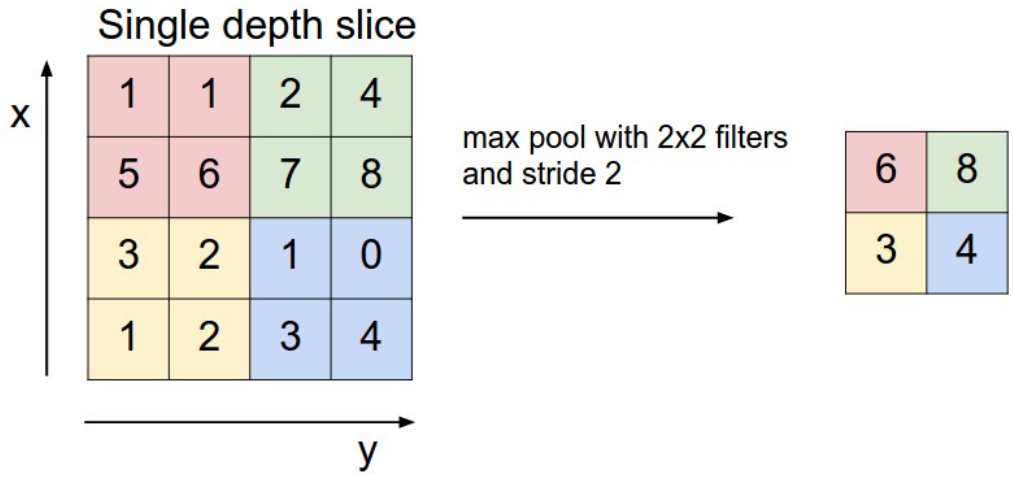

Figura 3.6. Operazione di max-pooling. (fonte: CS231n Covolutional Neural Networks for Visual Recognition)

strati precedenti e non solamente ad una piccola regione. Anche gli FC-layers, come gli strati convoluzionali, sono seguiti da uno strato di attivazione ReLU.

Strato di output Infine, l'output layer, anch'esso interamente connesso con il volume precedente, si occupa della classificazione vera e propria. Nella nostra rete, l'output sarà un volume di dimensione [1, 1, 2622], dove 2622 è il numero di classi (persone) presenti nel database con cui la rete VGG-16 è stata pre-allenata. La scelta viene effettuata tramite la funzione di attivazione *softmax*:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}} \quad i = 1, ..., K. \tag{3.2}$$

che, sostanzialmente, trasforma un vettore n-dimensionale di valori reali arbitrari in un vettore sempre di dimensione n di valori compresi in un intervallo (0,1) la cui somma è 1. Alla classe più probabile verrà assegnato il valore maggiore, mentre, al contrario, verrà minimizzato il valore delle classi meno probabili.

Come detto in precedenza, il modello di VGG-16 utilizzato in questa tesi è stato pre-allenato sulla prima versione del database di riconoscimento facciale VGGFace [25], creato dallo stesso gruppo di ricerca per classificare i volti appartenenti a 2622 individui.

3.3.2 Resnet-50

Il modello Resnet-50 appartiene ad una famiglia di reti artificiali neurali conosciute come reti residuali (Residual Networks). Le CNNs profonde hanno dimostrato di poter risolvere molto accuratamente gran parte dei compiti di classificazione di immagini e riconoscimento visivo. Per questo motivo, negli ultimi anni, c'è stata una tendenza ad aumentare sempre di più la profondità delle reti. Tuttavia, più si va in profondità, più l'addestramento delle reti convoluzionali classiche diventa lungo e complesso ed anche la precisione inizia a saturarsi o, peggio, a degradarsi. Le reti residuali si basano su un nuovo ed innovativo tipo di blocchi (detti *residual block*) e sul concetto di residual learning per cercare di risolvere entrambi i problemi. I blocchi residuali (fig 3.7) sottopongono un input x ad una sequenza di operazioni di convoluzione ed attivazione (Relu) ottenendo un output *F(x)*, al quale verrà sommato lo stesso input x, detto residuo. Le Residual Networks hanno dimostrato di essere più semplici da allenare rispetto alle CNN classiche ed, inoltre, non vanno incontro ad un degrado delle performance all'aumentare della profondità della rete.

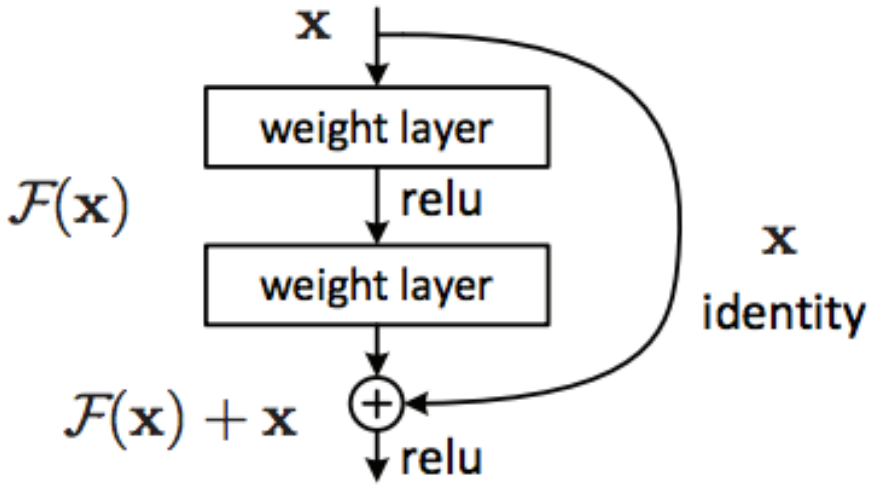

Figura 3.7. Blocco residuale. (fonte: Deep Residual Learning for Image Recognition)

Nello specifico la rete Resnet-50 utilizza due tipi di blocchi residuali (fig 3.8), il primo, detto blocco identità, si comporta come i blocchi residuali tradizionali descritti in precedenza, mentre il secondo, detto blocco di convoluzione, sottopone il tensore in input ad un diverso numero di strati convoluzionali in entrambi i rami, per poi sommarne i risultati.

Nonostante l'architettura di Resnet-50 sia completamente differente rispetto a quella di VGG-16, il tipo di strati utilizzati è sostanzialmente analogo. L'unica differenza sta nel tipo di layer utilizzati per la regolarizzazione della rete: in questa

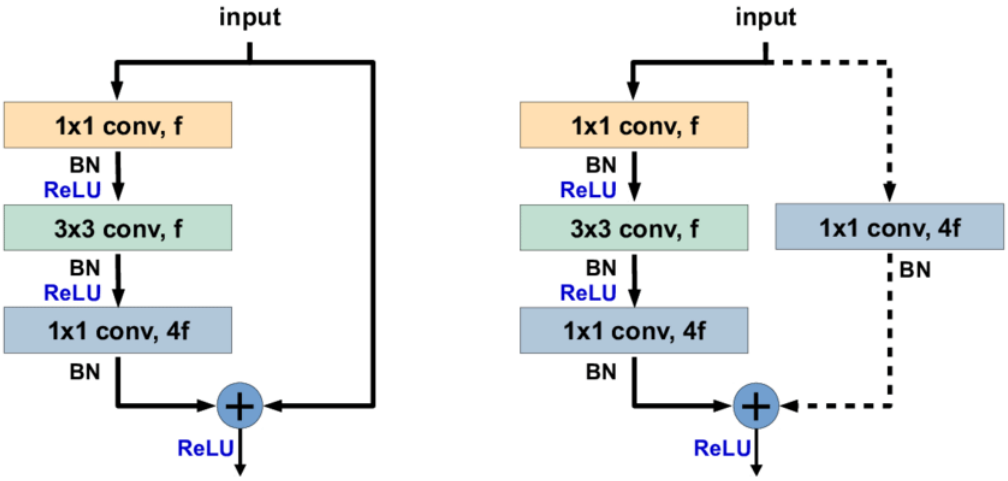

Figura 3.8. Blocco identità vs blocco di convoluzione.

architettura, infatti, gli strati di max-pooling sono stati sostituiti da un nuovo tipo di layer, detto di *batch normalization*.

Strati di Batch Normalization Questi layer si interpongono tra gli strati convoluzionali e quelli di attivazione e consentono la normalizzazione degli output ad ogni strato latente della rete. La normalizzazione delle batch, ossia l'insieme di campioni che vengono contemporaneamente dati in pasto alla rete, $B = \{x_1....x_m\}$ viene effettuata tramite la formula:

$$\hat{x_i} = \frac{x_i - \mu B}{\sqrt{\sigma_B^2 + \epsilon}}. \tag{3.3}$$

Il valore normalizzato viene poi moltiplicato e sommato per i parametri γ e β, che la rete apprenderà durante la fase di training. Il fatto che ad ogni step vengano introdotti i due valori variabili σ_B^2 e μB, che rappresentano la deviazione standard e la media di ogni mini-batch, fa sì che ogni layer acquisisca una certa robustezza ad una grande variabilità dei suoi input. Nella fase di test, la rete non calcolerà più i valori di media e deviazione standard, ma utilizzerà quelli appresi durante l'addestramento.

Un'ulteriore differenza tra le due architetture sta nel fatto che Resnet-50 non presenta alcuno strato interamente connesso in fondo alla rete, ad eccezione dello strato di output finale da 8631 nodi, con attivazione softmax. La figura 3.9 mostra nel dettaglio l'architettura di questa rete, evidenziando in rosso i blocchi di convoluzione ed in blu i blocchi identità.

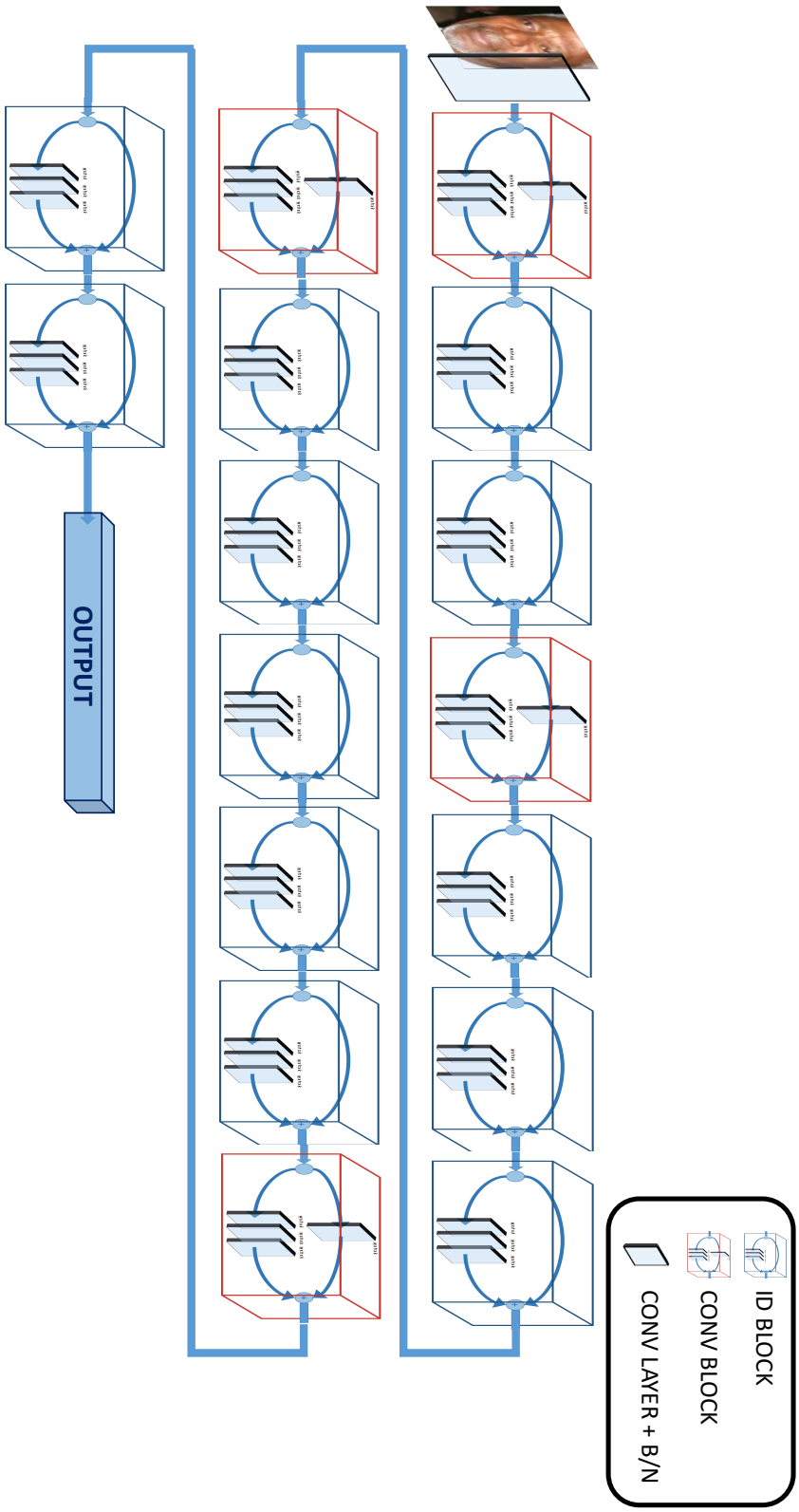

Figura 3.9. Architettura della rete Resnet-50.

Il training di questo modello è stato effettuato sulla versione più recente del dataset VggFace (VggFace2) [26], che comprende una media di 360 immagini per

ognuno dei 9131 individui presenti nel database (8631 per il training e 500 per la fase di testing). Il fatto che il numero di classi presenti in VGGFace2 siano circa quattro volte superiori rispetto alla prima versione del database presuppone che le features estratte da questo modello siano molto più generiche rispetto a quelle estratte da VGG-16 e, di conseguenza, ci siamo da subito aspettati che i risultati di Resnet-50 fossero più accurati.

3.4 Risoluzione tramite feature extraction

In questa sezione vengono descritti nel dettaglio i passaggi necessari per l'implementazione di un sistema di automatic kinship verification, a partire dai vettori di features estratti dalle due reti Vgg16 e Resnet50. La figura 3.10 schematizza la pipeline seguita in questa fase: nella parte in alto, vengono illustrati i passaggi necessari per la fase di addestramento del sistema, nella parte bassa, invece, viene riportata la fase di classificazione.

3.4.1 Estrazione delle caratteristiche

L'estrazione di caratteristiche, in inglese *feature extraction*, è una pratica molto comune nell'ambito del riconoscimento e l'elaborazione di immagini. Questa tecnica può essere vista come una forma di riduzione della dimensionalità, in quanto permette di descrivere tramite un singolo vettore (detto anche descrittore o feature vector), le principali informazioni presenti all'interno di un'immagine.

Il concetto di feature extraction non è necessariamente correlato a quello di rete neurale artificiale: è infatti possibile derivare un vettore di caratteristiche utilizzando diversi algoritmi scritti "a mano" (hand-crafted), come HOG, SIFT o LBP. I vettori di caratteristiche di questo tipo vengono definiti descrittori tradizionali, in contrapposizione con i descrittori profondi (*deep features*), estratti a partire da una ConvNet. La principale differenza sta nel fatto che nel primo caso è il progettista, a seconda dell'algoritmo utilizzato, a scegliere quali informazioni salvare nel vettore di caratteristiche, mentre nel secondo basta dare l'immagine in pasto alla rete e sarà poi lei a scegliere quali sono le informazioni più rilevanti, in base al problema da risolvere.

A partire da un'unica CNN è possibile, rimuovendo uno o più strati in testa alla rete, ottenere diversi descrittori profondi, a seconda del livello del layer da cui viene effettuata l'estrazione (fig 3.11). Nel nostro caso, partendo dal presupposto che le nostre reti sono state pre-addestrate per problemi di riconoscimento facciale, abbiamo inizialmente pensato di rimuovere dalle due CNNs solamente lo strato di output finale, che si occupa di distinguere tra le persone all'interno dello specifico database usato per il training, in modo tale da ottenere un feature vector che descriva, in generale, le principali caratteristiche del volto. Per quanto riguarda la rete Vgg-16 abbiamo ottenuto un descrittore da 4096 elementi, come il numero di nodi presenti nell'ultimo strato fully-connected (fc7), mentre per Resnet-50 un feature vector da 2048 elementi, ottenuto dall'operazione di average-pooling dell'output dell'ultimo blocco identità.

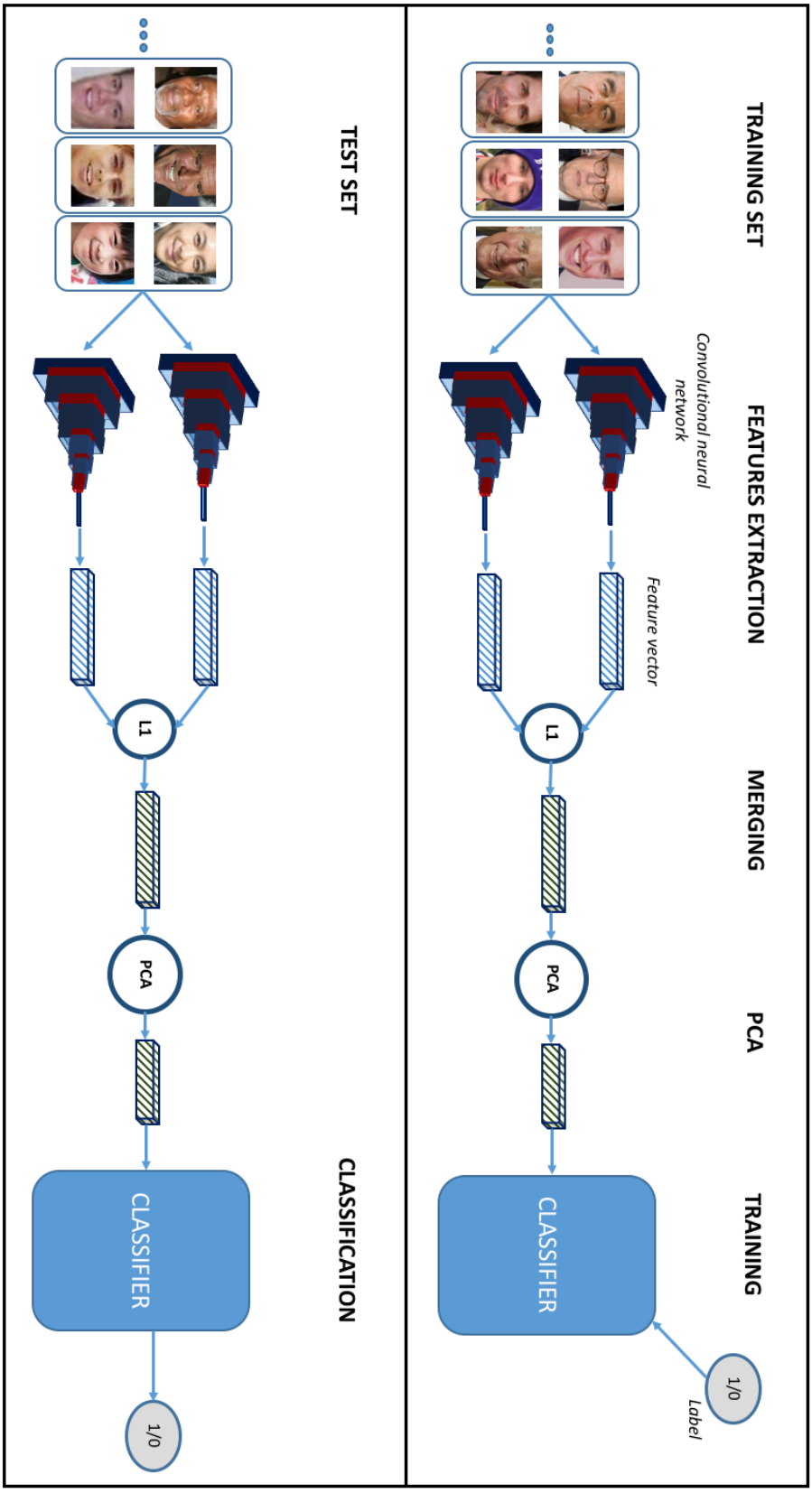

Figura 3.10. Pipeline seguita per l'implementazione tramite features extraction. In alto è rappresentata la fase di training, in basso quella di classificazione.

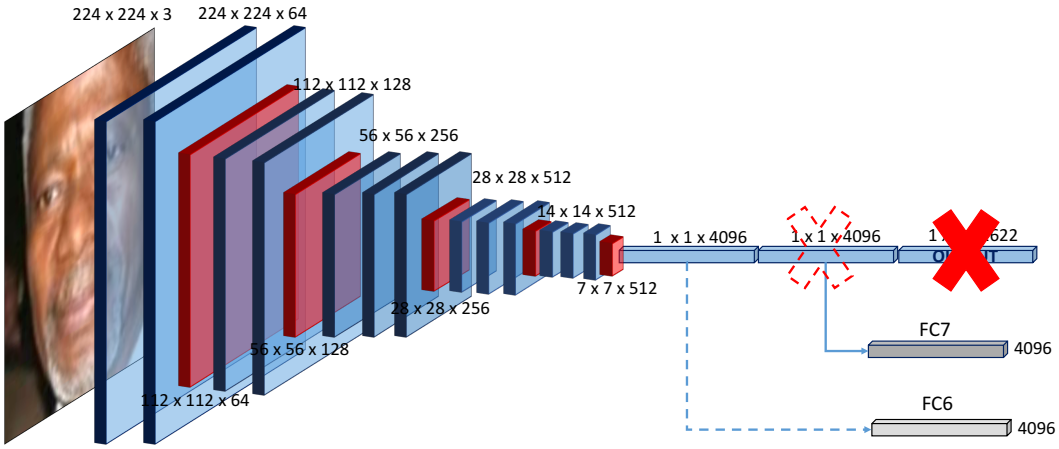

Figura 3.11. Esempio di estrazione delle features dai due fc-layers della rete Vgg-16 .

Nel corso della tesi abbiamo provato anche a testare le prestazioni dei feature vectors di diversi livelli in entrambe le reti. Come vedremo nell'apposito capitolo, i descrittori di secondo livello, intesi come l'output del penultimo layer fully-connected (fc6) in Vgg-16 e quello del terzultimo blocco identità nella rete Resnet-50, si sono rivelati molto competitivi, a differenza dei descrittori più profondi, con i quali abbiamo ottenuto risultati decisamente meno accurati.

3.4.2 Fusione delle features a più livelli

Nella fase precedente sono state valutate una ad una le prestazioni dei vari descrittori, estratti da diversi livelli della rete. Ognuno di questi vettori memorizza informazioni differenti presenti all'interno della stessa immagine: i layer più profondi individuano le features di basso livello, tipicamente angoli e bordi, mentre gli ultimi strati fully-connected permettono di rilevare le features di alto livello specifiche di ogni problema, nel nostro caso le informazioni riguardanti il volto. Sebbene i descrittori di basso livello, presi singolarmente, si siano rilevati poco precisi, abbiamo pensato che le loro informazioni, combinate con quelle dei descrittori di alto livello, potessero comunque dimostrarsi utili al fine della classificazione. Per questo motivo, abbiamo deciso di provare a concatenare i descrittori dei diversi layer, in modo da raccogliere in un unico vettore le features di basso, medio e alto livello.

Per quanto riguarda l'architettura Vgg-16, prendendo spunto dal lavoro "Good Practice in CNN Feature Transfer" di *Zheng et al.* [48], abbiamo estratto le features

sia dai due livelli fully-connected, sia dai cinque precedenti layer di convoluzione. A differenza degli strati f-c, i convolutional layers forniscono in output dei descrittori con un numero di features estremamente elevato, di dimensione *[w x h x n]*, dove w e h rappresentano le dimensioni dell'immagine in input o del volume precedente, mentre n sta per il numero di filtri presenti nello strato convoluzionale. Per procedere è stato quindi necessario effettuare un'operazione di average pooling su tutti i descrittori volumetrici, in modo da ridurre le features ad un singolo vettore di dimensione *[1 x 1 x n]* per ogni livello. A questo punto abbiamo concatenato i diversi descrittori, ottenendo in output un feature vector da 64 + 128 + 256 + 512 + 512 + 4096 + 4096 = 9664 elementi (fig 3.12).

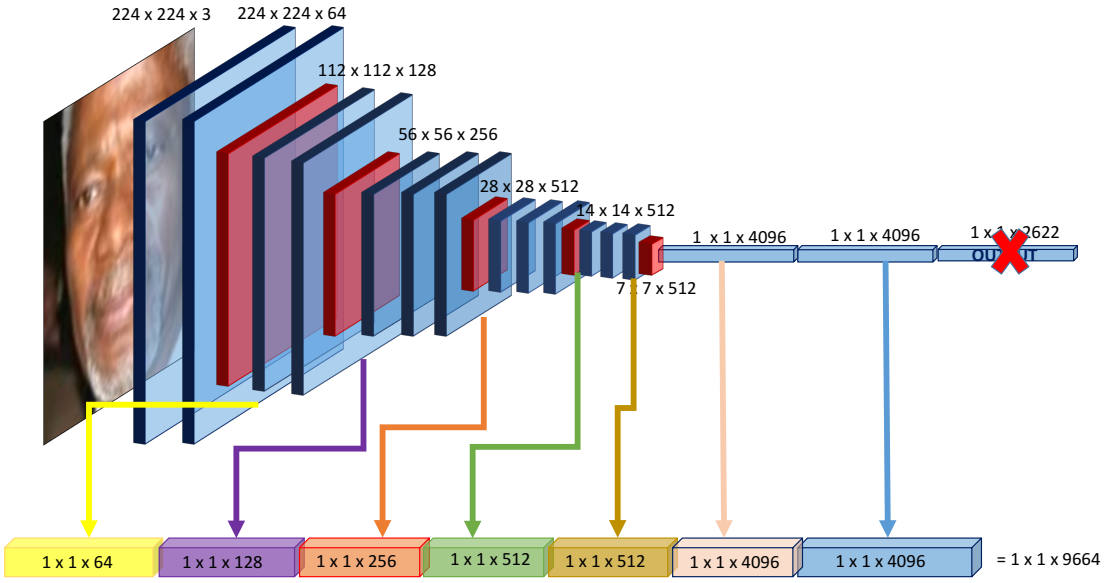

Figura 3.12. Schema di estrazione delle features a più livelli dalla rete Vgg16.

Nel caso della rete Resnet-50, non essendo possibile estrarre le features da ognuno degli oltre 50 strati convoluzionali, è stato scelto di considerare solamente i descrittori estratti dall'ultimo di una serie di blocchi identità, ossia i tre blocchi che precedono un blocco di convoluzione, più l'ultimo strato convoluzionale (fig 3.13). Anche in questo caso abbiamo prima effettuato l'average pooling delle features di basso livello, per poi concatenarle in un unico descrittore da 256 + 512 + 1024 + 2048 = 3840 elementi.

Nel corso di questa tesi ci riferiremo ai due descrittori concatenati, descritti in precedenza, con i nomi di Vgg16_fused e Resnet50_Fused. Essendo 9664 un numero di features difficilmente gestibile con le risorse a nostra disposizione, abbiamo deciso di concentrarsi sull'effetto della fusione delle features di diverso livello nella rete

Resnet-50, limitandoci a testare il descrittore Vgg-16_fused soltanto in determinate applicazioni.

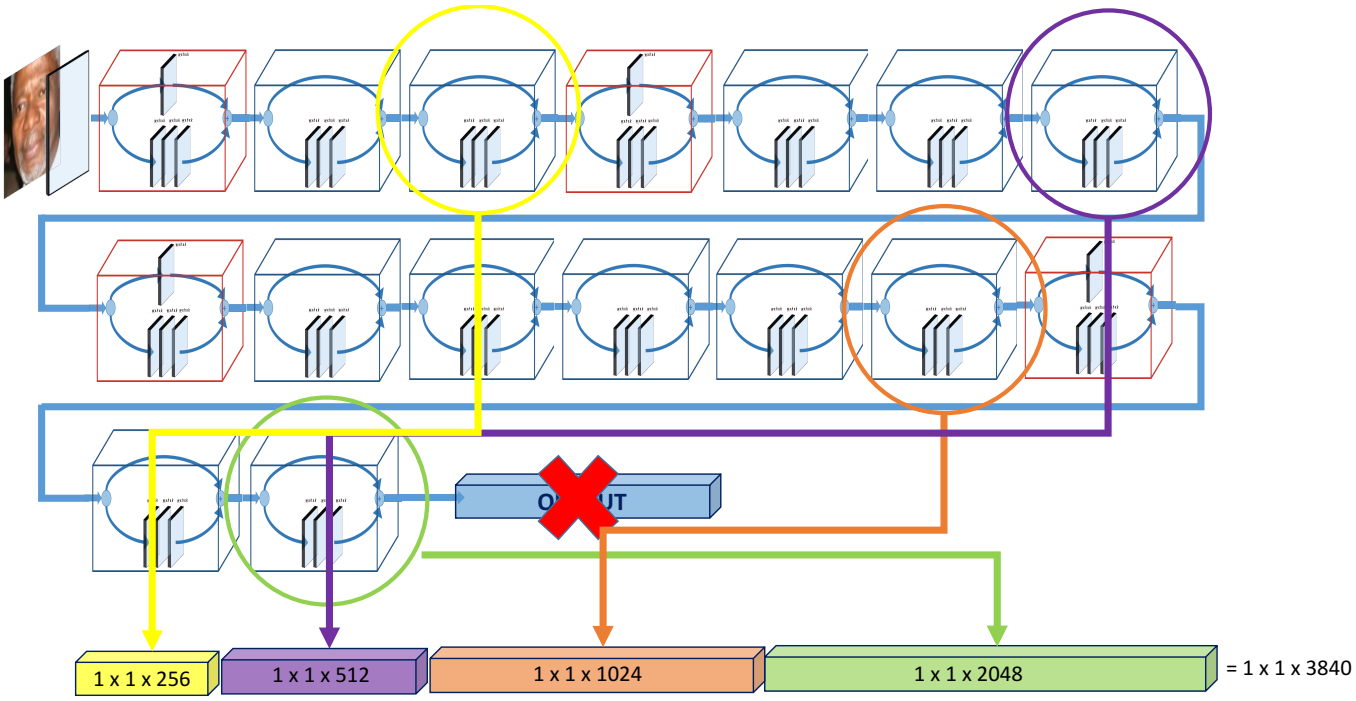

Figura 3.13. Schema di estrazione delle features a più livelli dalla rete.

3.4.3 Normalizzazione

Una volta ottenuto un descrittore per ogni immagine presente nel database, occorre procedere con la normalizzazione dei dati, in modo da limitare il valore assumibile dalle singole features ad un intervallo compreso tra 0 e 1. Questa è un'operazione necessaria per la maggior parte degli algoritmi di machine learning, in quanto permette, implicitamente, di assegnare uno stesso peso a tutte le features. Senza normalizzare i descrittori, infatti, le dimensioni in cui le features possono assumere i valori più alti, avrebbero un peso maggiore nella classificazione rispetto alle altre e, di conseguenza, verrebbero perse molte delle informazioni riguardanti le dimensioni con i range inferiori.

Per la normalizzazione dei dati, abbiamo scelto di utilizzare il metodo *Normalizer()* della classe *preprocessing* di *Scikit-Learn*. Questa funzione fa sì che ogni campione, che abbia almeno un componente diverso da zero, venga riscalato, indipendentemente dagli altri campioni, in modo tale che la sua norma abbia dimensione

unitaria. Tramite il parametro *norm* è possibile considerare diversi tipi di norma, nel nostro caso abbiamo deciso di utilizzare quella Euclidea (L2-norm):

$$\| x \|_2 = \sqrt{\left(\sum_{i=1}^{N} {x_i}^2\right)} \tag{3.4}$$

3.4.4 Merging dei descrittori

Dopo aver normalizzato i due vettori di caratteristiche di ogni coppia di immagini facciali, è necessario analizzarli per individuare eventuali legami. Inizialmente abbiamo provato semplicemente a calcolare la distanza tra i due descrittori, prima tramite la formula della distanza euclidea (L2):

$$L2(p,q) = \sqrt{\sum_{k=1}^{N} (p_k - q_k)^2}, \tag{3.5}$$

quindi con quella della distanza coseno:

$$d(p,q) = 1 - \frac{p \cdot q}{\| p \|_2 \cdot \| q \|_2} = 1 - \frac{\sum_{k=1}^{N} p_k \cdot q_k}{\sqrt{\sum_{k=1}^{N} p_k^2}\sqrt{\sum_{k=1}^{N} q_k^2}}. \tag{3.6}$$

In entrambi i casi il risultato dell'operazione è un singolo numero, che rappresenta la distanza tra i due vettori: tanto più sarà alto il valore, tanto più è improbabile che ci sia un legame di parentela tra i due volti. Per ognuno degli undici training set, corrispondenti ai diversi gradi di parentela, il classificatore ha calcolato la soglia che meglio dividesse le coppie positive da quelle negative; a questo punto tale soglia è stata utilizzata per la classificazione, kin o non-kin, tra le coppie di immagini facciali presenti nei diversi test set: tutte le coppie con distanza inferiore alla soglia sono state classificate positive, le altre negative (fig 3.14). Questo metodo, in particolare utilizzando Resnet-50 come base-network, ci ha permesso di ottenere risultati piuttosto accurati, per alcune parentele molto simili a quelli raggiunti dallo stato dell'arte.

Nonostante la bontà dei risultati ottenuti calcolando le distanze (euclidea e coseno) tra i due descrittori, abbiamo pensato che il fatto di utilizzare un solo valore per descrivere due vettori di migliaia di elementi potesse comportare la perdita di alcune importanti informazioni. Per questo motivo abbiamo deciso di testare due ulteriori strategie di merging (fig 3.15), prima semplicemente concatenando i due feature vectors della coppia, quindi calcolandone, elemento per elemento, la differenza in valore assoluto (L1). In questo caso, per ogni training set, abbiamo ottenuto in

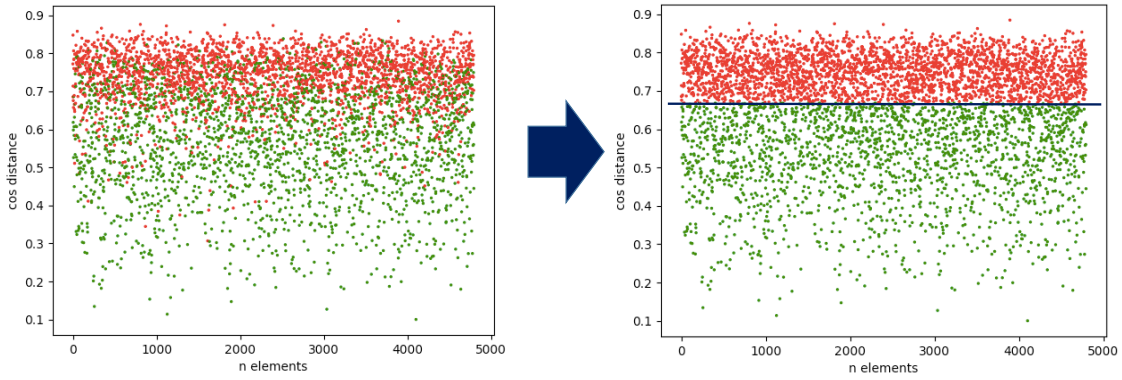

Figura 3.14. Operazione di sogliatura con distanza coseno nel test set SS: nell'immagine di sinistra viene mostrata la reale distribuzione delle coppie (in verde quelle positive, in rosso quelle negative), in quella di destra la classificazione finale

uscita una serie di descrittori multidimensionali, da *n*2* elementi nel primo caso e da *n* elementi nel secondo, i quali sono stati successivamente dati in pasto ad un classificatore lineare per addestrarlo a riconoscere i vettori corrispondenti ad una coppia di parenti. La tecnica del concatenamento dei due vettori, tuttavia, è stata presto abbandonata sia per una questione legata alla scarsa precisione dei risultati, sia per l'eccessiva quantità di tempo impiegata per addestrare i classificatori: abbiamo infatti calcolato che il tempo necessario per allenare un classificatore SVM (descritto in seguito) con il vettore concatenato è di gran lunga maggiore rispetto a quello impiegato con il vettore differenza L1.

3.4.5 PCA

Prima di dare in pasto al classificatore lineare i vettori risultanti dalle operazioni di merging, è possibile effettuare un passaggio ulteriore. Come abbiamo visto nel caso della concatenazione, il fatto di utilizzare descrittori di grandi dimensioni non è garanzia di precisione dei risultati. Anzi, molto spesso, gran parte degli elementi presenti nei feature vectors sono completamente inutili al fine della classificazione finale e risultano persino dannosi durante l'addestramento di una rete o, come nel nostro caso, di un classificatore lineare. Per ridurre questo rumore, abbiamo utilizzato una tecnica definita analisi in componenti principali o, più comunemente, *PCA* dall'inglese principal component analysis.

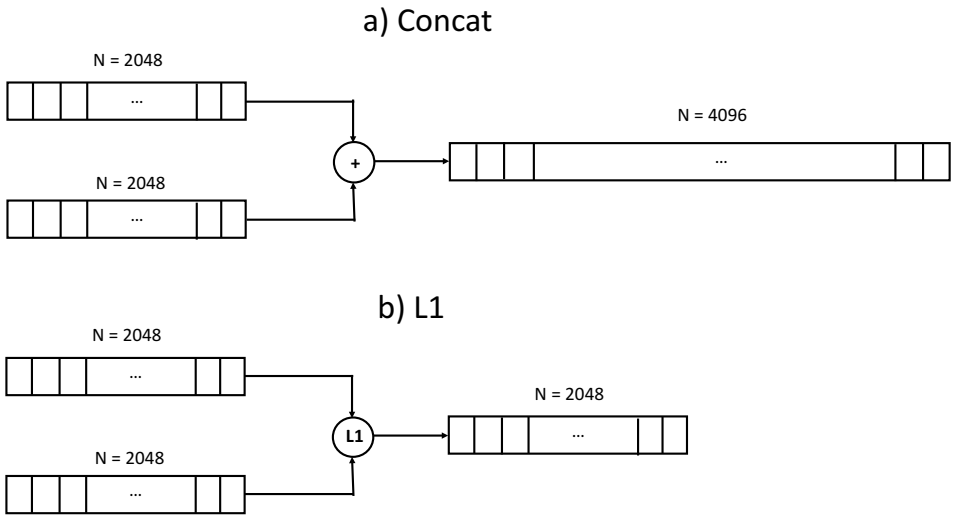

Figura 3.15. Concatenazione (a) vs Distanza L1 (b).

La PCA è una tecnica di features reduction, molto utile nel caso in cui il dataset abbia una certa ridondanza. Lo scopo primario è quello di trasformare il nostro set di descrittori di dimensione M x N, dove M corrisponde al numero di campioni e N al numero di features, in un nuovo insieme di variabili con lo stesso numero di campioni, ma con un numero di features inferiore. Per farlo i dati di partenza vengono proiettati in un nuovo sistema di riferimento, in cui viene massimizzata la varianza lungo gli assi: la variabile con varianza maggiore verrà proiettata sul primo asse, la seconda sul secondo asse e così via (fig 3.16).

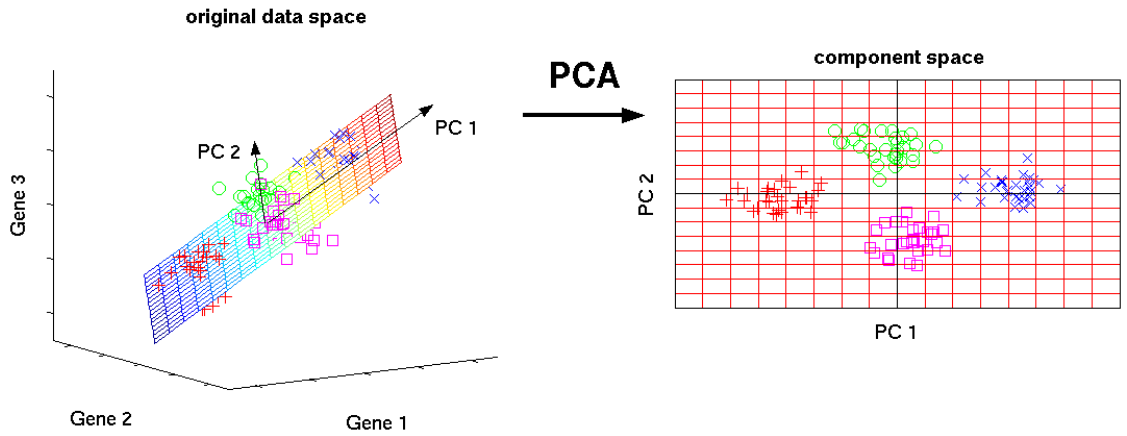

Figura 3.16. Esempio di funzionamento della Principal Component Analysis nel caso di descrittori tridimensionali.

A questo punto è possibile scegliere di analizzare unicamente i componenti più significativi, semplicemente non considerando le dimensioni con varianza minore.

La classe *decomposition* di scikit learn mette a disposizione il metodo *PCA()*, per sottoporre un dataset a principal component analysis. Il parametro *n_components* permette di scegliere il numero di componenti che si vogliono considerare, o come valore specifico, oppure, se indicato come un numero compreso tra 0 e 1, come la percentuale di varianza del dataset originario che si vuole mantenere. Solitamente, si usa scegliere un numero di componenti principali tale per cui venga mantenuta una percentuale della varianza originale compresa tra l'85% e il 99%. Abbiamo notato che questo parametro influiva significativamente sia sull'accuratezza dei risultati, sia sul tempo impiegato per il training del classificatore. Dopo diversi tentativi è stato deciso di utilizzare, per tutti i dataset, una percentuale del 95%, come miglior compromesso tra precisione e tempistiche.

Nella tabella 3.1 viene mostrata la riduzione del numero di features, in seguito al merging con distanza L1 e alla successiva analisi in componenti principali degli undici dataset, sia nel caso di VGG-16 come rete di base, sia nel caso di Resnet-50. In entrambe le reti è stato sottoposto a PCA il descrittore estratto dall'ultimo layer della rete. Ovviamente, è necessario utilizzare la tecnica PCA sia nei diversi training set, sia nei test set, in modo tale che i due insiemi utilizzino lo stesso numero di features.

A differenza degli step precedenti, la PCA non è un passaggio obbligatorio per la risoluzione del nostro problema, tuttavia ci ha permesso di accorciare notevolmente le tempistiche degli step successivi, soprattutto in quei casi in cui il numero di features era particolarmente elevato (es. concatenazione di due descrittori vgg16).

3.4.6 Classificazione

Una volta ridotta la dimensione dei vettori di caratteristiche è possibile procedere con l'addestramento di un classificatore, in modo da poter suddividere le coppie positive e negative presenti all'interno dei diversi test set. È bene ricordare che durante la fase di training è necessario dare in pasto al classificatore sia il descrittore ottenuto dai passaggi precedenti, sia la rispettiva etichetta (0/1): in questo modo sarà il classificatore stesso a trovare il confine che meglio separa le coppie di training positive da quelle negative e ad utilizzarlo per suddividere le coppie di test (fig 3.17).

Nell'ambito del machine learning esistono diversi tipi di classificatori; in questa tesi abbiamo provato ad utilizzarne tre tipologie differenti: una macchina a vettori di supporto (SVM), un'implementazione dell'algoritmo degli alberi di gradient boosting (XGBoost) ed infine una rete neurale artificiale.

	VGG-16 (4096)		RESNET-50 (2048)	
Parentela	Features dopo PCA	% riduzione	Features dopo PCA	% riduzione
BB	2172	46,97%	1174	42,68%
SS	2183	46,70%	1224	40,23%
SIBS	2292	44,04%	1280	37,50%
FD	2322	43,31%	1331	35,01%
FS	2286	44,19%	1286	37,21%
MD	2317	43,43%	1300	36,52%
MS	2264	44,73%	1282	37,40%
GFGD	1210	70,46%	745	63,62%
GFGS	1017	75,17%	697	65,97%
GMGD	1125	72,53%	737	64,01%
GMGS	983	76,00%	678	66,89%

Tabella 3.1. Numero di features e percentuale di riduzione della dimensione dei descrittori nei diversi dataset dopo PCA (parametro n_components = 95%)

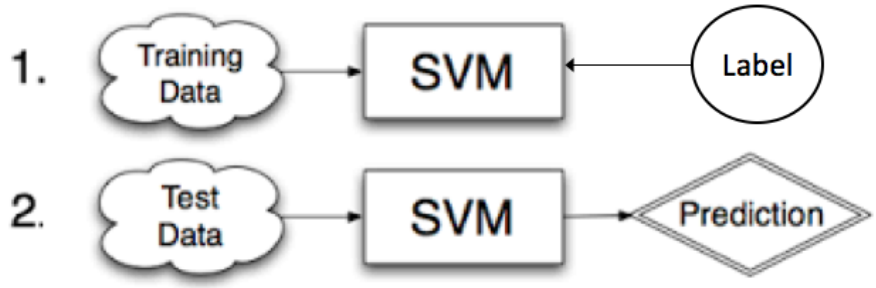

Figura 3.17. Fasi di training e testing di un classificatore SVM.

SVM

Le macchine a vettori di supporto, o SVM (support vector machines), rappresentano una particolare tipologia di classificatore lineare, in grado di trovare l'iperpiano ottimale per la separazione delle diverse classi, ossia quell'iperpiano per cui il margine, inteso come distanza tra il piano e il campione più vicino, sia equidistante per entrambe le classi. Nel caso in cui i campioni siano caratterizzati da una sola feature, come nel caso di merging tramite distanza L2 e coseno, l'iperipiano non sarà altro che una soglia, in due dimensioni sarà una retta, in tre un piano e così via. Nel nostro caso, avendo a che fare con descrittori da migliaia di features, le due classi

sono state suddivise da un'iperpiano n-dimensionale, dove n corrisponde al numero di features meno uno.

Nell'immagine 3.18 vengono mostrati due esempi di iperpiani calcolati da una support vector machine con kernel lineare. Come si può notare, in entrambi i casi l'iperpiano non suddivide correttamente le due classi, in quanto un campione per classe è stato etichettato in maniera errata. Per riuscire a classificare correttamente dati con comportamento non-lineare è necessario utilizzare una funzione di kernel differente. Con il termine kernel ci si riferisce alla funzione applicata ad ogni istanza, per mappare i dati originali non-lineari in un nuovo spazio in cui diventano separabili linearmente (fig 3.19); è possibile utilizzare diverse funzioni di kernel, le più comuni sono le seguenti:

- Lineare: $K(x, y) = x \cdot y$
- Polinomiale: $K(x, y) = (x \cdot y)^d$
- Gaussian Radial Basis function (rbf): $K(x, y) = exp(-|x - y|^2)/(2\sigma^2)$
- Sigmoid: $K(x, y) = tanh(\kappa x \cdot y - \delta)$

Sebbene le funzioni di kernel non-lineari permettano di risolvere problemi decisamente complessi, allo stesso tempo comportano un aumento esponenziale delle tempistiche per l'addestramento della support vector machine. In questa tesi è stato deciso di utilizzare prevalentemente un kernel di tipo lineare, tramite la funzione

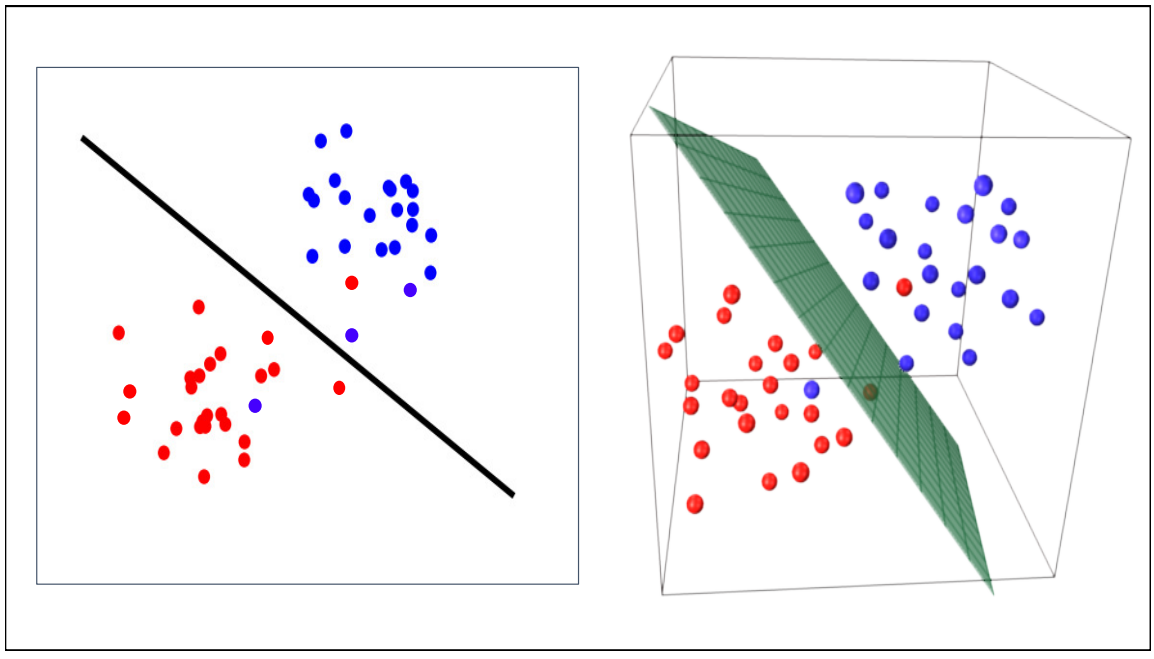

Figura 3.18. Esempio di iperpiano svm con kernel lineare in due e tre dimensioni.

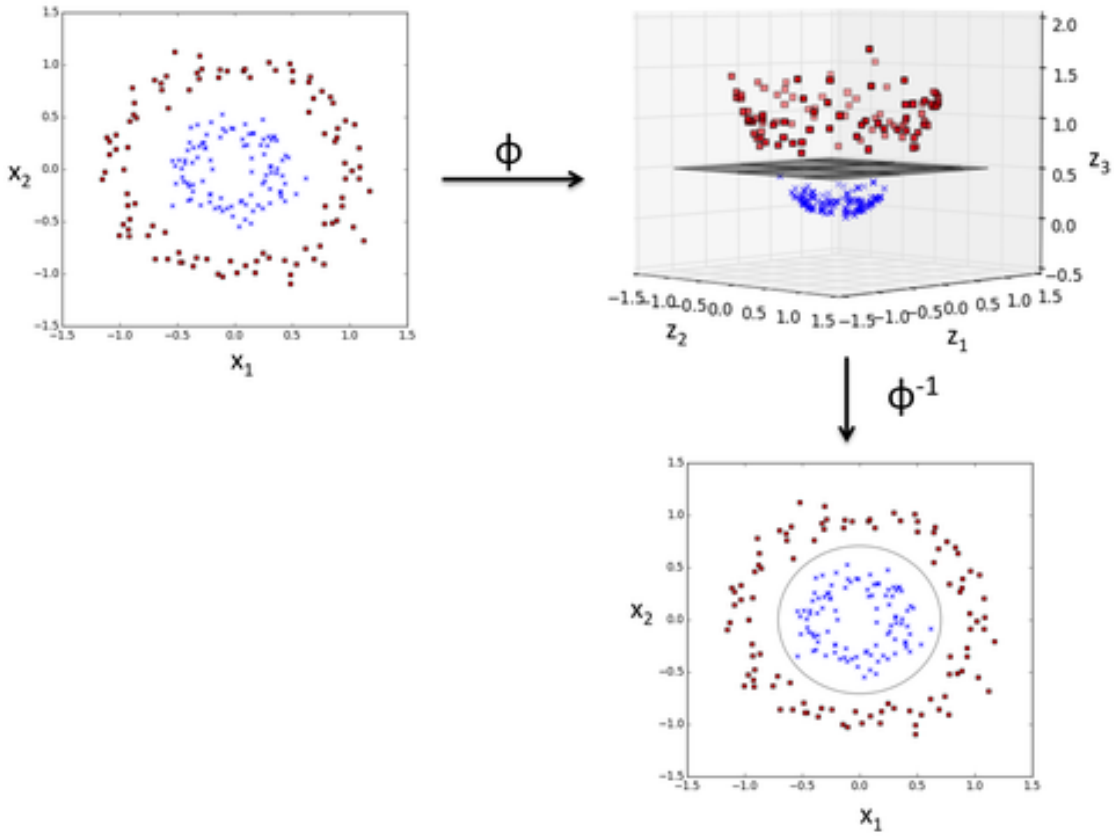

Figura 3.19. Esempio di svm con kernel non-lineare (fonte: KDnuggets).

LinearSVC() del framework *scikit-learn*; in seguito abbiamo provato a testare, in particolari condizioni, anche un kernel di tipo rbf, in modo da non fare nessun tipo di assunzione sulla distribuzione dei dati.

XGBoost

Il secondo tipo di classificatore testato nella nostra pipeline è il metodo *XGBClassifier()* del package XGBoost (dall'inglese eXtreme Gradient Boosting), un framework di Python per la gestione degli algoritmi di potenziamento del gradiente. L'idea che sta alla base della tecnica del gradient boosting è quella di implementare un buon classificatore, combinando le stime di una serie di classificatori più semplici e deboli, tipicamente alberi di decisione con un basso numero di split (fig 3.20). La particolarità, rispetto ad algoritmi simili come la Random Forest, sta nel fatto che la sequenza di modelli deboli viene costruita in modo adattivo, modificando, durante la fase di training, il peso delle singole istanze ad ogni iterazione, in modo da assegnare una probabilità maggiore ai dati mal classificati nel passo precedente.

Nella versione classica del gradient boosting, nel caso in cui il numero di alberi considerati sia troppo alto, c'è il rischio che il modello si sovradatti ai dati. Questo

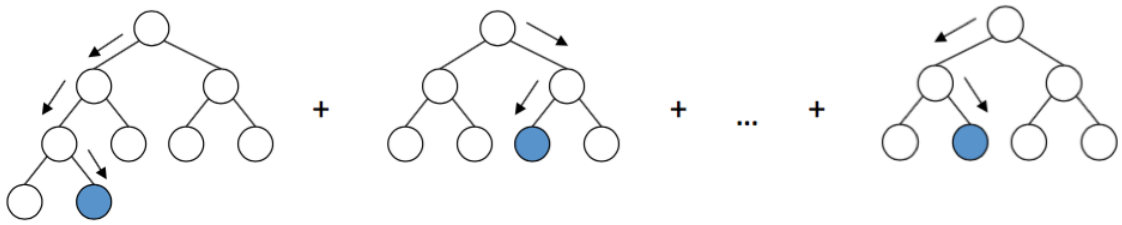

Figura 3.20. Esempio di gradient boosting (fonte: Medium).

rischio viene estremamente ridotto con l'implementazione di XGBoost, che utilizza una particolare funzione obiettivo regolarizzata, in modo da evitare il sovradattamento. Il classificatore XGBoost, inoltre, utilizza un metodo di stima approssimato, diverso dal metodo euristico classico dei precedenti algoritmi di discesa del gradiente, che permette di ottenere ottimi risultati in tempi e costi computazionali decisamente inferiori.

Rete neurale

Infine, abbiamo provato a sostituire il classificatore lineare con una rete neurale a due strati interamente connessi, rispettivamente da 512 e 128 neuroni ciascuno (fig 3.21).

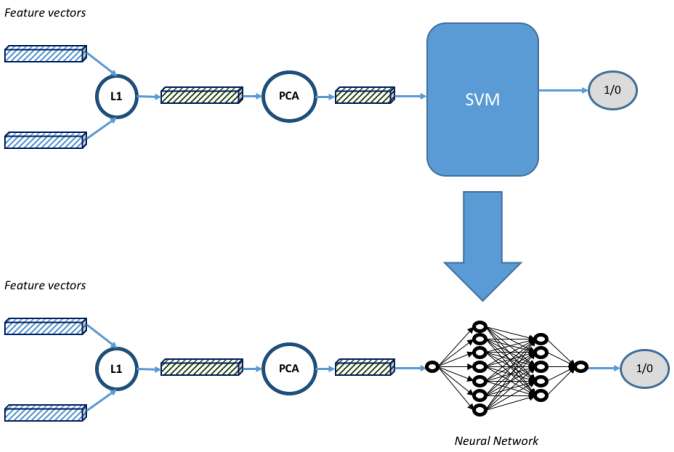

Figura 3.21. Classificazione tramite rete neurale artificiale.

3.5 Risoluzione tramite fine-tuning della rete

Nei paragrafi precedenti sono stati descritti gli step necessari per l'implementazione di un sistema di riconoscimento automatico delle parentele a partire dall'estrazione di vettori di caratteristiche da una rete preallenata. Nonostante i risultati ottenuti con questa tecnica si siano rivelati soddisfacenti, abbiamo deciso di provare a seguire anche una strada leggermente diversa: a partire dalle stesse reti preallenate testate in precedenza, abbiamo proseguito la fase di training utilizzando direttamente il database FIW, in modo tale che i pesi delle reti venissero modificati automaticamente per adattarsi al nuovo problema di kinship verification.

3.5.1 Finetuning

Come accennato precedentemente, addestrare una rete da zero è molto complicato sia per una questione di tempistiche (il training from scratch di una rete complessa su database di grandi dimensioni può richiedere giorni-settimane di computazione, anche su GPU di ultima generazione), sia per la difficoltà di trovare dataset di dimensioni adeguate. Allo stesso modo, estrarre semplicemente le features da una rete preallenata senza prima adattarla al nuovo problema, come è stato fatto in precedenza, potrebbe rivelarsi non sufficiente per raggiungere l'accuratezza desiderata. Il fine-tuning può essere visto come una via di mezzo tra queste due strade: a partire da una rete preallenata su un problema simile, si sostituisce lo strato di output con un nuovo livello di output softmax, adeguandolo al nuovo numero di classi; a questo punto, si eseguono nuove iterazioni di training, in modo da ottimizzare i pesi rispetto alle caratteristiche del nuovo dataset che, in questo caso, può anche non avere grandi dimensioni.

Una pratica molto comune quando si effettua il fine-tuning di una rete neurale è quella di 'congelare' i pesi degli strati convoluzionali più profondi, in modo che non vengano modificati durante la nuova fase di training. Questo perchè, come visto nel paragrafo 3.4.2, i primi layer di convoluzione catturano informazioni di tipo universale, come curve e contorni, che potrebbero essere rilevanti anche nel nuovo problema. In questo modo viene ridotto il numero di parametri da riallenare e, di conseguenza, viene ridotto anche il tempo di computazione. In *Keras*, è possibile modificare lo stato, allenabile o non-allenabile, di ogni layer tramite il parametro booleano *Trainable*. Per quanto riguarda le due architetture prese in esame in questa tesi, nella rete Vgg-16, inizialmente, è stato scelto di congelare tutti gli strati convoluzionali eccetto i due layer fully-connected finali (fig 3.22); in seguito, dopo aver notato che

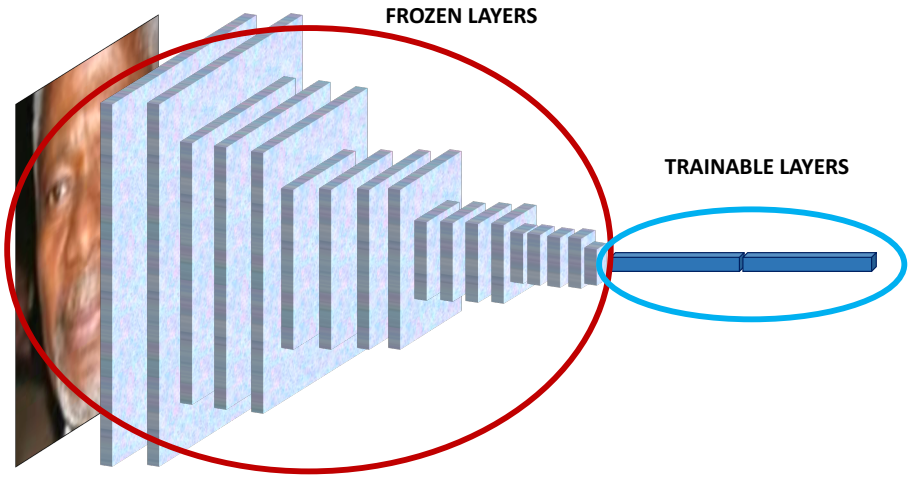

Figura 3.22. Congelamento degli strati convoluzionali per il finetuning della rete Vgg-16.

il descrittore estratto dal penultimo strato della rete aveva un comportamento migliore, abbiamo rimosso dalla rete anche l'ultimo fc layer, riducendo ulteriormente il numero di parametri modificabili. Per quanto riguarda Resnet-50, invece, è stato rimosso dalla rete solo lo strato di output e sono stati resi non-allenabili tutti i restanti layer, eccetto quelli all'interno dell'ultimo blocco di convoluzione. In tabella 3.2 viene riportato il numero di strati 'congelati' e il numero di parametri allenabili per la fase di fine-tuning delle due reti.

Network	***frozen layers/ tot layers***	***trainable parametres/ tot parametres***
Vgg-16	19/21	102.764.544/117.479.232
Resnet-50	141/180	14.964.7362/23.561.152

Tabella 3.2. Effetto del congelamento sul numero di parametri allenabili nelle reti Vgg-16 e Resnet-50.

Rispetto alla classica implementazione del fine-tuning, in cui è sufficiente rimuovere dalla rete originale lo strato di output e sostituirlo con uno nuovo di diversa dimensione, il transfer learning da face recognition a kinship verification richiede un passaggio ulteriore: nel primo caso, infatti, la rete riceve una singola immagine per volta, mentre nel secondo è necessario dare in input alla rete entrambe le immagini contemporaneamente. Per effettuare il finetuning delle due CNNs direttamente sugli undici training set del database Families in The Wild, è stato quindi necessario modificare leggermente l'architettura delle reti.

3.5.2 Reti Neurali Siamesi

Come detto nel paragrafo precedente, sia Vgg-16 che Resnet-50 sono state progettate per ricevere in input una sola immagine per volta e per classificarla in base ai volti presenti nei rispettivi database di training; per il nostro sistema di riconoscimento automatico, invece, abbiamo bisogno che la rete riceva più immagini nello stesso momento, in modo da trovare informazioni utili riguardo l'eventuale parentela. Per far sì che le reti accettassero input multipli, abbiamo pensato di costruire un'implementazione di tipo siamese delle stesse.

Le reti siamesi sono una particolare tipologia di rete neurale, composte da due o più sottoreti identiche. Con il termine identiche non si intende soltanto il fatto che le sottoreti debbano condividere la stessa architettura, ma anche gli stessi pesi e gli stessi parametri (fig 3.23). Le reti neurali siamesi sono state introdotte per la prima volta nel 1993, per verificare l'autenticità di alcune firme scritte a mano [47] e, da allora, vengono frequentemente utilizzate in tutti quei problemi in cui è richiesto di trovare una somiglianza o, comunque, una certo tipo di relazione tra due oggetti comparabili. Un'architettura di tipo siamese, permette di risparmiare la metà dei parametri da memorizzare, rispetto ad una stessa rete in cui i due rami vengono addestrati distintamente. Inoltre, nel caso in cui il modello fosse composto da due sottoreti distinte, sarebbe necessario dare ogni coppia di immagini in pasto alla rete per due volte, invertendo ogni volta l'ordine dei due input, in modo tale che la rete riesca a classificare allo stesso modo le coppie invertite *(x1, x2)* e *(x2, x1)*. In una rete siamese, il fatto che i rami condividano gli stessi pesi rende questa operazione non necessaria. Per riassumere, possiamo dire che le reti neurali siamesi sono modelli che imparano a riconoscere cosa renda due input simili tra loro, in contrapposizione con le reti di classificazione, che imparano a riconoscere le caratteristiche che permettono di associare un input ad una determinata classe.

3.5.3 Triplet Loss

Una caratteristica importante delle reti neurali siamesi sta nella funzione di costo (*loss function*), che deve essere minimizzata durante l'allenamento del modello. Nelle architetture tradizionali, questa funzione rappresenta solitamente l'errore che intercorre tra l'output predetto dal modello e l'output atteso. Le reti siamesi, invece, utilizzano una funzione di perdita basata sulla distanza tra una coppia di input: in questo caso, la rete cerca di minimizzare la distanza tra le coppie appartenenti alla stessa classe e, allo stesso tempo, di massimizzare la distanza tra le coppie di classi differenti.

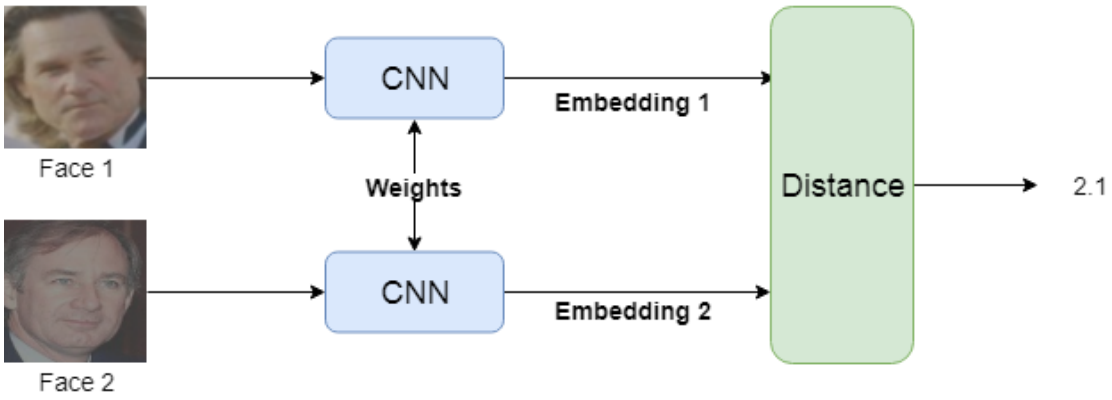

Figura 3.23. Esempio di architettura Siamese (fonte: Crafting adversarial faces).

Dopo aver provato a minimizzare la distanza euclidea tra le due immagini della coppia, è stato deciso di effettuare il finetuning delle reti utilizzando una loss function più efficiente. In particolare, abbiamo scelto una funzione di perdita denominata *triplet-loss* [45], definita dalla formula:

$$TripletLoss = \sum_{i=1}^{N} \left[\| f^a{}_i - f^p{}_i \|_2^2 - \| f^a{}_i - f^n{}_i \|_2^2 + \alpha \right]_+ \tag{3.7}$$

Per utilizzare questa funzione è necessario implementare una rete siamese a tre rami: il primo riceverà in input un'immagine di riferimento, denominata *anchor*, il secondo l'immagine di un parente della persona rappresentata nell'anchor (*positive embedding*), mentre l'ultimo ramo riceverà l'immagine di una terza persona, senza legami di parentela con le prime due (*negative embedding*), come rappresentato in figura 3.25. Durante il training, la rete imparerà a minimizzare la distanza tra anchor e positive embeddings e, allo stesso tempo, a massimizzare quella tra anchor e negative embeddings (fig3.24).

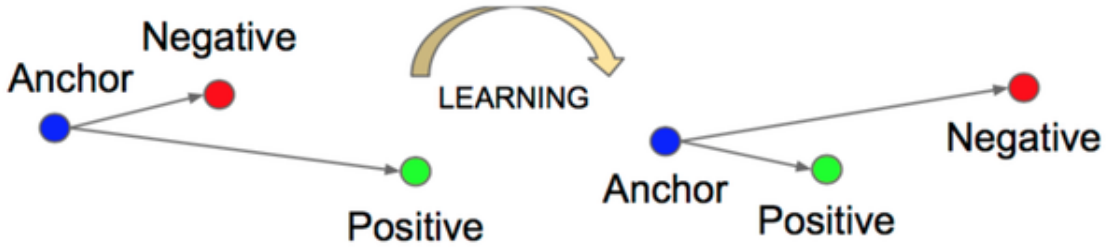

Figura 3.24. Triplet Loss (fonte: Towards Data Science.)

3.5.4 Training

Per l'addestramento di questo modello è stato prima necessario costruire dei nuovi set di training e validazione, in modo che non fossero più composti semplicemente da coppie di immagini, ma da triple del tipo [anchor, positive embedding, negative embedding]. Per farlo, prima abbiamo separato le coppie positive da quelle negative presenti in ogni set, quindi abbiamo considerato solamente le coppie di parenti come anchor e positive embedding del nuovo set. A questo punto abbiamo unito ad ogni coppia positiva una terza immagine, presa casualmente dal set di immagini negative, assicurandoci che la persona rappresentata non appartenesse alla stessa famiglia delle immagini della coppia.

Una volta definiti il modello e i nuovi set di training e validazione abbiamo potuto procedere con l'addestramento del sistema. In particolare, sono stati settati i seguenti parametri:

- *Learning Rate*: si tratta di un iper-parametro che controlla 'quanto' i pesi della rete vengono modificati per adattarsi al nuovo problema. Più basso è il parametro, più lento sarà il tempo di convergimento della rete. Un valore basso, tuttavia, permette di ridurre la possibilità che il modello si sovradatti ai dati (overfitting). Nel nostro caso, il parametro learning rate è stato impostato a $10^{-}5$.

- *Batch size*: questo parametro definisce il numero di campioni che verranno dati in pasto alla rete contemporaneamente. Molto spesso infatti, come nel nostro caso, non è possibile memorizzare l'intero training set in memoria, di conseguenza è necessario suddividere il dataset in un numero variabile di batch. Solitamente si usa scegliere come valore di questo parametro una potenza di 2, nel nostro caso $batch\_size = 128$.

- *Epochs*: questo parametro indica il numero di volte in cui una batch deve essere processata dal modello per considerare conclusa la fase di addestramento. Abbiamo deciso di impostare questo valore a 10 per tutte le tipologie di parentela.

Inoltre sono state utilizzate le due funzioni della classe *Callbacks* di *Keras*, *EarlyStopping()* e *ModelCheckPoint()*, da richiamare al termine di ogni epoch, rispettivamente per terminare l'addestramento in anticipo, nel caso si stia verificando una situazione di overfitting, e per salvare il modello migliore.

Una volta terminata la fase di fine-tuning delle reti, abbiamo ripetuto la pipeline descritta nel paragrafo 3.4, utilizzando i nuovi modelli come base-network per

l'estrazione dei descrittori. I risultati conseguiti con questa tecnica, come vedremo nell'apposito capitolo, ci hanno permesso di raggiungere i risultati più accurati; in particolare, con il fine-tuning della rete Resnet-50 sono stati conseguiti risultati molto simili a quelli dello stato dell'arte.

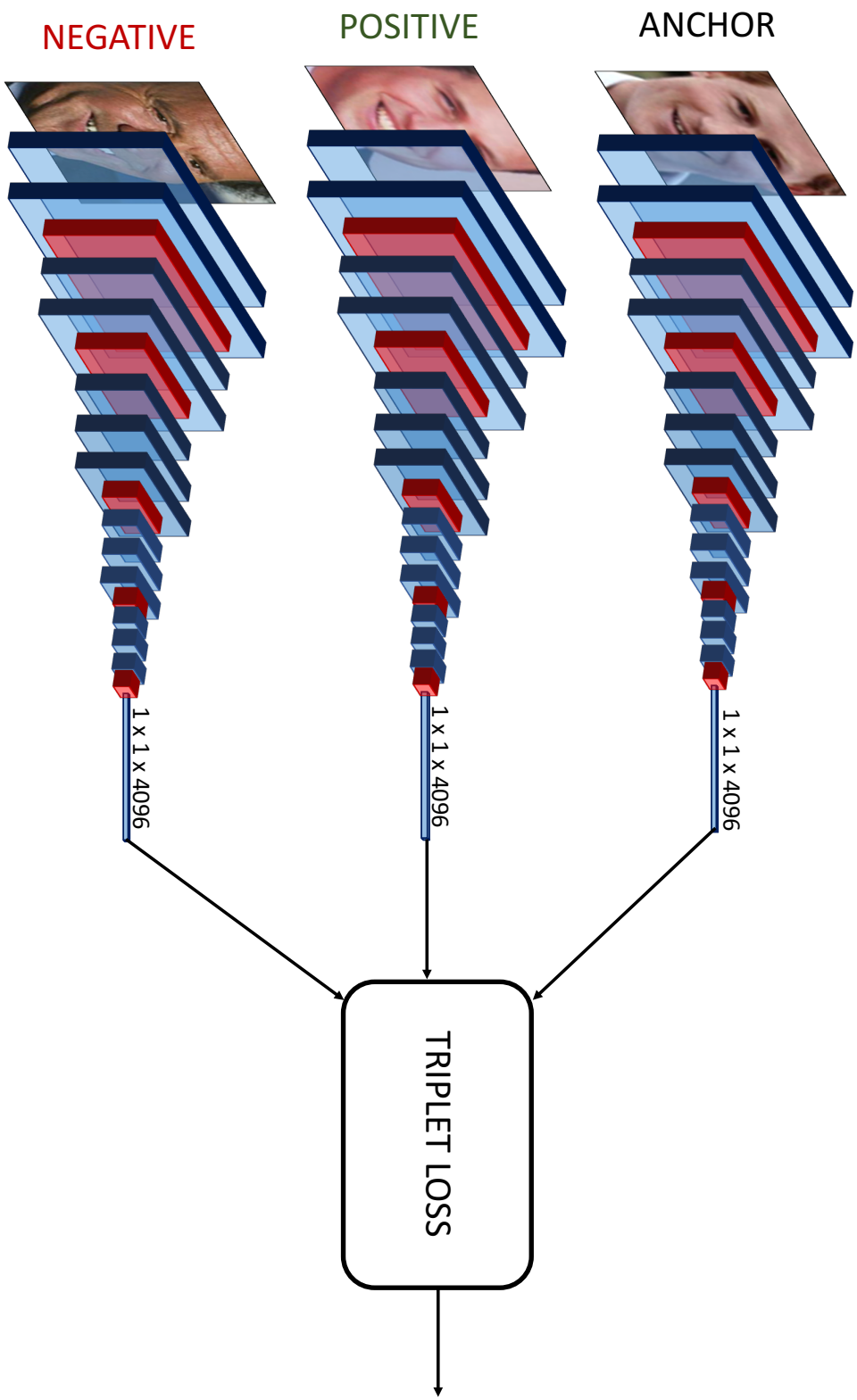

Figura 3.25. Architettura siamese a tre rami per la rete Vgg-16.

Capitolo 4

Risultati

In questo capitolo vengono riportati i risultati che abbiamo conseguito in questa tesi, con particolare attenzione al confronto tra le diverse strategie testate e all'impatto dei singoli step sulla precisione finale. Infine, i risultati migliori sono stati paragonati con quelli ottenuti dagli altri partecipanti alla sfida. La precisione dei risultati è indicata in termini di *accuratezza*, ossia la percentuale delle coppie, positive e negative, che sono state etichettate in maniera corretta dal sistema. Vale la pena ricordare che i risultati si riferiscono all'accuratezza ottenuta considerando solo le immagini presenti nei set di test (tab 4.1): le persone ritratte in queste immagini, non compaiono in nessuna delle fotografie utilizzate nella fase di training e, di conseguenza, sono totalmente nuove per il nostro sistema di riconoscimento.

Parentela	*# coppie di test*
BB	18196
SS	4796
SIBS	9716
FD	15040
FS	18166
MD	14394
MS	14806
GFGD	838
GFGS	1588
GMGD	952
GMGS	1470

Tabella 4.1. Numero di coppie all'interno dei diversi test set.

4.1 Confronto tra le reti

Ancor prima di implementare la nostra soluzione, abbiamo potuto confrontare le prestazioni ottenute dalle due diverse architetture della rete VGGFace, Vgg-16 e Resnet-50, nelle condizioni di benchmark, ossia estraendo i descrittori dall'ultimo layer nascosto e calcolandone la distanza coseno per la classificazione. Come prevedibile, la seconda versione della rete, essendo stata allenata su un numero di campioni decisamente maggiore rispetto alla prima, ha permesso di raggiungere un'accuratezza più alta per tutte le tipologie di parentela (tab 4.1). In particolare, con Resnet-50 è stato ottenuto un miglioramento medio superiore al 4% rispetto a Vgg-16, con picchi dell'8% per le parentele fratello-fratello (BB) e madre-figlio (MS). In generale, è possibile notare come in tutte e due le reti i risultati più accurati siano stati ottenuti sui dataset dei fratelli dello stesso sesso, mentre i meno precisi corrispondono alle quattro tipologie di parentela nonno-nipote: questo dato dimostra che la difficoltà di classificazione tra parenti e non-parenti cresce all'aumentare della differenza di età tra i due membri della coppia. In figura 4.1 viene valutata la qualità degli undici classificatori, per entrambe le reti, attraverso le curve ROC specifiche di ognuno. Una curva ROC è un grafico che rappresenta l'insieme delle coppie *True Positive* (TP) e False Positive (FP), al variare del parametro di soglia del classificatore: anche da questi grafici è possibile osservare come le curve in blu (Resnet-50) abbiano un comportamento migliore rispetto a quelle in arancione (Vgg-16), in quanto più distanti dalle linee immaginarie che tagliano i diversi grafici a 45°.

	BB	SS	SIBS	FD	FS	MD	MS	GFGD	GFGS	GMGD	GMGS	AVG
VGG-16	68,88 %	73,27 %	60,91 %	63,20 %	62,87 %	68,12 %	60,08 %	57,64 %	53,21 %	58,40 %	57,55 %	62.20 %
RESNET-50	75,98 %	78,82 %	65,22 %	68,08 %	67,08 %	71,64 %	68,01 %	57,64 %	57,62 %	60,50 %	60,48 %	66,46 %

Tabella 4.2. Accuratezza dei risultati calcolando la distanza coseno tra i descrittori di ultimo livello della rete.

Successivamente, sono stati valutati i risultati ottenuti dai due descrittori di ultimo livello in seguito ai vari passaggi della pipeline illustrata nel capitolo precedente: merging dei descrittori tramite distanza L1, Principal Component Analysis e classificatore SVM. Anche in questo caso i risultati migliori, per quasi tutte le parentele,

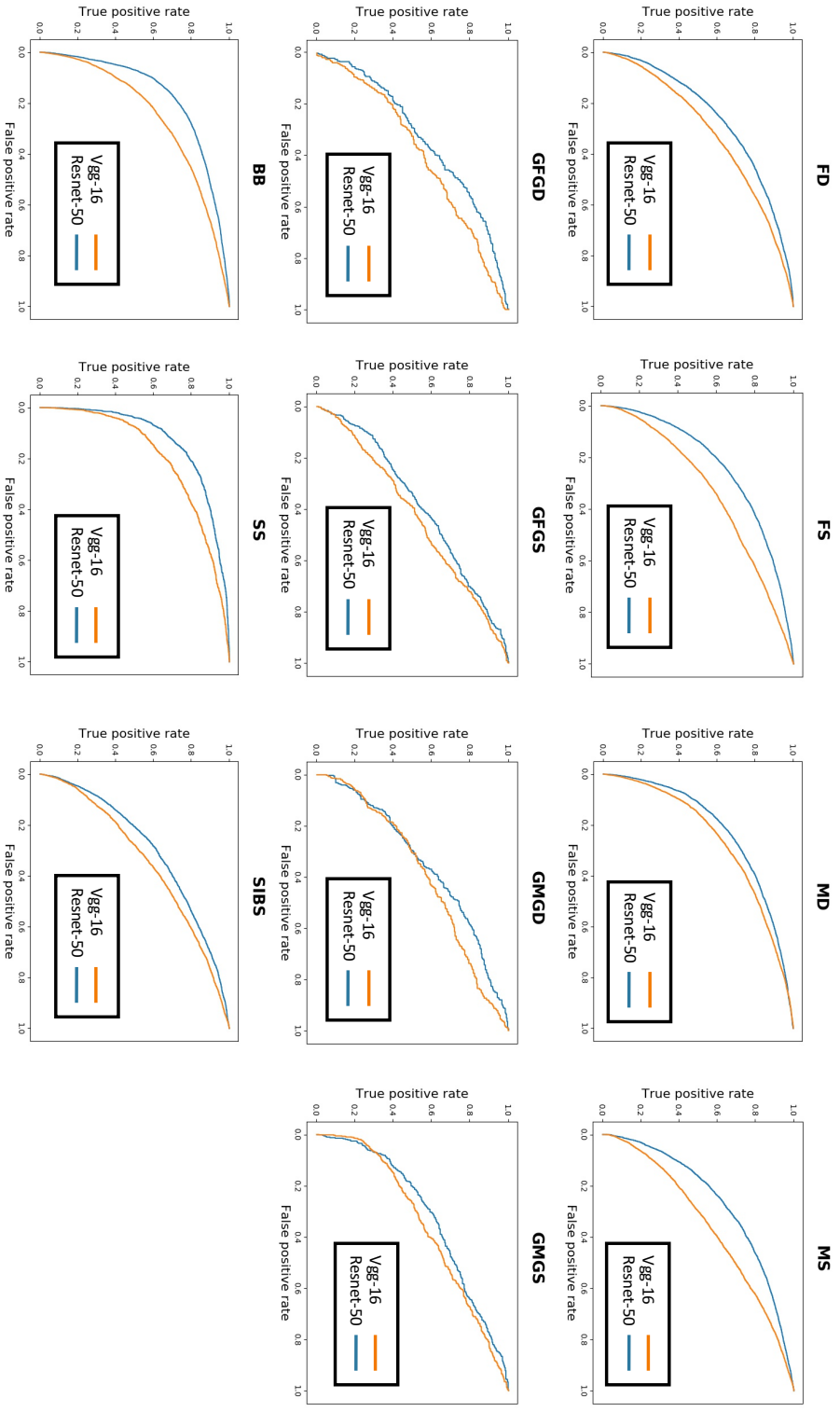

Figura 4.1. Curve ROC specifiche di ogni tipologia di parentela.

sono stati conseguiti utilizzando Resnet-50 come rete di base; in questa circostanza, tuttavia, lo scarto medio tra le prestazioni delle due reti è inferiore al 1% (tab 4.3). È molto interessante notare come, per entrambe le reti (in particolare per Vgg-16), i risultati abbiano subito un miglioramento sostanziale rispetto alle condizioni di benchmark per quasi tutte le tipologie di parentela. Le uniche eccezioni sono rappresentate dalle parentele fratello-fratello (BB), in cui l'accuratezza di Resnet-50 è scesa di oltre cinque punti percentuali, e da quella sorella-sorella (SS), in cui il calo di prestazioni è stato più lieve. Il miglioramento generale, comunque, è dovuto al fatto che il descrittore risultante dalla differenza in valore assoluto, contiene molte più informazioni rispetto al singolo valore ottenuto dalla distanza coseno: nel caso delle parentele tra fratelli dello stesso sesso, in cui il valore della distanza coseno è già sufficientemente discriminante, l'aumento del numero di elementi comporta un degrado dei risultati, mentre negli altri casi aiuta il classificatore a distinguere tra coppie positive e negative.

	BB	SS	SIBS	FD	FS	MD	MS	GFGD	GFGS	GMGD	GMGS	AVG
VGG-16	70,13 %	74,67 %	67,55 %	66,96 %	66,59 %	70,47 %	65,76 %	62,77 %	58,12 %	59,45 %	63,40 %	65,98 %
RESNET-50	70,20 %	78,63 %	70,12 %	68,45 %	67,08 %	72,34 %	68,01 %	57, 64%	58,00 %	64,71 %	62,99 %	66,87 %

Tabella 4.3. Accuratezza dei risultati sottoponendo i descrittori di ultimo livello delle reti alla pipeline L1 + PCA + SVM .

4.2 Confronto tra descrittori di livello diverso

Nella fase precedente abbiamo considerato solamente il descrittore estratto dall'ultimo livello delle due reti. In seguito, abbiamo constatato che, in molti problemi di computer vision, il feature vector estratto dal penultimo strato nascosto della rete ha permesso di ottenere risultati più accurati, o comunque comparabili col descrittore di ultimo livello. È stato quindi deciso di testare le prestazioni di questi due descrittori, sottoponendoli ai diversi passaggi descritti nella pipeline in figura 3.10, per effettuare un confronto con i risultati riportati nel paragrafo precedente. Ci riferiamo ai due descrittori di penultimo livello con il nome *fc6* (fully-connected 6)

per la rete Vgg-16 e *idB10* (identity block 10) per Resnet-50, dal nome del layer da cui sono stati estratti.

	BB	SS	SIBS	FD	FS	MD	MS	GFGD	GFGS	GMGD	GMGS	AVG
VGG-16 (FC-7)	70,13 %	74,67 %	67,55 %	66,96 %	66,59 %	70,47 %	65,76 %	62,77 %	58,12 %	59,45 %	63,40 %	65,98 %
VGG-16 (FC-6)	71,68 %	75,98 %	68,01 %	66,58 %	66,35 %	69,77 %	65,97 %	61,10 %	56,23 %	59,87 %	64,42 %	65,99 %
RESNET-50 (Last layer)	70,20 %	78,63 %	70,12 %	68,45 %	67,08 %	72,34 %	68,01 %	57, 64%	58,00 %	64,71 %	62,99 %	66,87 %
RESNET-50 (IdB-10)	69,94 %	78,38 %	68,73 %	68,03 %	66,85 %	71,02 %	68,47 %	56,68 %	58,88 %	59,45 %	61,77 %	66,20 %

Tabella 4.4. Confronto tra i descrittori di ultimo e penultimo livello delle reti (L1 + PCA + SVM).

In tabella 4.4 è possibile vedere come i risultati ottenuti con i due feature vectors di penultimo livello siano molto simili a quelli ottenuti coi descrittori classici. Per quanto riguarda l'architettura Vgg-16, in particolare, i risultati dei due descrittori di ultimo e ultimo livello sono pressocchè identici e, per questo motivo, nel momento di effettuare il fine-tuning, è stato deciso di unire i diversi rami della rete Siamese a questo livello, anziché all'ultimo, riducendo in questo modo il numero di parametri da ri-allenare. Successivamente abbiamo provato ad estrarre i descrittori anche dagli strati più profondi delle due reti, ma in questo caso i risultati si sono rivelati decisamente meno accurati, in quanto tali feature vectors memorizzano informazioni generiche di basso livello, come lati e angoli, che non si sono rivelate sufficientemente discriminanti per il nostro problema.

4.3 Effetto della fusione di features a più livelli

In questo paragrafo vengono valutate le prestazioni ottenute dai due features vectors denominati Vgg_Fused e Resnet_Fused, da 9664 e 3840 elementi ciascuno, in cui vengono concatenati i descrittori estratti da vari livelli delle due reti. Come dimostrato dal lavoro *"Good Practice in CNN Feature Transfer"* di *Zheng et al.* [48] la tecnica della fusione di features di basso e alto livello in un unico descrittore ha portato ad un significativo miglioramento dei risultati in 10 differenti database di *object classification*, rispetto all'utilizzo dei singoli descrittori (tab 4.5)

Datasets	Bird	Flower	Indoor	SUN	Cal-101	Cal-256	VOC'07	Holidays	Ukbench	Oxford
conv4+a/m pool.	53.20	88.01	67.81	50.71	80.44	63.86	67.55	70.25	3.23	38.10
conv5+a/m pool.	**73.40**	**94.73**	**75.67**	**58.88**	91.07	83.29	81.78	**80.71**	**3.77**	60.18
FC6+a/m pool.	72.78	94.07	75.32	57.76	92.24	84.20	82.31	78.46	3.69	62.77
FC7+a/m pool.	70.64	92.05	71.4	58.31	89.28	83.82	82.57	79.43	3.73	57.63
All layers	**76.35**	**95.62**	**78.42**	**63.71**	**92.31**	**85.99**	**83.66**	**84.2**	**3.75**	**71.30**

Tabella 4.5. Effetto della fusione di features di diversi livelli in 10 database di object classification (fonte: Good Practice in CNN Feature Transfer.)

Visti i promettenti risultati, abbiamo provato a testare questa tecnica nel nostro sistema di riconoscimento automatico delle parentele. Inizialmente, abbiamo valutato le prestazioni del descrittore Vgg_Fused, da 9664 elementi, nel dataset delle coppie sorella-sorella (SS): in questo caso l'accuratezza (74,08%) si è rivelata inferiore a quella ottenuta sia coi descrittori di ultimo livello (fc7), sia con quelli di penultimo (fc6). Il peggioramento dei risultati, oltre alle grandi capacità computazionali necessarie a gestire descrittori di tale dimensione, ci ha spinto a concentrarci sull'effetto della fusione dei descrittori di diversi livelli all'interno della rete Resnet-50.

In tabella 4.6 vengono riportati i risultati conseguiti utilizzando i descrittori *Resnet_Fused* per la classificazione. In questo caso i feature vectors non sono stati sottoposti a PCA prima di essere dati in pasto al classificatore SVM, in modo da valutare l'effettivo impatto dell'aumento del numero di features sui risultati finali. È possibile notare come siano stati ottenuti risultati leggermente migliori in tutte le undici tipologie di parentela: sebbene l'incremento medio sia di poco superiore allo 0.3%, il miglioramento si è rivelato costante su tutti i dataset, sintomo del fatto che anche i descrittori estratti dai layers più profondi contengono importanti informazioni sui diversi legami di parentela.

	Resnet-50	*Resnet-50_Fused*
BB	68,84%	69,11%
SS	78,46%	78,73%
SIBS	70,00%	70,06%
FD	68,52%	68,86%
FS	66,33%	66,85%
MD	72,83%	73,00%
MS	68,10%	68,77%
GFGD	58,95%	59,07%
GFGS	57,81%	58,19%
GMGD	65,44%	65,55%
GMGS	59,32%	59,73%
Avg	66,78%	67,09%

Tabella 4.6. Effetto della fusione dei descrittori di diversi livelli nella rete Resnet-50 per le varie tipologie di parentela.

4.4 Effetto della PCA

La scelta del numero dei componenti da selezionare nella fase di Principal Component Analysis ha avuto un ruolo centrale per lo svolgimento di questa tesi. Avendo notato che il valore assegnato al parametro *n_components* aveva un grandissimo impatto sull'accuratezza dei risultati, la maggior parte degli esperimenti sono stati fatti per trovare un valore ottimale per questo parametro. Inizialmente abbiamo provato a settare un numero fisso di componenti (es. n_components = 100), quindi abbiamo testato diverse percentuali di varianza del dataset originario da mantenere. In tabella 4.7 viene illustrata la percentuale di accuratezza per le diverse tipologie di parentela al variare di questo parametro, utilizzando Resnet-50 come rete neurale di base.

Si può notare come in alcuni casi (es. SS, MD) i risultati più accurati siano stati ottenuti considerando solamente i 100 componenti a varianza maggiore, mentre in altri (es. BB, SIBS) questo stesso valore del parametro *n_components* ha portato ad un netto peggioramento dei risultati. Dopo diversi tentativi, come detto nel paragrafo 3.4.5 è stato deciso di utilizzare, per tutti i dataset e per le features estratte da entrambe le reti, una percentuale del 95% della varianza del dataset originale, come miglior compromesso tra precisione e tempistiche per l'addestramento dei classificatori. Nella tabella 4.8 viene mostrata la riduzione del numero di features, in seguito al merging con distanza L1 e alla successiva analisi in componenti principali degli undici dataset.

RESNET-50			
	n_components = 100	n_components = 0.95	NO PCA
BB	65,45%	70,20%	68,84%
SS	80,44%	78,63%	78,46%
SIBS	66,98%	70,12%	70,00%
FD	65,91%	68,45%	68,52%
FS	65,80%	67,08%	66,33%
MD	72,97%	72,34%	72,83%
MS	66,87%	68,01%	68,10%
GFGD	57,16%	57,64%	58,95%
GFGS	57,62%	58,00%	57,81%
GMGD	59,66%	64,71%	65,44%
GMGS	58,44%	60,48%	59,32%

Tabella 4.7. Accuratezza per le diverse tipologie di parentela al variare del parametro di PCA *n_components*.

	VGG-16 (4096)		RESNET-50 (2048)	
Parentela	Features dopo PCA	% riduzione	Features dopo PCA	% riduzione
BB	2172	46,97%	1174	42,68%
SS	2183	46,70%	1224	40,23%
SIBS	2292	44,04%	1280	37,50%
FD	2322	43,31%	1331	35,01%
FS	2286	44,19%	1286	37,21%
MD	2317	43,43%	1300	36,52%
MS	2264	44,73%	1282	37,40%
GFGD	1210	70,46%	745	63,62%
GFGS	1017	75,17%	697	65,97%
GMGD	1125	72,53%	737	64,01%
GMGS	983	76,00%	678	66,89%

Tabella 4.8. Numero di features e percentuale di riduzione della dimensione dei descrittori nei diversi dataset dopo PCA (parametro n_components = 95%)

4.5 Effetto del fine-tuning delle reti

La tabella 4.9 mostra i risultati conseguiti con i descrittori estratti dalle nuove reti, modificate in seguito all'operazione di fine-tuning sul database *Families In the Wild*. Nell'ambito del riconoscimento automatico delle parentele, e in generale in gran parte dei problemi di computer vision, la tecnica del fine-tuning sul dataset di riferimento ha permesso di raggiungere i risultati più accurati. Nel nostro caso,

tuttavia, è stato molto complicato riuscire a trovare un insieme di parametri che ci consentisse di proseguire l'addestramento delle reti senza incappare in una situazione di sovradattamento ai dati e, di conseguenza, ad un deterioramento dei risultati.

	BB	SS	SIBS	FD	FS	MD	MS	GFGD	GFGS	GMGD	GMGS	AVG	% +/-
VGG-16 Fine-Tuned	71,46%	76,13%	65,22%	65,99%	66,31%	68,97%	68,21%	61,97%	58,82%	59,98%	64,01%	66,09 %	+0,11 %
RESNET-50 Fine-Tuned	75,10%	79,73%	70,21%	67,63%	70,56%	73,95%	72,78%	61,46%	58,25%	66,70%	63,95%	69,12 %	+1,48 %

Tabella 4.9. Accuratezza dei risultati dopo il fine-tuning delle reti.

Grazie all'utilizzo della funzione di perdita triplet-loss durante il fine-tuning delle reti, che minimizza la distanza tra coppie di parenti e massimizza quella tra coppie negative, è stato ottenuto un netto miglioramento per quanto riguarda l'architettura Resnet-50, nei dataset di quasi tutte le tipologie di parentela. Come riportato nell'ultima colonna della tabella 4.9, l'incremento medio dei risultati rispetto a quelli ottenuti con i descrittori estratti dalla rete di base è di circa l'1,5%. Per quanto riguarda la prima versione della rete VggFace (Vgg-16), invece, la nostra strategia per il fine-tuning non ci ha permesso di ottenere dei risultati altrettanto buoni: ad eccezione della parentela mamma-figlio (M-S), in cui si è verificato un miglioramento superiore al 3%, i risultati sono rimasti invariati, o, in alcuni casi, anche peggiorati. In questo caso, l'alto numero di parametri allenabili dell'architettura Vgg-16, in aggiunta al basso numero di triple utilizzate per il training, ha fatto sì che la rete andasse incontro ad una situazione di overfitting, causando un degrado dei risultati. In generale, studiando gli effetti del fine-tuning delle reti nell'ambito della kinship verification, ci saremmo aspettati dei risultati migliori: nel nostro caso, tuttavia, il set di parametri, non ottimale, utilizzato per il training, oltre al fatto di aver costruito casualmente le triple anchor, positive embedding e negative embedding, ha influito in modo negativo su questo risultato.

Sebbene ci fossimo aspettati un miglioramento dell'accuratezza superiore a quello ottenuto, possiamo comunque ritenerci soddisfatti dei risultati conseguiti con questa tecnica, in quanto, come verrà riportato nei paragrafi successivi, questi ultimi sono

molto simili o superiori ai risultati raggiunti dagli altri partecipanti alla RFIW 2018 e anche da quelli ottenuti dagli organizzatori durante gli esperimenti di benchmark.

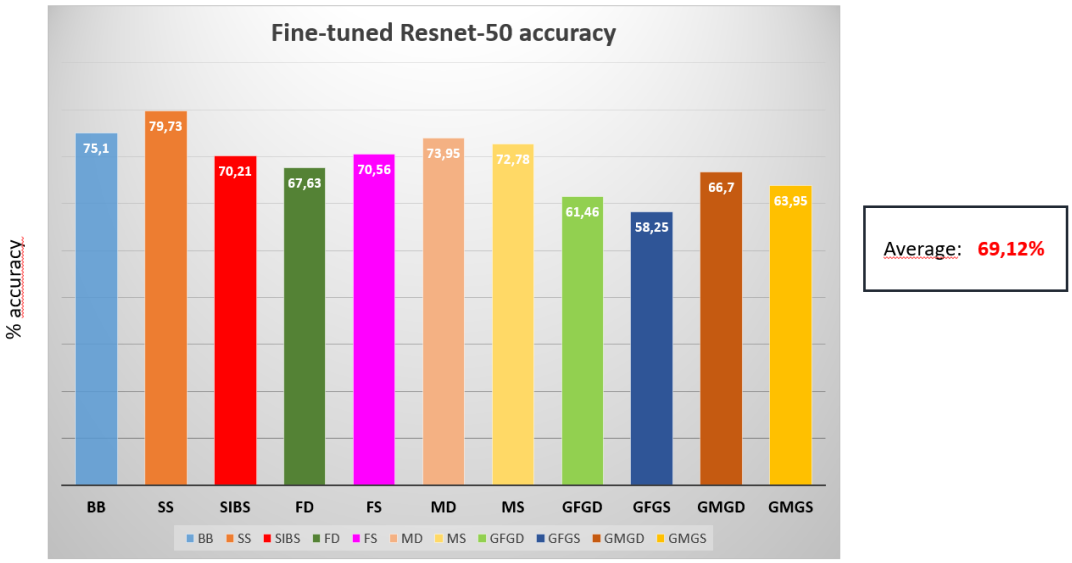

Figura 4.2. Accuratezza dei isultati ottenuti dopo il fine-tuning della rete Resnet-50.

4.6 Confronto con i risultati dei partecipanti alla RFIW 2018

In questo paragrafo abbiamo messo a paragone i risultati ottenuti in questa tesi, sia tramite features extraction che tramite fine-tuning delle reti, con quelli conseguiti dagli altri partecipanti alla *Recognizing Families in The Wild challenge 2018*, in modo tale da poter valutare la nostra ipotetica posizione in classifica. Purtroppo, come abbiamo accennato nel paragrafo 2.3.5, i metodi testati dai partecipanti non sono ancora stati resi disponibili pubblicamente e, di conseguenza, il confronto viene effettuato a scopo puramente quantitativo. Il paragone, tuttavia, è molto interessante per il fatto che i risultati ottenuti dai partecipanti, così come i nostri, sono riferiti all'accuratezza nei soli test set, a differenza dei risultati conseguiti dagli organizzatori, riferiti invece ad una media dell'accuratezza sui cinque sottogruppi in cui sono stati suddivisi gli interi dataset di ogni parentela. Dalla tabella 4.10 è possibile notare come i risultati ottenuti dai descrittori estratti dalle reti Vgg-16 e Resnet-50, con la pipeline illustrata in figura 3.10 (features extraction, differenza in valore assoluto, PCA e classificatore SVM), senza il fine-tuning delle reti, si siano

rivelati molto competitivi, ottenendo rispettivamente un secondo ed un terzo posto ipotetico nella sfida.

# Pos	User/method	BB	SS	SIBS	FD	FS	MD	MS	GFGD	GFGS	GMGD	GMGS	Avg
1°	eranda	69,14 %	77,25 %	68,64 %	68,92 %	68,89 %	73,79 %	70,85 %	61,93 %	63,66 %	63,86 %	63,26 %	68,20 %
2°	Resnet-50	70,20 %	78,63 %	70,12 %	68,45 %	69,94 %	72,34 %	68,11 %	60,62 %	58,00 %	64,71 %	62,99 %	67,64 %
3°	Vgg-16 (fc6)	70,13 %	74,67 %	67,55 %	66,96 %	66,59 %	70,47 %	65,76 %	62,77 %	58,12 %	59,45 %	63,40 %	65,98 %
4°	sizhangyu	63,84 %	71,12 %	65,28 %	65,55 %	66,94 %	66,18 %	67,56 %	59,18 %	52,58 %	59,45 %	52,65 %	62,76 %
5°	mariaivanovska	66,25 %	72,85 %	62,96 %	63,30 %	61,52 %	66,62 %	59,84 %	57,99 %	56,92 %	57,56 %	61,08 %	62,44 %
6°	mngyyu	68,44 %	70,20 %	61,62 %	63,07 %	61,22 %	65,88 %	60,46 %	59,42 %	56,36 %	56,93 %	62,78 %	62,40 %
7°	Bekhouche	62,70 %	68,01 %	59,01 %	58,21 %	58,15 %	61,55 %	60,48 %	57,87 %	53,27 %	54,51 %	52,17 %	58,72 %
8°	szl281	57,22 %	64,67 %	57,40 %	59,35 %	57,42 %	59,55 %	57,81 %	55,96 %	55,66 %	55,35 %	57,68 %	58,01 %

Tabella 4.10. Confronto con i risultati dei partecipanti alla RFIW 2018, in giallo sono evidenziati i risultati ottenuti con la pipeline features extraction, L1, PCA, SVM.

La tabella 4.11, invece, mostra il confronto tra i risultati dei partecipanti e quelli ottenuti in questa tesi in seguito all'operazione di fine-tuning delle reti: in questo caso i fetures vectors estratti dalla rete Resnet-50 ri-allenata, ci avrebbero permesso di classificarci al primo posto nella competizione.

# Pos	User/method	BB	SS	SIBS	FD	FS	MD	MS	GFGD	GFGS	GMGD	GMGS	Avg
1°	Fine-Tuned Resnet-50	75,10%	79,73%	70,21%	67,63%	70,56%	73,95%	72,78%	61,46%	58,25%	66,70%	63,95%	69,12 %
2°	eranda	69,14 %	77,25 %	68,64 %	68,92 %	68,89 %	73,79 %	70,85 %	61,93 %	63,66 %	63,86 %	63,26 %	68,20 %
3°	Fine-Tuned Vgg-16	71,46%	76,13%	65,22%	65,99%	66,31%	68,97%	68,21%	61,97%	58,82%	59,98%	64,01%	66,09 %
4°	sizhangyu	63,84 %	71,12 %	65,28 %	65,55 %	66,94 %	66,18 %	67,56 %	59,18 %	52,58 %	59,45 %	52,65 %	62,76 %
5°	mariaivanovska	66,25 %	72,85 %	62,96 %	63,30 %	61,52 %	66,62 %	59,84 %	57,99 %	56,92 %	57,56 %	61,08 %	62,44 %
6°	mngyyu	68,44 %	70,20 %	61,62 %	63,07 %	61,22 %	65,88 %	60,46 %	59,42 %	56,36 %	56,93 %	62,78 %	62,40 %
7°	Bekhouche	62,70 %	68,01 %	59,01 %	58,21 %	58,15 %	61,55 %	60,48 %	57,87 %	53,27 %	54,51 %	52,17 %	58,72 %
8°	szl281	57,22 %	64,67 %	57,40 %	59,35 %	57,42 %	59,55 %	57,81 %	55,96 %	55,66 %	55,35 %	57,68 %	58,01 %

Tabella 4.11. Confronto con i risultati dei partecipanti alla RFIW 2018, in giallo sono evidenziati i risultati ottenuti dopo il fine-tuning delle reti.

4.7 Confronto con benchmark e prestazioni umane

Infine, abbiamo confrontato i nostri risultati con quelli degli esperimenti di benchmark testati dagli organizzatori: in particolare, sono stati presi in considerazione i due approcci basati sul deep learning, Resnet-22 + CenterFace [38] e SphereFace [39], con i quali sono stati raggiunti i risultati più accurati. In questi esperimenti, come accennato nei paragrafi precedenti, gli organizzatori hanno suddiviso l'intero insieme delle coppie (train, validation e test set) in cinque sottogruppi ulteriori per ogni tipologia di parentela; l'accuratezza dei risultati è stata calcolata su ognuno di questi sottogruppi e la media delle cinque è stata riportata come risultato finale. In questo caso, quindi, il paragone con i nostri risultati, riferiti ai soli set di test, non è del tutto equo, ma è comunque indicativo per valutare la bontà delle nostre prestazioni.

Dalla tabella 4.12 è possibile notare come il nostro risultato migliore, raggiunto in seguito al fine-tuning della rete Resnet-50 sul database FIW, sia sostanzialmente identico a quello ottenuto dagli organizzatori con il fine-tuning della rete SphereFace. Un dato interessante sta nel fatto che, considerando soltanto la media dell'accuratezza nei sette database delle coppie di fratelli e di quelle genitore-figlio, i nostri

# Pos	User/method	BB	SS	SIBS	FD	FS	MD	MS	GFGD	GFGS	GMGD	GMGS	Avg
1°	SphereFace	71,94%	77,30%	70,23%	69,25%	68,50%	71,81%	69,49%	66,07%	66,36%	64,58%	65,40%	69,18%
2°	Fine-Tuned Resnet-50	75,10%	79,73%	70,21%	67,63%	70,56%	73,95%	72,78%	61,46%	58,25%	66,70%	63,95%	69,12 %
3°	Resnet-22 + CF	69,88%	69,54%	69,54%	68,15%	67,73%	71,09%	68,63%	66,37%	66,45%	64,81%	64,39%	67,87%
4°	Fine-Tuned Vgg-16	71,46%	76,13%	65,22%	65,99%	66,31%	68,97%	68,21%	61,97%	58,82%	59,98%	64,01%	66,09 %

Tabella 4.12. Confronto con i risultati degli esperimenti di benchmark della RFIW 2018, in giallo sono evidenziati i risultati ottenuti in questa tesi dopo il fine-tuning delle reti.

risultati superino dell'1,5% (72,77% vs 71,21%) quelli dello stato dell'arte. La situazione è completamente ribaltata per quanto riguarda le quattro coppie nonno nipote (GF-GD, GF-GS, GM-GD, GM-GS), per le quali la nostra accuratezza è risultata inferiore di quasi il 3% rispetto a quella della rete SphereFace: il motivo di questo calo è da ricercare, probabilmente, nel basso numero di triple con cui è stato proseguito l'addestramento delle reti sul dataset FIW per queste parentele. In generale, possiamo affermare che la tecnica del fine-tuning delle reti neurali è decisamente la strada più promettente per quanto riguarda il riconoscimento automatico delle parentele da un punto di vista visuale: sia i nostri risultati, che quelli ottenuti dagli organizzatori della RFIW Challenge, dimostrano un significativo aumento dell'accuratezza rispetto a tutte le altre tecniche descritte in letteratura (descrittori tradizionali, estrazione di features *off-the-shelf* da reti pre-allenate e apprendimento metrico).

Un'ultima considerazione può essere effettuata paragonando i nostri risultati, e quelli dei principali test di benchmark, con le prestazioni umane. Il grafico in figura 4.3 ci mostra come l'accuratezza media ottenuta dagli uomini tra tutte le tipologie di parentela (in giallo) sia del 57,85%, leggermente superiore a quella dei descrittori tradizionali SIFT ed LBP, ma inferiore di oltre 10 punti percentuali rispetto a quella raggiunta in seguito al fine-tuning delle reti.

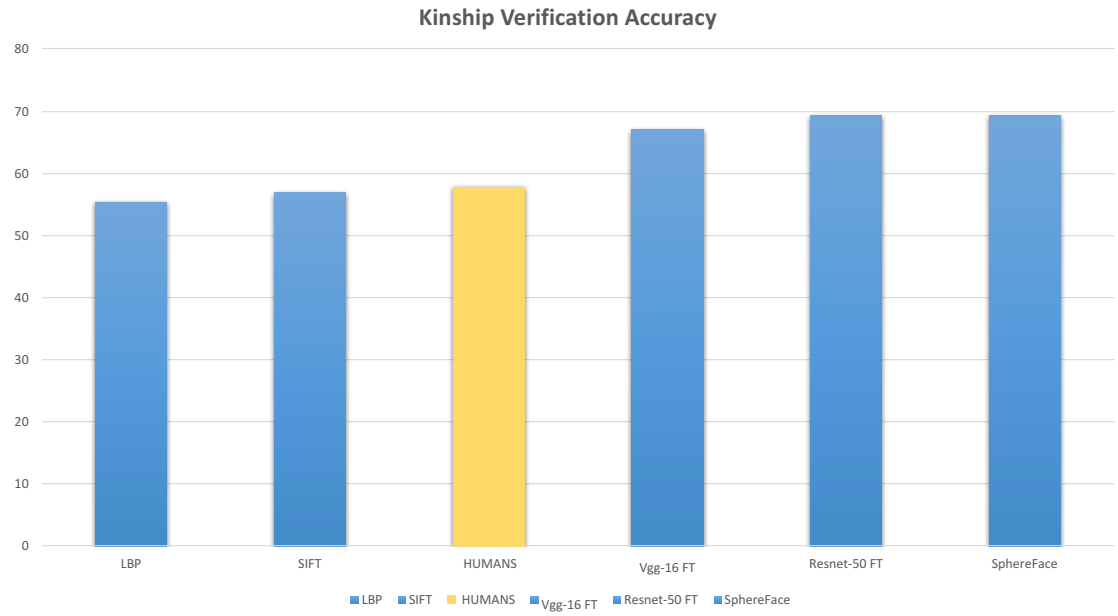

Figura 4.3. Confronto tra prestazioni umane, descrittori tradizionali e fine-tuning delle reti nell'ambito del riconoscimento automatico delle parentele.

Questi risultati, considerando l'abilità innata degli uomini nel riconoscere le somiglianze tra membri di una stessa famiglia, potrebbero essere considerati soddisfacenti; tuttavia, l'accuratezza media del 70% è ancora molto lontana da quella richiesta per poter implementare un sistema reale di riconoscimento automatico di questo tipo.

Capitolo 5

Conclusioni e sviluppi futuri

In questa tesi 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 100 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. In particolare, possiamo riassumere il contributo di questo lavoro in quattro punti chiave:

- innanzitutto, in base alle informazioni in nostro possesso, questa tesi rappresenta il primo tentativo di utilizzare la seconda versione della rete VggFace, con architettura residuale (Resnet-50), nell'ambito della kinship verification;
- in secondo luogo è stato dimostrato che, per quasi tutte le tipologie di parentela (S-S e B-B escluse), il vettore risultante dalla differenza in valore assoluto, elemento per elemento, tra i due descrittori di ogni coppia di immagini è più discriminativo rispetto al semplice valore ottenuto dalla distanza coseno tra i due;
- abbiamo inoltre provato ad estrarre le features da diversi livelli delle due reti e, per la prima volta nel campo del riconoscimento automatico della parentele, è stato testato l'effetto della fusione di vettori di caratteristiche di basso e alto livello in un unico descrittore.
- infine, i nostri esperimenti hanno confermato la supremazia della tecnica del fine-tuning, rispetto a tutte le altre strategie per la kinship verification.

Sebbene i nostri esperimenti ci abbiano permesso di raggiungere risultati molto simili a quelli dello stato dell'arte, riteniamo che ci siano diverse strade per ottenere un incremento ulteriore dell'accuratezza. Il metodo più diretto per migliorare la nostra pipeline potrebbe essere quello di modificare il set di parametri utilizzati per il fine-tuning delle reti, utilizzando un algoritmo genetico per trovare le impostazioni migliori. Un'altra idea è quella di incrementare il numero di immagini per il training attraverso traslazioni, rotazioni, scalamenti o modifiche nell'illuminazione e nei colori delle immagini originali (data augmentation): con questa tecnica il modello diventerà sempre più robusto rispetto alle variazioni nelle immagini di test. Distaccandoci dalla pipeline proposta in questa tesi, una strategia differente potrebbe essere quella di effettuare il training delle reti neurali per patches: in questo caso le immagini originali devono essere suddivise in diverse regioni (es occhi, naso, bocca...), che verranno poi utilizzate per addestrare separatamente diversi rami del modello. Infine, la soluzione che potrebbe portare ai risultati più accurati, sarebbe quella di costruire un modello ad hoc per il problema del riconoscimento delle parentele ed allenarlo *from scratch* sul dataset FIW.

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 la Recognizing Families in The Wild Challenge, grazie all'introduzione del database FIW, possa rappresentare un nuovo inizio per quest'area di ricerca, con la speranza di vederne presto sfruttate appieno tutte le sue potenzialità.

Bibliografia

[1] M.F.Dal Martello, L.T.Maloney, "Where are kin recognition signals in the human face?", Journal of Vision, Vol. 6, No. 2, November 2006, DOI 10.1167/6.12.2

[2] 2018 Recognizing Families in The Wild, https://web.northeastern.edu/smilelab/RFIW2018/

[3] Families in The Wild. A large-scale kinship recognition image database. https://web.northeastern.edu/smilelab/fiw/

[4] R.Fang, K.D.Tang, N.Snavely, T.Chen, "Towards computational models of kinship verification", Proceedings of the International Conference on Image Processing, ICIP 2010, Hong Kong (China) (Colorado), September 26-29, 2010 pp. 1577–1580, DOI 10.1109/ICIP.2010.5652590

[5] X.Zhou, J.Hu, J.Lu, Y.Shang, Y.Guan, "Kinship verification from facial images under uncontrolled conditions", Proceedings of the 19th ACM international conference on Multimedia, ACM (2011), Scottsdale (Arizona, USA), November 28 - December 01, 2011 pp. 953-956, DOI 10.1145/2072298.2071911

[6] X.Zhou, J.Hu, J.Lu, Y.Shang, Y.Guan, "Gabor-based gradient orientation pyramid for kinship verification under uncontrolled environments.", Proceedings of the 20th ACM international conference on Multimedia, ACM (2012), Nara (Japan), October 29 - November 02, 2012 pp. 725–728, DOI 10.1145/2393347.2396297

[7] G.Guo, X.Wang, " Kinship measurement on salient facial features.", IEEE Transactions on Instrumentation and Measurement , Vol. 61, No. 8, February 2012, pp. 2322 - 2325, DOI 10.1109/TIM.2012.2187468

[8] N.Kohli, R.Singh, M.Vatsa, " Self-similarity representation of weber faces for kinship classification.", 2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS), Arlington (Virginia, USA), September 23-27. 2012 pp. 2535–2545, DOI 10.1109/BTAS.2012.6374584

[9] H.Yan, J.Lu, X.Zhou, "Prototype-based discriminative feature learning for kinship verification.", IEEE Transactions on cybernetics, Vol. 45, No. 11, December 2014, pp. 245–250, DOI 10.1109/TCYB.2014.2376934

[10] J.Lu, V.E.Liong, X.Zhou, J.Zhou, "Learning compact binary face descriptor for face recognition.", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 37, No. 10, March 2015, pp. 2041-2056, DOI 10.1109/TPAMI.2015.2408359

[11] J.Lu, X.Zhou, Y.P.Tan, G.Wang, "Discriminative multimanifold analysis for face recognition from a single training sample per person.", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 1, April 2012, pp. 39-51, DOI 10.1109/TPAMI.2012.70

[12] H.Dibeklioglu, A.Ali Salah, T.Gevers, "Like father, like son: Facial expression dynamics for kinship verification.", Proceedings of the IEEE International Conference on Computer Vision, Sydney (Australia), December 1-8. 2013 pp. 1497–1504, DOI 10.1109/ICCV.2013.189

[13] J.Lu, X.Zhou, Y.P.Tan, Y.Shang, J.Zhou "Neighborhood repulsed metric learning for kinship verification.", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 36, No. 2, July 2013, pp. 331-345, DOI 10.1109/TPAMI.2013.134

[14] H.Yan, J.Lu, X.Zhou, W.Deng, " Discriminative multimetric learning for kinship verification.", IEEE Transactions on Information Forensics and Security, Vol. 9, No. 7, June 2014, pp. 1169 - 1178, DOI 10.1109/TIFS.2014.2327757

[15] J.Hu, J.Lu, J.Yuan, Y.P.Tan, "Large Margin Multi-metric Learning for Face and Kinship Verification in the Wild" nel libro "Discovering Harmony: A Hierarchical Colour Harmony Model for Aesthetics Assessment." a cura di P.Lu, X.Peng, 2014 pp. 252-267, DOI 10.1007/978-3-319-16811-1_17

[16] S.Xia, M.Shao, J.Luo, Y.Fu, " Understanding Kin Relationships in a Photo.", IEEE Transactions on Multimedia, Vol. 14, No. 4, February 2012, pp. 1046 - 1056, DOI 10.1109/TMM.2012.2187436

[17] S.Xia, M.Shao, Y.Fu,, "Kinship Verification through Transfer Learning.", Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcellona (Spain), July 16-22, 2011 pp. 2539–2544, DOI 10.5591/978-1-57735-516-8/IJCAI11-422

[18] Computational Models of Kinship Verification. http://chenlab.ece.cornell.edu/projects/KinshipVerification/

[19] UB KinFace Database. http://www1.ece.neu.edu/~yunfu/research/Kinface/Kinface.htm

[20] Tri-subject Kinship Face Database (TSKinFace). http://parnec.nuaa.edu.cn/xtan/data/TSKinFace.html

[21] Uva-NEMO Smile Database (TSKinFace). https://www.uva-nemo.org/

[22] Kinship Classification by Modeling Facial Feature Heredity People. http://chenlab.ece.cornell.edu/projects/KinshipClassification/index.html

[23] KinFaceW. http://www.kinfacew.com/datasets.html

[24] MS-Celeb-1M. Challenge of Recognizing One Million Celebrities in the Real World. https://www.msceleb.org

[25] VGG Face Dataset.http://www.robots.ox.ac.uk/~vgg/data/vgg_face/

[26] VGGFace2. A large scale image dataset for face recognition.http://www.robots.ox.ac.uk/~vgg/data/vgg_face2/

[27] IEEE Conference on Automatic Face and Gesture Recognition. https://fg2018.cse.sc.edu/

[28] D.G.Lowe " Distinctive image features from scale-invariant keypoints", International journal of computer vision, Vol. 60, No. 2, November 2004, pp. 91-110, DOI 10.1023/B:VISI.0000029664.99615.94

[29] T.Ahonen, A.Hadid, M.Pietikainen "Distinctive image features from scale-invariant keypoints", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 12, October 2006, pp. 91-110, DOI 10.1109/TPAMI.2006.244

[30] O.M.Parkhi, A.Vedaldi, A.Zisserman, "Deep Face Recognition.", British Machine Vision Conference, January, 2015 DOI 10.5244/C.29.41

[31] Y Wen, K.Zhang, Z.Li, Y.Qiao, "A discriminative feature learning approach for deep face recognition.", European Conference on Computer Vision, Amsterdam (Netherlands), October 11-14, 2016 pp. 409-515, DOI 10.1007/978-3-319-46478-7_31

[32] J.V.Davis, B.Kulis, P.Jain, S.Sra, I.S.Dhillon, "Information- theoretic metric learning.", Proceedings of the 24th international conference on Machine learning, Corvalis (Oregon, USA) June 20-24, 2007 pp. 209–216, DOI 10.1145/1273496.1273523

[33] Z.Ding, S.Suh, J.J.Han, C.Choi, Y.Fu, "Discriminative low-rank metric learning for face recognition.", 11th IEEE In- ternational Conference and Workshops on Automatic Face and Gesture Recognition (FG), Ljubljana (Slovenia), May 4-8, 2015 pp. 1–6, DOI 10.1109/FG.2015.7163088

[34] X.Niyogi, X.He, "Locality preserving projections.", Neural information processing systems, 2015, Vol. 16, pp. 153,

[35] K.Q.Weinberger, L.K.Saul, "Distance metric learning for large margin nearest neighbor classification.", Journal of Machine Learning Research, Vol. 10, February 2009, pp. 207-244,

[36] Y.Peng, S.Wang, B.L.Lu, "Marginalized denoising autoencoder via graph regularization for domain adaptation", International Conference on Neural Information Processing, Daegu (Korea), November 3-7, 2013. pp. 156–163, DOI 10.1007/978-3-642-42042-9_20

[37] S.Wang, J.P.Robinson, Y.Fu, "Kinship verification on families in the wild with marginalized denoising metric learning", IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), Washington, DC (USA), 30 May - 3 June, 2017 DOI 10.1109/FG.2017.35

[38] J.P.Robinson, M.Shao, H.Zhao, Y.Wu, T.Gillis, Y.FU, "Recognizing families in the wild (rfiw)", Proceedings of the 2017 ACM Workshop on RFIW17, Mountain View (California, USA) — October 27 - 27, 2017 pp. 5–12 DOI 10.1145/3134421.3134424

[39] W.Liu, Y.Wen, Z.Yu, M.Li, B.Raj, L.Song, "Sphereface: Deep hypersphere embedding for face recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), April, 2017 DOI 10.1109/CVPR.2017.713

[40] 2017 Recognizing Families in The Wild, https://web.northeastern.edu/smilelab/RFIW2017/

[41] Y.Li, J.Zeng, J.Zhang, A.Dai, M.Kan, S.Shan, X.Chen "KinNet: Fine-to-Coarse Deep Metric Learning for Kinship Verification", Proceedings of the 2017 ACM Workshop on RFIW17, Mountain View (California, USA) — October 27 - 27, 2017 pp. 13–20 DOI 10.1145/3134421.3134425

[42] Scikit-Learn. Machine learning in Python. http://scikit-learn.org/stable/

[43] Keras: The Python Deep Learning library. https://keras.io/

[44] Floydhub. Fastest way to build, train, and deploy deep learning models. https://www.floydhub.com

[45] F.Schroff, D.Kalenichenko, J.Philbin,, "FaceNet: A Unified Embedding for Face Recognition and Clustering", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston (MA, USA), June 7-12, 2015 DOI 10.1109/CVPR.2015.7298682

[46] K.Simonyan, A.Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur (Malaysia), November 3-6, 2015 DOI 10.1109/ACPR.2015.7486599

[47] J.Bromley, I.Guyon, Y.LeCun, E.Sickinger, R.Shah, "Signature verification using a "Siamese" time delay neural network", NIPS'93 Proceedings of the 6th International Conference on Neural Information Processing Systems, Denver (Colorado), November 29 - December 02, 1993 pp. 737-744,

[48] L.Zheng, Y.Zhao, S.Wang, Q.Tian, "Good Practice in CNN Feature Transfer", April, 2016