

Università degli Studi di Salerno

Dipartimento di Informatica



Progetto Fondamenti di Visione Artificiale e Biometria

Professore

Michele Nappi

Studenti

Antonio Cimino 0522500922

Vincenzo De Martino 0522500966

Diego Varriale 0522500910

Sommario

Abstract	2
Introduzione	3
Tecnologie Utilizzate	5
Prima Parte: studio e utilizzo della rete neurale Siamese	7
Seconda parte: transfer learning e nuovo dataset	11
Terza parte: puntate dei soliti ignoti e test	18
Parte Facoltativa: Scale Invariant Feature Transform (SIFT)	22
Parte Facoltativa: User Interface	27
Conclusioni	29

Abstract

Il seguente studio è stato affrontato per trovare la parentela tra due o più persone confrontando i volti tramite immagini. Questo esperimento è stato svolto in diverse fasi. Una prima fase nella quale addestriamo una rete neurale a riconoscere se due persone sono parenti con relativi test. La seconda parte riguarda l'addestramento della stessa rete con un altro dataset contenente meno immagini ma con più classi, per diversi tipi di parentela. La terza fase riguarda il programma “i soliti ignoti”, abbiamo preso i video delle puntate, analizzando i concorrenti della fase riguardante il “parente misterioso” per individuare tra le 8 persone il parente del concorrente. Sono state poi effettuate altre due fasi, una dove sperimentiamo la tecnica Scale-Invariant Feature Trasform, vedendo come cambiano i risultati senza utilizzare una rete neurale, e seconda per la creazione di un'interfaccia grafica.

Introduzione

Il progetto svolto è incentrato sul problema della Kinship Recognition, ovvero la verifica di un'eventuale parentela tra due o più persone. La parentela è un vincolo costituito da legami biologici, sociali, culturali e giuridici, tra le persone che hanno in comune uno stipite. Il volto risulta essere uno dei fenotipi più forti per il riconoscimento della parentela tra umani e per lo più ciò accade a livello inconscio. Per questo motivo quindi abbiamo analizzato, tramite immagini o parti di video, i volti delle persone per confrontare e valutare se tra questi individui ci sia un possibile tipo di parentela. Come primo passo per analizzare e svolgere questo tipo di tematica abbiamo utilizzato il dataset offerto dal sito: “<https://www.kaggle.com/c/recognizing-faces-in-the-wild/data>”, da una challenge creata nel 2019. Per poter analizzare questo dataset abbiamo utilizzato una base di codice recuperato da un progetto di Kaggle al seguente link: “<https://www.kaggle.com/jiangstein/a-very-simple-siamese-network-in-pytorch>”.

In questo progetto l'autore utilizza la rete neurale Siamese e utilizza Pytorch, un framework di machine learning open source che accelera il percorso dalla prototipazione della ricerca alla distribuzione in produzione. Dopo aver analizzato il primo dataset e allenato la rete neurale Siamese siamo passati ad un secondo dataset molto simile al precedente, ma che non indica solo la parentela tra due persone ma anche il tipo, precisamente se si tratta di padre-figlio/a o madre-figlio/a. Con l'aumentare delle caratteristiche sono state aumentate anche le classi da assegnare alle coppie di immagini. Per svolgere questo tipo di operazione abbiamo utilizzato il transfer learning e allenato il nuovo dataset sulla rete neurale Siamese adottata in precedenza. Terminato il training sul secondo dataset abbiamo effettuato un test con la rete allenata e riportato i risultati del test in un documento. Dai risultati del test abbiamo creato una matrice di confusione con cui siamo riusciti ad effettuare un'analisi delle prestazioni dell'algoritmo di apprendimento utilizzato. Nell'ambito dell'Intelligenza artificiale, la matrice di confusione, detta anche “tabella di errata classificazione”, restituisce una rappresentazione dell'accuratezza di classificazione statistica. Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. Sono stati presi i volti per un ulteriore test dalle puntate dei “I Soliti Ignoti”, uno dei programmi/quiz televisivi a premi in onda su Rai 1 nel quale il gioco finale consiste nel “*parente misterioso*”, nella quale il concorrente deve indovinare quale delle otto identità ha una qualsiasi parentela con il parente misterioso. Noi abbiamo preso in considerazione alcune puntate recuperate da RaiPlay e successivamente rilevati i volti delle otto identità e del parente misterioso. Attraverso la nostra rete neurale Siamese precedentemente addestrata, siamo andati a verificare la parentela tra le otto identità del programma e il parente misterioso,

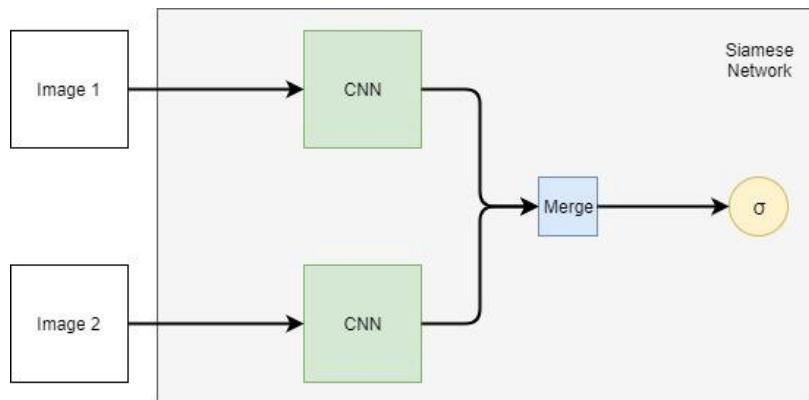
protagonista di questa ultima fase del programma. Quindi tutte le operazioni per portare a termine il progetto consistono in tre parti differenti:

1. nella prima parte abbiamo analizzato e appreso il codice dato ed effettuato gli allenamenti, compresi di test;
2. nella seconda parte abbiamo effettuato il transfer learning sul nuovo dataset;
3. nella terza parte abbiamo utilizzato la rete precedentemente allenata per rilevare le possibili parentele tra il parente misterioso e le otto identità nelle puntate del programma televisivo descritto in precedenza.

A queste tre fasi poi aggiungiamo una fase in più dove facciamo il confronto tra uno studio sulle parentele fatto con le tecniche della SNN e la SIFT. Per completezza abbiamo preferito inserire una interfaccia grafica per mostrare quello che potrebbe essere un esempio per facilitare l'utilizzo di questo software.

Tecnologie Utilizzate

Nella prima parte del progetto abbiamo scaricato il dataset da Kaggle e inserito il codice con repository chiamata dall'autore “A very simple siamese network in Pytorch”. Nella prima settimana il lavoro ci siamo focalizzati sul capire il codice. Abbiamo provato successivamente a studiare la rete Siamese.



Una rete neurale siamese è una rete neurale artificiale che utilizza gli stessi pesi mentre lavora in tandem su due diversi vettori di input per calcolare vettori di output comparabili. Spesso uno dei vettori di output è precalcolato, formando così una linea di base rispetto alla quale viene confrontato l'altro vettore di output.

Un ulteriore studio è stato fatto per comprendere meglio il framework Pytorch, usando il sito “<https://pytorch.org/> ”. Questo sito è stato molto utile poiché ci ha permesso di acquisire familiarità con i concetti e i moduli di PyTorch. Diverse porzioni di codice che abbiamo trovato nella repository studiata combaciavano con questo sito, rendendo facilitato l'apprendimento.

Colaboratory o, in breve, "Colab" permette di scrivere ed eseguire codice Python nel tuo browser con i seguenti vantaggi: nessuna configurazione necessaria, accesso gratuito alle GPU, condivisione semplificata. La scelta di usare Google Colab è stata fatta poiché abbiamo computer con schede video non troppo performanti, offrendoci nella versione gratuita una GPU con 16 gb di ram. Colab, inoltre, ci facilita il lavoro perché non abbiamo bisogno di installare le librerie e permette di eseguire solo determinati blocchi di codice, così da eseguire solo i blocchi di codici che ci servono.

Kaggle è una comunità online di data scientist e professionisti dell'apprendimento automatico . Kaggle consente agli utenti di trovare e pubblicare set di dati, esplorare e costruire modelli in un ambiente di scienza dei dati basato sul Web, collaborare con

altri scienziati dei dati e partecipare a concorsi per risolvere le sfide della scienza dei dati.

Google Drive è un servizio web, in ambiente cloud computing, di memorizzazione e sincronizzazione online. E' stato molto utile nel nostro studio poiché abbiamo caricato tutti i file necessari, come le immagini delle persone o i file csv contenenti i risultati dei test, per poter lavorare in cloud contemporaneamente su diversi computer e avere i file condivisi tra di noi.

Oltre alle librerie classiche di Python come pandas, matplotlib, csv, ecc.. abbiamo utilizzato una libreria chiamata "face recognition" perchè nella terza parte, dello studio riguardante le puntate del programma "i soliti ignoti", i video vengono analizzati frame per frame e, poiché i volti si ripetono più volte, face recognition ci permette di prendere solo una volta i volti dei concorrenti così da non avere ridondanze.

Prima Parte: studio e utilizzo della rete neurale Siamese

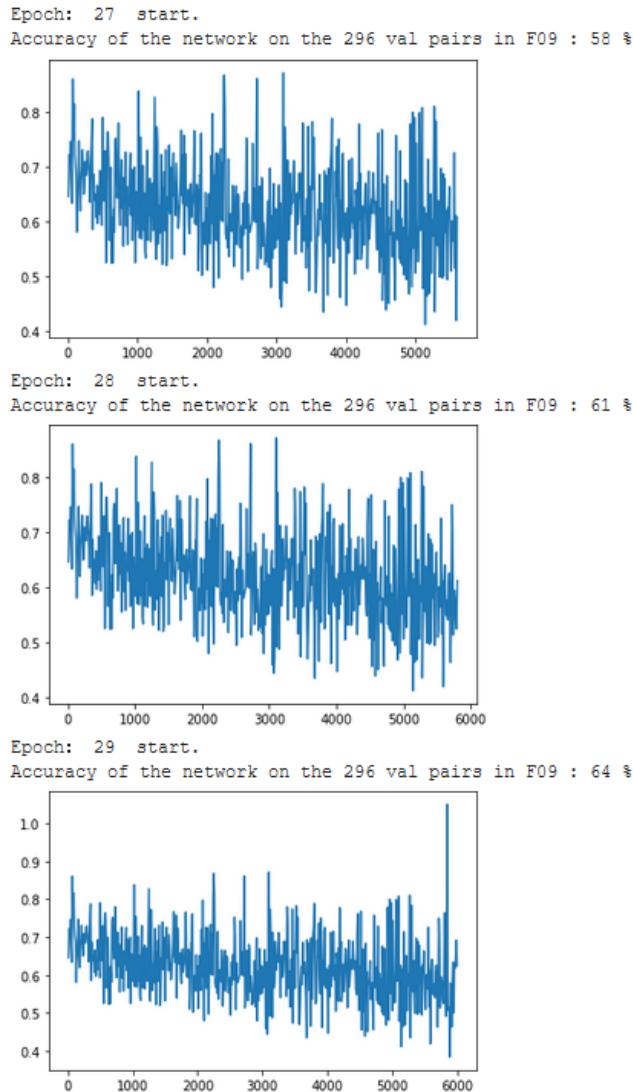
Dopo aver studiato questi aspetti, abbiamo importato il codice su Google Colab, fatto l'upload dei file e cambiato i path. In media, per caricare tutti i file occorrevano tra i 20 ed i 30 minuti. Successivamente, abbiamo intuito che non era l'approccio migliore e per questo motivo abbiamo cercato una soluzione online: abbiamo scoperto che i file potevano essere gestiti tramite Google Drive e quindi abbiamo caricato tutto lì e importato Drive su Colab. Utilizzando i file direttamente da Drive però, le fasi di training impiegavano ore ed ore per essere svolte completamente e per questo abbiamo cercato un ulteriore approccio basato sulla copia dei file in locale su Colab così da velocizzare le operazioni. Ciò ha reso il tutto più rapido e infatti, in media, ogni allenamento durava non più di un'ora. Qui sottostante possiamo osservare la tabella dei risultati ottenuti con i vari esperimenti fatti modificando i vari parametri per poter migliorare la percentuale di train.

BATCH_SIZE	Learning rate	N° epoches	Percentuali finali	Picchi massimi	Tempo	Accuratezza test (kaggle)
16	0,01	30	48-54-48	55	37:29	0.500
16	0,001	30	65-56-59	66	8:12	0.639
16	0,0001	30	58-61-64	65	39:15	0.619
16	0,001	50	58-59-62	67	30:34	0.603
16	0,0001	50	56-58-57	66	23:48	0.629
16	0,0001	20	60-57-60	63	20:50	0.609
16	0,0001	100	54-51-52	67	47:33	0.655
32	0,0001	30	60-62-60	66	5:34	0.618
32	0,01	30	56-49-53	56	10:41	0.500
32	0,001	30	61-62-63	63	10:47	0.633
32	0,0001	50	60-64-62	65	18:01	0.639
32	0,001	50	57-58-57	65	17:59	0.632
32	0,0001	20	65-57-60	68	7:36	0.633
32	0,0001	100	61-56-56	66	37:00	0.653
64	0,0001	30	51-59-62	65	5:24	0.622
64	0,01	30	58-58-60	60	5:20	0.580
64	0,001	30	63-54-55	66	5:23	0.621
64	0,0001	50	55-63-61	67	8:53	0.632
64	0,001	50	57-56-57	64	15:16	0.642
64	0,0001	20	57-64-57	64	6:05	0.636
64	0,0001	100	57-56-61	65	31:00	0.500
128	0,0001	30	66-60-58	66	8:38	0.622

128	0,01	30	59-59-57	62	8:39	0.623
128	0,001	30	56-58-56	66	8:40	0.611
128	0,0001	50	62-56-55	69	14:29	0.634
128	0,001	50	57-52-62	64	14:30	0.646
128	0,0001	20	61-56-60	67	5:55	0.610
128	0,0001	100	63-58-62	67	29:26	0.658

In una prima fase abbiamo cambiato solo il learning rate nella rete neurale Siamese, per non stravolgere troppo il codice e per capire come realmente si comportasse la rete neurale, e abbiamo notato da subito che col valore tendente ad 1 la rete diminuiva l'accuracy del train, viceversa allontanandoci aumentava. Nella tabella non ci sono troppi esperimenti su questa prima fase di sperimentazione perché ancora dovevamo definire un modello di raccolta dati che poi successivamente abbiamo definito nella tabella che possiamo osservare. Ma a sostegno della tesi, per la quale un lr più vicino a 0 dà un risultato migliore, possiamo notare dalla tabella che con un lr = 0.001 il valore di accuracy finale è maggiore di un lr = 0.01 con lo stesso numero di epoch e con lo stesso numero di batch size, nella tabella possiamo osservare questo fenomeno diverse volte ma già nelle prime due righe si nota. Studiare il rendimento del numero di epoch è stato poi il passo successivo. Assodato che la rete neurale Siamese è una rete non grande ed essendo anche il dataset di piccole dimensioni, usando un numero troppo grande di epoch rischiavamo di incontrare problemi di overfitting e di conseguenza, non migliorare l'accuracy. Questa problema però si presenta nel momento in cui associato al numero elevato di epoch abbiamo un batch size basso, mentre quando il batch size è grande l'aumentare delle epoch ha portato anche dei miglioramenti, come per esempio nelle ultime righe della tabella dove abbiamo un batch size di 128 con un numero di epoch uguale a 100 abbiamo ottenuto risultati migliori rispetto a quando abbiamo utilizzato un numero minore di epoch a differenza come già detto nella prima parte in cui con un batch size di 16 o 32 è stato meglio il train con un numero basso di epoch. Quindi non è detto che aumentare troppo o abbassare troppo i parametri si tende sempre a migliorare, dipende spesso anche dalla combinazione con gli altri. Quando abbiamo aggiunto alla batch size un valore maggiore o uguale di 256, Colab avvisava con un warning un problema di memoria e per questo motivo, tutti gli esperimenti con questo valore non sono registrati in tabella. Altra cosa che abbiamo notato è che anche riprovando con gli stessi parametri succede che l'accuracy varia, e questo ci ha spinto a salvare il risultato alla fine del train arrivati ad una buona percentuale per non perderla perchè magari con gli stessi parametri la volta successiva avremmo avuto un risultato inferiore. In diversi test che abbiamo eseguito, quando le percentuali di training erano

molto basse, abbiamo preferito non continuare la fase di training per procedere diversamente modificando i parametri e trovare risultati migliori. Il risultato migliore per di train è stato ottenuto con 16 di batch size, lr uguale a 0,0001 e 30 epoche ottenendo 64% come risultato finale, questo lo possiamo osservare nella foto sottostante



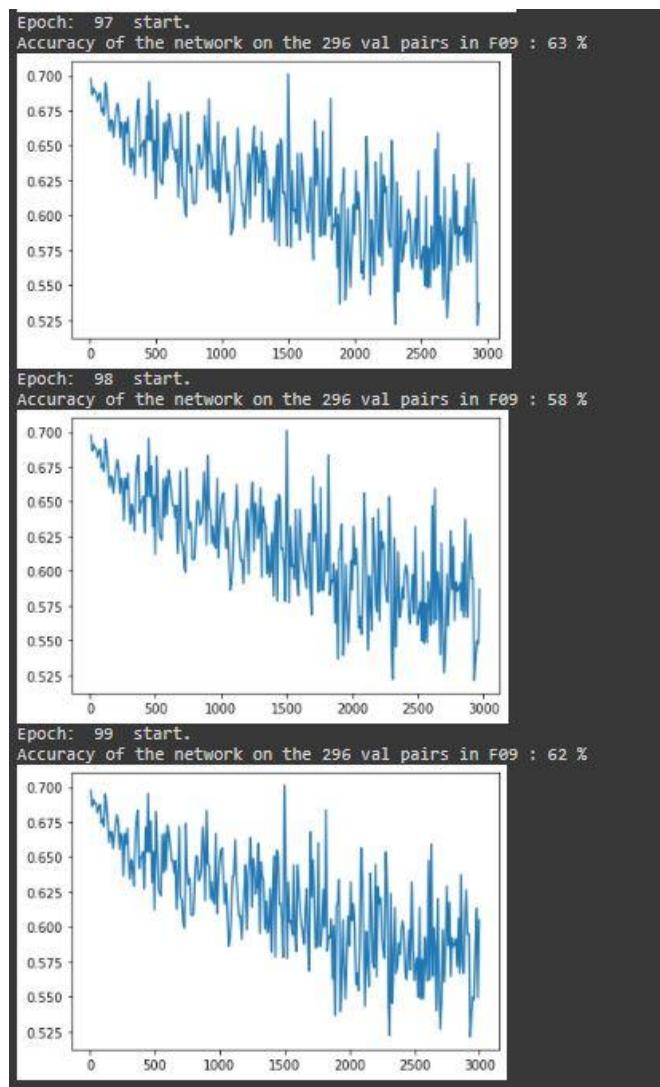
Il fatto che però questo sia stato il miglior train, a livello di accuratezza di test non ha dato grandi soddisfazioni infatti abbiamo ottenuto 0.619.

Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
submission(2).csv	just now	1 seconds	0 seconds	0.619
Complete				
Jump to your position on the leaderboard ▾				

Questa percentuale risulta essere più basso del caso di 128 batch size, lr uguale a 0,0001 e 100 epoche, dove raggiungiamo una percentuale del 62% quindi un train peggiore ma il test migliora, infatti otteniamo 0.658, che è l'accuratezza nei test migliore che abbiamo raggiunto.

Name	Submitted	Wait time	Execution time	Score
submission (19).csv	just now	1 seconds	0 seconds	0.658
Complete				

Qui troviamo i grafici del caso sopra citato.



Quindi abbiamo salvato il modello che ci ha portato il miglior caso di accuratezza sui test per poi utilizzarlo nella seconda fase.

Seconda parte: transfer learning e nuovo dataset

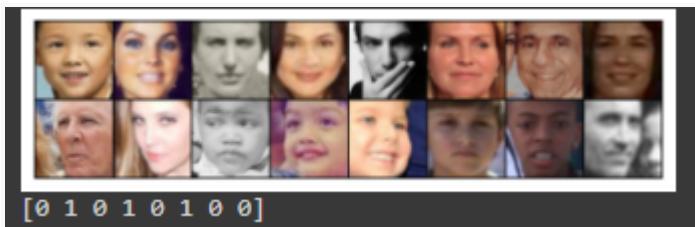
Nella seconda parte di questo lavoro abbiamo considerato un secondo dataset, che rispetto al primo dataset, con un numero inferiore di immagini e con la differenza che le possibili parentele tra due persone non dessero come risultato solo 1 o 0 se due persone sono parenti o meno ma anche il grado di parentela tra due individui, precisamente le relazioni individuabili sono: padre-figlio, madre-figlia, padre-figlia e madre-figlio. La nuova fase però voleva comunque sfruttare il lavoro fatto in precedenza quindi l'obiettivo è stato quello di allenare la nuova rete sfruttando il modello precedentemente addestrato e salvato, quindi effettuare il cosiddetto transfer learning. L'idea alla base del transfer learning prevede di modificare l'apprendimento della rete per risolvere un compito diverso, a patto che il dato iniziale abbia la stessa dimensione/forma. Visto che i nostri dataset avevano le stesse caratteristiche, abbiamo potuto utilizzare il transfer learning.

Il primo passo svolto è stato salvare su Google Drive la rete neurale sul primo dataset con il risultato migliore, così da poterla utilizzare per i training del secondo dataset, azione svolta durante il lavoro nella prima fase del progetto. All'inizio abbiamo provato un approccio in cui carichiamo la rete e aggiungiamo solo i layer fully connected, approccio definito estrazione delle feature. I primi due layer fully connected mantengono i valori come nella prima rete neurale e gli altri vengono modificati. Abbiamo quindi proceduto a fare dei test utilizzando solo questo tipo di modifica. Successivamente però abbiamo pensato che il metodo non era abbastanza per ottenere buoni risultati e quindi abbiamo provato ad applicare fine tuning, infatti di seguito troveremo gli esperimenti effettuati sia prima che dopo aver applicato il metodo del fine tuning insieme al freeze, quindi abbiamo effettuato una tecnica combinata. Quello che abbiamo applicato inizialmente è il metodo dell'estrazione delle feature senza nè freeze nè fine tuning. Successivamente, con l'aggiunta del nuovo codice abbiamo modificato il parametro all'interno dei layer "require_grad" a True o a False, per applicare rispettivamente il fine tuning o il freeze (siamo arrivati a questa conclusione grazie l'ausilio del forum pytorch in cui abbiamo trovato una discussione in cui si parlava di questo parametro al link: <https://discuss.pytorch.org/t/what-does-param-requires-grad-true-or-false-do-in-the-pretrained-model/99026>). Per il training, oltre al settaggio della rete su cui lavorare, abbiamo creato un unico file csv sul secondo dataset per lavorare con lo stesso codice con cui abbiamo lavorato sul precedente, ed è stato necessario traslare i dati contenuti nei file math del secondo dataset in un file csv con la stessa struttura, o comunque simile, al file csv che conteneva le info per il primo dataset. Nel nuovo file csv abbiamo inoltre inserito le classi di relazioni delle coppie di immagini indicate con valori da 1 a 4, o con valore 0 indicante la non parentela. Ora, poiché il nuovo dataset è diverso (per via dell'aggiunta di 3 classi rispetto al primo in cui ne avevamo solo

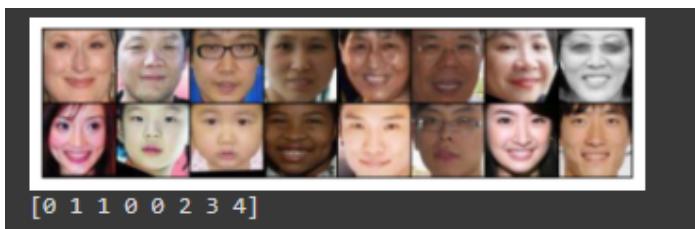
due) abbiamo modificato la classe “`trainingDataset`” per adattarlo al nuovo caso di studio e per testare se la logica di esecuzione venisse svolta correttamente.

Abbiamo usato il codice precedente che ci permetteva di mostrare a schermo le immagini e i risultati di confronto sono i seguenti.

- Primo dataset:



- Secondo dataset:



Come possiamo notare i valori ottenuti vanno da 0 a 4, indicanti il grado di parentela, e non più solo valori 0 e 1 come nel primo dataset. Completato ciò abbiamo allenato la nuova rete neurale e abbiamo ottenuto i seguenti risultati:

BATCH_SIZE	Learning rate	N° epoche	Percentuali finali	Picchi massimi	Tempo	Accuratezza test
16	0,001	30	52-43-41	54	2:40	0.50
16	0,0001	50	38-37-31	50	15:40	0.52
16	0,0001	100	34-40-41	45	11:57	0.51
32	0,0001	30	34-44-33	47	6:14	0.53
32	0,0001	50	40-34-41	47	10:56	0.50
32	0,0001	25	38-43-33	52	2:50	0.51
64	0,0001	50	41-44-37	51	7:58	0.50
64	0,001	50	36-33-40	50	3:44	0.50
64	0,0001	20	43-38-31	48	1:38	0.49
128	0,0001	50	33-37-37	51	4:38	0.50
128	0,001	50	47-48-34	50	3:35	0.50
128	0,0001	100	33-37-37	48	8:31	0.51

Questa tabella corrisponde ai risultati degli esperimenti fatti senza effettuare fine tuning o freeze, risultati che non ci hanno soddisfatto, quindi abbiamo pensato come precedentemente detto, di lavorare con una tecnica combinata sia con fine tuning che

con freeze, rispettivamente freezando i primi 10 livelli della rete, impostando il parametro required_grad = False per questi primi 10 livelli, invece per i restanti abbiamo impostato il parametro a True per il fine tuning, quindi questi livelli verranno riaddestrati utilizzando anche l'allenamento precedente, invece quelli freezati non vengono riaddestrati. Il codice è il seguente:

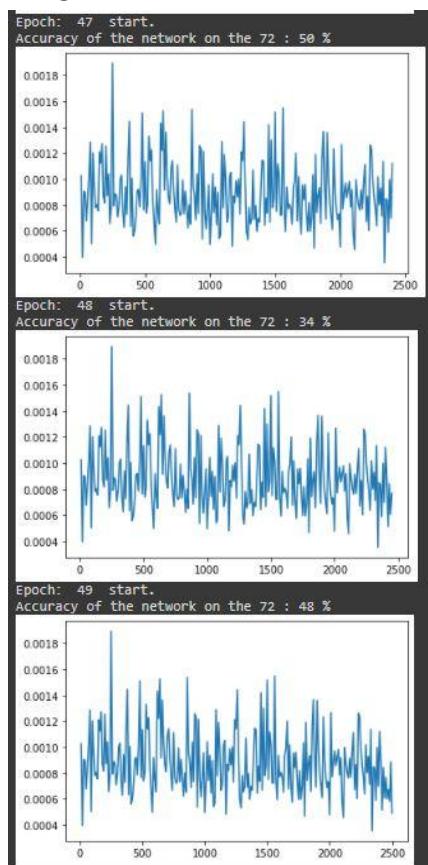
```
ct = 0
for param in model.parameters():
    print(ct)
    if ct > 9 and ct<=14: param.requires_grad = True #per settare il fine tuning
    else: param.requires_grad = False #per settare il freeze
    ct = ct + 1
```

Da ciò abbiamo effettuato altri test con la rete impostata in questo modo ottenendo i seguenti risultati:

BATCH_SIZE	Learning rate	N° epoche	Percentuali finali	Picchi massimi	Tempo	Accuratezza test
16	0,0001	40	31-37-33	40	3:34	0.54
16	0,0001	30	24-45-44	47	8:10	0.51
16	0,001	30	50-47-47	54	2:41	0.54
16	0,0001	50	50-34-48	50	4:36	0.55
16	0,0001	100	36-40-36	50	9:14	0.54
16	0,001	35	44-38-43	52	3:08	0.52
16	0,001	25	38-30-33	48	2:14	0.54
32	0,001	35	41-31-38	52	2:05	0.54
32	0,001	25	38-38-29	44	1:29	0.53
32	0,001	30	33-41-30	45	1:09	0.56
32	0,0001	50	40-40-36	45	2:54	0.50
32	0,001	50	37-47-36	51	2:52	0.52
32	0,0001	20	36-26-41	54	1:13	0.52
32	0,0001	100	29-34-36	43	6:13	0.50
64	0,0001	50	26-27-29	43	2:32	0.45
64	0,001	50	34-27-27	44	2:32	0.50
64	0,0001	20	31-41-41	41	1:00	0.51
128	0,0001	50	40-31-34	44	4:39	0.48
128	0,001	50	36-34-29	44	4:37	0.50
128	0,0001	100	41-40-33	40	9:20	0.50

Si può vedere che impostando la rete in questo modo i risultati dei test sono migliorati, quindi abbiamo deciso di scegliere questo metodo per questo secondo allenamento e, in particolare, abbiamo scelto il modello che ci ha dato un'accuratezza dello 0.55 sul

test perchè oltre ad essere il secondo valore di accuratezza più alto, risulta avere anche una accuratezza nel train più vicina a quella del test rispetto al miglior esperimento sul test col quale arriviamo ad un accuratezza di 0.56 ma avente nel train le percentuali finali con valori bassi. Il metodo di train dopo aver effettuato il transfer learning è stato praticamente identico a quello a cui è stato applicato nella parte precedente, quindi andando a modificare i parametri di batch size, numero epoche e tr. Un'altra cosa che abbiamo valutato rispetto alla prima fase è stata quella di applicare la tecnica di data augmentation, per aumentare il numero delle immagini su cui effettuare il train, perchè, essendo il nuovo dataset più piccolo rispetto al precedente, abbiamo pensato che poteva dipendere da ciò la diminuzione dell'accuratezza sul test, ma così non è stato poichè leggendo in rete questa tecnica si applica in una situazione di overfitting, ovvero quando il sistema da una percentuale di train maggiore rispetto a quella del test, perché il sistema durante il train tende a memorizzare dei risultati piuttosto che apprendere come distinguere le immagini in base a determinati valori delle caratteristiche. Siccome nel nostro caso abbiamo una percentuale di train minore abbiamo reputato che non fosse necessaria questa tecnica. Di seguito possiamo osservare gli ultimi tre risultati grafici del test selezionato.

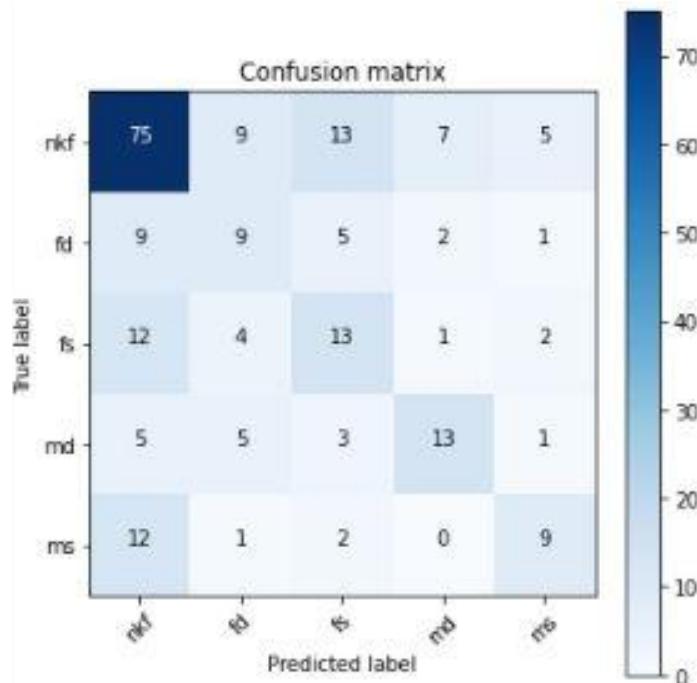


Al fine di accertarsi della qualità del train, abbiamo effettuato dei test i cui risultati sono stati salvati nel file “submission2.csv“ contenente le seguenti caratteristiche:

- **img_pair**: indica i nomi delle immagini confrontate
- **gr_p**: indica il grado di parentela e ha valori da 0 a 4
- **tp_r**: indica se esiste una parentela e ha valori 0 o 1
- **is_related**: è la risposta della rete neurale al confronto tra le immagini

Svolta la fase di testing abbiamo calcolato la matrice di confusione che restituisce una rappresentazione dell'accuratezza di classificazione statistica. Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. Della matrice di confusione abbiamo inoltre calcolato le metriche di valutazione delle prestazioni come l'accuracy, la recall, la precision e l'f1 score. I parametri sono stati calcolati per ogni tipo di classe (avendo un molteplicità di classi) e di sotto oltre alla matrice di confusione si può osservare anche la tabella che sintetizza i valori di queste metriche, ovviamente mostriamo la tabella che ci ha restituito lo studio con il modello selezionato ovvero quello con l'accuratezza sul test dello 0.55.

```
Confusion matrix, without normalization
Accuracy tot: 0.55
```



	Recall	Precision	Accuracy	F1_score
nkf	0.66	0.63	0.59	0.65
fd	0.32	0.08	0.18	0.12
fs	0.36	0.11	0.22	0.17
md	0.57	0.11	0.16	0.18
ms	0.50	0.08	0.13	0.13

Per calcolare questi parametri abbiamo innanzitutto calcolato la matrice di confusione poi da lì abbiamo calcolato i seguenti valori:

- **True positive** (TP, o veri positivi): sono i casi in cui abbiamo previsto che tra la coppia c'è un tipo di parentela ed esiste effettivamente quel tipo di relazione. Tramite la tabella otteniamo la quantità dei true positive di una classe recuperando il valore della cella che interseca la stessa classe sulle ascisse e sulle ordinate.
- **True negative** (TN, o veri negativi): il modello ha previsto che la parentela non c'è e non esiste veramente. Tramite la tabella otteniamo la quantità dei true negative di una classe sommando i valori delle celle sulla diagonale tranne il valore contenuto nella cella che interseca la stessa classe sulle ascisse e sulle ordinate.
- **False positive** (FP, o falsi positivi): sono i casi in cui abbiamo previsto che c'è un certo tipo di parentela, ma in realtà non è corretto. Tramite la tabella otteniamo la quantità dei false positive di una classe sommando i valori delle celle nella colonna corrispondente a quella classe, tranne il valore contenuto nella cella che interseca la stessa classe sulle ascisse e sulle ordinate.
- **False negative** (FN, o falsi negativi): il modello ha previsto che la parentela non c'è, anche se in realtà esiste. Tramite la tabella otteniamo la quantità dei false positive di una classe sommando i valori delle celle nella riga corrispondente a quella classe, tranne il valore contenuto nella cella che interseca la stessa classe sulle ascisse e sulle ordinate.

Recall che è la capacità di un classificatore di trovare tutte le istanze positive. Per ogni classe è definito come il rapporto tra i veri positivi e la somma dei veri positivi e dei falsi negativi. Precision che è l'abilità di un classificatore di non etichettare un'istanza positiva che è in realtà negativa. Per ogni classe è definito come il rapporto tra veri positivi e la somma di veri e falsi positivi. F-score è una media armonica ponderata delle metriche *Precision* e *Recall* in modo tale che il punteggio migliore sia 1 e il peggiore sia 0. Accuracy indica l'accuratezza del modello come dice il nome. Pertanto, la migliore accuratezza è 1, mentre la peggiore è 0. L'accuracy totale è stata

calcolata sommando i risultati giuste delle predizioni confrontandoli con quelli effettivi (ovvero sommando tutta la diagonale della matrice) diviso il numero di casi di test. I valori non sono incoraggianti dato dal fatto che anche se un'accuratezza del 55% con 5 valori come risposta non è così disastrosa, contando sempre il fatto che la rete è piccola, queste risposte corrette si concentrano sulla classe della non parentela. I risultati ottenuti dalla matrice di confusione, come possiamo notare, hanno percentuali maggiori su alcuni parametri, quali accuracy, F1-score e precision riguardanti la prima riga dove troviamo la non-relazione tra due individui poiché il numero di risultati del test in cui esiste la non relazione è maggiore rispetto alle restanti classi, quindi questo ci fa capire che il sistema si è fossilizzato sul restituire troppo spesso come risultato la soluzione di non relazione, una motivazione per cui questo potrebbe avvenire secondo noi perché quando generiamo il valore casuale nel nostro codice, questo viene generato sempre tra 0 e 1, è vero che il train tra parenti e non parenti sarà sempre 50 e 50 ma il 50% delle parentele comprenderà tutti i tipi di classi. Mentre la classe 0 verrà allenata il 50% delle volte le altre quattro dovranno dividersi il restante 50%. Quindi il sistema studia una maggioranza di casi della classe 0 piuttosto che delle altre e questo fa sì che il modello generato dalla rete sia più propenso a restituire tale risultato.

Per la matrice abbiamo seguito la seguente guida:
<https://www.lorenzogovoni.com/matrice-di-confusione/>

Terza parte: puntate dei soliti ignoti e test

Nella terza parte dello studio abbiamo scaricato le puntate dei “soliti ignoti” dal sito “<https://www.raisplay.it/>”. Da qui sono state scelte 10 diverse puntate, questi video poi sono stati divisi in frame con le funzioni `convert_game_parents()` e `convert_initial()`, rispettivamente la prima salva i frame del gioco durante la fase del “parente misterioso” e l’altra invece salva i frame dai primi 30 secondi per poter riconoscere correttamente il presentatore e il concorrente. Le funzioni effettuano un lavoro di divisione frame by frame da un preciso minutaggio fino al minuto da quel minutaggio, salvando il frame solo se contenente un volto al suo interno. Sono state poi utilizzate funzioni apposite di detection del volto per poter tagliare i volti dai frame dei video. Inizialmente abbiamo avuto un problema riguardante i frame, poiché venivano salvati tutti i volti delle persone nel video e non solo, anche della stessa persona veniva salvato più volte lo stesso volto ma da angolazioni diverse, arrivando ad ottenere anche migliaia di immagini dello stesso volto. Questo problema è stato risolto utilizzando la libreria `Face_Recognition`, questa libreria ci permette di prendere un volto salvato e confrontarlo. Ogni volto nei frame viene confrontato con quelli già salvati in precedenza così da ridurre i volti ridondanti e immagazzinare quindi una quantità notevolmente inferiore di immagini, ottenendo quindi un'unica copia di ogni volto. Verranno raggruppati alla fine 11 volti, 8 riguardanti i partecipanti e il parente misterioso, il presentatore e il concorrente, anche se non sempre tutti i volti vengono prelevati infatti abbiamo degli episodi in cui ci troviamo con un partecipante in meno. Un ulteriore problema riguardante la cattura dei volti stava nel fatto che i volti venivano catturati anche di lato, perchè all’inizio il “parente misterioso” veniva mostrato ruotando di 180° e siccome viene catturato il primo frame in cui è visibile il volto veniva catturato di lato, poiché la nostra rete neurale è stata addestrata solo su volti frontali non riusciva a riconoscere l’eventuale parentela, o comunque poteva essere una delle motivazioni, quindi c’è stato un lavoro per far partire l’analisi dei frame da quando il “parente misterioso” era già ruotato. Di seguito inseriamo alcune analisi delle puntate osservate con relativi commenti:



[2, 2, 2, 2, 2, 2, 2, 2]

- in questa prima immagine possiamo osservare che la relazione di parentela è stata riconosciuta positivamente ma in maniera parziale poichè il parente misterioso, in questo caso, è una ragazza e la soluzione del test invece è stata la relazione 2, ovvero ‘padre-figlio’. Oltre a ciò, è stata erroneamente segnalata la parentela con tutte le identità, infatti l’identità numero 2 era l’unica corretta;



[2, 2, 2, 2, 2, 2, 2, 2]

- nella seconda immagine osserviamo una situazione molto simile al caso precedente poiché abbiamo ottenuto gli stessi risultati su ogni coppia di persone e quindi anche in questo caso la relazione ‘padre-figlio’ è corretta per la relazione con l’identità numero 1 ed errata per tutte le altre identità. A differenza però del primo caso, il figlio è stato riconosciuto correttamente;



[2, 2, 2, 0, 2, 2, 2, 2]

- anche in questo caso, con i risultati ottenuti il parente misterioso è stato riconosciuto correttamente come *figlio* ma non è stato associato correttamente visto che il *padre* era l’identità numero 2 (la prima coppia della nostra immagine) e invece è stato associato anche con le altre 6 identità erroneamente;



[0, 2, 2, 2, 2, 4, 4, 2, 0]

- in questo quarto test la relazione è stata rilevata correttamente ma in maniera parziale tra parente misterioso e identità numero 3 poiché il risultato è stato ‘padre-figlio’ ma il parente misterioso era una ragazza. Negli altri casi, 6 parentele errate e l’ultima correttamente segnalata come ‘non parentela’;



[2, 2, 2, 4, 2, 2, 2]

- nella quinta immagine che vediamo, si può osservare che la relazione tra parente misterioso e sesta identità è stata rilevata come ‘padre-figlio’ ma in realtà sarebbe dovuta essere ‘madre-figlio’. Le restanti, ovviamente, sono tutte

scorrette. Da aggiungere anche un piccolo errore di riempimento della matrice, probabilmente perché sono stati rilevati più volti di quanti il sistema se ne aspettava;



[3, 2, 2, 3, 2, 2, 2, 2]

- qui la relazione corretta era tra il parente misterioso e l'identità numero 2 con relazione di tipo ‘padre-figlia’. Il nostro risultato ottenuto, invece, è stata la relazione di tipo ‘padre-figlio’ e quindi errata riguardo al sesso del parente misterioso. Tutte le altre sono scorrette. Eccezione in questo caso particolare, Amadeus che è stato inserito all’interno del gruppo delle 8 identità anche con controllo effettuato;



[2, 2, 2, 2, 2, 2, 2]

- in questo caso la relazione vincente è quella con l’identità numero 7 (coppia numero 5 nel nostro caso) dove è stato rilevato correttamente il padre ma non la *figlia*, infatti il risultato è stato ‘padre-figlio’. Le restanti sono scorrette. Da segnalare, in questo caso, degli errori di ritaglio del frame probabilmente dovuti a complicazioni video della puntata o lo stesso problema della quinta immagine;



[2, 2, 2, 2, 2, 2, 0, 2, 2, 0]

- in quest’altra puntata l’unica ‘non parentela’ rilevata è corretta, mentre la relazione vincente è la quarta che nel nostro caso è stata rilevata correttamente a metà perchè il risultato è stato ‘padre-figlio’ ma in realtà sarebbe dovuta essere ‘madre-figlio’. Le restanti, invece, sono tutte erroneamente rilevate;



[2, 0, 2, 2, 2, 2, 2, 2, 0]

- in questo penultimo test, purtroppo evidenziamo che tutti i risultati sono errati poiché la relazione vincente è quella tra parente misterioso e seconda identità, rilevata nel nostro caso come ‘non parentela’. Negli altri casi (tranne l’ultima coppia), il parente misterioso è stato correttamente evidenziato come *padre*;



[2, 2, 2, 2, 2, 2, 2, 2]

- in questo ultimo caso è stata rilevata perfettamente la relazione vincente, cioè la settima coppia tra l’identità numero 7 e il parente misterioso. Tutte le altre, però, sono scorrette.

Conclusa l’analisi di tutte le puntate possiamo concludere che i risultati del lavoro non sono soddisfacenti, ma comunque non ci sorprendiamo particolarmente perché partiamo pur sempre da una percentuale molto bassa di accuratezza, dipeso anche dalla qualità della rete, che dopo averla spinta più in là possibile possiamo concludere non essere adatta a tale lavoro, a differenza di altre reti che abbiamo visto da alcune nostre ricerche in rete ottenere risultati migliori. Supponiamo che la relazione di tipo 2, ovvero ‘padre-figlio’, è quella più presente perché il modello probabilmente riconosce qualcosa di visto durante il train di quel tipo di classe all’interno delle immagini della puntata dei soliti ignoti, come per esempio potrebbe essere la presenza di sfondi all’interno delle immagini, i volti non sono centrati pienamente ma alcune volte si vede altro. Questa idea è sorta dai seguenti test , quando il volto è centrato la rete riesce a restituire risultati migliori, su quella che potrebbe essere la possibile relazione, invece quando i volti non sono centrati o presentano sfondi tende a restituire il valore 2. Infatti mentre nel primo set di immagini sottostanti il parente misterioso ha sempre lo sfondo, e quindi ci restituisce sempre due, nel secondo set possiamo vedere che il parente misterioso si vede più centrato e in corrispondenza con un volto che si vede bene tanto quanto (1° colonna) la relazione restituita è giusta, infatti la classe 3 è la classe madre-figlia.

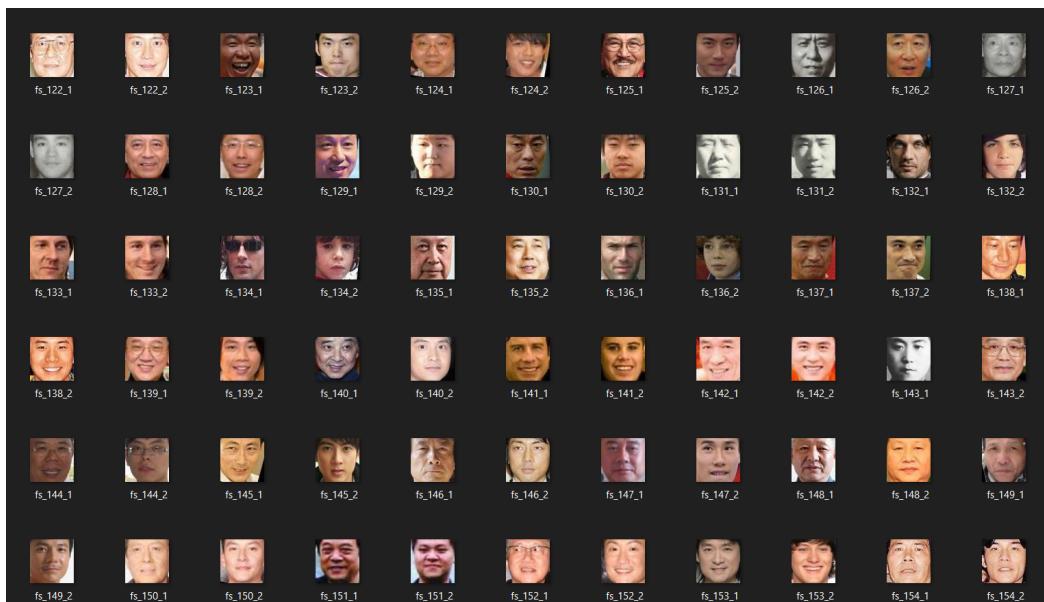


[2, 2, 2, 2, 2, 2, 2, 2]



[3, 2, 2, 3, 2, 2, 2, 2]

Le immagini seguenti sono le immagini utilizzate per il training nella relazione padre-figlio e come possiamo notare diverse immagini non sono centrate e hanno sfondi di diversi colori e forse per questo la rete neurale ha accostato questo tipo di caratteristica alla classe 2. Immagini con sfondo sono presenti anche nelle altre cartelle ma con minore rilevanza.



Parte Facoltativa: Scale Invariant Feature Transform (SIFT)

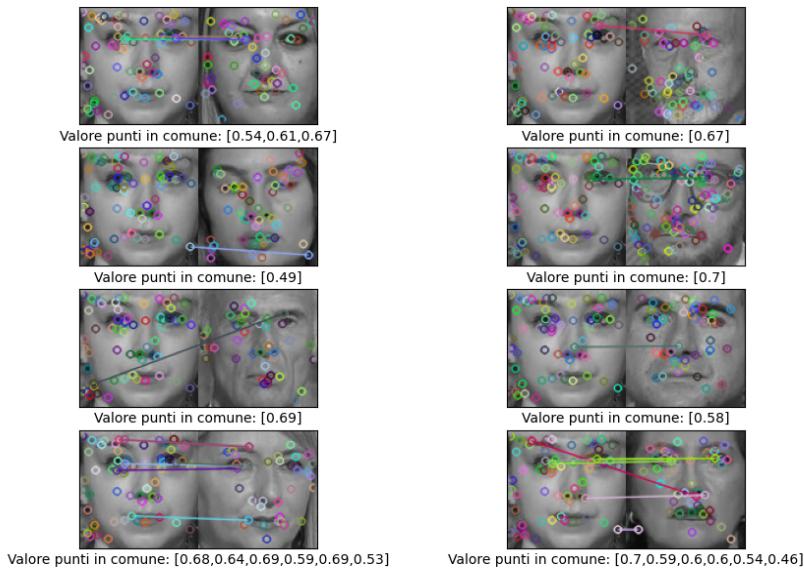
Lo Scale-Invariant Feature Transform (SIFT) è un algoritmo piuttosto complicato. Ci sono principalmente quattro passaggi coinvolti nell'algoritmo:

1. **Selezione dei picchi in scala spaziale:** posizione potenziale per trovare le caratteristiche.
2. **Localizzazione dei punti chiave:** individuazione accurata dei punti chiave della funzione.
3. **Assegnazione dell'orientamento:** assegnazione dell'orientamento ai punti chiave.
4. **Descrittore del punto chiave:** descrive i punti chiave come un vettore ad alta dimensione.
5. **Keypoint Matching**

I punti chiave tra due immagini vengono abbinati identificando i loro vicini più prossimi. Ma in alcuni casi, la seconda corrispondenza più vicina potrebbe essere molto vicina alla prima; può accadere a causa del rumore o per altri motivi. In tal caso, viene preso il rapporto tra la distanza più vicina e la seconda distanza più vicina.

SIFT in OpenCV è utilizzabile tramite la classe `xfeature2d` e la funzione `SIFT_create()`, ovviamente facendo l'import di `cv2`. Successivamente, dopo aver eseguito l'algoritmo sulle coppie di immagini di riferimento, in genere si creano dei plot descrittivi del risultato ottenuto in base al Keypoint Matching.

Considerando i confronti che hanno avuto maggior rilievo nel nostro studio, come possiamo osservare nelle immagini successive, abbiamo le previsioni delle puntate dove la macchina ha trovato le possibili parentele. Nell'utilizzo del SIFT la maggior parte delle volte vengono assegnati diversi punti alle figure aventi lo stesso sesso. Inoltre, il SIFT tende ad assegnare la maggior parte delle volte punti in comune tra persone anche se non esiste la parentela e dal nostro studio non riesce mai ad assegnare il maggior numero di punti e di percentuale dei valori in comune alle effettive parentele. Quindi, possiamo concludere dicendo che la tecnica SIFT fallisce nel problema della Kinship Recognition. Vogliamo ora mostrare tre diversi esperimenti svolti utilizzando il SIFT per mostrare ciò che è stato ottenuto tramite questo algoritmo.



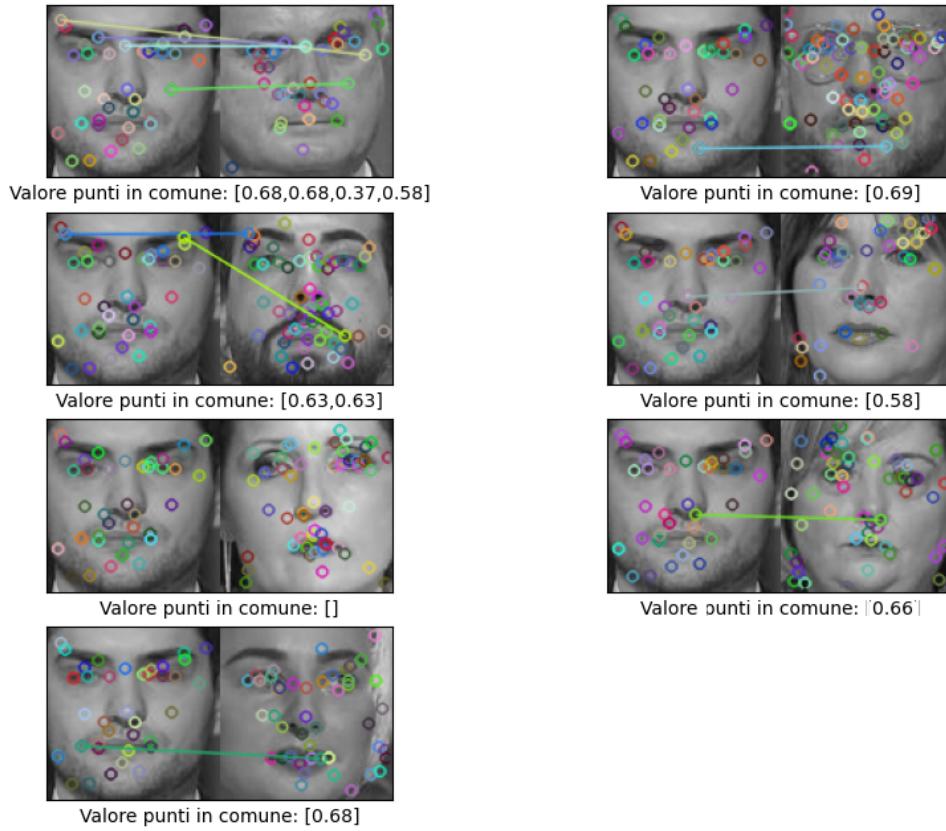
Tramite questo confronto notiamo che le figure femminili analizzate tendono ad avere diversi punti in comune e porta erroneamente a considerare la parentela madre-figlia. La seconda immagine a sinistra, forse perché ha gli occhi chiusi, ha dato un unico punto in comune con una percentuale bassa. La parentela reale però è padre-figlia ed è la prima immagine a destra che ha un solo punto in comune con una percentuale del 67%, che non è nemmeno la più alta percentuale di match, considerando tutti i punti trovati e che quindi ci fa dedurre che magari non sono loro i parenti tra le otto coppie, trovando comunque risultati migliori su altri tipi di confronti.



[2, 2, 2, 2, 2, 2, 2, 2]

Confrontando questi risultati con la rete neurale possiamo notare che la rete neurale non è riuscita a trovare la relazione padre-figlia, ma perlomeno ha individuato la parentela tra i due. Sottolineando che la rete neurale non ha funzionato per nulla bene il SIFT sembra funzionare ancor peggio, perchè mentre con la rete neurale l'effettivo parente non viene escluso con il SIFT si, perchè appunto associamo la parentela ad altre coppie che hanno ottenuto match migliori. Con questo primo confronto possiamo pensare che forse il SIFT tende a dare punti in comune tra persone dello stesso sesso. Ora consideriamo questo esempio riguardante la parentela tra madre e figlio, la corretta parentela è nell'ultima immagine a destra, il SIFT è riuscito a trovare un

punto in comune, riguardante il naso tra madre e figlio.



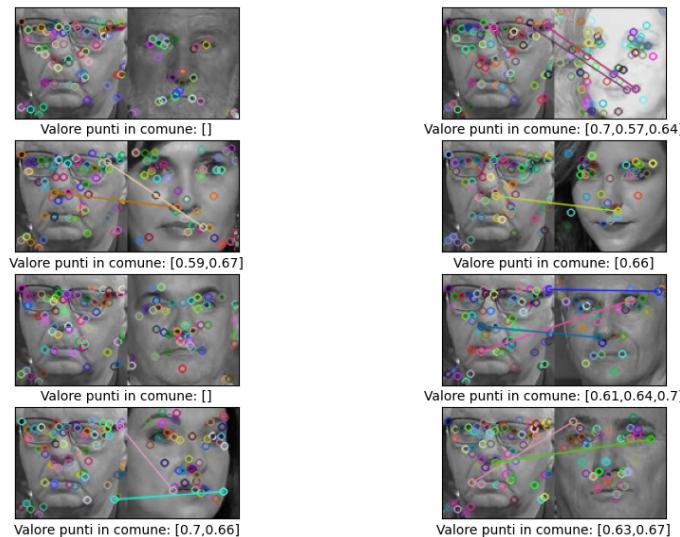
Come possiamo notare però le percentuali più alte e il maggior numero di punti in comune sono ottenuti nei confronti tra figure dello stesso sesso, quindi la tesi precedentemente realizzata sta prendendo sempre più piede. Questo forse perchè il SIFT è una tecnica che tende ad individuare somiglianza tra le immagini, e forse in un'immagine è più semplice per il sistema trovare somiglianze dovute a tratti maschili o femminili in un volto piuttosto che similitudini tra parenti.



[2, 2, 2, 4, 2, 2, 2]

La rete neurale non è riuscita a trovare la relazione madre-figlio, restituendo nella maggior parte dei casi come previsione la relazione padre-figlio. Quindi sia la rete neurale che il SIFT, considerando anche che le immagini da testare non sono complete, non hanno dato risultati soddisfacenti. Abbiamo provato a determinare altri tipi di schemi di match del SIFT, visto che quello riguardante il sesso di una persona è

una caratteristica generica abbiamo provato a analizzare un'altra caratteristica generica riconoscibile in queste immagini, ovvero l'età. Proviamo, quindi, a vedere se il SIFT dipendente anche dal progresso dell'età delle persone per vedere se particolari segni dell'invecchiamento tende a renderli come punti in comune. Il seguente esperimento del SIFT però nega ciò che abbiamo ipotizzato: la prima immagine a sinistra non riconosce neanche un punto in comune, guardando anche la terza immagine a sinistra alla quale non vengono associati punti in comune, pur essendo in entrambi i casi entrambe le persone anziane. Considerando anche altri partecipanti notiamo che il SIFT assegna tre punti in comune alla prima (nonostante la qualità dell'immagine) e alla terza immagine a destra, la prima è la reale parentela padre-figlia.



L'immagine della figlia non è stata acquisita correttamente ma il SIFT è riuscito ugualmente ad assegnare diversi punti in comune, ma considerando le percentuali il risultato migliore è il terzo a destra. Il SIFT è riuscito a dare diversi punti in comune sia alle figure maschili che femminili, assegnando anche punti con percentuali superiori al 66% alle figure femminili, contrariamente a come abbiamo spiegato in precedenza dicendo che il SIFT assegna percentuali migliori e un maggior numero di punti in comune a figure dello stesso sesso.



La rete neurale non è riuscita a trovare la relazione padre-figlia, non riconoscendo proprio la parentela, comunque è da considerare la qualità dell'immagine non ottimale, e assegnando a diverse coppie la relazione padre-figlio anche se in alcuni casi sono donne. In ogni caso entrambe non sono riuscite a trovare la reale parentela e ad assegnare valori nulli alle altre coppie, ma in questo caso il SIFT ci è andato più vicino. Quindi in conclusione, avendo analizzato anche altri casi oltre a quelli mostrati, possiamo dire che il SIFT non è una tecnica utilizzabile per questo tipo di lavoro, probabilmente perché è una tecnica rivolta più a riconoscere soggetti uguali all'interno di immagini diverse, quindi più i soggetti in entrambe le figure sono uguali migliore sarà il match. La parentela tra due persone tende a far somigliare quelle due persone e questo metodo non riesce a cogliere questa somiglianza, anche non è sempre semplice da cogliere da una singola foto di un volto.

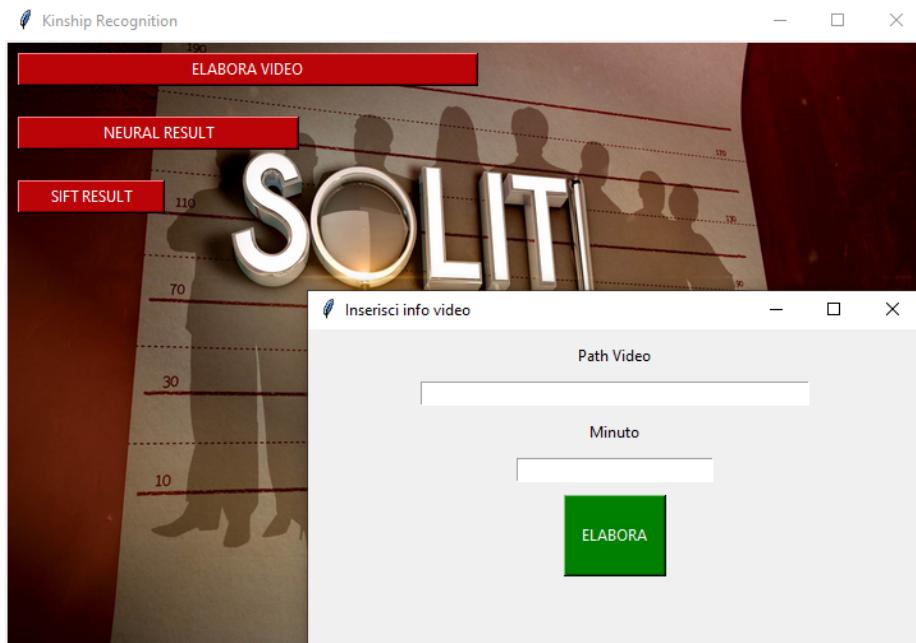
Parte Facoltativa: User Interface

L'interfaccia creata è composta da un'immagine di sfondo caratteristica del programma su cui si basa il progetto e tre bottoni.

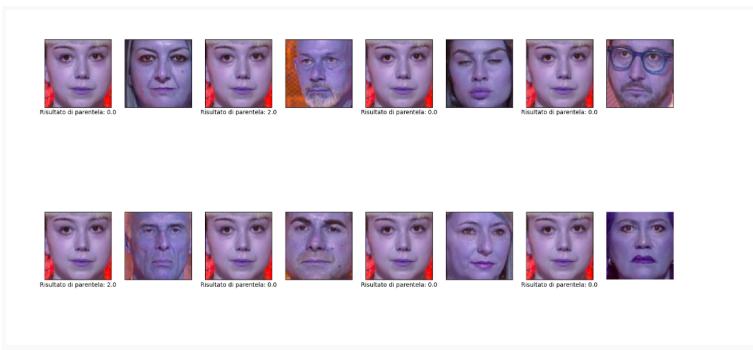


I bottoni hanno diverse funzionalità:

- **ELABORA VIDEO:** per aprire una dialog box che consente di inserire il link della puntata da analizzare e il minuto preciso in cui inizia il gioco del parente misterioso;



→ NEURAL RESULT per visualizzare il plot dei risultati dall'analisi effettuata con la rete neurale.



→ SIFT RESULT: per visualizzare il plot dei risultanti dall'analisi effettuata con il metodo SIFT.



Conclusioni

Lo studio di questo esperimento ci ha permesso di affrontare diverse tematiche. Abbiamo appreso e utilizzato una rete neurale per poter allenare un modello attraverso l'utilizzo di dati prelevati da un dataset, nel nostro caso immagini, e successivamente con la stessa rete attraverso tecniche di transfer learning, altra tematica studiata per portare a termine il progetto, per allenare nuovamente lo stesso modello con un ulteriore dataset. Sono stati svolti diversi test per concludere il progetto e questo ci ha permesso di capire diversi aspetti del machine learning, come per esempio l'importanza di un cambiamento su un singolo parametro, soprattutto la maggior parte dei dubbi sono stati risolti grazie alle guide di pytorch e le discussioni nei diversi forum online. Un altro argomento affrontato è stato di cercare all'interno di un video determinati elementi, che nel nostro caso risultavano essere volti, e ci riteniamo molto soddisfatti del risultato, anche se, in alcuni casi, alcuni volti nelle puntate non sono stati bene riconosciuti. Anche lo studio sul SIFT è stato molto interessante, è una tecnica molto utilizzata per riconoscere la presenza di oggetti simili in due immagini, ma evidentemente il volto umano non fornisce abbastanza punti in comune al SIFT per riconoscere la parentela. Per quanto riguarda invece l'esperimento svolto, per vedere se due persone sono parenti e indicando anche il grado di parentela, reputiamo che i risultati ottenuti, utilizzando la rete neurale Siamese, non siano particolarmente utili ai fini dell'individuazione della parentela e del grado di parentela, pur avendo svolto molti test. Probabilmente utilizzando modelli pre addestrati come vgg face, resnet, etc.. saremmo riusciti ad ottenere risultati migliori. Reputiamo che per i tempi tra studio e applicazione effettiva dei concetti siamo riusciti a massimizzare i valori ottenuti dai train e dai test ma questi valori non sono abbastanza utili per individuare la parentela tra due individui.