SAPIENZA
UNIVERSITÀ DI ROMA

Rilevamento dell'inganno attraverso caratteristiche facciali

Facoltà di Ingegneria dell'informazione, Informatica e Statistica

Corso di Laurea in Informatica

Candidato

Ruben Seror
Matricola 1815399

Relatore

Prof. Luigi Cinque

Correlatore

Prof. Daniele Pannone

Anno Accademico 2020/2021

Tesi non ancora discussa

Rilevamento dell'inganno attraverso caratteristiche facciali
Tesi di Laurea. Sapienza – Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: rubser17@gmail.com

Indice

Capitolo 1

Introduzione

In questo capitolo viene introdotto il lavoro presentato e le possibili applicazioni (Paragrafo 1.1). I metodi proposti dallo stato dell'arte sono riassunti (Paragrafo 1.2) . Infine, vengono descritti i contributi del lavoro e come è organizzata la tesi (Paragrafo 1.3).

1.1 Rilevamento dell'inganno

La bugia è una falsa affermazione per trarre altri in errore, o per nascondere una propria colpa. É un evento comune nelle nostre vite, si stima infatti che una persona, in media, menta due volte al giorno [2]. Una menzogna può essere innocua, ma, in alcuni casi, le conseguenze da essa derivanti possono essere pesanti. Per esempio una bugia può portare una persona innocente a essere condannata in tribunale, o viceversa, un imputato colpevole può essere assolto. La capacità di discriminare un comportamento ingannevole risulta, quindi, cruciale in situazioni in cui la posta in gioco è alta, come per esempio interrogatori della polizia o processi in tribunale. L'abilità di un essere umano di discernere un inganno, senza usare strumenti, è pressoché aleatorio, con una precisione di circa il 54% [4]. Per questa ragione nasce il bisogno di utilizzare sistemi in grado di rilevare un inganno. L'approccio standard usato dalle forze dell'ordine è il poligrafo. Tuttavia, in alcuni casi può essere impraticabile poiché richiede il contatto con la persona e una interpretazione dei dati. L'esito è quindi soggetto a errori e a bias dello strumento e del giudizio umano [5]. Sono stati elaborati dei sistemi che analizzano attività cerebrali in cerca di alterazioni, come ad esempio l'elettroencefalogramma, la risonanza magnetica funzionale e la spettroscopia funzionale nel vicino infrarosso. Questi sistemi ottengono risultati ma hanno delle limitazioni: l'equipaggiamento richiede la supervisione di un esperto e ciò potrebbe rendere troppo costo l'utilizzo quotidiano; il soggetto in analisi deve indossare l'equipaggiamento rendendolo consapevole di essere in esame e quindi potrebbe alterare il suo comportamento. Questi sistemi, negli ultimi anni, si sono evoluti utilizzando tecniche di Machine Learning per esaminare i dati prodotti. Nonostante questa evoluzione, che ha reso i sistemi più robusti, le criticità precedentemente mostrate non sono risolte. Negli ultimi la ricerca in questo campo si è spostata in sistemi basati sulla Computer Vision, ovvero sull'analisi di video. L'utilizzo di questi sistemi permette di individuare reazioni istintive e involontarie dei soggetti. Visto che basta l'installazione di una camera, non serve indossare strumenti specifici, rendendo quindi il soggetto inconsapevole di essere esaminato da sistemi automatici. Precedenti ricerche affermano che esistono differrenze nel modo di comunicare quando si mente e quando si dice la verità [6,7], come ad esempio il modo

in cui si parla, le espressioni facciali o il movimento degli occhi. Queste differenze posso essere individuate da sistemi di Machine Learning. Basati sulle ricerche di Paul Ekman [8–10] riguardo le caratteristiche facciali nel momento in cui si mente, il lavoro di questa tesi propone un sistema di rilevamento dell'inganno basato su video RGB. Il metodo proposto analizza i video in modo da individuare sequenze di espressioni facciali o composizione di esse e l'asimmetria che distinguono un comportamento bugiardo con uno sincero. Per mostrare l'efficacia del sistema proposto, i test sono stati eseguiti sul Real-life Trial dataset, introdotto da Perez-Rosas et al. [35], dove sono contenuti video di processi in tribunale.

1.2 Stato dell'arte

Nella letteratura, sono state proposte numerose soluzioni riguardo il rilevamento dell'inganno. Questo paragrafo riassume i metodi proposti dallo stato dell'arte e ne descrive una tassonomia. Due fasi principali caratterizzano l'attività di deception detection:

1. **Feature extraction:** individuazione ed estrazione delle caratteristiche che permettono la distinzione di un comportamento bugiardo da uno veritiero.
2. **Apprendimento e classificazione:** un modello di classificazione binario è allenato per avere una precisione ottimale sulla base delle informazioni precedentemente estrapolate.

La natura eterogenea delle caratteristiche fisiologiche e psicologiche che caratterizzano l'atto del mentire ne permettono di descrivere una tassonomia dei suoi metodi di individuazione.

1.2.1 Analisi cerebrale

Con questo approccio, il soggetto in analisi è consapevole di essere sotto esame. L'utilizzo di elettrodi e altri strumenti di monitoraggio può essere molto invasivo e costoso. Tuttavia, l'uso di elettroencefalogramma (EEG), risonanza magnetica funzionale (fMRI) e spettroscopia funzionale nel vicino infrarosso (fNIRS) è diffuso. Esaminando la variabilità dell'EEG, con un approccio fuzzy, è possibile identificare bande di frequenza sensibili utili per misurare lo stato mentale nel momento della menzogna con l'analisi spettrale [11]. In [12] i segnali EEG, con una rappresentazione bi-spettrale, vengono classificati attraverso una Support Vector Machine con funzione kernel a base radiale (RBF-SVM). Con la fMRI, è possibile individuare quale regione del cervello si attiva nel momento in cui si mente [13, 14]. Con la fNIRS, l'attività cerebrale viene tracciata analizzando i livelli di ossigeno nel sangue, riuscendo cosi a rilevare l'inganno [15, 16].

1.2.2 Analisi di faccia e occhi

A differenza degli approcci precedentemente descritti, l'analisi attraverso caratteristiche visuali come la faccia e gli occhi è meno invasiva e riduce la consapevolezza del soggetto di essere sotto esame, evitando così di snaturare il proprio comportamento. Determinate espressioni facciali sono involontarie o difficili da controllare, quindi, possono rivelarsi importanti indizi per determinare la verità o la bugia. Proudfoot et al. [25], con la latent growth modeling, si traccia l'evoluzione del diametro della pupilla dei soggetti innocenti e colpevoli durante un concealed information test (CIT).

In Su e Levine [3] i movimenti delle sopracciglia, i movimenti della bocca, le rughe del volto e lo sbattere delle palpebre sono usati nella classificazione implementata con la Random Forest (RF). In Khan et al. [21] vengono combinati i movimenti del viso e degli occhi per distinguere un comportamento sincero da uno menzognero. In Avola et al. [1], la forma dei muscoli facciali, le sue caratteristiche temporali e le traiettorie della faccia sono modellate. La classificazione è poi implementata con una tecnica di ensemble learning. Logistic Regression (LR), SVM, Extreme Gradient Boosting (XGBoost) e Multilayer Perceptron (MLP) agiscono parallelamente come classificatori deboli. I risultati vengono poi concatenati e dati in input in un secondo MLP con la tecnica della stacked generalization [22]. Bhaskaran et al. [24] esamina il cambiamento nei movimenti degli occhi rispetto al comportamento normale applicando un modello bayesiano dinamico.

1.2.3 Analisi verbale

Il modo di parlare di una persona può far trasparire all'esterno ciò che una persona prova veramente nel suo interno. Le caratteristiche linguistiche e vocali possono essere quindi rilevanti al fine di determinare se un individuo stia mentendo o meno. In Zhou et al. [17] vengono analizzate le caratteristiche prosodiche (intonazione, ritmo, durata, etc.) estraendo parametri come l'altezza del suono e Mel-Frequency Cepstrum Coefficient (MFCC) tra gli altri. Viene poi implementata una variante della SVM, la Relevance Vector Machine per la classificazione delle feature. In Tao et al. [18] il zero-crossing rate, pitch rate, short-term energy e MFCC vengono usati per allenare la SVM. In Nasri et al. [19] viene presentato un dataset contenente dichiarazioni sia veriterie che mendaci, registrate in ambienti all'aperto e al chiuso. Da esso vengono poi estratti MFCC e il tono della voce, la SVM con funzione kernel lineare, infine, riesce a individuare bugia e verità con una precisione del 88,23% e 84% rispettivamente. In Mihalcea et al. [20] viene utilizzato un dataset formato da 140 video, caricati da utenti attraverso un ambiente di crowdsourcing. Vengono poi raccolti dal dataset, manualmente e attraverso specifici software di speech recognition, le trascrizioni dei video. Con una rappresentazione bag-of-word delle trascrizioni è stato costituito un vocabolario. La SVM poi raggiunge una precisione del 73,7% nel rilevamento dell'inganno.

1.2.4 Analisi multimodale

Nell'analisi multimodale sono prese in considerazione caratteristiche eterogenee, estratte da sorgenti diverse. Negli ultimi anni questo approccio sta diventando sempre più diffuso, poiché, generalmente, è in grado di fornire prestazioni migliori rispetto ad approcci unimodali. In Rill-Garcia et al. [27] e Krishnamurthy et al. [30] si analizzano le peculiarità visuali, acustiche e la trascrizione delle parole. In Mathur e Matarić [23], oltre alle informazioni precedentemente citate, sono estratte la valenza delle emozioni e lo stato di eccitazione dimostrati con le espressioni facciali. Valenza negativa può indicare rabbia o paura, valenza positiva può significare gioia. La classificazione è stata poi performata da una metodo ensemble, chiamato Adaboost, ottenendo una precisione leggermente migliore escludendo le informazioni testuali. In Perez-Rosas et al. [28] sono estratte feature verbali e non verbali. Le caratteristiche facciali sono estratte sia automaticamente sia con annotazioni manuali, con la codifica MUMIN. La classificazione è poi implementata con la RF, RBF-SVM e una Neural Network (NN) ottenendo risultati migliori con quest'ultima (72,88%). In Lu et al. [29] è analizzata la correlazione tra le espressioni facciali e il linguaggio del corpo. Solitamente, negli approcci multimodali, la Data Fusion delle informazioni

provenienti da modali differenti è effettuata attraverso la concatenazione in un singolo feature vector. Carissimi et al. [31] propone un approccio multiview, dove ogni feature di natura diversa è considerata una view. Ogni vista è modellata da una singola funzione e poi tutte le funzioni sono ottimizzate congiuntamente per migliorare le performance di apprendimento. Uno studio fatto sulle differenze di comportamento delle persone di sesso maschile da quelle di sesso femminile [32]. Sono prese in considerazione feature linguistiche, fisiologiche, come ad esempio la frequenza respiratoria, e termali. In Karimi et al. [26] informazioni visuali e vocali sono classificate con la Large Margin Nearest Neighbor (LMNN), un algoritmo di machine learning che classifica un'istanza con la classe ottenuta dal major vote delle sue k istanze più vicine, il cui calcolo della distanza è ottimizzato durante la fase di training.

1.3 Contributo e Outline

In questa tesi, il rilevamento dell'inganno è implementato analizzando le caratteristiche facciali di un individuo. La principale novità è l'analisi di sequenze di espressioni facciali, invece di analizzare singoli frame chiave. I principali contributi del metodo proposto possono essere riassiunti in:

- L'utilizzo della asimettria facciale, calcolata in un determinato istante;
- L'utilizzo di una particolare Recurrent Neural Network (RNN), chiamata Long Short-Term Memory (LSTM), per analizzare le sequenze di espressioni facciali.

Il resto della tesi è organizzato come segue. Nel Capitolo 2, Feature Extraction, si introduce il concetto di feature e quali sono utilizzate nel riconoscimento di espressioni facciali. Il Capitolo 3, Deep Learning, descrive la struttura delle reti neurali artificiali e come esse apprendono. Il Capitolo 4, Architettura del Sistema, illustra il metodo proposto. Nel Capitolo 5, sono descritti il dataset utilizzato e i risultati ottenuti confrontandoli con il corrente stato dell'arte. Infine, nel Capitolo 6, sono descritte le considerazioni finali e i possibili sviluppi futuri.

Capitolo 2

Feature Extraction

In questo capitolo viene introdotto il concetto di feature e perché è importante nella classificazione (Paragrafo 2.1). Viene poi descritto il processo di individuazione di viso e punti di interesse facciali (Paragrafi 2.2 e 2.3). Sono poi illustrati principali metodi di lavorazione delle immagini (Paragrafi 2.4 e 2.5) per ottimizzare la fase di estrazione delle feature (Paragrafo 2.6). Infine, un riepilogo del capitolo è descritto (Paragrafo 2.7).

2.1 Feature

Informazioni non rilevanti, o ridondanti, possono compromettere le prestazioni di un sistema di rilevamento dell'inganno automatico. Per questo la fase di estrazione delle caratteristiche, chiamate feature, diventa fondamentale. In questa fase, si eliminano dati che fanno rumore ottendendo un insieme di dati di potenziali feature discriminanti. Le feature da estrarre si possono dividere in due categorie:

- **Feature globali**: caratteristiche che permettono di analizzare una immagine, come ad esempio la distribuzione dei colori. Tali caratteristiche permettono di descrivere la faccia nella sua interezza;
- **Feature locali**: caratteristiche che permettono di analizzare punti specifici di una immagine, come ad esempio, il rilevamento di punti di interesse. Nell'esempio di analisi di caratteristiche facciali, esse permettono di analizzare diverse regione facciali.

In questa fase possono essere combinate sia feature locali che globali. Una solo feature difficilmente riesce a discriminare una bugia da una verità, per la classificazione si utilizza, quindi, un vettore delle feature. Per quanto concerne il rilevamento dell'inganno tramite espressioni facciali, sono prese rilevanti le ricerche di Ekman. Ekman e Friesen introdussero il Facial Action Coding System (FACS), un sistema che analizza le espressioni facciali scomponendole nelle più piccole unità d'azione, chiamate Action Unit (AU) (Figura 2.1).

2.2 Individuazione della faccia

Per analizzare le espressioni facciali di una persona bisogna prima localizzarne il viso. Uno degli algoritmi più utilizzati è quello introdotto da Viola - Jones [59] basato sul Machine Learning, applicabile nel generico campo dell'Object Detection. Ogni faccia umana ha proprietà comuni che possono essere descritte dalle Haar

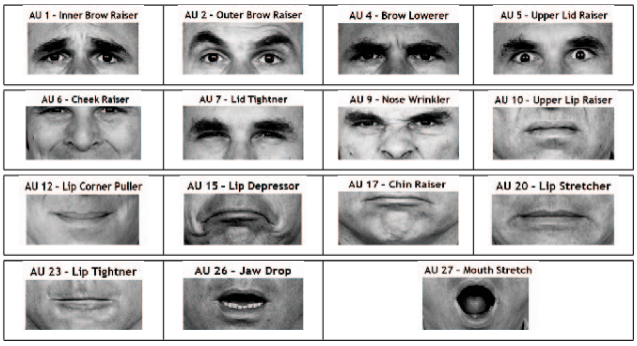

Figura 2.1. Alcuni esempi di Action Unit [36].

features. Il nome deriva dalla somiglianza con la Haar wavelet (Figura 2.2). La

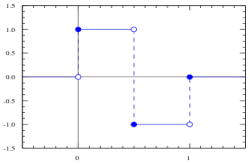

Figura 2.2. La wavelet Haar Fonte: Wikipedia.

regione degli occhi è più scura rispetto a quella delle guance, il ponte del naso è più luminoso rispetto agli occhi. La composizione di queste proprietà permette di costituire delle feature abbinabili. L'algoritmo riesce ad abbinare in base alla differenza di luminosità tra i rettangoli neri e bianchi applica su una specifica area dell'immagine (Figura 2.3). L'immagine viene analizzata e l'algoritmo individua se

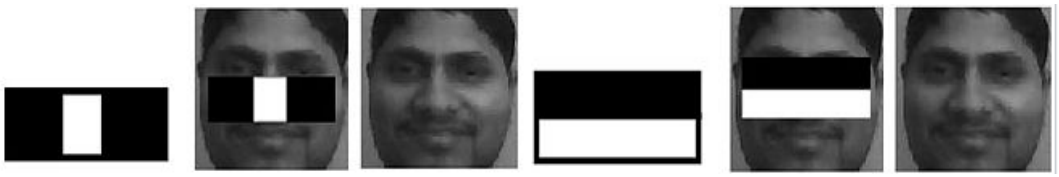

Figura 2.3. A sinistra viene applicata al viso una Haar feature simile al ponte del naso, a destra è applicata una Haar feature simile alla regione degli occhi. Fonte: Wikipedia

in ogni area è presente o meno una faccia. La Figura 2.4 mostra la localizzazione delle facce in una foto. Nell'algoritmo è presente una funzione a cascata allenata da immagini positive (con facce presenti) e immagini negative (senza volti presenti), le Haar features sono applicate alle immagini per individuare la migliore soglia che classifica il rilevamento del viso come positivo o negativo. Le Haar features sono selezionate utilizzando Adaboost.

2.3 Punti di interesse facciali

Per analizzare più dettagliatamente determinate regioni facciali si individuando i punti di interesse. L'algoritmo adottato in questa tesi è il Convolutional Experts Constrained Local Model (CE-CLM) [40]. Un Constrained Local Model (CLM) è

una classe di metodi per individuare un insieme di punti, vincolato da un modello statistico, in una immagine. Una regione dell'immagine vine campionata alla stima corrente, proiettandola nel frame di riferimento. Per ogni punto, viene generata una immagine di risposta dando un costo per avere quel determinato punto ad ogni pixel. Si ricerca poi la combinazione di punti che ottimizza il costo totale, modificando i parametri del modello. Il CLM è usato per modellare la presenza di ogni punto di interesse individualmente usando rilevatori locali e un modello di forma per una ottimizzazione vincolata. La CE-CLM si divide in due parti fondamentali: calcolo della response map usando il Convolutional Experts Network (CEN), e aggiornare i parametri di forma. Nel primo passaggio, i singoli punti facciali di interesse sono individuati valutando la probabilità di localizzazione ad ogni pixel. Formalmente:

$$p^* = \arg\min_p \left[\sum_{i=1}^{n} -D_i(x_i, I) + R(p) \right] \tag{2.1}$$

dove p^* è l'insieme di parametri che controllano la posizione del punto di interesse; p è la stima corrente; D_i è la probabilità di allineamento del punto i nella posizione x_i dell'immagine I e $R(p)$ è il peso aggiornato da un Point Distribution Model (PDM). Questo calcolo è eseguito dal CEN che prende in input una Regione di Interesse (ROI) lungo la stima corrente del punto, e restituisce come output una response map. Successivamente, ogni punto di interesse è controllato nuovamente dalla PDM. Le forme irregolari sono penalizzate dal parametro $R(p)$. Un esempio visivo dei punti di interesse facciali è mostrato nella figura 2.4.

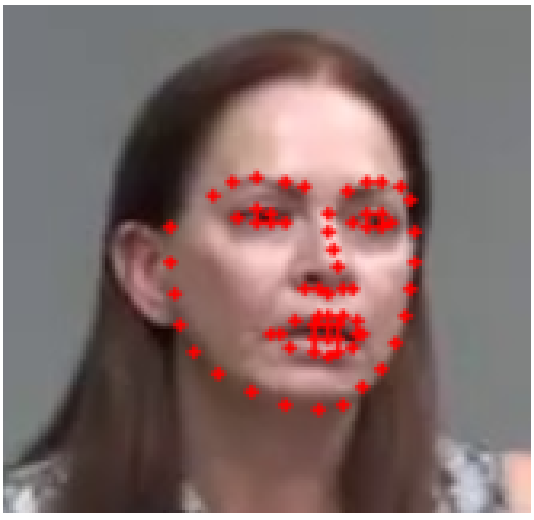

Figura 2.4. Esempio di punti di interesse facciali tracciati con l'algoritmo CE-CLM dal Real-life Trial Dataset.

2.4 Trasformazioni geometriche

Anche se la faccia è stata individuata non è detto che sia nelle condizioni adatte per essere analizzata. Alcuni problemi che si possono affrontare sono la rotazione dell'immagine, la scala o il rumore. In alcuni lavori sono state applicate delle trasformazioni geometriche alle facce che sono state rilevate in condizioni non ottimali. Per quanto concerne il problema della rotazione, una delle soluzioni più

popolari è l'utilizzo dei punti di interesse facciali, che si vedranno più in dettaglio nel Paragrafo 2.5. Si considerano due punti di interesse facciali che formano un angolo di zero gradi lungo l'asse orizzontale quando la faccia è allineata. Quindi, si ruota l'immagine finché l'angolo formato da questi due punti non sia zero. La Figura 2.5 mostra la correzione della rotazione. Avere immagini in scala diversa

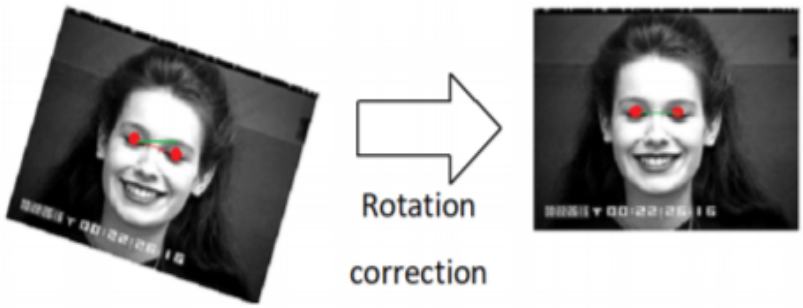

Figura 2.5. Esempio di correzione della rotazione in una faccia presa dal CK+ Dataset [33].

crea problemi perché le distanze per individuare il volto sono differenti. Le ROI di una faccia rilevata più vicina rispetto ad un'altra sono più grandi. Una tecnica per ovviare a questo problema è normalizzare lo spazio portando le ROI ad una grandezza predefinita. Nell'area dell'immagine dove viene individuata una faccia è presente anche lo sfondo e questo crea rumore. Uno degli approcci più utilizzati e quello di usare i punti di interesse per individuare il perimetro della faccia e ritagliare l'immagine escludendo lo sfondo.

2.5 Image processing

La fase di trasformazione geometrica dell'immagine potrebbe non essere sufficiente per avere dati da dare al classificatore. Sono state ideate molto tecniche di image processing per accentuare determinate feature. Facendo lo smoothing di una immagine, è possibile rilevare dei pattern mentre si filtra il rumore. I metodi più popolari di lisciamento delle immagini sono il filtro bilaterale [41] o il Gaussian Filter [42] Un filtro bilaterale è efficace nella rimozione del rumore mantenendo i bordi nitidi, poiché utilizza una funzione gaussiana dello spazio per livellare solo i pixel vicini e una funzione gaussiana di intensità per levigare i pixel che hanno un'intensità simile al pixel centrale. In questo modo, il filtro può preservare i bordi, poiché di solito hanno variazioni di alta intensità. Il Gaussian Filter analizza i pixel che circondano un dato pixel centrale e individua il peso gaussiano medio. Questo filtro smusserà anche i bordi. Gli istogrammi mostrano la distribuzione di intensità di una immagine. La possibile differenza di intensità che le facce estratte possono presentare, potrebbe ridurre la precisione. Ci sono algoritmi basati su istogrammi sono in grado di estendere questa distribuzione, il che significa che le regioni del viso sovraesposte o sottoesposte avranno la loro intensità uniforme. Questo metodo migliora il contrasto dell'immagine, evidenziando le caratteristiche della faccia, riducendo l'interferenza causata da differenti condizioni di luminosità. Uno dei metodi, basato su questo approccio, più utilizzato è la Histogram equalization [43] (Figura 2.6).

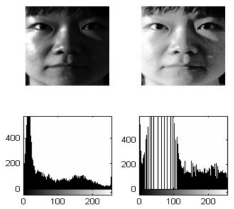

Figura 2.6. A sinistra l'immagine originale, a destra l'immagine dopo la histogram equalization. [44].

2.6 Feature Extraction

Una volta terminata la fase di pre-lavorazione delle immagini bisogna estrarre le feature più rilevanti per poter avere una classificazione ottimale. Il Local Binary Pattern (LBP) [45] è un descrittore di immagini che segue un pattern a griglia che può essere usato per modellare la forma dei muscoli facciali. Il processo di costruzione del vettore delle feature è il seguente. L'immagine viene divisa in celle (e.g. celle di 3x3 pixel), per ogni pixel nella cella vengono esaminati gli 8 pixel che lo circondano. Quando il centro del pixel è maggiore rispetto al pixel esaminato si scrive 0 nella posizione, 1 altrimenti. Questa operazione permette di fornire una codifica binaria a 8 bit convertita in un numero decimale. Successivamente, si calcola l'istogramma che rappresenta la distribuzione dell'occorrenza di ogni numero. Infine, si concatenano gli istogrammi di ogni cella ottenendo il vettore delle feature dell'immagine. L'Optical Flow è un pattern che modella il movimento di

Figura 2.7. Esempio di applicazione del LBP su un viso. Fonte: advancesourcecode

un oggetto tra due frame consecutivi. L'algoritmo di optical flow lavora assumendo che l'intensità di pixel dell'oggetto in esame non cambi in due frame consecutivi e che pixel vicini hanno un moto simile.Un optical flow è un'approssimazione del cosiddetto campo di moto, cioè la proiezione del vettore velocità di un punto nello spazio 3D sul piano immagine 2D. Una variante dell'optical flow è il Dense Optical Flow [46] che calcola il flusso per ogni punto del frame. Questo metodo può essere efficacemente utilizzato nelle analisi delle espressioni facciali, poiché quando si passa da una espressione neutra ad una massima espressione facciale c'è un movimento che può essere stimato (Figura 2.8). I parametri di animazione facciale (FAPs) [47] rappresentano 66 spostamenti e rotazioni di determinati punti da una espressione neutrale. I FAPs si basano sui movimenti facciali determinati dalle azione dei muscoli. Permettono la rappresentazione di azioni facciali basiche. Un esempio è illustrato nella Figura 2.9. Il Gabor filter [48] è usato per rappresentare le informazioni

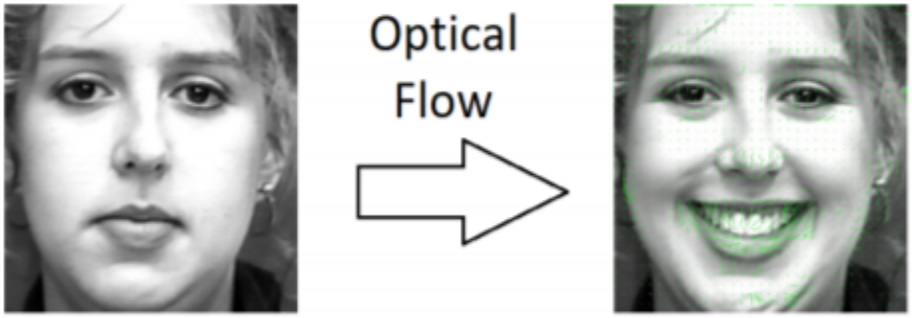

Figura 2.8. Esempio di applicazione dell'optical flow su una sequenza [33].

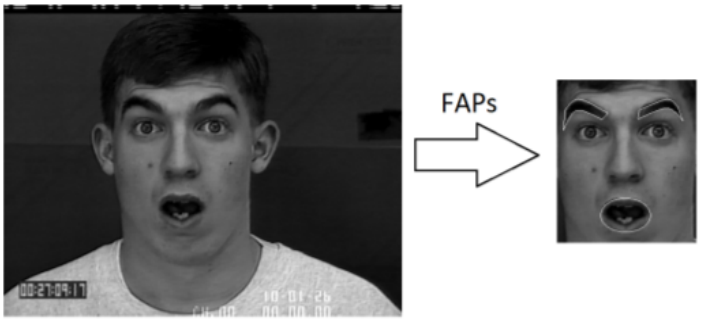

Figura 2.9. Esempio di estrazione dei FAPs riguardanti bocca e sopracciglia [33].

riguardo le texture essendo robusto in condizioni di illuminazione difficili. È in grado di catturare informazioni spaziali di frequenza, posizione e orientamento da un'immagine ed estrae sottili trasformazioni locali in modo efficace. Tuttavia, gli alti costi computazionali escludono l'utilizzo del filtro per applicazioni real-time. Nella Figura 2.10 un esempio di applicazione del Gabor filter su un viso.

Figura 2.10. A sinistra l'immagine originale a destra l'applicazione del Gabor filter. Fonte: programmersought

2.7 Conclusioni

Nel capitolo sono state riassunte le principali attività e criticità che si incontrano quando si analizzano le caratteristiche facciali di una persona. Una volta individuato un volto, occorre omologare e rendere ottimali le caratteristiche delle immagini in analisi (e.g. allineamento scala del viso) per migliorare la precisione del classificatore. Successivamente, sono riassunte le principali feature per rappresentare al meglio le espressioni facciali. Le feature utilizzate in questa tesi verranno viste nel dettaglio quando verrà presentata l'architettura del metodo proposto, nel Capitolo 4.

Capitolo 3

Deep Learning

In questo capitolo sarà introdotto il Deep Learning (Paragrafo 3.1) descrivendo la struttura delle reti neurali (Paragrafo 3.2). Sono poi presentati i paradigmi di apprendimento delle reti (Paragrafo 3.3) e le architetture delle reti neurali ricorrenti più utilizzate (Paragrafo 3.4). Infine un riassunto di quanto descritto nel capitolo è presentato (Paragrafo 3.5).

3.1 Introduzione

Il Machine Learning è quel campo di studi che dà ai computer l'abilità di imparare senza essere programmato esplicitamente (Arthur Samuel, 1959), generalizzando dalla propria esperienza [50]. Per generalizzazione si intende l'abilità di una macchina di portare a termine in maniera accurata esempi o compiti nuovi, che non ha mai affrontato, dopo aver fatto esperienza su un insieme di dati di apprendimento. Il Deep Learning, campo di ricerca del Machine Learning, è un insieme di tecniche basate su reti neurali artificiali organizzate in diversi strati (Figura 3.1), dove ogni strato calcola i valori per quello successivo affinché l'informazione venga elaborata in maniera sempre più completa. Queste architetture sono applicate e si sono rivelate efficaci in numerosi campi tra cui Computer Vision, Natural Language Processing, Speech Recognition e Bioinformatica. Le reti neurali artificiali sono ispirate dall'elaborazione delle informazioni e dai nodi di comunicazione distribuiti nei sistemi biologici.

3.2 Rete neurale artificiale

La rete neurale artificiale è un modello computazionale composto di "neuroni" artificiali, ispirato dalla semplificazione di una rete neurale biologica. La rete eredita dal cervello umano, il modello di apprendimento basato sul funzionamento parallelo e non lineare di molte unità simili tra loro, dove nel caso del cervello umano le unità rappresentano i neuroni.

3.2.1 Neurone artificiale

Un neurone, rappresentato nella figura 3.2, è una cella cerebrale la cui funzione principale è raccogliere, elaborare e propagare segnali elettrici. Si ritiene che la capacità del cervello di processare informazioni derivi principalmente da reti di neuroni. Ogni neurone consiste in un corpo cellulare, o soma, che contiene il nucleo. Dal corpo si dirama una quantità di fibre chiamate dendriti e una singola, lunga

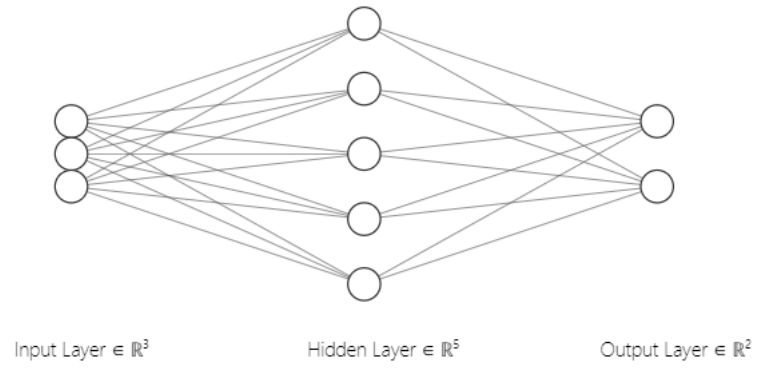

Figura 3.1. Esempio generico di rete neurale. Immagine generata con NN-SVG.

fibra che prende il nome di assone. L'assone si prolunga per una grande distanza. Un neurone si collega collega con un numero che va da 10 a 10.000 altri neuroni, utilizzando punti di congiunzione chiamati sinapsi. I segnali si propagano grazie a una complicata reazione elettrochimica e controllano l'attività cerebrale nel breve periodo, ma permettono anche dei cambiamenti nel lungo termine nella posizione e nella connessione tra neuroni. Si ritiene che questo meccanismo formi la base dell'apprendimento.

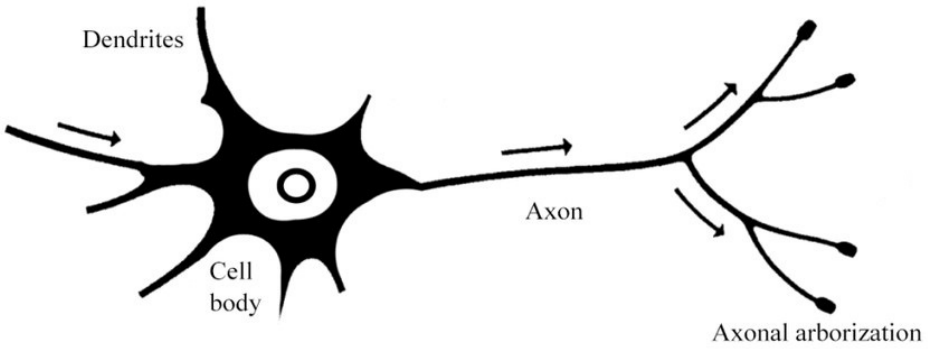

Figura 3.2. Struttura di un neurone biologico

La Figura 3.3 mostra il modello matematico del neurone inventato da McCulloch e Pitts (1943), comunemente chiamato percettrone elementare. Il neurone si attiva quando una combinazione lineare dei suoi input supera una certa soglia. Le reti neurali sono composti da unità unite da collegamenti diretti. Un collegamento dall'unità j alla unità i serve a propagare l'attivazione a_j da j a i. A ogni collegamento è anche associato un peso numerico $W_{j,i}$, che determina la forza e il segno della

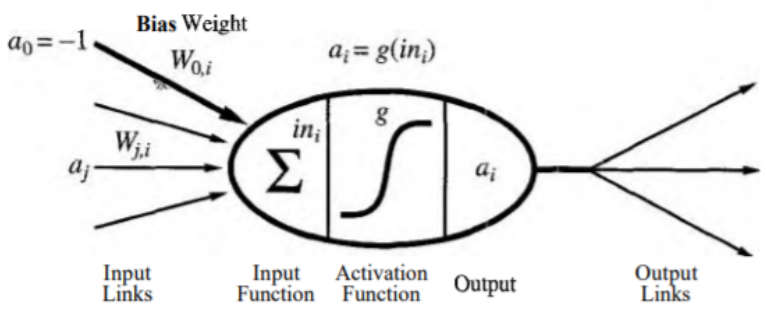

Figura 3.3. Modello matematico di un neurone. Fonte: Stuart Russel e Peter Norvig [39].

connessione. Ogni unità i calcola una somma pesata dei propri input:

$$in_i = \sum_{j=0}^{n} W_{j,i} \cdot a_j \tag{3.1}$$

Si applica poi una funzione di attivazione g alla somma per derivare l'output:

$$a_i = g(in_i) \tag{3.2}$$

3.2.2 Funzione di attivazione

La funzione di attivazione è progettata per attivare l'unità quando sono forniti i dati "corretti", e viceversa, mantenerla inattiva quando gli input sono "sbagliati". Il peso di bias $W_{0,i}$ determina la soglia effettiva dell'unità, nel senso che l'unità si attiva quando la somma pesata degli input (3.1) supera $W_{0,i}$. Il percettrone elementare, con funzione di attivazione a gradino di Heaviside (Figura 3.4), è un classificatore in grado di risolvere problemi con dati linearmente separabali. La funzione di Heaviside è così definita:

$$g(in_i) = \begin{cases} 1, & \text{se } in_i \geq 0. \\ 0, & \text{altrimenti.} \end{cases} \tag{3.3}$$

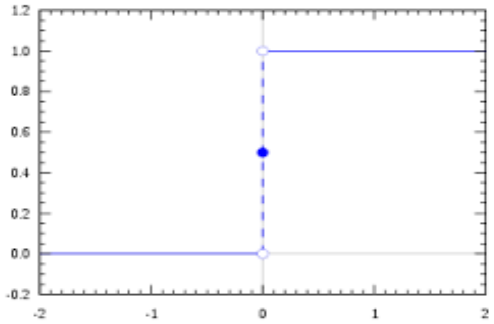

Figura 3.4. Funzione a gradino di Heaviside. Fonte: Wikipedia

Per problemi con dati non linearmente separabili, occorre utilizzare funzioni di attivazione differenziabili. Storicamente, le più utilizzate sono la funzione sigmoidea

(Figura 3.5) e la tangente iperbolica (Figura 3.6).

$$g(in_i) = \frac{1}{1 + e^{-in_i}} \tag{3.4}$$

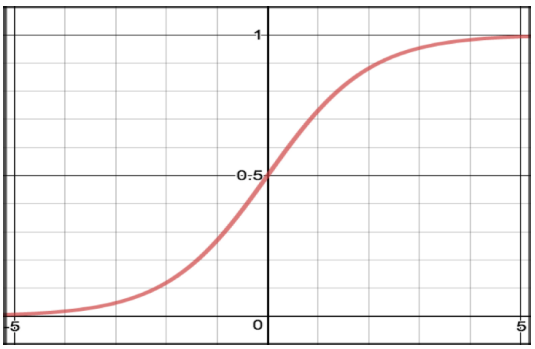

Figura 3.5. Funzione sigmoidea. Fonte: ichi.pro

$$g(in_i) = \frac{e^{2in_i} - 1}{e^{2in_i} + 1} \tag{3.5}$$

La funzione più utilizzata attualmente è la funzione a rampa (Figura 3.7) così

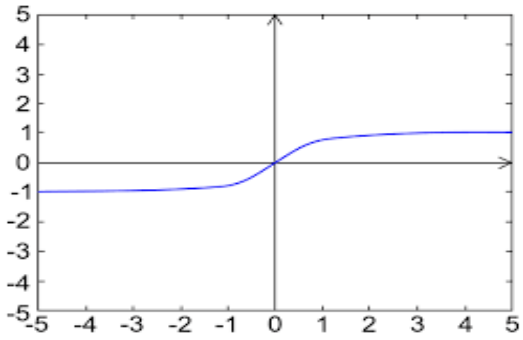

Figura 3.6. Funzione tangente iperbolica. Fonte: YouMath

definita:

$$g(in_i) = max(0, in_i) \tag{3.6}$$

I neuroni che fanno uso di una funzione a rampa, denominati Rectified Linear Unit o ReLU, sono divenuti popolari poiché, il tempo di apprendimento della rete diminuisce e le funzioni di ottimizzazione basate su gradiente vengono accelerate.

3.2.3 Multi-Layer Perceptron

Una rete in cui tutti gli input sono collegati direttamente agli output è chiamata rete neurale a strato singolo, o rete di percettroni. Il Multi-Layer Perceptron (MLP) [51] è un tipo di rete neurale alimentata in avanti (feed-forward) dove ogni nodo di un livello è connesso a tutti i nodi del livello successivo (Figura 3.8). Le informazioni si muovono in una sola direzione, dai nodi input verso quelli di output. I pesi delle connessioni vengono aggiornati nella fase di training, attraverso

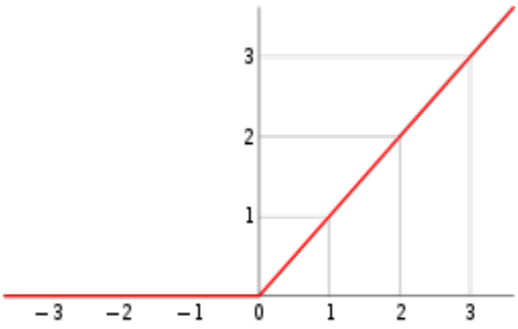

Figura 3.7. Funzione a rampa. Fonte: Wikipedia

l'algoritmo della retropropagazione dell'errore [52]. Con questa tecnica, una funzione di ottimizzazione è utilizzata per aggiornare i pesi con lo scopo di minimizzare l'errore quadratico medio (MSE) tra il valore dell'output della rete con il valore dell'output reale. L'errore quadratico medio è definito dalla seguente formula:

$$E = \sum_{t=1}^{T} \sum_{j=1}^{n} \frac{1}{2} (o_{t,j} - y_{t,j})^2 \tag{3.7}$$

dove E è il valore da ottimizzare; T è un insieme non vuoto di campioni di training; j indica la j-sima unità di output; $o_{t,j}$ e $y_{t,j}$ indicano rispettivamente l'output predetto dalla rete e il valore atteso.

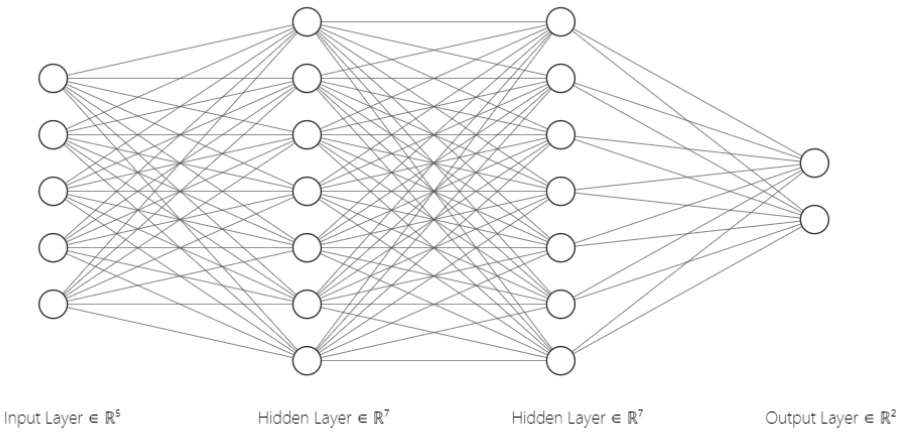

Figura 3.8. Esempio di rete neurale multistrato (MLP). In particolare ha 5 nodi di input, 7 nodi in entrambi i livelli nascosti e 2 unità di output. Rete generata con NN-SVG.

3.3 Algoritmi di apprendimento

Per apprendimento si intende il modo in cui si addestrano i neuroni affinché la rete apprenda. Gli algoritmi di apprendimento modificano i pesi delle connessioni che compongono la rete a seconda del paradigma di apprendimento scelto. I paradigmi di apprendimento si suddividono in:

- *apprendimento supervisionato*: un training set composto da coppie X_t e Y_t, rispettivamente t-esimo valore di input e t-esimo valore di output atteso, sono forniti alla rete per addestrarla. Nella fase di training si confronta il valore predetto dalla rete con il valore atteso, i pesi delle connessioni vengono aggiornati per ridurre l'errore;
- *apprendimento non supervisionato*: sono forniti solo i valori di ingresso. L'obiettivo è di raggruppare i dati secondo caratteristiche comuni, mediante metodi topologici o stocastici, individuando cluster che rappresentino i dati. Questo paradigma è ottimo quando si ha una mole di dati che non è possibile da etichettare manualmente;
- *apprendimento per rinforzo*: punta a realizzare agenti autonomi in grado di scegliere azioni da compiere per il conseguimento di determinati obiettivi tramite interazione con l'ambiente in cui sono immersi.

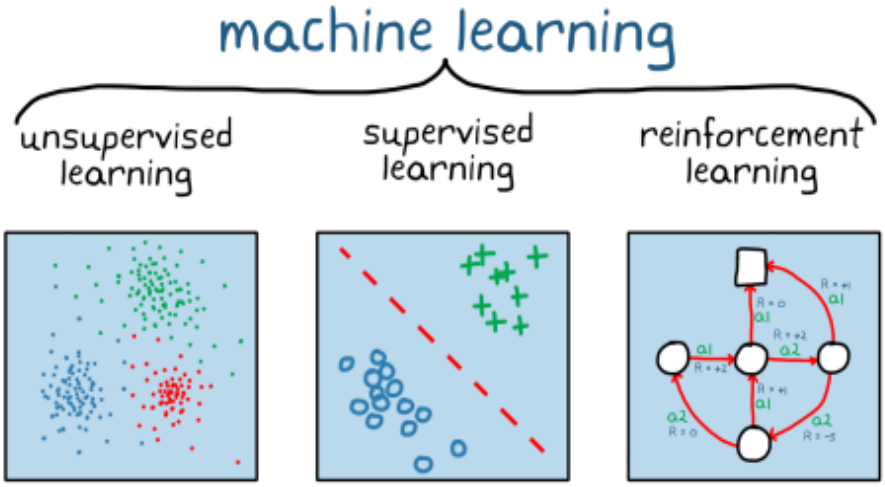

Figura 3.9. Esempi di apprendimento non supervisionato, supervisionato e per rinforzo.

Un'epoca è un ciclo di apprendimento della rete effettuato sull'intero training set. Solitamente, per migliorare le prestazioni, gli elementi del training set sono raggruppati in batch. Il primo obiettivo, nella fase di addestramento, è la convergenza dei dati cercando di massimizzare la precisione delle predizioni e quindi minimizzare l'errore. Il numero di epoche che servono a raggiungere questa convergenza è dipendente dal training set. Lo scopo principale è però la generalizzazione della predizione, cercando di mantenere un'accuratezza alta sui dati mai visti dalla rete.

3.3.1 Regola delta

La regola delta è un algoritmo di apprendimento supervisionato, utilizzato dal percettrone elementare per risolvere problemi con dati linearmente separabili, con lo scopo di minimizzare l'errore di predizione tramite una procedura di discesa del gradiente. Si definisce E l'errore quadratico medio tra il valore di uscita predetto e il valore di uscita atteso (Formula 3.7). Occorre poi minimizzare la funzione obiettivo per far sì che la rete rispondi correttamente ai dati in input. Scelto un punto casuale nello spazio multidimensionale, scendere lungo il gradiente significa muoversi verso

il minimo locale più vicino. Il gradiente di una funzione, mostra la direzione di massima crescita, o decrescita, di quest'ultima a partire da un certo punto. In questo caso, il gradiente corrisponde al vettore che ha per elementi le derivate parziali della funzione E rispetto ai pesi delle connessioni della rete:

$$\nabla E(w) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_n}] \tag{3.8}$$

la regola di aggiornamento dei pesi è quindi:

$$\Delta w_i = -\alpha \frac{\partial E}{\partial w_i} \tag{3.9}$$

dove α è una costante chiamata learning rate, che definisce la velocità di apprendimento del percettrone. Infine la formula completa di aggiornamento del peso è:

$$w_{i,t} = w_{i,t-1} + \Delta w_i \tag{3.10}$$

3.3.2 Retropropagazione dell'errore

Per quanto riguarda il MLP, l'algoritmo di apprendimento più utilizzato è la retropropagazione dell'errore, che estende la regola delta a reti composte da più layer e con funzioni di attivazione non lineari. Lo scopo resta quello di minimizzare la funzione obiettivo E definita dall'errore quadratico medio. Attraverso il confronto tra i valori di uscita desiderati presenti nel training set e quelli predetti dalla rete possiamo conoscere l'errore dei neuroni dello strato di output. Non abbiamo però alcuna indicazione su quali debbano essere i valori di uscita desiderati dei neuroni nascosti. Si effettua quindi la retropropagazione dell'errore dal livello di output ai livelli nascosti. Poiché l'aggiornamento di un peso varia a seconda se il neurone fa parte di uno strato di output o uno nascosto, si utilizza la regola delta generalizzata:

$$\Delta w_{j,k} = -\alpha \delta_j o_k \tag{3.11}$$

dove $\Delta w_{j,k}$ è la variazione del peso della connessione tra i neuroni k e j; α è il learning rate; o_k è l'output del neurone k corrispondente all'input del neurone e δ_j è l'errore generato dal neurone j. Per un neurone sullo strato di uscita l'errore δ_j è definito come:

$$\delta_j = y_j(1 - y_j)(d_j - y_j) \tag{3.12}$$

dove y_j è il valore predetto e d_j il valore desiderato. Per un neurone in uno strato nascosto, invece, δ_j è definito come:

$$\delta_j = y_j(1 - y_j) \sum_k \delta_k w_{k,j} \tag{3.13}$$

dove $\sum_k \delta_k w_{k,j}$è la somma pesata di tutti gli errori generati dai nodi a cui il nodo j è connesso. Infine, la formula di aggiornamento pesi è:

$$w_{jk,t} = w_{jk,t-1} + \Delta w_{jk} \tag{3.14}$$

3.3.3 Overfitting e regolarizzazione

L'overfitting [53], ovvero l'eccessivo adattamento al training set, è un problema nell'addestramento delle reti neurali, poiché, quest'ultima, avrà difficoltà a classificare dati mai visti. Sono state sviluppate alcune tecniche che, operando sulla funzione costo o sull'architettura della rete, aiutano a ridurre gli effetti del sovraallenamento di una rete neurale. Tali tecniche prendono il nome di regolarizzazione. Una di queste, chiamata Dropout [54], toglie una certa percentuale di neuroni in ogni hidden layer in modo casuale (Figura 3.10). La rete risultante dal processo di Dropout corrisponde ad una media tra molteplici reti diverse con la capacità di sovra-adattarsi in modi differenti. Un percettrone viene costretto ad apprendere caratteristiche più robuste, utili se unite a diversi sottoinsiemi casuali degli altri percettroni, poiché non può basarsi sulla presenza di un altro specifico neurone, e questo, si traduce in una migliore generalizzazione della rete finale. Un'altra delle tecniche più utilizzate è chiamata regolarizzazione L2 o decadimento dei pesi [55]: utilizza la somma dei quadrati dei pesi in addizione alla funzione obiettivo E_0:

$$E = E_0 + \frac{\lambda}{2n} \sum_i w_i^2 \tag{3.15}$$

dove $\lambda > 0$ è il parametro di regolarizzazione e n è la cardinalità del training set. Così si apprendono dei valori bassi per i pesi delle connessioni della rete permettendo l'apprendimento di valori più alti solo se migliorano la funzione costo originale. Analoga alla L2, è la regolarizzazione L1 dove a E_0 è aggiunta la somma del valore assoluto dei pesi:

$$E = E_0 + \frac{\lambda}{n} \sum_i |w_i| \tag{3.16}$$

Tale equazione, analogamente alla regolarizzazione L2, fa in modo che la rete apprenda pesi abbastanza bassi. La differenza tra le due tecniche di regolarizzazione consiste nel modo in cui il valore dei pesi viene ridotto. Nella regolarizzazione L1 i pesi sono ridotti di una quantità costante tendente a zero, mentre, nella regolarizzazione L2 sono ridotti di una quantità proporzionale a w.

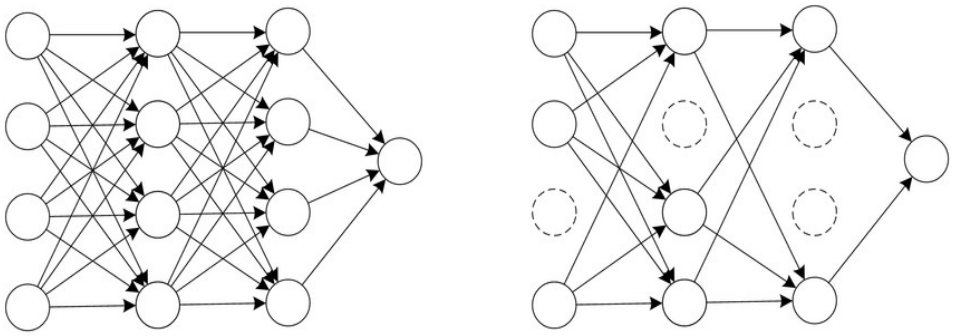

Figura 3.10. A sinistra un esempio di MLP, a destra la rappresentazione della MLP applicandogli la tecnica del Dropout.

3.4 Reti neurali ricorrenti

Una rete neurale ricorrente (RNN) [56] è una classe di rete neurale artificiale che include neuroni collegati tra loro in un ciclo (Figura 3.11). La RNN può essere

vista come t copie di una stessa rete che si ripetono nel tempo e tra le quali i pesi vengono condivisi. Come mostrato, l'unità A analizza, in un generico tempo t, il rispettivo input x_t e genera l'output h_t. Il ciclo permette di passare le informazioni elaborate nel tempo t al tempo t+1. Questa architettura permette di simulare la memoria a breve termine del cervello umano, poiché la presenza di cicli permette la gestione di sequenze temporali di dati. Tuttavia, la RNN non sembra essere in grado di correlare dati di una sequenza quando il gap temporale è troppo alto, poiché, in lunghe sequenze temporali, il gradiente della funzione errore decade o cresce esponenzialmente rispetto al tempo causando un problema noto come vanishing/explosion gradient.

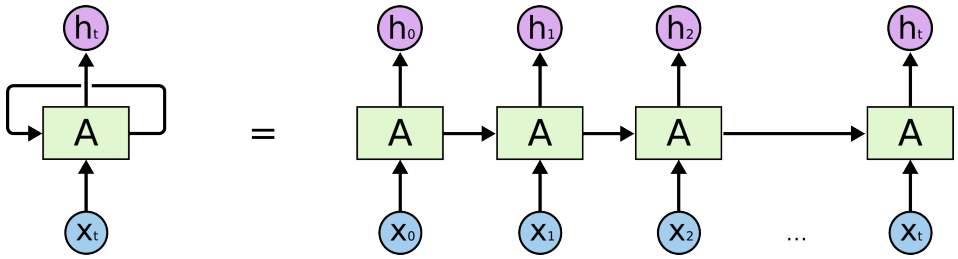

Figura 3.11. A sinistra l'architettura di una RNN, a destra la sua corrispondente rappresentazione non ciclica. Fonte: qastack.it

Formalmente, una RNN riceve una sequenza $x_1, x_2, ..., x_t \in R^n$ e calcola l'output $h_1, h_2, ..., h_t \in R^m$ per ogni elemento della sequenza, da cui poi vengono fatte le predizioni $y_1, y_2, ..., y_t \in R^k$ iterando le seguenti equazioni t volte:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{3.17}$$

$$y_t = softmax(W_{hy}h_t + b_y) \tag{3.18}$$

dove h_t è lo stato dello strato nascosto al tempo t; σ rappresenta la funzione sigmoidea; W_{hh} è la matrice dei pesi dello strato nascosto in istanti consecutivi; W_{xh} è la matrice dei pesi tra lo strato di input e lo strato nascosto; W_{hy} è la matrice dei pesi tra lo strato nascosto e lo strato di output e i vettori b_h e b_y sono i pesi di bias. L'algoritmo di apprendimento utilizzato nelle RNN è un'estensione della retropropagazione dell'errore, aggiungendo la componente temporale. L'errore deve essere calcolato e accumulato per ogni istante di tempo. Considerando la fase di apprendimento della rete dal tempo t_0 a t_n, la funzione obiettivo si calcola come segue:

$$E_{tot}(t_0, t_n) = \sum_{i=0}^{n} E(t_i) \tag{3.19}$$

e l'aggiornamento dei pesi ad ogni istante t è così definita:

$$\Delta w_{j,k} = -\alpha \frac{\partial E_{tot}(t_0, t_n)}{\partial w_{j,k}} \tag{3.20}$$

Per evitare il problema del decadimento del gradiente è stata ideata la retropropagazione dell'errore nel tempo troncato, limitando la propagazione dell'errore in un determinato lasso di tempo.

3.4.1 LSTM: Long Short-Term Memory

La Long Short-Term Memory (LSTM) [57] è un'architettura nata con lo scopo di risolvere le dipendenze temporali a lungo termine. Una rete LSTM prevede delle unità di memoria (Figura 3.12) per il salvataggio di informazioni riguardanti lunghe sequenze temporali al posto dei percettroni. L'unità di memoria è in grado

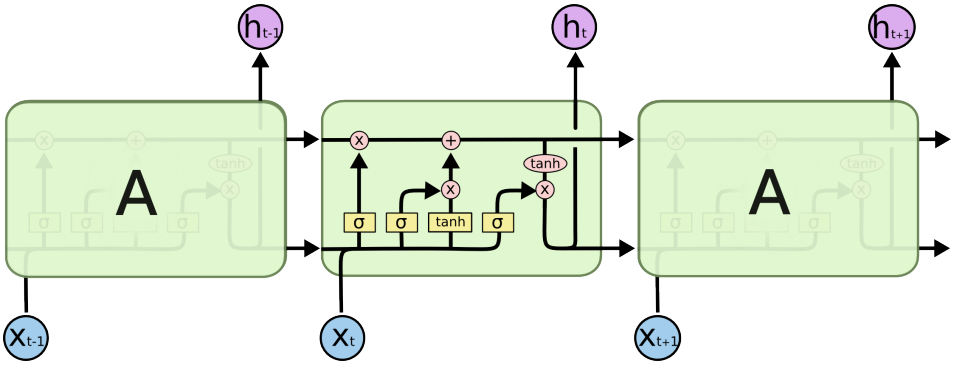

Figura 3.12. Architettura di una LSTM. Le frecce che convergono indicano una concatenazione di vettori, frecce che divergono indicano una copia. Fonte: colah's blog

di trasmettere l'informazione all'unità successiva tramite operazioni lineari (Figura 3.13). La LSTM ha la capacità di stabilire quali informazioni memorizzare e quali dimenticare nella propria unità di memoria grazie alla presenza di cancelli, chiamati gate. Questi gate sono composti da un livello sigmoidale e da un operazione di moltiplicazione. Applicando la sigmoide, si hanno valori tra zero e uno e indica quanto una informazione debba proseguire, zero significa che nessuna informazione debba passare, al contrario uno indica che essa prosegui per intero.

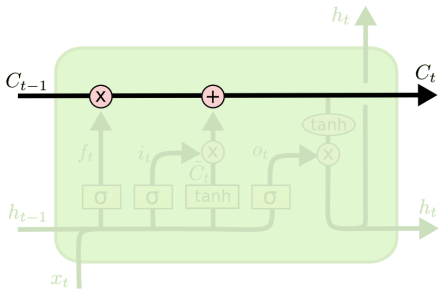

Figura 3.13. Percorso dello stato dell'unità di memoria. Fonte: colah's blog

I gate presenti nella cella di memoria sono 3:

- *Forget gate*: Questo gate decide quali informazioni mantenere nella cella. La sigmoide viene applicata al t-simo input della sequenza x_t e al precedente strato nascosto h_{t-1}. La forget gate è rappresentatata nella Figura 3.14.
- *Input gate*: Questo gate decide quali informazioni memorizzare nell'unità di memoria. Il gate è formato da due livelli: il primo è una funzione sigmoide per determinare come aggiornare le informazioni, il secondo è una funzione *tanh* per creare il vettore con i valori candidati ad aggiornare lo stato della

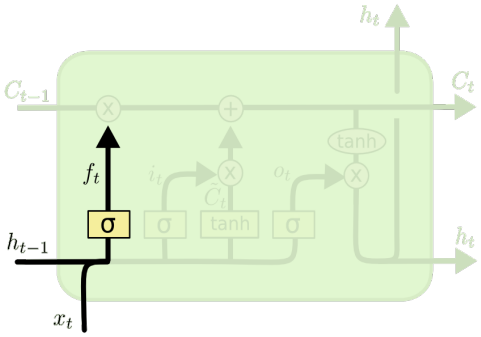

Figura 3.14. Forget gate. Fonte: colah's blog

cella. Entrambi i layer prendono in input x_t e h_{t-1}. Gli output dei due livelli sono combinati con una moltiplicazione e sommati poi allo stato della cella precedente c_{t-1} (Figura 3.15).

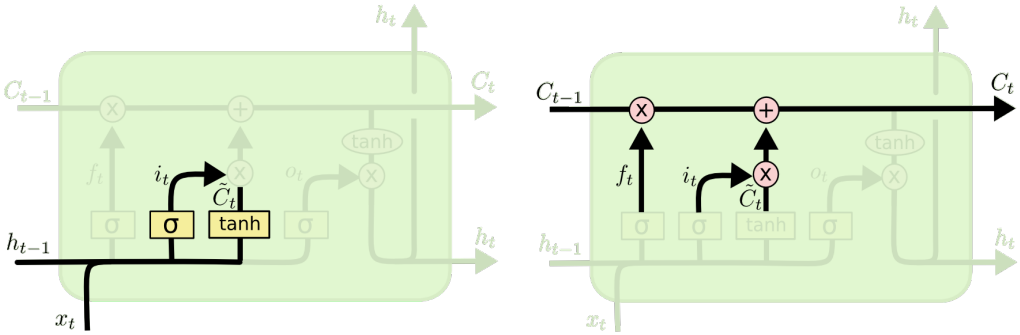

Figura 3.15. Input gate e percorso di aggiornamento dello stato della cella. Fonte: colah's blog

- *Output gate*: Questo gate filtrata lo stato della cella per determinarne l'output. Una funzione sigmoidale è applicata per decidere quale componente dello stato della cella usare come output, mentre una funzione *tanh* è applicato allo stato della cella aggiornato per normalizzare i valori in [-1,1]. Successivamente, questi due elementi sono moltiplicati.

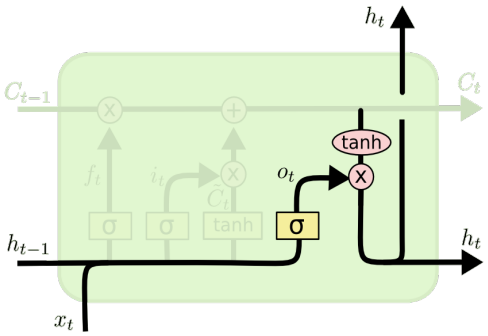

Figura 3.16. Output gate. Fonte: colah's blog

Le equazioni che definiscono le unità precedentemente descritte sono:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{3.21}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{3.22}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3.23}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{3.24}$$

$$h_t = o_t \odot tanh(c_t) \tag{3.25}$$

dove i,f,c e o sono rispettivamente input gate, forget gate, stato della cella e output gate; h_t è lo stato dello strato nascosto al tempo t; W_{xi}, W_{xf}, W_{xo}, W_{xc} sono i pesi rispetti tra il livello di input e, rispettivamente, input gate, forget gate, output gate e cell state; W_{hi}, W_{hf}, W_{ho}, W_{hc} sono i pesi rispetti tra il livello nascosto e, rispettivamente, input gate, forget gate, output gate e cell state; b_i, b_f, b_c, b_o sono rispettivamente i pesi di bias dell'input gate, forget gate, cell state e output gate; infine σ indica la funzione sigmoidale e $\odot$ il prodotto di Hadamard tra vettori.

3.4.2 GRU: Gated Recurrent Unit

Parallelamente alla LSTM, è stata ideata la Gated Recurrent Unit (GRU) [58], anch'essa in grado di gestire il problema dell'esplosione o decadimento del gradiente. La GRU, ha una architettura più semplice rispetto alla LSTM (Figura 3.17), input e forget gate sono combinati in unico update gate. Anche lo stato della cella e lo stato dello strato nascosto sono combinati, rendendo la rete più veloce da allenare. Le seguenti equazioni definiscono formalmente la GRU:

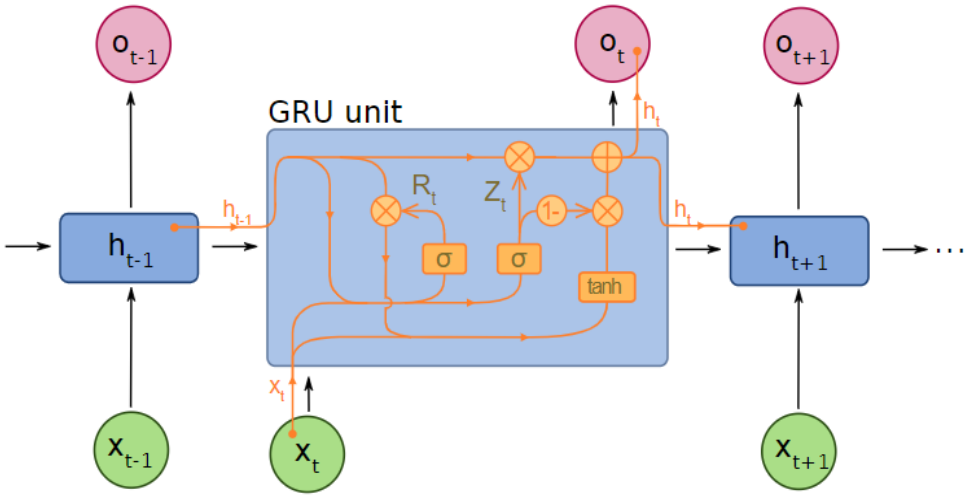

Figura 3.17. Gated Recurrent Unit. Fonte: Wikipedia

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{3.26}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{3.27}$$

$$\tilde{h}_t = tanh(W x_t + U(r_t \odot h_{t-1})) \tag{3.28}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t \tag{3.29}$$

dove z_t è l'update gate; $W_z x_t$ e $U_z h_{t-1}$ sono, rispettivamente, l'input della t-esima cella e l'output della cella precedente, moltiplicati con i rispettivi pesi; r_t è il reset gate; h_t è la funzione di attivazione basata sull'interpolazione lineare tra il precedente stato h_{t-1} e la candidata attivazione $\tilde{h}_t$; infine σ rappresenta la funzione sigmoidale e $\odot$ il prodotto di Hadamard tra vettori.

3.5 Conclusioni

In questo capitolo è stato introdotto il Deep Learning a partire dai paradigmi di apprendimento principali, per capire meglio il funzionamento di una rete neurale. L'elemento fondamentale di una rete neurale è stato descritto, e visto come interagisce con altri elementi in una architettura più complessa come la MLP. Sono stati poi descritte le principali reti neurali ricorrenti e come sono in grado di gestire sequenze temporali di dati. Come descritto, le RNN classiche hanno problemi nel gestire lunghe dipendenze temporali tra dati e quindi è stata introdotta la LSTM, in grado di farlo. Per questa ragione è stata adottata in questa tesi.

Capitolo 4

Architettura del Sistema

In questo capitolo è presentato l'architettura del sistema proposto in questa tesi (Paragrafo 4.1). In dettaglio, si descriveranno le feature estratte (Paragrafo 4.2) e il modello di apprendimento e classificazione (Paragrafo 4.3). Infine, le considerazioni sull'architettura implementata sono riepilogate (Paragrafo 4.4).

4.1 Metodo proposto

L'idea alla base di questa architettura è di analizzare una sequenza di espressioni facciali per individuare composizioni di quest'utltime e pattern che potenzialemente possano discernere un comportamento bugiardo da uno sincero. L'architettura si divide in due componenti principali: il modulo di Feature Extraction e il modulo di Apprendimento e Classificazione. Il metodo proposto è illustrato nella Figura 4.1. Il modulo di Feature Extraction prende in input una sequenza di frame di un video $F = \{f_0, f_1, ..., f_T\}$ e per ogni elemento della sequenza calcola l'istogramma dei gradienti orientati (HOG) e l'asimmetria facciale producendo un insieme di vettori delle feature per ogni elemento della sequenza $X = \{x_0, x_1, ..., x_T\}$. Maggiori dettagli si vedranno nel Paragrafo 4.2. Successivamente, un training set formato dagli insiemi di vettori delle feature X, calcolati nella fase precedente, è dato in input alla LSTM per addestrarla. La LSTM è costruita per agire come classificatore binario, maggiori dettagli sull'architettura della rete si vedranno nel Paragrafo 4.3.

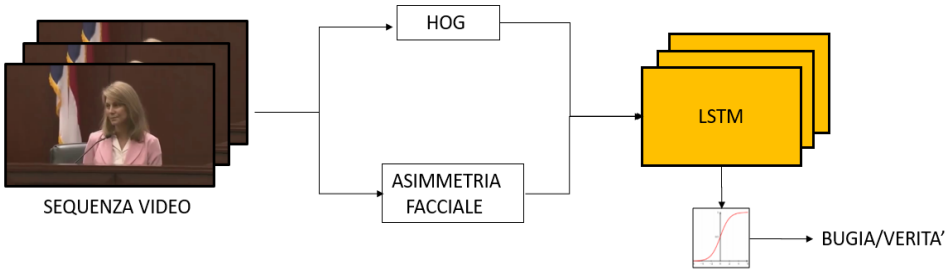

Figura 4.1. Pipeline dell'architettura. Per ogni elemento della sequenza video sono calcolate HOG e asimmetria facciale e i valori sono concatenati in un vettore. La sequenza è poi data in input alla LSTM che la classifica in bugia o verità

4.2 Feature Extraction

Le feature utilizzate in questa tesi sono HOG e asimmetria facciale. L'HOG è un descrittore di caratteristiche di immagini che si è dimostrato efficace nel modellare le espressioni facciali [49]. Quando una persona esprime delle emozioni poco spontanee attraverso le espressioni facciali è molto frequente che ci sia una contrazione muscolare più forte da un lato creando una asimmetria [37]. Nei prossimi sottoparagrafi gli algoritmi per calcolare queste feature per ogni elemento della sequenza sono presentati.

4.2.1 Istogramma dei gradienti orientati

L'HOG è un descrittore di caratteristiche, in grado di evidenziare e modellare la forma dei muscoli facciali, permettendo quindi al sistema di analizzare le espressioni facciali. Il descrittore calcola le occorrenze di orientamento dei gradienti in una determinata porzione dell'immagine. Questo metodo segue un pattern a griglia, dividendo l'immagine in celle. Per ogni cella viene calcolata la pixel magnitude e l'orientamento del gradiente come segue:

$$g = \sqrt{g_x^2 + g_y^2} \tag{4.1}$$

$$\theta = arctan(\frac{g_x}{g_y}) \tag{4.2}$$

dove g è la pixel magnitude e θ l'orientamento del grandiente; g_x e g_y sono la variazione dell'intensità di pixel, rispettivamente, della componente orizzontale e verticale. Per la cella viene poi calcolato l'istogramma dei gradienti orientati. Questi istogrammi vengono poi normalizzati e raggruppati in un blocco più grande, applicandogli la L2-norm. Questa fase permette di rendere il descrittore più robusto ai cambi di illuminazione e ombreggiatura. Sia v il vettore degli istogrammi di un blocco, $||v||_2$ la L2 norm, e ϵ è una piccola costante. Il fattore di normalizzazione f è definito come segue:

$$f = \frac{v}{||v||_2^2 + \epsilon^2} \tag{4.3}$$

Infine, gli istogrammi vengono ordinati e concatenati in un singolo vettore che rappresenta il descrittore dell'immagine.

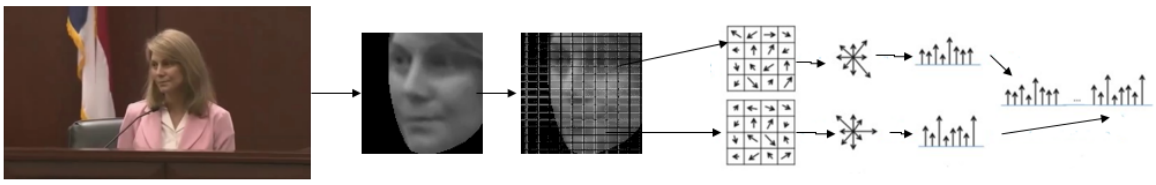

Figura 4.2. Pipeline del processo di estrazione del'HOG feature vector. La faccia viene isolata e rappresentata in scala di grigio. L'immagine viene divisa in celle e per ogni cella è calcolato l'istogramma che rappresenta la distribuzione dei gradienti orientati della variazione di intesità dei pixel.Infine gli istogrammi sono concatenati per formare il vettore finale.

4.2.2 Asimmetria facciale

Per calcolare l'asimmetria facciale è stato adottato il metodo proposto in [38]. Nella prima fase si tracciano i punti di interesse facciali per individuare le ROI da analizzare. Le regioni di interesse in esame sono la zona delle sopracciglia e della bocca studiando separatamente le parti destre e sinistre. Individualmente, ad ogni regione facciale viene applicato il dense optical flow per modellare il loro movimento in frame consecutivi. Il calcolo del movimento totale di una regione facciale tra due frame consecutivi è così definito:

$$V_S = \frac{1}{M} f(r,t) \tag{4.4}$$

$$M = (x_{max} - x_{min})(y_{max} - y_{min}) \tag{4.5}$$

dove V_S è il movimento totale, *f(r,t)* è il flow magnitude calcolato con l'optical flow di una regione facciale r tra il frame t-1 e t; M è il numero di pixel della regione facciale individuato calcolando l'area del rettangolo avente come base la differenza tra la coordinata x più grande x_{max} e la x più piccola x_{min}, e come altezzala differenza tra la coordinata y più grande y_{max} e la più piccola y_{min}. Per il primo frame questo valore è stato preimpostato a 0. Infine, il valore di simmetria di una regione facciale è il seguente:

$$S = 1 - \lambda \left| V_{Sleft} - V_{Sright} \right| \tag{4.6}$$

dove S è il valore di simmetria di una regione facciale; λ è un coefficiente di asimmetria stimato da esperti, il cui valore in questa tesi è 3.8; V_{Sleft} e V_{Sright} sono, rispettivamente il movimento totale della regione sinistra e destra. Quando il valore di S supera 1 il valore viene impostato a 1 e, viceversa, quando il valore è minore di 0 viene impostato a 0.

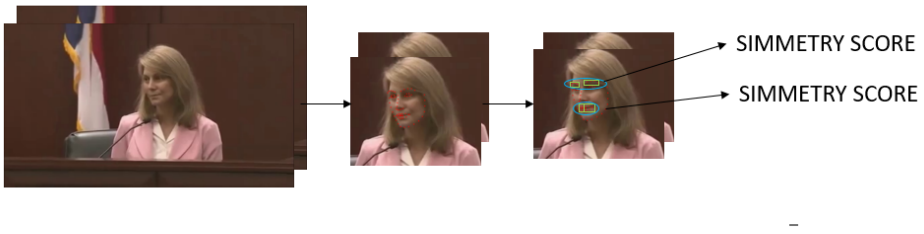

Figura 4.3. Pipeline del processo di calcolo dell'asimmetria delle labbra e delle sopracciglia su due frame consecutivi.

4.3 Architettura della rete

L'architettura della rete utilizzata in questa tesi è la LSTM, con un solo livello nascosto, per la sua capacità di gestire sequenze di input. La LSTM prende in input una sequenza di T vettori X = $\{x_0,x_1,...,x_T\}$ data dalla concatenazione delle feature calcolate nel modulo precedentemente descritto. Di seguito un riepilogo delle equazioni che definiscono la LSTM, estrapolate dal Paragrafo 4.3.1:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{4.7}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{4.8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4.9}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{4.10}$$

$$h_t = o_t \odot tanh(c_t) \tag{4.11}$$

dove i,f,c e o sono rispettivamente input gate, forget gate, stato della cella e output gate; h_t è lo stato dello strato nascosto al tempo t; W_{xi}, W_{xf}, W_{xo}, W_{xc} sono i pesi rispetti tra il livello di input e, rispettivamente, input gate, forget gate, output gate e cell state; W_{hi}, W_{hf}, W_{ho}, W_{hc} sono i pesi rispetti tra il livello nascosto e, rispettivamente, input gate, forget gate, output gate e cell state; b_i, b_f, b_c, b_o sono rispettivamente i pesi di bias dell'input gate, forget gate, cell state e output gate; infine σ indica la funzione sigmoidale e $\odot$ il prodotto di Hadamard tra vettori. Per rendere la LSTM un classificatore binario viene applicata una sigmoide al valore di uscita del livello di output della rete. La classificazione della sequenza è definita dalla seguente equazione:

$$y(x) = \begin{cases} 1, & \text{se } x \geq 0.5, \\ 0, & \text{se } x < 0.5 \end{cases} \tag{4.12}$$

dove y è la funzione di classificazione e x è il valore di uscita della LSTM. Come funzione obiettivo è utilizzata la Binary Cross-Entropy (BCE) definita con la seguente equazione:

$$L(y, \hat{y}) = -y \cdot log(\hat{y}) - (1 - y) \cdot log(1 - \hat{y}) \tag{4.13}$$

dove y è il valore atteso e $\hat{y}$ è il valore predetto dalla LSTM. Come funzione che ottimizza la discesa del gradiente è stato utilizzato l'algoritmo Adam [60].

4.4 Conclusioni

In questo capitolo è stata descritta l'architettura del metodo proposto dettagliando i moduli che la compongono. In particolare sono stati presentati i metodi per calcolare l'HOG e l'asimmetria facciali e come la LSTM le utilizza per apprendere e classificare.

Capitolo 5

Test e Valutazioni

In questo capitolo le librerie utilizzate per la progettazione (Paragrafo 5.1) e il dataset (Paragrafo 5.2) sono presentati. I risultati ottenuti e il confronto con il corrente stato dell'arte sono descritti (Paragrafo 5.3). Infine un riepilogo del capitolo è presentato (Paragrafo 5.4).

5.1 Dettagli di progettazione

Il linguaggio di programmazione utilizzato per realizzare il metodo proposto è Python con l'ausilio delle seguenti librerie:

- **OpenCV**: libreria utilizzata per la Computer Vision, sviluppata da Intel. In questa tesi è stata utilizzata per leggere le immagini e per implementare il dense optical flow;
- **OpenFace**: libreria sviluppata da Baltrusaitis et al. [61] per analizzare i comportamenti facciali. In questa tesi è stata utilizzata per individuare i punti di interesse facciali e per implementare l'HOG;
- **Pytorch**: è una libreria di machine learning open source basata sulla libreria Torch, sviluppata principalmente dal laboratorio di ricerca di intelligenza artificiale di Facebook. In questo lavoro è stata utilizzata per implementare la LSTM.

Inoltre per operazioni sui feature vector è stata utilizzata la libreria NumPy, per collezionare i dati calcolati nel modulo di feature extraction in file csv, la libreria Pandas, e per il calcolo di accuracy e divisione del dataset in training e test la libreria Scikit Learn. Il sistema utilizzato per le sperimentazioni e l'apprendimento della LSTM è un Intel-Core i5 1.80GHz CPU di ottava generazione con 16GB di RAM.

5.2 Dataset

Il Real-life Trial deception dataset è un dataset pubblico ideato da Perez-Rosas et al. [35], dove sono raccolti scenari ad alto rischio come ad esempio i processi in tribunale. Per collezionare il set di dati, sono stati individuati contenuti multimediali pubblici dove le registrazioni dei processi erano disponibili e dove i comportamenti ingannevoli e bugiardi erano osservati e verificati. Per la maggior parte della durata del video è la faccia dell'imputato o del testimone è visibile. La qualità del video e dell'audio è abbastanza per analizzare le espressioni facciali e per capire chiaramente

ciò i soggetti stanno dicendo. Per etichettare i video sono stati analizzati tre differenti risultati dei processi: colpevole, innocente ed esoneri comprovati dalla polizia. Il dataset contiene 121 video, divisi nelle classi verità e bugia con rispettivamente 60 e 61 elementi. La durata media dei video è di 28.0 secondi e i soggetti ripresi sono 21 donne e 35 uomini con una età che varia tra i 16 e i 60 anni. La Figura 5.1 mostra alcuni campioni presi dal dataset. Per un miglior calcolo dell'asimmetria facciali

Figura 5.1. Alcuni esempi presi dal Real-life Trial deception detection dataset.

alcuni elementi del dataset non sono stati presi in considerazione, per un totale di 110 video equamente divisi in bugia e verità.

5.3 Configurazioni sperimentali e risultati

Il dataset utilizzato è stato diviso in training set, per l'apprendimento, e test set per la fase di test. Il training set è il 70% del dataset totale. La fase di apprendimento è stata eseguita usando 50 epoche, learning rate inizialmente impostato a 0.01 e poi aggiornato dall'ottimizzatore Adam e una batch size di 10 elementi. Il tempo medio di apprendimento della LSTM sul Real-life Trial dataset è di circa 25 minuti. Nell'analisi delle sequenze è stata utilizzata la tecnica dello "zero padding" per uniformare la grandezza di esse. Inserire il padding dopo la sequenza riduce le prestazioni di accuratezza della LSTM mentre inserirle prima non altera significativamente i risultati [62]. Per questo motivo il padding è stato aggiunto prima della sequenza. I risultati ottenuti sono illustrati nella Tabella 5.1. Un confronto con i classificatori

Metodo	Features	Accuracy
LSTM	HOG	79,9%
LSTM	HOG + asimmetria	**80,8%**

Tabella 5.1. Risultati ottenuti sul Real-life trial dataset, ottenuti utilizzando la 10-cross validation sui video del dataset.

utilizzati nel lavoro proposto da Avola et al. [1] per analizzare l'HOG dimostra l'efficacia della LSTM (Tabella 5.2). Nella Tabella 5.3 il confronto con i lavori proposti dallo stato dell'arte che utilizzano lo stesso dataset sono presentati. La maggior parte dei lavori proposti sono sistemi multimodali, quindi, verranno considerati solo i risultati ottenuti analizzando caratteristiche visuali. I metodi usati per il confronto sono quello che hanno ottenuto i risultati migliori, nel nostro caso la LSTM che analizza HOG e simmetria. Nel Real-life Trial dataset il numero di soggetti è minore al numero di video. Gli autori hanno adottato diversi protocolli di test per evitare

Algoritmo	Accuracy
LSTM	**79,9%**
SVM	70.24%
MLP	70.22%
Logistic Regression	68.46%
XGBoost	67.71%

Tabella 5.2. Confronto con alcuni algoritmi proposti in Avola et al. per analizzare l'HOG.

al sistema di apprendere caratteristiche specifiche di una persona trasformando il rilevamento dell'inganno in un sistema di re-identificazione. Perez-Rosas et al. [28] hanno utilizzato la Leave One Subject Out (LOSO) cross validation. Lu et al [29] e Krishnamurthy et al. [30] hanno implementato la 10-fold cross validation in base ai soggetti del dataset facendo in modo che i soggetti nel test set non si sovrapponessero a quelli del training set. I risultati presi in considerazione in Avola et al. [1] sono quelli ottenuti utilizzando il protocollo di test appena citato. Mathur et al. [23] utilizza una 5-fold stratified cross validation. Karimi et al. [26] valida i test facendo la media su dieci esecuzioni. Il confronto con lo stato dell'arte è, in parte, dipendente

Metodo	Accuracy	Algoritmo
Perez-Rosas et al. [28]	80.79%	NN
Lu et al. [29]	93.16%	FFCSN
Mathur et al. [23]	76%	SVM
Avola et al. [1]	92.01%	Stacked generalization
Krishnamurthy et al. [30]	93.08%	MLP
Karimi et al. [26]	75.00%	RF
Metodo proposto	80,8%	LSTM

Tabella 5.3. Confronto con lo stato dell'arte dei risultati ottenuti sul Real-life Trial Dataset.

dal protocollo di test adottato. Nonostante i risultati siano inferiori a parte dei lavori proposto dallo stato dell'arte, dimostrano, comunque, l'efficacia del metodo proposto.

5.4 Conclusioni

In questo capitolo le librerie utilizzate per la realizzazione e la sperimentazione del metodo proposto sono state presentate. La configurazione sperimentale della LSTM è stata descritta e i risultati ottenuti messi a confronto con lo stato dell'arte sono mostrati nella Tabella 5.3. Per rendere il sistema più robusto si potrebbero analizzare

Capitolo 6

Conclusioni

In questa tesi è stato presentato un sistema di rilevamento dell'inganno automatico analizzando video. Il metodo analizza la composizione delle espressioni facciali su una sequenza di immagini per individuare comportamenti sinceri o bugiardi. Gli esperimenti sono stati condotti sul Real-life Trial dataset, maggiormente utilizzato dallo stato dell'arte. I risultati ottenuti su questo dataset dimostrano l'efficacia del metodo proposto. Per rendere il sistema più robusto, si potrebbe renderlo multimodale, analizzando feature di natura diversa rispetto alle caratteristiche facciali, come il modo di parlare e il discorso di una persona . Per migliorare la ricerca in questo campo si necessita di un dataset più ampio e di qualità maggiore.

Bibliografia

[1] Danilo Avola, Marco Cascio, Luigi Cinque, Alessio Fagioli and Gian Luca Foresti, LieToMe: An Ensemble Approach for Deception Detection from Facial Cues, International Journal of Neural Systems, Vol. 31, No. 02, 2050068 (2021)

[2] DePaulo, B. M., Kashy, D. A., Kirkendol, S. E., Wyer, M. M., & Epstein, J. A. (1996). Lying in everyday life. Journal of Personality and Social Psychology, 70(5), 979–995.

[3] Lin Su, Martin Levine, Does “lie to me” lie to you? An evaluation of facial clues to high-stakes deception, Computer Vision and Image Understanding, Volume 147, 2016.

[4] Bond, C. F. and DePaulo, B. M. (2006) ‘Accuracy of Deception Judgments’, Personality and Social Psychology Review, 10(3), pp. 214–234.

[5] T. Gannon, A. Beech, and T. Ward, Risk Assessment and the Polygraph. John Wiley and Sons Ltd, 2009, pp. 129–154.

[6] P. Ekman, Telling lies: Clues to deceit in the market- place, politics, and marriage (revised edition) 2009.

[7] M. L. Newman, J. W. Pennebaker, D. S. Berry and J. M. Richards, Lying words: Predicting deception from linguistic styles, Personality and Social Psychology Bulletin 29(5) (2003) 665–675.

[8] P. Ekman and W. V. Friesen, Detecting deception from the body or face, Journal of Personality and Social Psychology 29(3) (1974) 288–298.

[9] P. Ekman and E. L. Rosenberg, What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS) 1997.

[10] P. Ekman, Darwin, deception, and facial expres- sion, Annals of the New York Academy of Sciences 1000(1) (2003) 205–221.

[11] Lai, YF., Chen, MY. & Chiang, HS. Constructing the lie detection system with fuzzy reasoning approach. Granul. Comput. 3, 169–176 (2018).

[12] R. Alazrai, F. Alqasem, S. Alaarag, K. M. Ahmad Yousef and M. I. Daoud, "A Bispectrum-based Approach for Detecting Deception using EEG Signals," 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom), 2018.

[13] F. Andrew Kozel, Kevin A. Johnson, Qiwen Mu, Emily L. Grenesko, Steven J. Laken, Mark S. George, Detecting Deception Using Functional Magnetic Resonance Imaging, Biological Psychiatry, Volume 58, Issue 8, 2005.

[14] Simpson JR. Functional MRI lie detection: too good to be true? J Am Acad Psychiatry Law. 2008.

[15] Li, Fang et al. “Lie Detection Using fNIRS Monitoring of Inhibition-Related Brain Regions Discriminates Infrequent but not Frequent Liars.” Frontiers in human neuroscience vol. 12 71. 13 Mar. 2018

[16] M. R. Bhutta, K. Hong, N. Naseer and M. J. Khan, "Spontaneous lie detection using functional near-infrared spectroscopy in an interactive game," 2015 10th Asian Control Conference (ASCC), 2015.

[17] Yan Zhou, Heming Zhao, Xinyu Pan, Li Shang, Deception detecting from speech signal using relevance vector machine and non-linear dynamics features, Neurocomputing,2015.

[18] H. Tao, P. Lei, M. Wang, J. Wang and H. Fu, "Speech Deception Detection Algorithm Based on SVM and Acoustic Features," 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), 2019.

[19] H. Nasri, W. Ouarda and A. M. Alimi, "ReLiDSS: Novel lie detection system from speech signal," 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), 2016.

[20] Rada Mihalcea, Verónica Pérez-Rosas, and Mihai Burzo. 2013. Automatic Detection of Deceit in Verbal Communication. In Proceedings of the 15th ACM on International Conference on Multimodal Interaction. 131–134.

[21] Wasiq Khan, Keeley Crockett, James O’Shea, Abir Hussain, Bilal M. Khan, Deception in the eyes of deceiver: A computer vision and machine learning based automated deception detection, Expert Systems with Applications, Volume 169, 2021.

[22] D. H. Wolpert, Stacked generalization, Neural Net- works 5(2) (1992) 241–259.

[23] Leena Mathur and Maja J. Matarić. 2020. Introducing Representations of Facial Affect in Automated Multimodal Deception Detection. In Proceedings of the 2020 International Conference on Multimodal Interaction (ICMI ’20).

[24] N. Bhaskaran, I. Nwogu, M. G. Frank and V. Govindaraju, "Lie to Me: Deceit detection via online behavioral learning," 2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG).

[25] J. G. Proudfoot, J. L. Jenkins, J. K. Burgoon and J. F. Nunamaker, "Deception is in the eye of the communicator: Investigating pupil diameter variations in automated deception detection interviews," 2015 IEEE International Conference on Intelligence and Security Informatics (ISI).

[26] H. Karimi, J. Tang and Y. Li, "Toward End-to-End Deception Detection in Videos," 2018 IEEE International Conference on Big Data (Big Data)

[27] R. Rill-García, H. J. Escalante, L. Villaseñor-Pineda and V. Reyes-Meza, "High-Level Features for Multimodal Deception Detection in Videos," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)

[28] U. M. Sen, V. Perez-Rosas, B. Yanikoglu, M. Abouelenien, M. Burzo and R. Mihalcea, "Multimodal Deception Detection using Real-Life Trial Data," in IEEE Transactions on Affective Computing

[29] M. Ding, A. Zhao, Z. Lu, T. Xiang and J. Wen, "Face-Focused Cross-Stream Network for Deception Detection in Videos," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

[30] G. Krishnamurthy, N. Majumder, S. Poria and E. Cambria, A deep learning approach for multimodal deception detection, Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing (CICLing), 2018.

[31] N. Carissimi, C. Beyan and V. Murino, "A Multi-View Learning Approach to Deception Detection," 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)

[32] Mohamed Abouelenien, Verónica Pérez-Rosas, Bohan Zhao, Rada Mihalcea, and Mihai Burzo. 2017. Gender-based multimodal deception detection. In Proceedings of the Symposium on Applied Computing (SAC '17).

[33] Canedo, D.; Neves, A.J.R. Facial Expression Recognition Using Computer Vision: A Systematic Review. Appl. Sci. 2019

[34] Zhe Wu and Bharat Singh and Larry S. Davis and V. S. Subrahmanian, Deception Detection in Videos, arXiv, 2017.

[35] Veronica Perez-Rosas, Mohamed Abouelenien, Rada Mihalcea, and Mihai Burzo. 2015. Deception Detection using Real-life Trial Data. In Proceedings of the 2015 ACM on International Conference on Multimodal Interaction (ICMI '15). ACM, New York, NY, USA, 59-66.

[36] S. Velusamy, H. Kannan, B. Anand, A. Sharma and B. Navathe, "A method to infer emotions from facial Action Units," 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011

[37] Ekman, P., Hager, J.C. and Friesen, W.V. (1981), The Symmetry of Emotional and Deliberate Facial Actions. Psychophysiology, 18: 101-106.

[38] A. M. N. Taufique, A. Savakis and J. Leckenby, "Automatic Quantification of Facial Asymmetry Using Facial Landmarks," 2019 IEEE Western New York Image and Signal Processing Workshop (WNYISPW), 2019

[39] Artificial Intelligence: A Modern Approach (vol.2). Stuart Russel and Peter Norvig. Pearson, Prentice Hall.

[40] Amir Zadeh, Tadas Baltrušaitis and Louis-Philippe Morency,Convolutional Experts Constrained Local Model for Facial Landmark Detection, arXiv, 2017.

[41] Tomasi, C.; Manduchi, R. Bilateral filtering for gray and color images. ICCV 1998.

[42] Lindenbaum, M.; Fischer, M.; Bruckstein, A. On Gabor's contribution to image enhancement. Pattern Recognit. 1994

[43] Pizer, S.M.; Amburn, E.P.; Austin, J.D.; Cromartie, R.; Geselowitz, A.; Greer, T.; ter Haar Romeny, B.; Zimmerman, J.B.; Zuiderveld, K. Adaptive histogram equalization and its variations. Comput. Vis. Graph. Image Process. 1987

[44] Vishwakarma, Virendra Pandey, Sujata Gupta, M. (2009). Adaptive Histogram Equalization and Logarithm Transform with Rescaled Low Frequency DCT Coefficients for Illumination Normalization. International Journal of Recent Trends in Engineering. 1.

[45] Ojala, T.; Pietikäinen, M.; Mäenpää, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Analy. Mach. Intell. 2002;

[46] Farnebäck G. (2003) Two-Frame Motion Estimation Based on Polynomial Expansion. In: Bigun J., Gustavsson T. (eds) Image Analysis. SCIA 2003. Lecture Notes in Computer Science, vol 2749. Springer, Berlin, Heidelberg.

[47] . Pakstas, A.; Forchheimer, R.; Pandzic, I.S. MPEG-4 Facial Animation: The Standard, Implementation and Applications; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2002.

[48] Jain, A.K.; Farrokhnia, F. Unsupervised texture segmentation using Gabor filters. Pattern Recognit. 1991

[49] Carcagnì, P., Del Coco, M., Leo, M. et al. Facial expression recognition and histograms of oriented gradients: a comprehensive study. SpringerPlus 4, 645 (2015).

[50] Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 0-387-31073-8

[51] C. M. Bishop, Neural Networks for Pattern Recognition. New York, NY, USA: Oxford University Press, Inc., 1995.

[52] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in Neural networks for perception, pp. 65–93, Elsevier, 1992.

[53] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," pp. 402–408, 01 2001.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014.

[55] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, (New York, NY, USA), pp. 78–, ACM, 2004.

[56] L. Medsker and L. C. Jain, Recurrent neural networks: design and applications. CRC press, 1999.

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[58] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv preprint arXiv:1412.3555, 2014.

[59] Paul Viola and Michael Jones, Robust Real-time Object Detection, International Journal of Computer Vision, 2001.

[60] Diederik P. Kingma and Jimmy Ba, Adam: A Method for Stochastic Optimization, arXiv, 2017.

[61] T. Baltrusaitis, A. Zadeh, Y. C. Lim and L. Morency, "OpenFace 2.0: Facial Behavior Analysis Toolkit," 2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018).

[62] Mahidhar Dwarampudi and N V Subba Reddy, Effects of padding on LSTMs and CNNs, 2019.