Universitá degli studi di Salerno

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Progetto di corso - Biometria e Visione Artificiale

BABELE: Riconoscimento multilingua senza audio tramite approccio probabilistico

Docente

Prof. **Michele Nappi**

Tutor

Dott.ssa **Lucia Cascone**

Team Member

Carmine D'Angelo

Matricola: 0522500881

Matteo Della Rocca

Matricola: 0522501500

Francesco Aurilio

Matricola: 0522500979

Anno Accademico 2022-2023

Indice

Elenco delle figure

CAPITOLO 1

Introduzione

La fonetica si occupa dello studio dei suoni del linguaggio umano. Il numero di fonemi utilizzati in una lingua varia da 15 a 50 con una media di 30 fonemi per lingua. Al variare della lingua parlata alcune sequenze fonetiche sono più probabili di altre. Ogni suono corrisponde ad una determinata articolazione legata ai movimenti facciali. La fonetica articolatoria studia proprio l'articolazione dei suoni del linguaggio, osservando i movimenti dei diversi organi vocali coinvolti nella produzione dei suoni. Essa analizza come i suoni vengono formati attraverso la posizione della lingua, delle labbra, della faringe e di altri organi. Babele è un progetto che, a partire da un dataset di soggetti che parlano, si pone come obiettivo quello di riconoscere la lingua parlata considerando esclusivamente i movimenti articolari della bocca eliminando la componente audio. L' obiettivo di questo documento è di descrivere il lavoro svolto e i risultati ottenuti. Il preprocessing dei video è stato effettuato sperimentando diverse metodologie mentre la classificazione è stata realizzata tramite una rete neurale probabilistica (PNN) personalizzata ad hoc. Sono stati infine calcolati anche i risultati prendendo in considerazione i dataset CH-SIM e CMU-MOSEI in modo da validare i risultati ottenuti con il dataset principale.

CAPITOLO 2

Lavori correlati

Come dimostrato nel lavoro "Analysis of Anomaly Detection Techniques in Video Surveillance"[1], l'utilizzo delle Probabilistic Neural Network (PNN) è molto correlato all' individuazione di anomalie. Lo studio condotto nel paper ha utilizzato una PNN per cercare anomalie nei video di sorveglianza; in questo caso, le anomalie che hanno considerato sono gli incidenti che potrebbero coinvolgere i cittadini; nel nostro caso, quando si tratta di lingue, abbiamo considerato come anomalie i movimenti delle labbra che possono esserci durante la pronuncia di una determinata lettera o parola. Ad esempio nell'italiano abbiamo la presenza di un determinato movimento delle labbra per la lettera "l", ma questo movimento è totalmente assente nella lingua giapponese, anche se in quest'ultima esiste una consonante chiamata "r", che ha una pronuncia simile ad un suono tra una "r" e una "l" dell'inglese. La ricerca intitolata "Sparse Coding Based Lip Texture Representation For Visual Speaker identification"[2] si concentra anche sulla scelta di utilizzare solo le labbra per identificare una lingua specifica senza utilizzare l'audio. Il documento dimostra che molte informazioni sulla lingua possono essere conservate solo attraverso il movimento delle labbra.
Un ultimo lavoro che utilizza una PNN ma identifica un soggetto utilizzando le caratteristiche delle labbra è stata pubblicata nel seguente articolo: "Sparse coding based lip texture representation for visual speaker identification"[3], nel paper sono descritte varie feature utili per identificare le labbra, viene anche mostrata una tecnica chiamata PSO per ottimizzare la ricerca di un sigma ottimo in una PNN.

CAPITOLO 3

Descrizione dataset utilizzati

3.1 BABELE

Il dataset BABELE è una vasta raccolta di migliaia di video di soggetti di diverse lingue, generi ed età che parlano. Il dataset include video in otto lingue diverse e copre una vasta gamma di variazioni linguistiche e di pronuncia. Molti video sono già suddivisi in set di Train, Test e Validation.

3.1.1 Nomenclatura

Ogni video è identificato da una nomenclatura precisa che ne indica la lingua, il genere, l'età e un id univoco, oltre al frame-rate.

Lingua_genere_età_id-univoco_frame-rate

dove:

- **Lingua**: è un numero che va da 1 a 8, in particolare:
    1. Italiano;
    2. Inglese;
    3. Tedesco;
    4. Spagnolo;
    5. Olandese;
    6. Russo;
    7. Giapponese;
    8. Francese.

- **Genere**: un numero che va da 1 a 2, in particolare:
  1. Uomo;
  2. Donna.
- **Età**: un numero che va da 1 a 2, in particolare:
  1. Meno di 30 anni;
  2. Più di 30 anni.
- **Id-univoco**: un numero per indicare in modo univoco il soggetto nella categoria di appartenenza.
- **Frame-rate**: un numero che rappresenta il frame-rate del video.

3.1.2 SubjectIndipendent - Full Video

Questa sottocategoria di video, appartenente a BABELE, è stata selezionata in quanto presenta una serie di caratteristiche che la rendono particolarmente adatta per il nostro scopo: i video sono a schermo intero, mostrano il volto delle persone e non richiedono il riconoscimento del soggetto per individuare la lingua parlata. Una volta selezionati, i video sono stati utilizzati principalmente per l'estrazione delle distanze. Su questo particolare dataset sono state effettuate due tipi di classificazione: una classificazione multiclasse ed una binaria.

3.1.3 SubjectIndipendent - Lips Video

Quest'altra sottocategoria, sempre appartenente a BABELE, presenta video che riprendono solo le labbra delle persone che parlano, senza altri fattori che potrebbero influenzare l'analisi. Anche in questi video non è richiesto il riconoscimento del soggetto per individuare la lingua.
Una volta selezionati i video, ad essi è stata applicata la codifica sparsa per l'estrazione di caratteristiche distintive, come la forma e il movimento delle labbra, senza l'influenza di fattori esterni come l'illuminazione o il contesto visivo. Oltre all'estrazione di caratteristiche distintive tramite la codifica sparsa, è stato utilizzato anche l'algoritmo Optical Flow per analizzare il movimento del volto e delle labbra nei video, discriminando così informazioni utili da quelle irrilevanti. Su questo particolare dataset è stata effettuata una classificazione multiclasse.

3.2 CH-SIM CMU-MOSEI

Questo dataset è composto da video di persone che parlano cinese (CH-SIMS) e inglese (CMU-MOSEI), che sono stati utilizzati per estrarre delle misure dalle labbra per l'individuazione del linguaggio parlato. In particolare, su questo dataset è stata effettuata una classificazione binaria. È importante notare che il dataset CH-SIMS CMU-MOSEI non presenta una nomenclatura precisa.

CAPITOLO 4

Preprocessing

Nel presente capitolo, esploreremo dettagliatamente il preprocessing effettuato sui vari video del dataset BABELE e CH-SIM e CMU-MOSEI. Saranno presentati i vari test eseguiti al fine di determinare le configurazioni ottimali. Tali configurazioni sono state ottenute mediante diverse esecuzioni della PNN, i risultati di tali esecuzioni saranno ulteriormente approfonditi nel successivo Capitolo 6 "Risultati ottenuti".

4.1 Scelta del numero di frame da estrarre

La scelta del numero di frame da utilizzare per tagliare un video è un aspetto fondamentale nel contesto del nostro studio. Durante i nostri test, abbiamo esaminato diversi valori, tra cui 30, 40, 48, 50, 60, 80, 100, 120 e 240. Abbiamo stabilito 240 come tetto massimo, poiché il nostro dataset richiede un bilanciamento tra la durata del video e il numero di frame. Tenendo presente che il minimo numero di fotogrammi al secondo (FPS) nel nostro dataset è 24, e considerando che tutti i video hanno una durata di 10 secondi, abbiamo selezionato 240 per assicurarci di avere un buon bilanciamento.

Inizialmente, abbiamo condotto diverse prove empiriche per valutare l'impatto del numero di frame sulla nostra rete PNN. Durante questi test, abbiamo osservato che l'utilizzo di 60 frame totali era una scelta appropriata in termini di prestazioni. Tuttavia, abbiamo anche notato che, all'interno di ogni secondo di video, venivano sempre considerati i primi 6 frame. Non siamo riusciti a identificare un criterio universale per determinare quali frame fossero più importanti degli altri.

Di conseguenza, abbiamo adottato un approccio randomico per selezionare i 6 frame da considerare in ogni secondo di video. Questa scelta si basa sul principio del quicksort[4], un algoritmo di

ordinamento in cui la scelta del pivot è migliorata utilizzando un approccio randomico. Applicando un processo simile, abbiamo selezionato casualmente 6 frame all'interno di ogni secondo di video e abbiamo constatato dei miglioramenti modesti nelle prestazioni complessive del nostro sistema.

L'approccio randomico nella selezione dei frame ci ha consentito di affrontare la sfida di determinare una sequenza di frame ottimale per l'analisi dei video. Tuttavia, è importante sottolineare che questa scelta potrebbe comportare una variazione nei risultati durante l'esecuzione del nostro algoritmo. Nonostante ciò, i risultati ottenuti attraverso l'approccio randomico sono stati promettenti e hanno contribuito a migliorare l'accuratezza delle nostre valutazioni.

4.2 Estrazione delle feature

In questa sezione descriveremo quali feature delle labbra abbiamo utilizzato, di come sono state estratte e di quali problemi si sono affrontati.

4.2.1 Landmark utilizzati

Per l'estrazione dei landmark, abbiamo utilizzato la libreria Mediapipe, una libreria open source sviluppata da Google che fornisce una vasta gamma di modelli e strumenti per l'elaborazione dei dati multimodali, compresa l'analisi delle immagini e dei video. La libreria offre una serie di moduli preaddestrati per il rilevamento e il tracciamento di oggetti e parti del corpo umano, compresi i punti chiave del volto (figura 4.1).

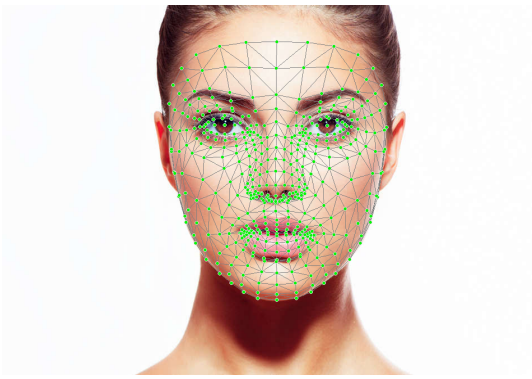

Figura 4.1: Landmark facciali di media pipe

Nel nostro studio, ci siamo basati sul paper "Using a Probabilistic Neural Network for lip-based biometric verification"[3] per selezionare specifiche feature facciali. Abbiamo scelto di utilizzare le misurazioni della larghezza e dell'altezza delle labbra come indicatori significativi per le nostre analisi che corrispondono alla "feature 2" e "feature 3" del paper sopra citato. Per individuare queste feature, abbiamo identificato specifici punti chiave all'interno della regione delle labbra, utilizzando le coordinate predefinite fornite dalla libreria Mediapipe.

I punti chiave che abbiamo selezionato per le nostre misurazioni delle labbra sono i seguenti:

(306,402), (61,402), (17,314), (0,267), (37,402), (267,402), (84,402), (314,402), (61,402), (17,314), e corrispondono alla seguente rappresentazione:

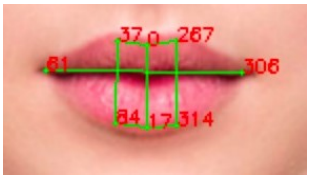

Figura 4.2: Rappresentazione feature scelte

4.2.2 Distanze utilizzate

La fase successiva del nostro preprocessing è il calcolo delle distanze dei vari punti chiave individuati precedentemente. Per calcolare le distanze tra i vari punti chiave, abbiamo utilizzato diverse metriche di distanza, tra cui la distanza cityblock, cosine, chebyshev ed euclidea. Di seguito è una breve descrizione di ciascuna metrica di distanza:

- **Distanza Cityblock (Manhattan)**: La distanza cityblock prende il nome dal fatto che calcola la distanza come il numero di blocchi che si devono attraversare in una griglia cittadina per raggiungere una destinazione. Immagina di spostarti su una griglia stradale, dove puoi muoverti solo in orizzontale o verticale. La distanza cityblock misura la somma delle differenze assolute tra le coordinate dei punti lungo gli assi x e y. Questa metrica di distanza è utile quando si considerano spostamenti limitati a percorsi rettilinei lungo gli assi x e y.

$$d_{\text{cityblock}} = |x_2 - x_1| + |y_2 - y_1|$$

- **Distanza Cosine**: La distanza coseno è basata sul concetto di angolo tra due vettori. I vettori sono considerati come punti nello spazio e la distanza coseno calcola la dissimilarità tra di essi. La metrica di distanza coseno utilizza il coseno dell'angolo tra i due vettori normalizzati. Se i vettori sono simili, l'angolo sarà piccolo e la distanza coseno sarà vicina a zero. Al contrario, se i vettori sono diversi, l'angolo sarà ampio e la distanza coseno sarà vicina a uno.

$$d_{\text{cosine}} = 1 - \frac{A \cdot B}{\|A\|\|B\|}$$

- **Distanza Chebyshev**: La distanza di Chebyshev prende il nome dal matematico Pafnuty Chebyshev. Questa metrica di distanza calcola la deviazione massima tra le coordinate dei punti lungo gli assi x e y. Immagina di spostarti su una griglia e di cercare il percorso più breve

tra due punti, consentendo spostamenti diagonali. La distanza di Chebyshev calcola la massima differenza tra le coordinate x e y dei due punti, considerando spostamenti in tutte le direzioni.

$$d_{\text{chebyshev}} = \max(|x_2 - x_1|, |y_2 - y_1|)$$

- **Distanza Euclidea**: La distanza euclidea è basata sul teorema di Pitagora nel piano. Questa metrica di distanza calcola la lunghezza del vettore diretto tra due punti nello spazio euclideo. È la distanza più comune e ampiamente utilizzata. La distanza euclidea calcola la radice quadrata della somma dei quadrati delle differenze tra le coordinate dei punti lungo gli assi x e y. Rappresenta la lunghezza del percorso più breve tra i due punti nello spazio euclideo.

$$d_{\text{euclidean}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Dalle varie prove effettuate la distanza cityblock è risultata quella più accurata.

4.2.3 Eliminazione valori not a number (NaN)

Durante la creazione dei dataset, abbiamo riscontrato numerosi problemi relativi al calcolo della distanza quando la zona labiale non veniva identificata correttamente. Ciò ha comportato l'inserimento di valori NaN nel dataset delle feature, aumentando così la probabilità di errore della PNN. Per risolvere questo problema, inizialmente abbiamo pensato di non includere nel dataset i video con valori NaN, ma questo ha creato uno sbilanciamento tra i vari dataset. Pertanto, abbiamo deciso di sostituire i valori NaN utilizzando i seguenti passaggi:

1. Inizialmente si è effettuato una media dei valori sulla riga per poter eliminare eventuali valori NaN all'inizio del dataset;
2. Successivamente si è eseguita un interpolazione lineare per eliminare tutti i valori NaN presenti in posizioni randomiche dei dataset.

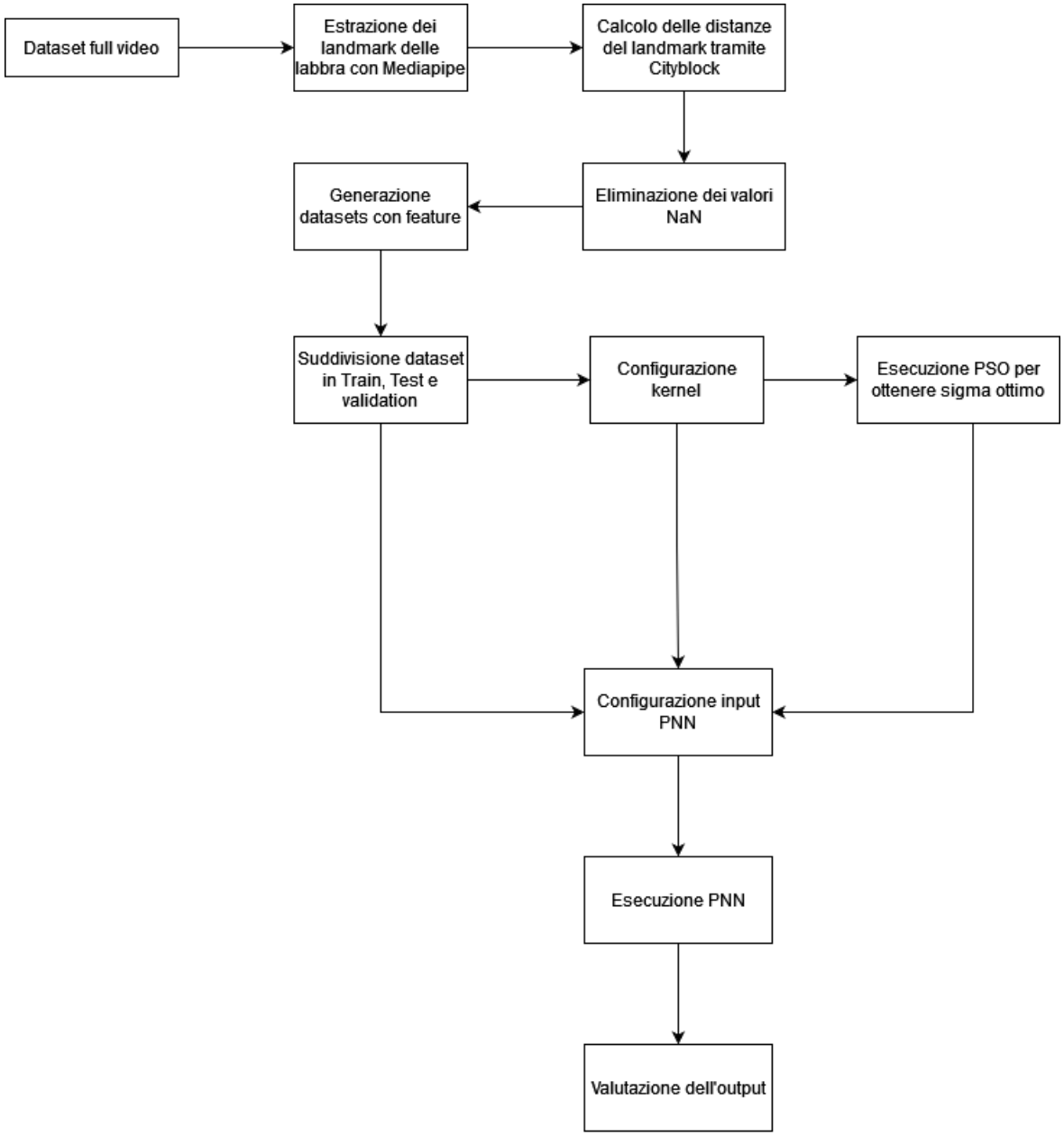

Figura 4.3: Schema preprocessing utilizzando i landmark

4.3 Preprocessing tramite codifica sparsa

Per tentare di migliorare ulteriormente le prestazioni della PNN, abbiamo esplorato un'alternativa al preprocessing dei dati mediante l'adozione della codifica sparsa. Questa tecnica, ampiamente utilizzata nell'elaborazione del segnale e nell'apprendimento automatico, permette una rappresentazione efficiente dei dati sfruttando la loro natura sparsa. Ciò implica che solamente un limitato numero di componenti del dizionario contribuiscono in maniera significativa alla rappresentazione di un dato

specifico.
Nel nostro lavoro, abbiamo implementato l'algoritmo *MiniBatchDictionaryLearning*, disponibile nella libreria Python Scikit-learn, per apprendere il dizionario. Questo algoritmo di apprendimento non supervisionato mira a identificare un dizionario sparsificato in grado di rappresentare efficacemente un insieme di dati. A tale scopo, l'algoritmo sfrutta la codifica sparsa per esprimere i dati come una combinazione lineare delle componenti del dizionario.

L'applicazione di questa metodologia ha portato alla creazione di un dataset con circa 60mila componenti per ogni riga, per poter dare in pasto il dataset alla PNN si è applicata una tecnica di riduzione dei dati chiamata **Principal Component Analysis (PCA)**.

4.3.1 Utilizzo della Principal Component Analysis

La PCA è una tecnica molto utilizzata per l'analisi di grandi set di dati, questa tecnica infatti cerca di ridurre la dimensione dei dati senza però perdere informazioni. Nel nostro studio, abbiamo applicato la PCA su un insieme di componenti specifiche, comprese [4, 8, 20, 40, 60, 80, 120, 160]. Abbiamo creato dataset basati su tali componenti al fine di esplorare l'effetto della riduzione della dimensionalità sulle prestazioni della PNN. Tuttavia, i risultati ottenuti hanno evidenziato una riduzione significativa delle prestazioni della PNN. Sulla base di questa evidenza, giungiamo alla conclusione che l'applicazione dell'estrazione di feature tramite la codifica sparsa, in questo specifico contesto non è stata utile ma potrebbe essere molto utile per il riconoscimento di un soggetto tramite le labbra. In effetti, nel nostro caso, come precedentemente menzionato, l'applicazione della codifica sparsa ha introdotto un rumore aggiuntivo che ha compromesso le prestazioni complessive del sistema.

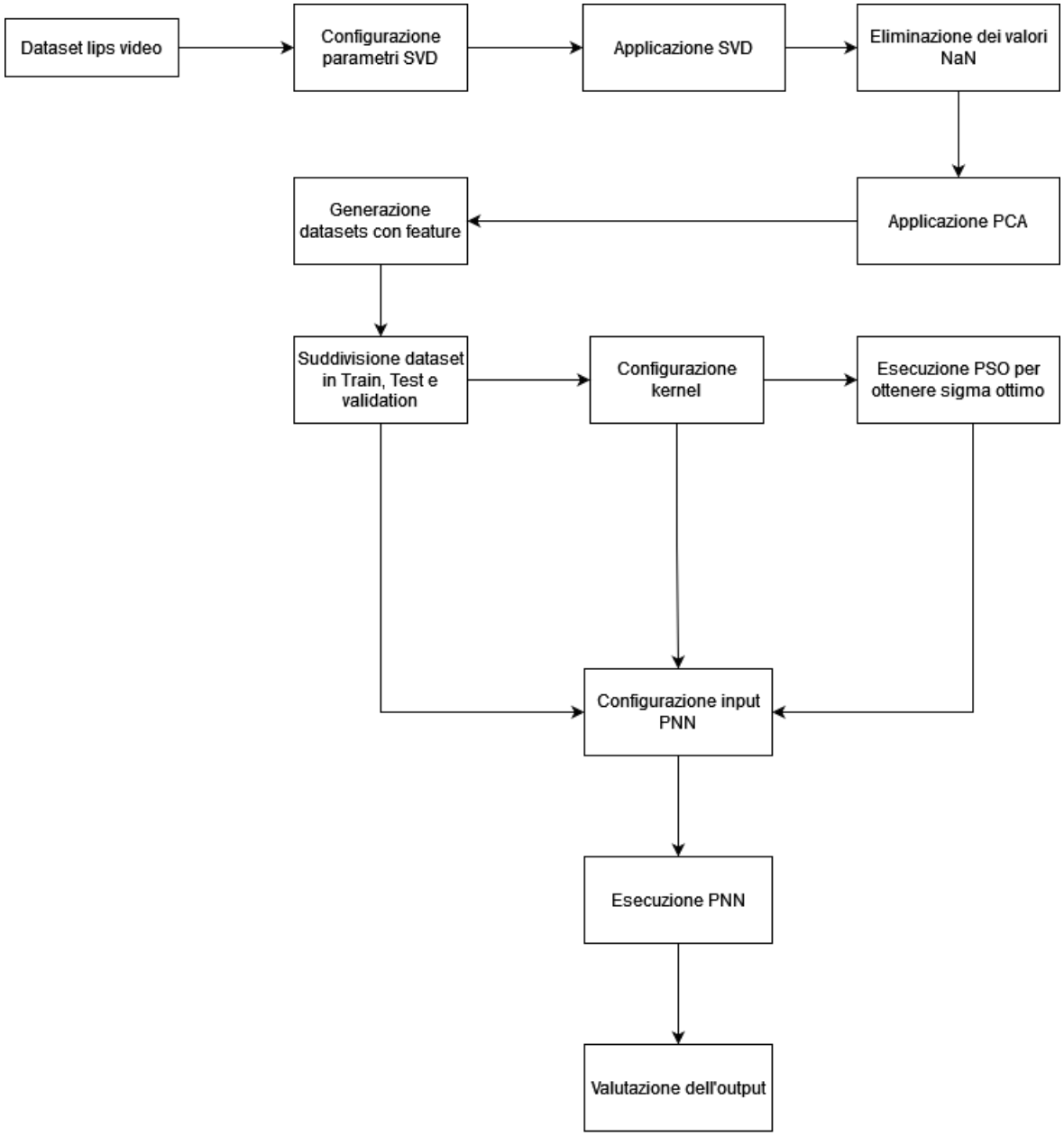

Figura 4.4: Schema preprocessing utilizzando l'codifica sparsa

4.4 Preprocessing tramite optical flow

L'optical flow, o flusso ottico, è un concetto utilizzato per descrivere il movimento apparente dei pixel nell'immagine. Si basa sull'assunzione che i pixel corrispondenti in due frame successivi di un video abbiano spostamenti relativamente piccoli tra di loro. Il suo obiettivo è quindi determinare la direzione e la velocità di spostamento dei punti tra i frame consecutivi. Queste informazioni possono essere utilizzate per diverse applicazioni, come il tracciamento del movimento degli oggetti, la stima della profondità o la rilevazione dei contorni dell'immagine. L'optical flow viene rappresentato come un campo vettoriale, dove ogni vettore rappresenta la direzione e la velocità di spostamento di un punto nell'immagine. Questa rappresentazione fornisce informazioni sul movimento dei punti e può essere utilizzata per analizzare il flusso del movimento in un'immagine o in un video. Il movimento di un pixel è rappresentato sulle dimensione x e y e funziona nel seguente modo:

- Asse x: rappresenta di quanto il pixel si è spostato sull'asse orizzontale dal frame precedente, se ha valore negativo si è mosso verso sinistra, se è 0 non si è spostato e se è positivo si è mosso verso destra;
- Asse y: rappresenta di quanto il pixel si è spostato sull'asse verticale dal frame precedente, se ha valore negativo si è mosso verso il basso, se è 0 non si è spostato e se è positivo si è mosso verso l'alto;

Esistono diversi metodi per calcolare l'optical flow. Uno dei più comuni è il metodo di **Farneback**, utilizzato per calcolare l'optical flow tra due frame consecutivi di un video. Si basa sulla correlazione di intensità locale dei pixel dell'immagine. Utilizza un modello polinomiale per approssimare le distribuzioni dei valori dei pixel nei due frame analizzati e stima l'optical flow calcolando la differenza tra questi modelli. Inoltre, l'algoritmo utilizza una piramide gaussiana per gestire le scale spaziali dell'immagine, consentendo di considerare il flusso ottico a diverse risoluzioni. È stato scelto di utilizzare come metodo di estrazione di feature l'optical flow dopo aver visionato il seguente paper "Visual Speech Recognition Using Optical Flow and Hidden Markov Model"[5].

4.4.1 Utilizzo optical flow su Babele

Il paper precedentemente citato si concentra sull'estrazione delle feature utilizzando esclusivamente video in cui è presente la zona labiale. La scelta di utilizzare solo la zona labiale è dovuta al funzionamento dell'algoritmo di optical flow. Questo sistema rileva gli spostamenti dei pixel all'interno di un'immagine, ma l'array di spostamenti in output rappresenta soltanto l'oggetto con il maggior movimento tra i due frame forniti in input.

Nel nostro sistema, abbiamo preso il dataset di Babele che contiene solo le immagini delle labbra e abbiamo applicato la funzione di optical flow fornita dalla libreria OpenCV: "calcOpticalFlowFarneback()". Questa funzione richiede diversi parametri in input:

- **prev_frame**: È il frame precedente (o l'immagine precedente) dalla sequenza di input. Deve essere un'immagine in scala di grigi (un singolo canale);
- **next_frame_gray**: È il frame successivo (o l'immagine successiva) dalla sequenza di input. Anch'esso deve essere un'immagine in scala di grigi;
- **prev_flow**: Questo parametro può essere utilizzato per fornire un'ipotesi iniziale del flusso ottico. Se non si dispone di una stima iniziale, si può passare None;
- **pyr_scale**: È un parametro che specifica la scala dell'immagine per la piramide a livelli multipli. Di solito è impostato a 0.5, il che significa che ogni livello della piramide successivo è la metà delle dimensioni del livello precedente;
- **levels**: Rappresenta il numero di livelli nella piramide a livelli multipli utilizzata per il calcolo. Un numero tipico di livelli è 3;
- **winsize**: Specifica la dimensione della finestra di media mobile utilizzata per approssimare la derivata spaziale. Un valore comune è 15;
- **iterations**: Rappresenta il numero di iterazioni dell'algoritmo di ottimizzazione per ogni livello della piramide. Di solito si usa 3;
- **poly_n**: È il grado del polinomio espanso per approssimare la distribuzione dei pixel circostanti. Un valore tipico è 5;
- **poly_sigma**: Specifica la deviazione standard del kernel gaussiano usato per approssimare le derivate spaziali. Un valore comune è 1.2;
- **flags**: Sono dei flag opzionali che possono essere utilizzati per modificare il comportamento dell'algoritmo. In questo caso, il valore 0 indica l'uso delle impostazioni predefinite.

Successivamente abbiamo calcolato l'optical flow in due modi:

- **Metodo 1**: viene estratta una sequenza di 240 frame dal video. L'algoritmo di optical flow viene applicato alla coppia di frame consecutivi (frame0 e frame1), ottenendo un array contenente gli spostamenti dei pixel, viene successivamente applicato a tutte le coppie successive di frame (frame1 e frame2, frame2 e frame3, ..., frame238 e frame239), ottenendo 240 array in totale. Quindi, per ogni video, si avranno 240 array di spostamenti dei pixel corrispondenti alle diverse coppie di frame.;
- **Metodo 2**: viene estratta una sequenza di 240 frame dal video. I frame vengono raggruppati in array di 24 frame ciascuno, ottenendo quindi 10 array. L'algoritmo di optical flow viene applicato a ciascun array di 24 frame, ottenendo un unico vettore di spostamenti dei pixel per

ogni array. Quindi, per ogni video, si avranno 10 array contenenti un vettore di spostamenti dei pixel corrispondente a ciascun gruppo di 24 frame.

Invece di utilizzare direttamente gli array di spostamenti dei pixel, sono state calcolate le medie orizzontali e verticali degli array. Di conseguenza, per ogni frame del video, sono state estratte solo due feature:

1. Per il primo metodo descritto, in cui vengono estratti 240 frame e l'algoritmo di optical flow viene applicato alle coppie di frame consecutivi, si ottengono 2 x 240 feature per ogni video;
2. Nel secondo metodo, in cui vengono estratti 240 frame e raggruppati in array di 24 frame ciascuno, si ottengono 2 x 10 feature per ogni video.

Dopo aver ottenuto i dataset dai due metodi descritti, li abbiamo utilizzati come input per l'addestramento della PNN, e come per il caso in cui abbiamo utilizzato la matrice codifica sparsa, non abbiamo ottenuto un miglioramento delle prestazioni, ma anzi abbiamo riscontrato un peggioramento. Questo risultato potrebbe essere attribuito all'introduzione di un elevato livello di rumore nei dati.

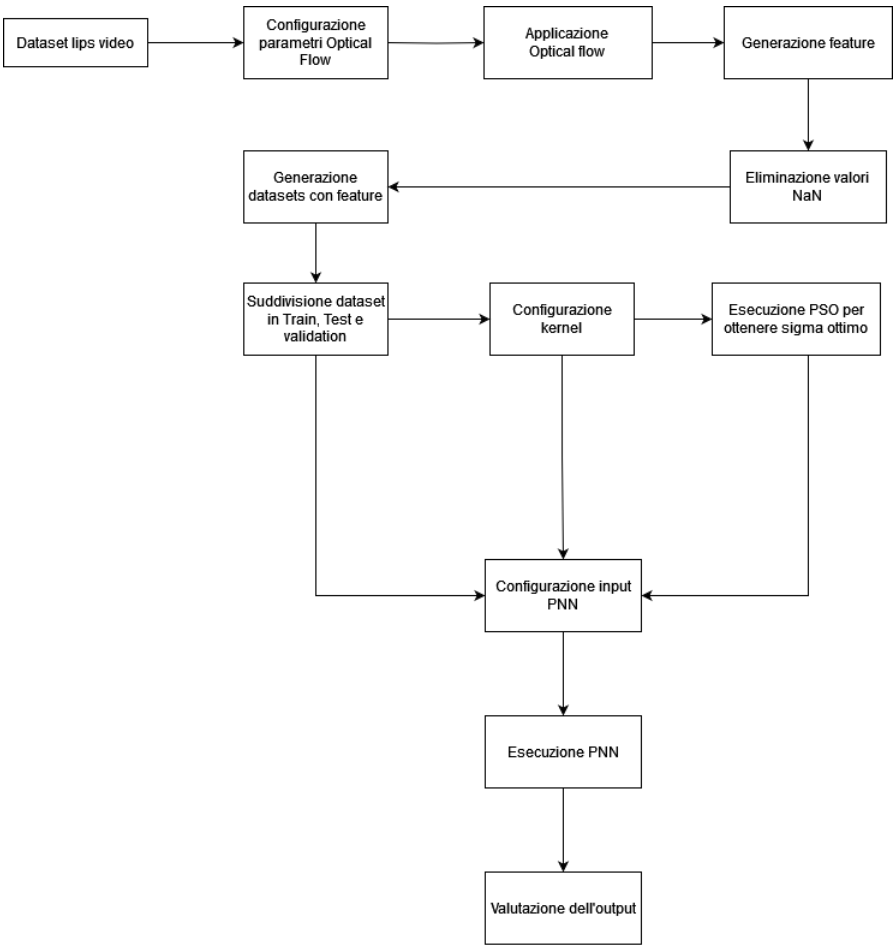

Figura 4.5: Schema preprocessing utilizzando optical flow 240 frame

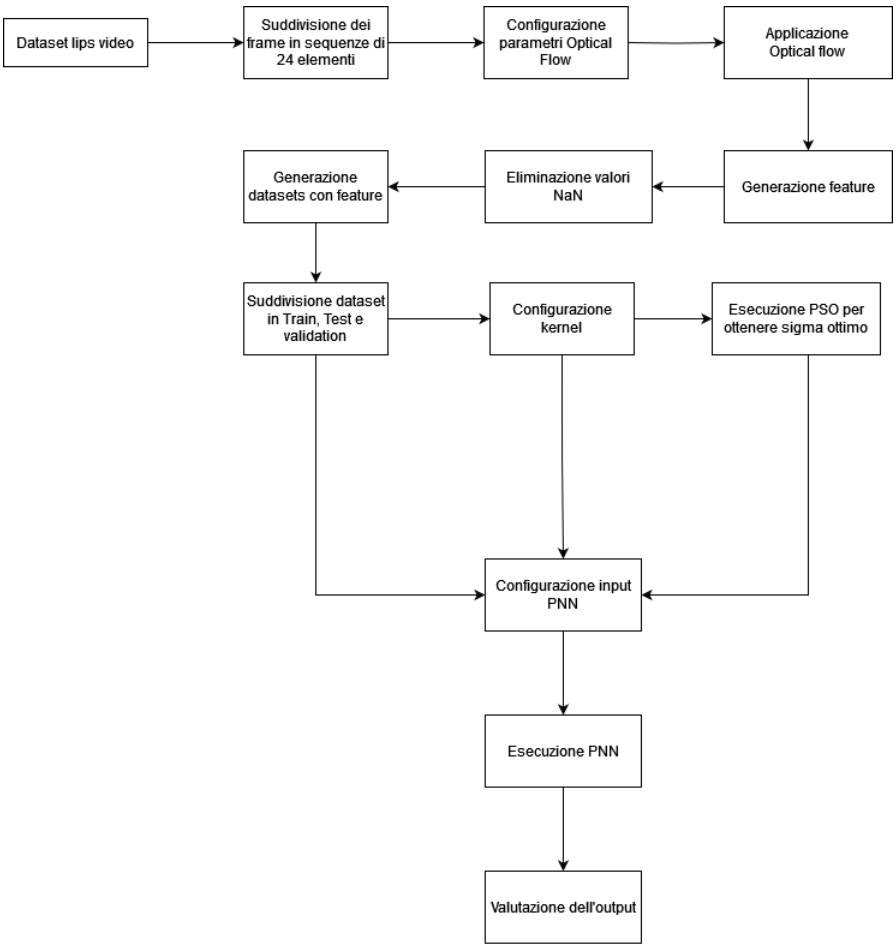

Figura 4.6: Schema preprocessing utilizzando optical flow con sequenze di 24 frame

CAPITOLO 5

PNN

5.1 PNN: aspetti generali

La Probabilistic Neural Network (PNN) è un modello di rete neurale descritto da D.F. Specht in [6], basato sul metodo di approssimazione della densità di probabilità Parzen Windows [7], che si concentra principalmente sulla classificazione dei dati.
Risulta avere un'architettura *feedforward*, le informazioni si muovono in una sola direzione, in avanti, dai neuroni di input ai neuroni di output. Ciò significa che ogni strato di neuroni riceve input solo dallo strato precedente e alimenta output solo allo strato successivo. Non ci sono cicli o collegamenti all'indietro nella rete, quindi l'informazione fluisce in una direzione determinata e non vi è alcuna retroazione o modifica del segnale in ingresso una volta che è stato elaborato da uno strato.
Uno dei suoi principali vantaggi rispetto ad altre tecniche di apprendimento automatico è la sua velocità di addestramento, che è stata riportata essere circa 200.000 volte più veloce della retropropagazione per un livello di prestazioni equivalente.

5.1.1 Parzen Windows

Il metodo delle finestre di Parzen (anche conosciuto come metodo delle finestre di Parzen-Rosenblatt) è una tecnica non parametrica ampiamente utilizzata per stimare la densità di probabilità di una variabile casuale continua a partire da un insieme di dati campionati. La finestra, nel pratico, può essere vista come un'area attorno al punto di interesse, e la sua forma appunto dipende dalla scelta della funzione kernel utilizzata.

Parametri critici della Parzen Windows

I due parametri critici nelle tecniche delle finestre di Parzen sono:

1. **la larghezza della finestra**: parametro che controlla la larghezza della finestra utilizzata per stimare la densità di probabilità. In pratica, rappresenta la dimensione della finestra utilizzata per calcolare la densità di probabilità in un determinato punto. Se la larghezza di banda è troppo piccola, la stima della densità di probabilità sarà molto sensibile ai dettagli nei dati, ma potrebbe essere influenzata dal rumore. Se la larghezza di banda è troppo grande, la stima della densità di probabilità sarà meno sensibile ai dettagli nei dati, ma potrebbe mancare di precisione.

2. **il kernel**: è la funzione utilizzata per definire la finestra di Parzen. Deve essere una funzione continua, non negativa e integrabile, con un'area totale uguale a 1.

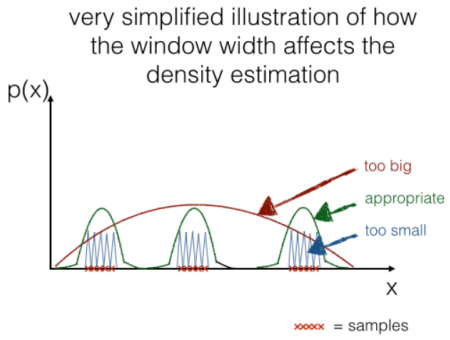

Figura 5.1: Come la larghezza della finestra può influenzare la stima di densità

5.1.2 Struttura base della rete

La struttura e l'architettura della PNN rappresentata nella Figura 5.2 comprende quattro strati:

- **Input Layer**: questo strato accetta in input le caratteristiche dei dati da classificare. Le unità di input sono indipendenti tra loro e forniscono gli stessi valori di input a tutte le unità dello strato di pattern.

- **Pattern Layer**: questo strato è composto da neuroni che calcolano la somiglianza tra l'input e il campione di addestramento. La somiglianza viene calcolata come la distanza euclidea tra i due vettori di input, divisa per il parametro di larghezza del kernel (sigma) e quindi trasformata dalla funzione kernel in una misura di somiglianza. Il parametro di smoothing (sigma) viene utilizzato nel kernel per regolare l'effetto della distanza sui punti vicini e lontani dal punto di interesse *(vedi sez. Parametri critici della Parzen Windows)*. Il valore di somiglianza così ottenuto viene utilizzato come output del neurone corrispondente. Il numero di neuroni in questo strato è uguale al numero di campioni di addestramento.

- **Summation Layer**: in questo strato per ogni classe di addestramento, viene calcolata la media delle somiglianze tra l'input e i campioni di addestramento associati a quella classe, pesata dai valori di somiglianza calcolati dai neuroni del Pattern Layer; le medie pesate successivamente sono date in input al livello successivo.

- **Output Layer**: questo strato confronta le medie ponderate accumulate nel livello di Summation per ogni categoria e utilizza la media più grande per prevedere la categoria di appartenenza dell'input.

5.1.3 Training della rete

La fase di training nella PNN c'è, ma la procedura è differente rispetto ad altre reti neurali. Nella PNN, l'addestramento è "one-shot", ovvero richiede solo un passaggio attraverso il set di addestramento per calcolare i pesi nell'Output Layer. Ciò significa che la PNN non richiede un addestramento iterativo come altre reti neurali, come ad esempio la retropropagazione.
Durante l'addestramento della PNN, per ogni nuovo pattern di addestramento, viene aggiunta un'unità di pattern e i suoi pesi di input vengono adattati in modo che coincidano con il pattern stesso. Ciò significa che la PNN impara le caratteristiche del singolo pattern di addestramento e le utilizza per la classificazione dei pattern di test.

5.2 PNN utilizzata nel progetto

5.2.1 Implementazione scelta

Durante la fase di implementazione del sistema di classificazione, sono state considerate tre opzioni di implementazione della PNN per il nostro caso di studio. Di seguito, elenchiamo le caratteristiche principali di ciascuna opzione:

- La **prima opzione** di implementazione della PNN era facile da comprendere, ma durante le prove si è dimostrata difficile da adattare al caso di studio;

- La **seconda opzione** [8] di implementazione della PNN si basava su una funzione kernel e un parametro di larghezza del kernel (sigma) personalizzabili, il che la rendeva facilmente comprensibile ed adattabile al nostro problema in questione;

- La **terza opzione** [9] di implementazione della PNN era facile da comprendere ed adattabile al nostro problema, ma si è rivelata essere meno modificabile. Infatti, questa implementazione utilizzava una funzione kernel fissa e un parametro di larghezza del kernel (sigma) predefinito.

Complessivamente, la seconda implementazione della PNN ci ha fornito una soluzione robusta ed efficiente per il nostro problema di classificazione. La sua flessibilità e facilità d'uso ci hanno permesso di testare e ottimizzare rapidamente le prestazioni della rete, ottenendo risultati migliori rispetto alle altre opzioni considerate. Riteniamo che la nostra scelta di questa implementazione possa fornire utili spunti per futuri studi in questo campo.

5.3 Scelta del kernel della PNN

Nel nostro caso come kernel abbiamo testato varie funzioni:

- **Kernel uniforme**: assegna lo stesso peso a tutti i punti all'interno di una determinata distanza. Quindi, se un punto si trova entro un raggio specificato, riceve un contributo pari al peso uniforme. Al di fuori di tale raggio, il contributo è zero. Questo kernel è caratterizzato da una forma rettangolare

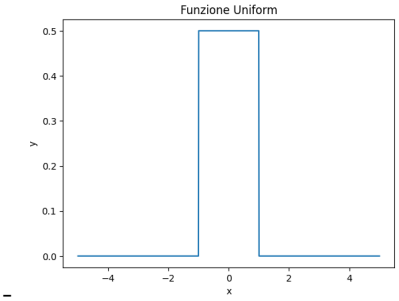

Figura 5.2: Grafico kernel uniforme

- **Kernel triangolare**: assegna un peso crescente ai punti all'interno di una distanza specificata rispetto a un punto centrale dove raggiunge il valore massimo. Dopo di che, il peso diminuisce in modo simmetrico. Assume quindi una forma triangolare

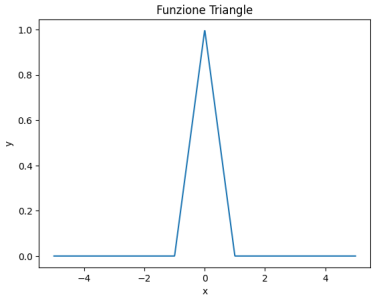

Figura 5.3: Grafico kernel triangolare

- **Kernel gaussiano**: assegna un peso decrescente ai punti in base alla loro distanza da un punto centrale, seguendo una curva gaussiana. Il punto centrale ha il peso massimo, mentre i punti

lontani hanno un peso sempre minore. Questo kernel produce una distribuzione di probabilità continua, simmetrica e a campana

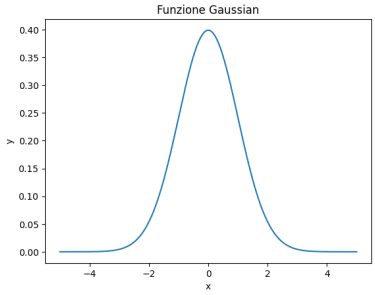

Figura 5.4: Grafico kernel gaussiana

- **Kernel laplaciano**: assegna un peso decrescente ai punti in base alla loro distanza da un punto centrale, seguendo una curva simile a quella della distribuzione laplaciana. Questo kernel è caratterizzato da una forma a picco con code pesantemente decrescenti. È spesso utilizzato per evidenziare cambiamenti bruschi o discontinuità nei dati

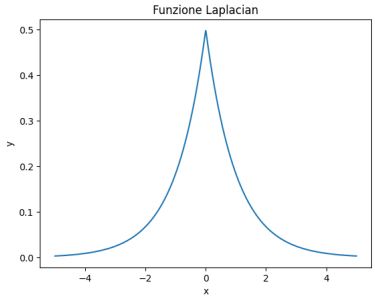

Figura 5.5: Grafico kernel laplaciana

- **Kernel Epanechnikov**: assegna un peso crescente ai punti all'interno di una distanza specificata rispetto a un punto centrale dove raggiunge il valore massimo. Dopo di che, il peso diminuisce in modo simmetrico allontanandosi dal punto centrale, seguendo una curva parabolica

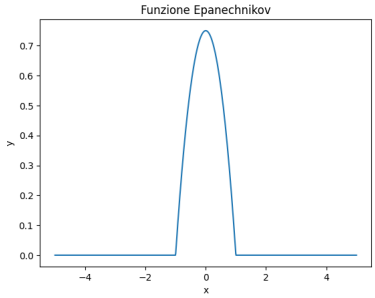

Figura 5.6: Grafico kernel Epanechnikov

Dopo una serie di test e sperimentazioni, è stato determinato che i kernel migliori per l'analisi dei dati sono quelli basati sulla funzione gaussiana e sulla funzione di Epanechnikov. Entrambi i kernel hanno dimostrato di avere prestazioni simili durante le prove. Tuttavia, si è deciso di utilizzare esclusivamente il kernel gaussiano poiché in alcuni casi ha mostrato prestazioni superiori rispetto all'altro kernel.

5.4 Scelta del sigma ottimo

Il parametro sigma controlla la larghezza delle funzioni di base utilizzate per stimare le densità di probabilità condizionate. La scelta corretta di sigma è importante per ottenere una modellazione accurata delle distribuzioni di probabilità dei dati.

In tutti i test che abbiamo eseguito per capire su quanti frame basarci e su quale kernel utilizzare ci siamo basati su un range di valori di sigma che andava da 0.05 a 1.4, e verificare poi da noi quale kernel per quale sigma aveva prestazioni migliori. Ci siamo chiesto quindi se esistesse un modo per trovare efficientemente il sigma ottimo senza eseguire ogni volta la PNN. Leggendo attentamente il paper "Using a Probabilistic Neural Network for lip-based biometric verification"[3] abbiamo notato che si faceva accenno ad una tecnica chiamata Particle Swarm Optimization (PSO) per trovare efficientemente il valore del sigma ottimo.

5.4.1 Particle Swarm Optimization

L'algoritmo PSO è un algoritmo di ottimizzazione basato sul concetto di intelligenza di sciame, che si ispira al comportamento di un gruppo di individui in natura, come uno stormo di uccelli o uno sciame di api, che collaborano per trovare la migliore soluzione possibile per un determinato problema di ottimizzazione. PSO utilizza una popolazione di particelle che rappresentano le potenziali soluzioni del problema di ottimizzazione. Ogni particella tiene traccia della sua posizione nel dominio del problema e della sua velocità corrente. Durante l'esecuzione di PSO, le particelle comunicano tra loro per scambiare informazioni sulle soluzioni migliori trovate fino a quel momento. In base a queste informazioni, ogni particella aggiorna la propria velocità e posizione utilizzando una combinazione di conoscenza individuale e conoscenza sociale.
L'obiettivo di PSO è di ottimizzare una funzione obiettivo, cercando di trovare la soluzione migliore o la migliore combinazione di parametri all'interno dello spazio di ricerca specificato. In particolare, nel nostro caso, abbiamo utilizzato PSO per trovare il valore ottimale del sigma, utilizzando il kernel Gaussiano, che massimizzino l'accuratezza della PNN sul nostro set di dati. Nel dettaglio, abbiamo utilizzato la funzione *single.GlobalBestPSO* della libreria *pyswarms* per eseguire l'algoritmo PSO e trovare il valore ottimale del sigma.

N.B: è importante considerare che ci possono essere più di un valore ottimale del sigma in base alla definizione dell'obiettivo di ottimizzazione e alle caratteristiche del dataset.

5.4.2 Schema generale

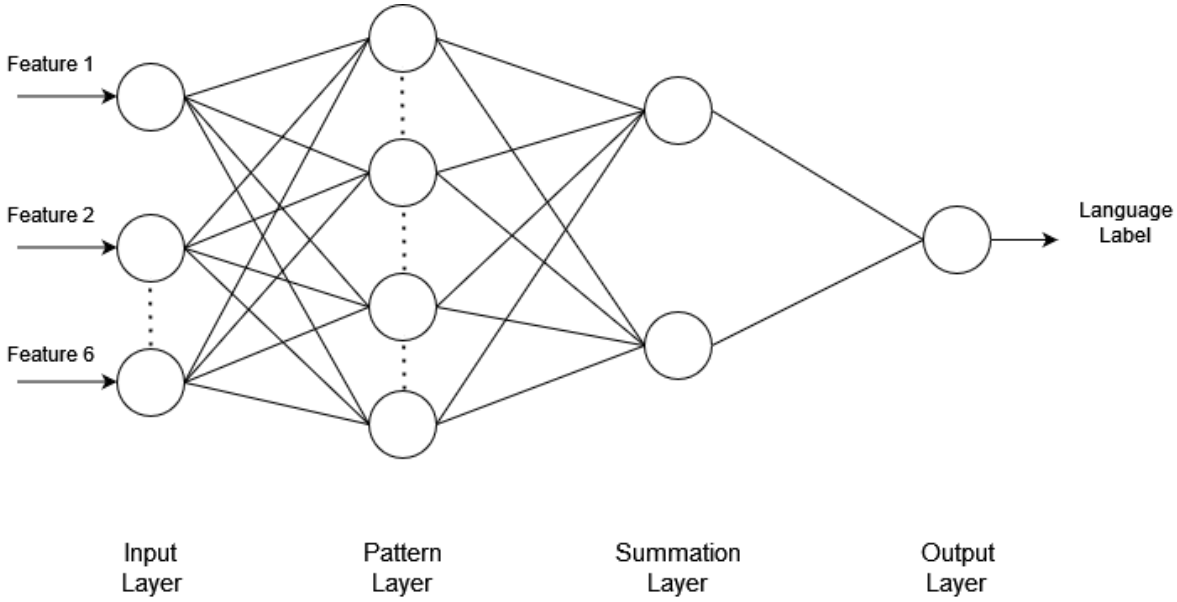

Figura 5.7: Schema rete PNN sul nostro problema

CAPITOLO 6

Risultati Ottenuti

In questo capitolo verranno analizzati e commentati i risultati ottenuti per tutte le prove che sono state effettuate. Tutti i test sono stati eseguiti con la medesima configurazione della PNN che, dopo le varie prove di cui sopra, è risultata essere quella ottimale.

6.1 BABELE - Features

Mostreremo per prima i risultati conseguiti utilizzando il dataset Babele preprocessato tramite l'estrazione delle features delle labbra. I test sono stati effettuati utilizzando la stessa configurazione di preprocessing Numero di frame - Landmark - Distanza (cap. 4) in modo da ottenere risultati confrontabili. In partiolare La PNN è stata utilizzata per classificare le lingue sia in modalità "MondoVsMondo" che in modalità "xVsMondo". In "MondoVsMondo" sono state valutate tutte le lingue passando quindi alla PNN tutte le varie lingue ognuna con un'etichetta diversa e osservandone le predizioni. In "xVsMondo" è stato valutato il riconoscimento di una lingua per volta rispetto a tutte le altre.

6.1.1 BABALE - MondoVsMondo

Per quanto riguarda MondoVsMondo i risultati ottenuti sono visualizzabili tramite la seguente matrice di confusione:

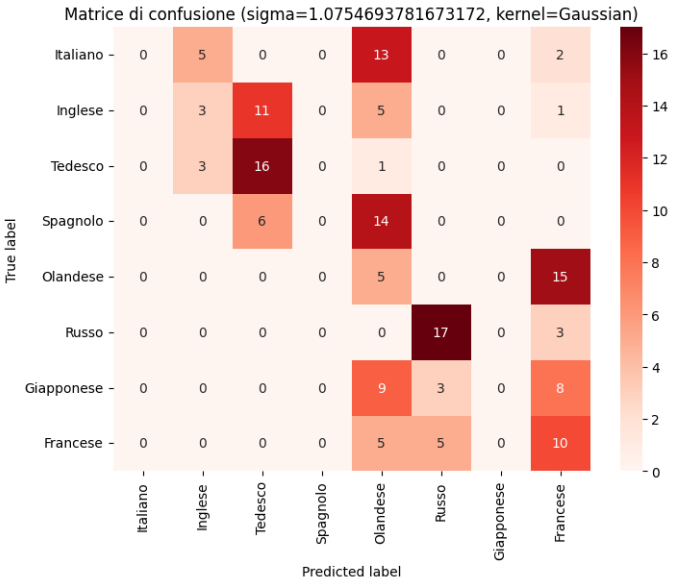

Figura 6.1: Matrice di confusione BABELE - MondoVsMondo

Tramite la matrice di confusione è possibile notare come la PNN fa molta confusione tra diverse lingue come ad esempio tra Francese e Olandese mentre le lingue che vengono riconosciute meglio risultano essere il Tedesco e il Russo. In questo caso l'accuracy è risultata essere del 32.5%.

6.1.2 BABALE - FranceseVsMondo

Per quanto riguarda il riconoscimento della lingua francese i risultati ottenuti sono i seguenti:

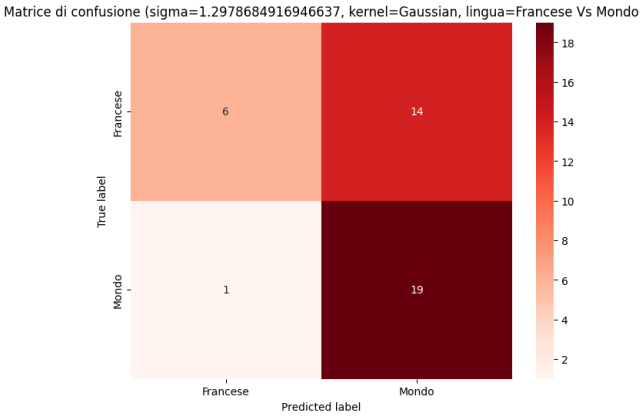

Figura 6.2: Matrice di confusione BABELE - FranceseVsMondo

Per quanto riguarda la lingua Francese è possibile notare dalla figura sopra essa viene molto spesso confusa ottenendo infatti un numero elevati di falsi positivi, l'accuracy infatti è stata del: 62.5%.

6.1.3 BABALE - GiapponeseVsMondo

Per il riconoscimento del Giapponese i risultati sono apprezzabili osservando la seguente figura:

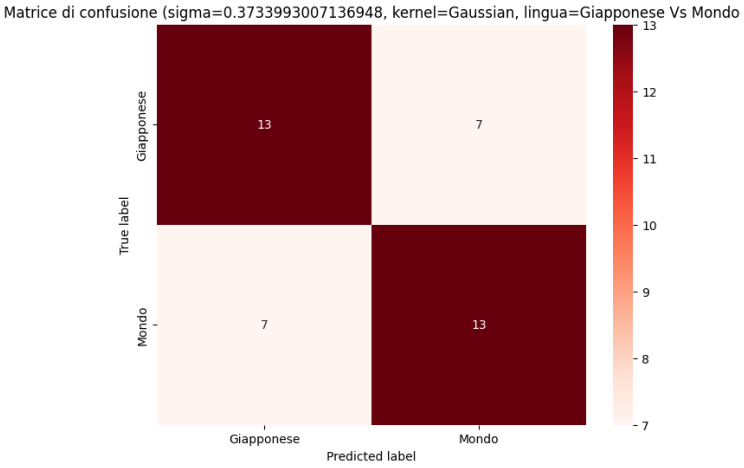

Figura 6.3: Matrice di confusione BABELE - GiapponeseVsMondo

La matrice di confusione del Giapponese evidenzia come ci sia un equilibrio tra falsi positivi e falsi negativi che risultano essere più della metà dei veri positivi e veri negativi il ché porta l'accuracy ad essere del 65%.

6.1.4 BABALE - RussoVsMondo

Eseguendo la PNN in modalità RussoVsMondo sono stati ottenuti i seguenti risultati:

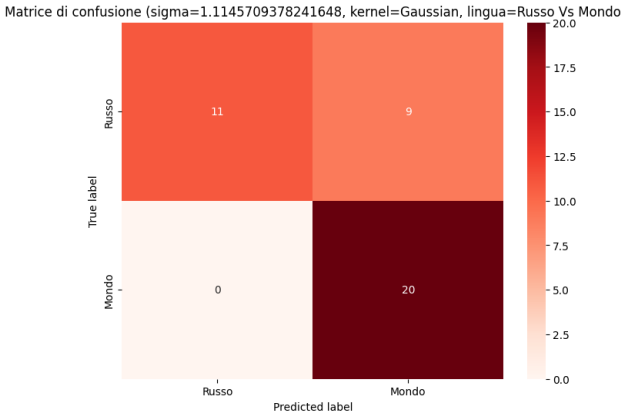

Figura 6.4: Matrice di confusione BABELE - RussoVsMondo

È possibile notare come le altre lingue non vengono mai confuse con il Russo tuttavia nel riconoscimento del Russo esso viene spesso confuso con altre lingue ottenendo un accuracy del 77.5%.

6.1.5 BABALE - OlandeseVsMondo

Per il riconoscimento della lingua Olandese i risultati possono essere acquisiti tramite la seguente matrice di confusione:

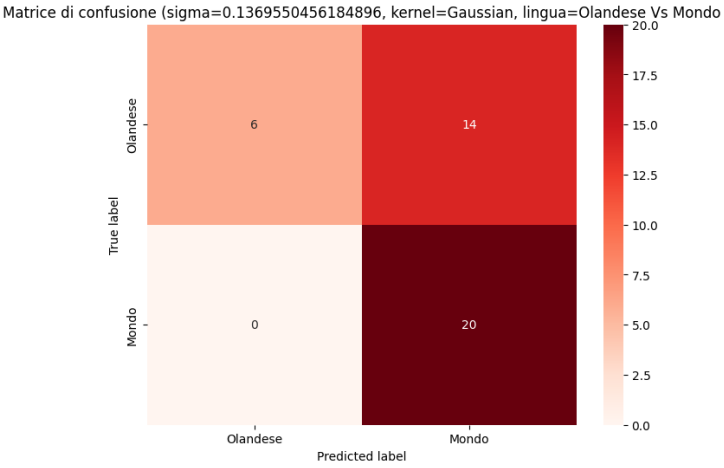

Figura 6.5: Matrice di confusione BABELE - OlandeseVsMondo

Tale matrice evidenzia come il numero dei falsi positivi è addirittura doppio rispetto alle predizioni corrette mentre i falsi negativi a 0 fa risalire l'accuracy fino al 65%.

6.1.6 BABALE - SpagnoloVsMondo

Eseguendo la PNN per riconoscere lo Spagnolo sono stati ottenuti i seguenti risultati:

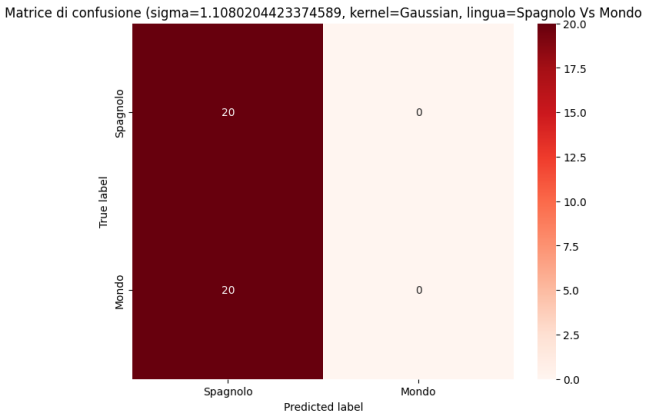

Figura 6.6: Matrice di confusione BABELE - SpagnoloVsMondo

La matrice di confusione per lo spagnolo evidenzia come esso viene sempre riconosciuto tuttavia possiamo vedere come c'è un numero altissimo di falsi negativi il che fa pensare che la rete confonde tutte le lingue con lo spagnolo infatti l'accuracy è del 50% che, confrontata con i risultati delle altre lingue, risulta essere molto bassa.

6.1.7 BABALE - ItalianoVsMondo

I risultati ottenuti valutando l'Italiano sono apprezzabili analizzando la seguente matrice di confusione:

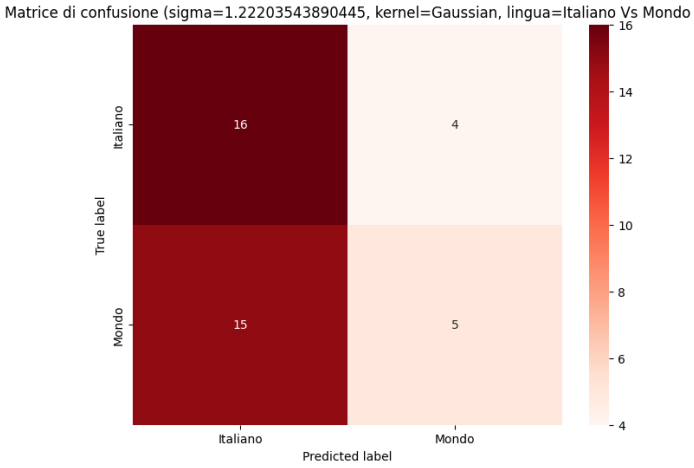

Figura 6.7: Matrice di confusione BABELE - ItalianoVsMondo

Tale matrice è molto simile rispetto a quella dello Spagnolo (leggermente migliore) e questo è un risultato che ci aspettavamo in quanto è possibile notare una innegabile somiglianza tra le due lingue in questione. Di conseguenza anche l'accuracy è simile a quella ottenuta con lo Spagnolo infatti essa risulta essere del 52.5%.

6.1.8 BABALE - TedescoVsMondo

Per quanto riguarda il tedesco la matrice di confusione ottenuta è la seguente:

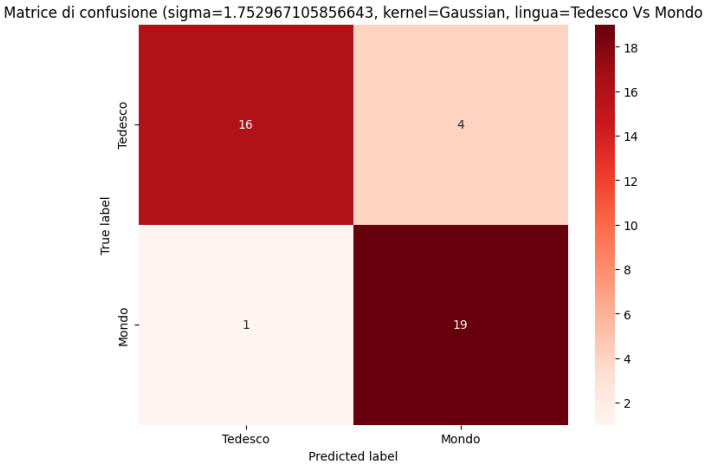

Figura 6.8: Matrice di confusione BABELE - TedescoVsMondo

Possiamo notare come i risultati ottenuti per il Tedesco sono i migliori in assoluto. Infatti tale lingua risulta essere quella meglio riconosciuta e molto difficilmente confusa raggiungendo quindi un accuracy dell'87.5%.

6.1.9 BABALE - IngleseVsMondo

Eseguendo la rete in modalità IngleseVsMondo sono stati ottenuti i seguenti risultati:

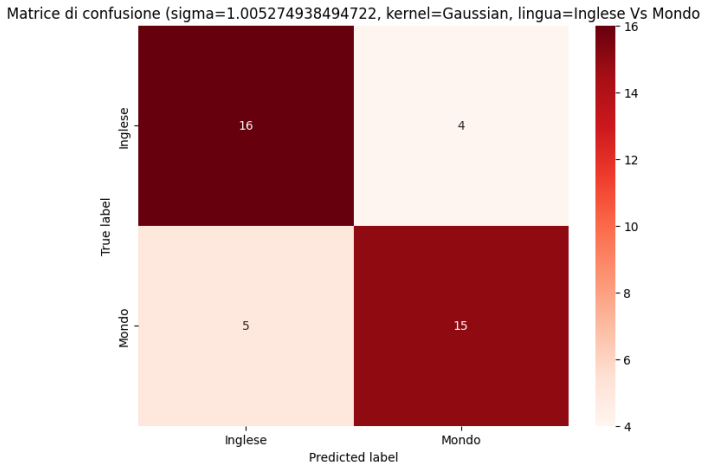

Figura 6.9: Matrice di confusione BABELE - IngleseVsMondo

Tale matrice di confusione evidenzia come anche l'Inglese viene riconosciuto abbastanza bene. Infatti, dopo il Tedesco, l'Inglese è la lingua meglio riconosciuta seppur viene confusa un numero di

volte leggermente superiore. Tali risultati vengono confutati dall'accuracy che è risultata essere del 77.5%.

6.2 BABELE - CODIFICA SPARSA

Per quanto riguarda il preprocessing tramite codifica sparsa i risultati ottenuti sono i seguenti:

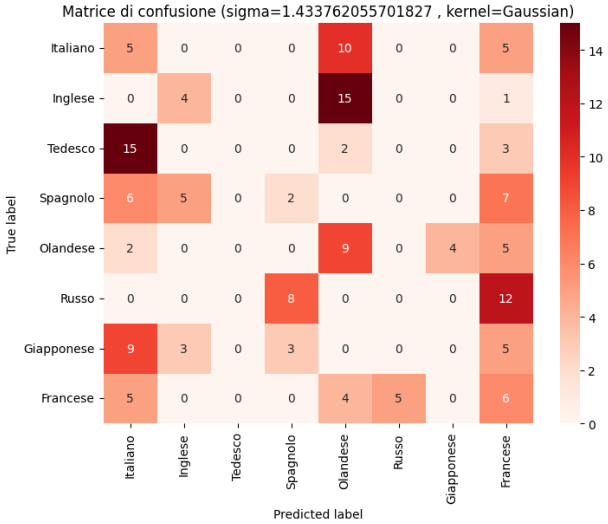

Figura 6.10: Matrice di confusione BABELE - codifica sparsa

Come si può evincere dalla matrice di confusione i risultati ottenuti sono tutt'altro che buoni infatti le lingue vengono spesso confuse tra di loro con un accuracy che si ferma solo al 16%. Probabilmente un risultato così basso è dovuto al fatto che la codifica sparsa prende in input tutta la zona labiale che include quindi altre parti del volto oltre alle labbra che nel nostro problema potrebbero rappresentare del rumore che si traduce in un peggioramento dell'accuracy.

6.3 BABELE - CODIFICA SPARSA + PCA

L'applicazione di codifica sparsa e PCA in modo da ridurre la dimensione dei dati senza perdita di informazione è stata effettuata su 120 componenti ed è possibile visualizzare i risultati tramite la seguente matrice di confusione:

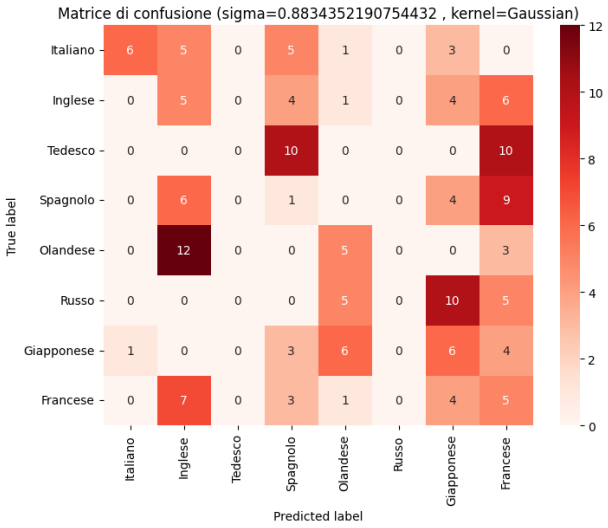

Figura 6.11: Matrice di confusione BABELE - codifica sparsa + PCA

Come si può notare da tale matrice i risultati risultano essere molto simili a quelli ottenuti con la sola applicazione dell'codifica sparsa confermando quindi l'inadeguatezza di tale algoritmo per il nostro problema e marcando invece la bontà dell'algoritmo PCA.L'accuracy in questo caso è stata del 17%.

6.4 BABELE - CODIFICA SPARSA + FEATURES

L'ultima prova effettuata con codifica sparsa consiste in una combinazione di SVD ed estrazione delle features:

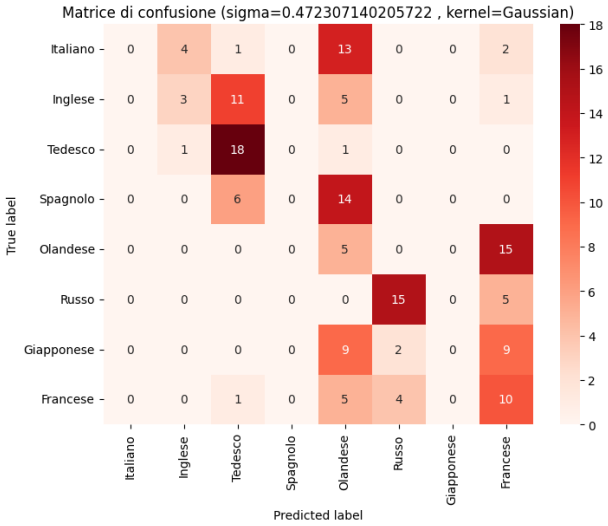

Figura 6.12: Matrice di confusione BABELE - codifica sparsa + FEATURES

In questo caso la PNN produce dei risultati nettamente migliore a riprova ancora una volta che per il nostro problema il preprocessing tramite estrazione delle features determina risultati nettamente migliori della codifica sparsa. L'accuracy in questo caso è risultata essere del 32%.

6.5 BABELE - OPTICAL FLOW

Come già detto in precedenza per quanto riguarda optical flow esso è stato applicato sia partendo da coppie di frame che raggruppando i frame in array di 24 frame. La prima prova è stata dunque effettuata utilizzando optical flow su coppie di frame e i risultati ottenuti sono i seguenti:

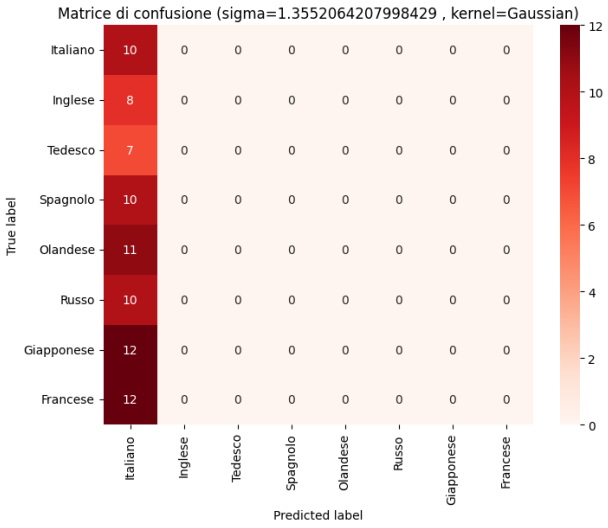

Figura 6.13: Matrice di confusione BABELE - OPTICAL FLOW COPPIE

Dalla matrice di confusione è possibile notare come i risultati ottenuti siano pessimi infatti tale prova ha fatto registrare l'accuracy più bassa che si è fermata solo al 12%.
Successivamente sono stati ricalcolati i risultati utilizzando optical flow su array di 24 frame e si possono analizzare i risultati dalla seguente matrice di confusione:

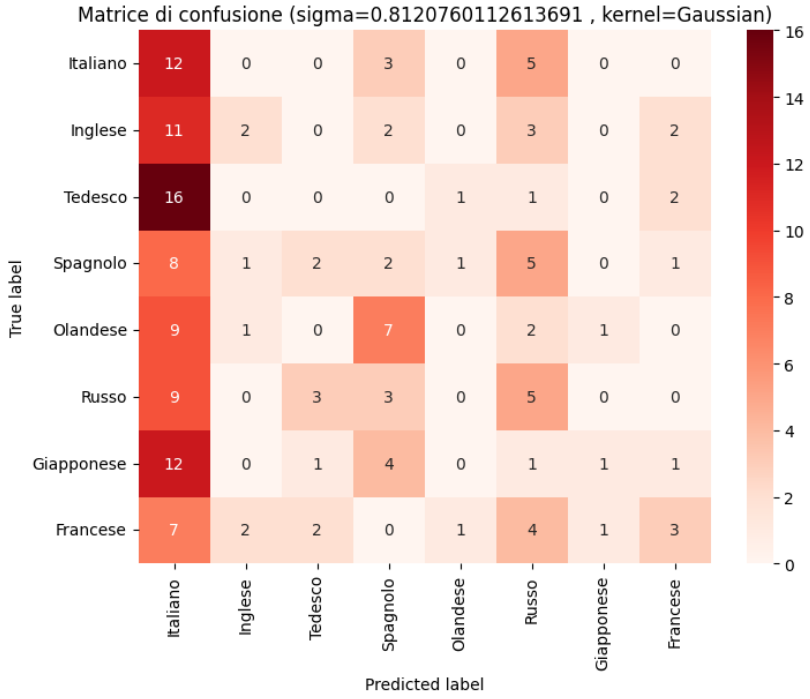

Figura 6.14: Matrice di confusione BABELE - OPTICAL FLOW ARRAY

Tale matrice evidenzia come i risultati ottenuti, seppur migliori del test precedente, non siano buoni infatti spesso le lingue vengono confuse tra di loro il che porta l'accuracy al 16%.

6.6 CH-SIM CMU-MOSEI

In questa sezione verranno analizzati i risultati ottenuti utilizzando i dataset CH-SIM e CMU-MOSEI contenenti video in lingua Inglese e Cinese. Anche in questo caso tali risultati sono stati ottenuti con la medesima configurazione utilizzata per BABELE che è risultata essere quella ottimale. I risultati possono essere apprezzati tramite la seguente matrice di confusione:

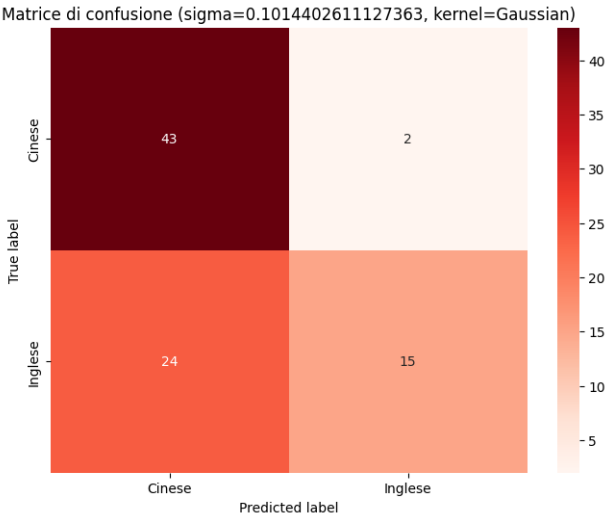

Figura 6.15: Matrice di confusione BABELE - CH-SIM CMU-MOSEI

Analizzando tale matrice è possibile notare come il cinese viene riconosciuto mediamente bene ma spesso confuso con l'Inglese. Quest'ultimo invece viene ben riconosciuto e molto raramente confuso con il Cinese. L'accuracy ottenuta in questo caso è risultata essere del 69%.

CAPITOLO 7

Conclusioni

Dopo aver effettuato diverse prove ed eseguito la PNN su diversi input è possibile affermare che tale rete ottiene risultati simili a quelli ottenuti da altre reti per il riconoscimento della lingua a partire dai movimenti delle labbra. Tuttavia essendo la PNN un concetto meno esplorato di altri, il lavoro svolto ha ampiamente raggiunto tutti gli obiettivi preposti che erano quelli di esplorare il problema e la rete in maniera più ampia possibile in modo da poter osservare in quali condizioni e con quali input la PNN si comporta meglio nell'ambito del problema in questione. Riteniamo che tutte le prove effettuate sia per il preprocessing che per la configurazione della rete possano risultare preziose per degli sviluppi futuri. Il lavoro svolto infatti, vuole sicuramente essere anche uno spunto per eventuali lavori futuri che, a questo punto, potrebbero partire sia con una conoscenza migliore del problema che soprattutto con una conoscenza migliore delle PNN e delle correlazioni che intercorrono tra il problema illustrato e questo tipo di reti.

Bibliografia

[1] K. B. Ovhal, S. S. Patange, R. S. Shinde, V. K. Tarange, and V. A. Kotkar, "Analysis of anomaly detection techniques in video surveillance," in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 596–601, 2017.

[2] J.-Y. Lai, S.-L. Wang, X.-J. Shi, and A. W.-C. Liew, "Sparse coding based lip texture representation for visual speaker identification," in *2014 19th International Conference on Digital Signal Processing*, pp. 607–610, 2014.

[3] K. Wrobel, R. Doroz, P. Porwik, J. Naruniec, and M. Kowalski, "Using a probabilistic neural network for lip-based biometric verification," *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 112–127, 2017.

[4] C. A. R. Hoare, "Quicksort," *The Computer Journal*, vol. 5, pp. 10–16, 01 1962.

[5] U. Sharma, S. Maheshkar, A. Mishra, and R. Kaushik, "Visual speech recognition using optical flow and hidden markov model," *Wireless Personal Communications*, vol. 106, 06 2019.

[6] "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[7] E. Parzen, "On Estimation of a Probability Density Function and Mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065 – 1076, 1962.

[8] R. Gupta, "Bayesian networks - probabilistic neural network (pnn)," Apr 2023.

[9] Antoniogrv, "Antoniogrv/probabilistic-neural-network: Implementation of a generic-purpose pnn.."