Identificazione attraverso labiale della lingua parlata

Giuseppe Genito, Rosaria Leone

11 febbraio 2025

Indice

Sommario

Questo paper presenta il lavoro di classificazione della lingua parlata dal solo movimento delle labbra, (senza l'ausilio dell'audio), attraverso l'uso del modello ConvLSTM2D e della Feature Level Fusion.

1 Introduzione

Nel presente lavoro, lo scopo è stato quello di capire mediante il solo movimento delle labbra, senza l'ausilio di una traccia audio, quale fosse la lingua parlata dalla persona; in particolare i soggetti potevano essere classificati nelle seguenti 8 lingue: italiano, inglese, tedesco, spagnolo, olandese, russo, giapponese e francese. Il riconoscimento della lingua è stato effettuato grazie all'addestramento di una rete neurale nota come ConvLSTM2D su un dataset di training chiamato BABELE, Successivamente, sono stati utilizzati classificatori addestrati sui dati dei landmark del movimento delle labbra per ottenere le migliori accuratezze. Le previsioni dei classificatori sono state unite con quelle della rete neurale utilizzando la tecnica di features level fusion per creare un nuovo dataset, con il quale sono stati riaddestrati i classificatori per raggiungere accuratezze più elevate. Lo stesso processo è stato ripetuto utilizzando un altro dataset chiamato CH-SIMS CMU-MOSEI, in cui le lingue da classificare sono limitate a inglese e cinese. Nei prossimi paragrafi verrà fornita una descrizione dettagliata di ogni fase del processo, evidenziando le scelte effettuate e le motivazioni adottate.

2 Dataset pre-processato in canale RGB

Sono state create due versioni del lavoro, una utilizzando il canale di colore RGB e l'altra in scala di grigi. Esamineremo innanzitutto la configurazione a colori.

2.1 Fase di preprocessing

È stata effettuata una fase di preprocessing al fine di generare gli input appropriati per la rete. È stato preso in considerazione il dataset presente nella cartella *"Lips_video_10_seconds_division_train_test_val split_subject_indipendent"* del drive, il quale include i set di Validazione, Train e Test.
Suddivisi come segue:

- Train: 960 video
- Validation: 160 video
- Test: 160 video

Ognuno di questi video è composto da circa 300/350 frame, in formato ".avi" con i canonici tre canali di colore RGB. In primis vengono letti i file CSV:

- file_train.csv
- file_val.csv
- file_test.csv

Essi contengono i nomi dei video, dove il primo carattere del nome indica la lingua parlata. Nel passaggio successivo dell'algoritmo di pre-processing, vengono generate tre funzioni denominate *"prepare_train"*, *"prepare_validation"*, *"prepare_test"*. Attraverso l'utilizzo dei nomi dei video, queste funzioni assemblano i percorsi per accedere ai video dal drive e restituiscono due liste. La prima lista contiene tutti i nomi dei video, mentre la seconda lista consiste nel primo carattere del nome di ciascun video, fornendo così l'informazione sulla lingua parlata in ogni video. Gli output saranno passati in input alle funzioni:

- pre_processing_train
- pre_processing_validation
- pre_processing_test

Sono state create queste funzioni per uniformare il numero di frame in ogni video, riducendolo da 300/350 a 150 frame per video. La decisione è stata presa affinché i frame siano consecutivi, e quindi il video verrà tagliato dal 150º frame. In alternativa, si poteva optare per l'acquisizione di un frame ogni 4, ma tale approccio è stato scartato poiché causava una riproduzione a scatti dei video, con la conseguente perdita di informazioni sul movimento delle labbra. È comunque un'operazione necessaria diminuire il numero di frame per video perché la potenza di calcolo a disposizione è limitata, inoltre passare da 150 a 200/300 frame incide molto sul tempo di caricamento e di elaborazione di ciascun video, ma potrebbe portare a migliori prestazioni. Inoltre sono stati ridimensionati i frame e portati ad una dimensione 50x50, questa ulteriore standardizzazione è stata fatta con lo scopo di ridurre ancor di più il carico computazione. Prima di adottare il formato 50x50, è stato considerato l'utilizzo del formato 128x128. Tuttavia, questa scelta ha comportato la creazione di un dataset significativamente più grande, imponendo restrizioni sulla dimensione del batch e su altri iperparametri della rete neurale. Al termine del processo di preprocessing, i dataset sono stati salvati nel formato numpy array. Questa scelta è stata motivata dal fatto che il formato numpy array offre dei vantaggi per questo scopo. In particolare, i dati possono essere rappresentati in modo efficiente e compatto, consentendo un rapido accesso anche per future manipolazioni dei dati, senza la necessità di dover rieseguire il preprocessing. Inoltre, salvando i dati nel formato numpy array, diventa possibile fornire direttamente tali dati al modello senza la necessità di ulteriori trasformazioni o conversioni. Prima di spiegare il preprocessing che ci ha portato alla più alta accuratezza mostriamo due tecniche che sono state provate ma che non hanno riscosso successo.

- Gray Scale: È stato constatato che con la scala di grigi si perde qualche punto percentuale di accuratezza, in particolare se viene riutilizzata la stessa rete del dataset a colori le accuratezze scendono molto, mentre a parità di capacità computazionali, modificando gli iperparametri si raggiungono risultati con quale punto percentuale di accuratezza inferiore.

- Mean Shift: È stato condotto un esperimento sull'utilizzo del metodo Mean Shift, tuttavia, a causa dell'elevato tempo di esecuzione richiesto, che ammontava a circa 4 giorni per completare l'operazione sul solo training set, si è deciso di abbandonare l'impiego di tale tecnica, nonostante fosse computazionalmente fattibile, dunque può essere considerata come idea in un preprocessing futuro dell'applicazione. Il risultato del mean shift su un singolo frame di un video è mostrato in figura 1:

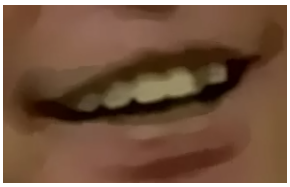


Figura 1: Frame di un video trasformato con il Mean Shift

Come si può vedere i colori nell'immagine sono evidentemente artificiali, rendendo i valori dei pixel simili.

2.2 Rete ConvLSTM2D con RGB

La costruzione della rete è avvenuta seguendo le indicazioni presenti nel seguente riferimento bibliografico [Gé19]. Si è partiti con la più semplice configurazione possibile, ossia una rete formata da:

1. ConvLSTM2D con 3 filtri
2. Flatten
3. Denso con 8 neuroni

Con un early stopping pari a 3 sull'accuratezza e una batch size di 2.
Ottenendo questo risultato:

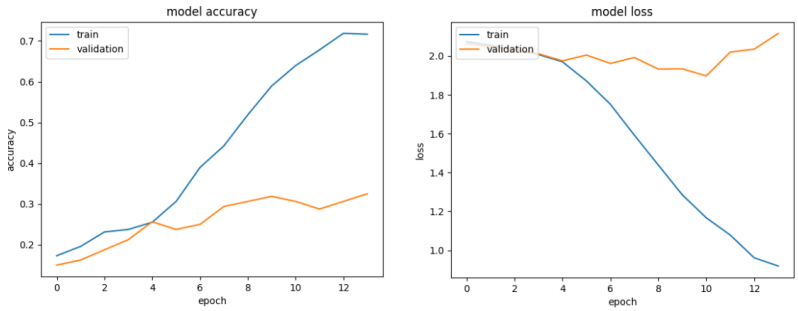


Figura 2: Accuratezza e Loss generati dalla rete neurale

Diamo una spiegazione della rete e degli iperparametri associati: la ConvLSTM2D come spiegato nel paper: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting [SCW+15], è un tipo di rete neurale ricorrente che combina le proprietà delle reti LSTM, (Long Short Term Memory) con le reti convoluzionali, ed è una delle reti più utilizzate per l'elaborazione di immagini. In input alla ConvLSTM2D va fornito un tensore 4D, dove il primo valore sta ad indicare il numero di frame, nel caso in questione 150, successivamente l'altezza del frame e la larghezza, ossia 50x50 ed in fine il canale di colori, che in questa sezione sarà 3, che sta ad indicare il canale RGB. Per capire il funzionamento dei filtri si rimanda al capitolo 14 del libro [Gé19], in sostanza quello che fa la convoluzione è un'operazione locale, dunque un filtro, ossia una maschera 3x3, scorre sull'immagine, ed essa andrà a mappare delle caratteristiche o dei pattern, dunque all'aumentare del numero di filtri potremmo avere un aumento dell'accuratezza del modello, a discapito di difficoltà computazionali ed alta probabilità di fare overfitting. È un problema avere un modello che fa overfitting in quanto perde la capacità di generalizzare i risultati quando viene portato nel mondo reale questo perché ha imparato troppo dai dati di training. Un modo per risolvere questo problema potrebbe essere l'aumento del numero di video nel dataset di addestramento. Il layer Flatten si occupa di ristrutturare i dati in vettori di dimensioni minori, viene utilizzato come ponte tra la fase di estrazione delle caratteristiche, che avviene nella ConvLSTM2D e la fase di classificazione di una rete neurale. Infine il layer denso con 8 neuroni, il quale si occupa di fare classificazione multi-classe, il fatto che i neuroni siano 8 non è un caso in quanto ognuno si occupa di rappresentare una delle 8 lingue che cerchiamo di classificare. Il layer denso ha una funzione di attivazione del tipo "softmax" che serve per restituirci la probabilità associata ad ogni classe, ogni lingua avrà associato un valore tra 0 e 1, ed infatti la somma finale di tutte le probabilità ci darà 1. Avere un early stopping pari a 3 sull'accuratezza significa che quando il valore di accuratezza di 3 epoche consecutive sarà simile oppure abbastanza vicino l'uno all'altro, l'addestramento verrà terminato. L'early stopping è un alleato importante per non cercare il numero di epoche ideale andando a tentativi, inoltre è importante anche per evitare l'overfitting, avere valori alti porterà sicuramente ad un miglioramento dell'accuratezza, ma altrettanto sicuramente il modello farà overfitting, dunque va identificato il valore ideale affinché il trade-off del modello tra accuratezza ed overfitting sia ottimale. Per concludere l'ultimo iperparametro è la batch size, ossia la grandezza del sottoinsieme del dataset di training che viene passato alla rete, entro la fine dell'epoca saranno stati passati tutti i sottoinsiemi e nella successiva epoca verranno utilizzati sottoinsiemi diversi, dunque combinazioni di dati differenti. Anche questo iperparametro è determinante nelle prestazioni del modello, utilizzare una batch size grande permette di addestrare il modello velocemente, a discapito di un maggior utilizzo della memoria e ciò comporta difficoltà computazionali. Inoltre,

l'uso di una batch size ampia riduce la possibilità di fare overfitting, d'altra parte l'uso di una batch size ridotta rende il tempo di addestramento maggiore, il vantaggio risiede nelle accuratezze maggiori, ovviamente a discapito di un'alta probabilità di fare overfitting.
Passiamo adesso a valutare il grafico ottenuto da questa configurazione. Al termine di ogni epoca il modello ci darà come output 4 valori:

1. Accuratezza sul training set
2. Accuratezza sul validation set
3. Loss, (funzione di perdita), sul training set
4. Loss sul validation set

L'accuratezza serve a misurare le prestazioni del modello ed indica la percentuale di previsioni corrette effettuate dal modello. La funzione di perdita, anche chiamata loss, indica quanto i valori previsti dal modello si discostano da quelli presenti nel training set o nel validation set. L'obiettivo di un buon addestramento è ottenere un'alta accuratezza sul validation set e sul test set, accompagnati da una funzione di loss decrescente. La funzione di perdita è uno strumento importante per capire quanto overfitting il modello sta facendo. Nella figura 2 il grafico a sinistra indica l'accuratezza e il grafico a destra la funzione di perdita, con il blu viene indicato il valore del training set e con l'arancione quello del validation set. Nonostante sia stata raggiunta un'accuratezza del 35% sul test set, un risultato promettente data la situazione attuale, si è riscontrato un eccessivo overfitting del modello. In particolare, l'andamento della curva arancione, che idealmente dovrebbe essere in diminuzione, risultava quasi lineare con una tendenza crescente verso la fine.
Quindi proviamo a modificare la batch size, passando da 2 a 32, ottenendo cosi un'accuratezza del 21% sul test set e una funzione di perdita ancora non decrescente, e peggiore della precedente. Di seguito il grafico:

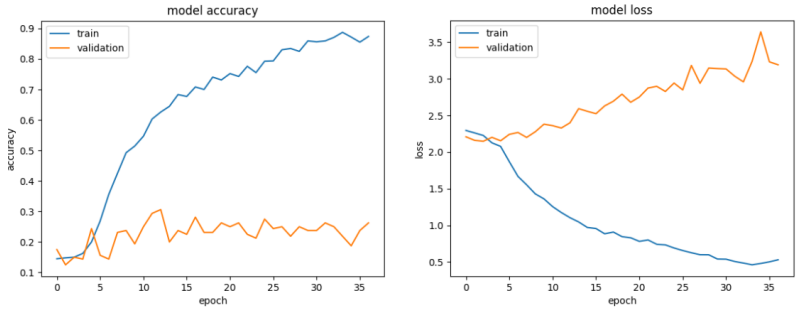


Figura 3: Accuratezza e Loss generati dalla rete neurale nel secondo tentativo

Evitando di narrare tutti i numerosi tentativi, saranno illustrati i punti salienti che hanno permesso l'ottenimento del modello finale. Seguendo il filone di pensiero secondo cui aumentare la batch size permette la diminuzione dell'overfitting, il seguente tentativo è stato effettuato con la medesima configurazione ma con una batch size di 136 ed una pazienza sull'early stopping di 2 invece che di 3, ottenendo così il seguente grafico:

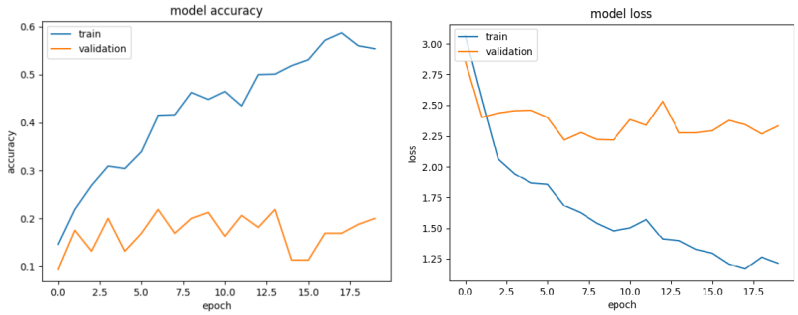


Figura 4: Grafico con la prima funzione di perdita decrescente

Dal grafico a sinistra è poco evidente la funzione di perdita decrescente, ma comunque è stato considerato il miglior risultato fino a quel momento, come c'era da aspettarsi a discapito di una

diminuzione dell'overfitting c'è stata anche una diminuzione dell'accuratezza, in particolare si arriva ad un valore del 21% sul test set.
Dunque ottenuto un ragionevole risultato con l'overfitting bisognava lavorare ad un miglioramento dell'accuratezza, per farlo come detto in precedenza bisognava aumentare il numero di filtri, questo aumento ha comportato una diminuzione nella batch size in quanto le capacità della macchina erano limitate, ottenendo questa configurazione:

1. ConvLSTM2D con 3 filtri
2. Flatten
3. Denso con 8 neuroni

Con un early stopping pari a 1 sull'accuratezza e una batch size di 86. Il risultato raggiunto è stato il seguente:

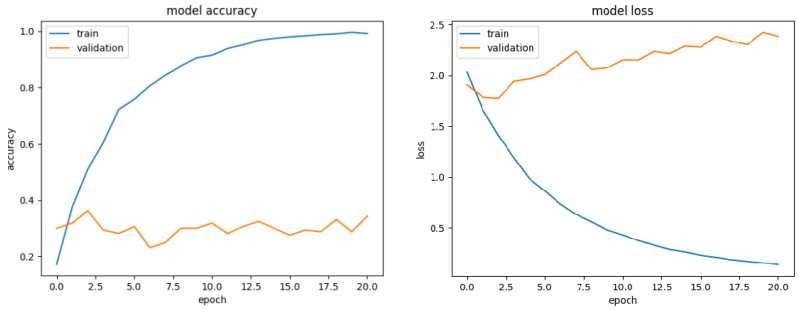


Figura 5: Grafico per migliorare l'accuratezza

Con questa configurazione l'accuratezza sul test set è stata del 33% la più alta ottenuta fino a quel momento, ma la curva della funzione di perdita era tornata crescente, nonostante la batch size fosse alta, e l'early stopping il più basso possibile. Nei tentativi successivi sono state provate varie impostazioni, per agevolare il lavoro è stato utile costruire una mappa concettuale che verrà esposta di seguito:

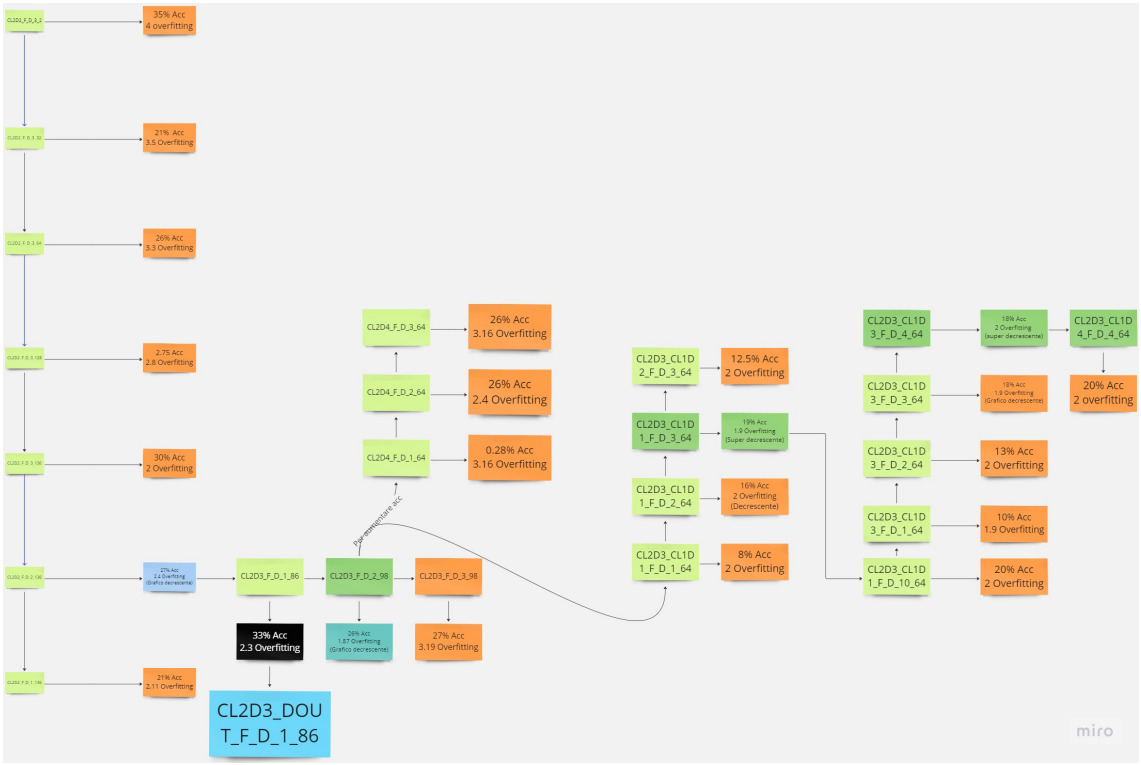


Figura 6: Mappa che mostra il processo logico di alcuni dei tentativi

I rettangoli arancioni sono dei punti ciechi, i rettangoli verde scuro indicano una strada da percorrere, il rettangolo blu successivo è stato il miglior tentativo raggiunto per accuratezza ed overfitting.
Il rettangolo nero rappresenta la configurazione finale esposta, a partire dalla quale si è sviluppato un successivo susseguirsi di prove, suddivise in due percorsi distinti. Da un lato, si è proceduto all'aumento del numero di filtri, mentre dall'altro si è aggiunto uno strato di complessità alla rete, come ad esempio l'implementazione di un modello ConvLSTM1D o l'inclusione di strati di Dropout, responsabili dell'attivazione e disattivazione dei neuroni al fine di ridurre l'overfitting. Di seguito viene presentata la configurazione che ha dimostrato la più bassa e decrescente funzione di perdita:

1. ConvLSTM2D con 3 filtri
2. ConvLSTM1D con 1 filtro
3. Flatten
4. Denso con 8 neuroni

Con un early stopping pari a 3 sull'accuratezza e una batch size di 64, di seguito il grafico associato:

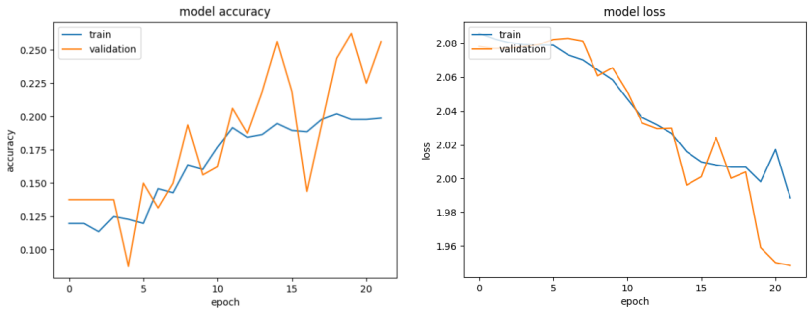


Figura 7: La configurazione con la funzione di perdita migliore

Con il suddetto modello, è stata raggiunta un'accuratezza del 19%. In prospettiva futura, si potrebbe considerare la continua esplorazione di questa configurazione al fine di migliorare ulteriormente l'accuratezza, mediante l'aggiunta di filtri e l'aumento della dimensione del batch. Tuttavia, è importante notare che abbiamo riscontrato limitazioni in termini di capacità computazionali che ci hanno impedito di procedere oltre in questa direzione.

2.3 Il miglior risultato

Dopo aver affrontato l'ostacolo delle numerose strade chiuse, è stata presa la decisione di retrocedere ai progressi più significativi ottenuti in termini di precisione, al fine di esplorare alternative volte a mitigare l'overfitting. È stata quindi adottata la decisione di integrare uno strato di Dropout proprio nella configurazione che aveva precedentemente garantito un'elevata precisione. Dopo vari tentativi, tra cui la selezione della dimensione del batch, la disposizione del layer di Dropout, il numero di filtri e la pazienza dell'early stopping, siamo giunti al modello ottimale in termini di accuratezza e overfitting. Il modello ha conseguito un tasso di precisione del 35%, con una funzione di perdita che, sebbene non manifesti un declino pronunciato, non evidenzia una significativa crescita. La configurazione è stata la seguente:

1. ConvLSTM2D con 3 filtri
2. Dropout
3. Flatten
4. Denso con 8 neuroni

Con un early stopping pari a 1 sull'accuratezza e una batch size di 86, in basso il grafico associato:

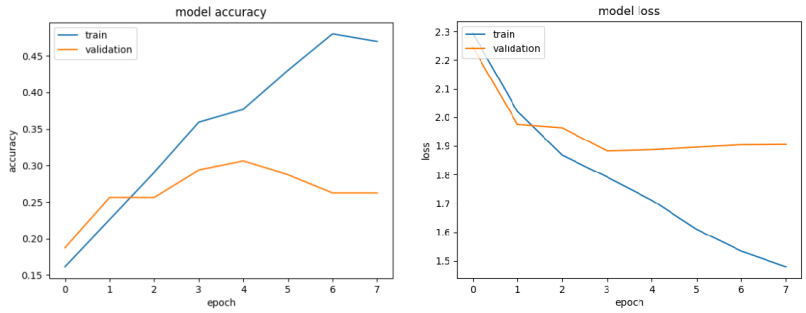


Figura 8: La configurazione con la funzione di perdita migliore

Ecco una tabella riassuntiva di tutte le configurazioni:

	Numero di filtri	Batch size	Accuratezza	Livelli della rete
1° Tentativo	3	2	35.00%	ConvLSTM2D + Flatten + Dense
2° Tentativo	3	32	21.00%	ConvLSTM2D + Flatten + Dense
3° Tentativo	1	136	21.00%	ConvLSTM2D + Flatten + Dense
4° Tentativo	3	86	33.00%	ConvLSTM2D + Flatten + Dense
5° Tentativo	ConvLSTM2D 3 filtri ConvLSTM1D 1 filtro	64	19.00%	ConvLSTM2D + ConvLSTM1D + Flatten + Dense
6° Tentativo	3	86	35.00%	ConvLSTM2D + Dropout + Flatten + Dense

Figura 9: Rete neurale

Di seguito è rappresentata la struttura della rete finale:

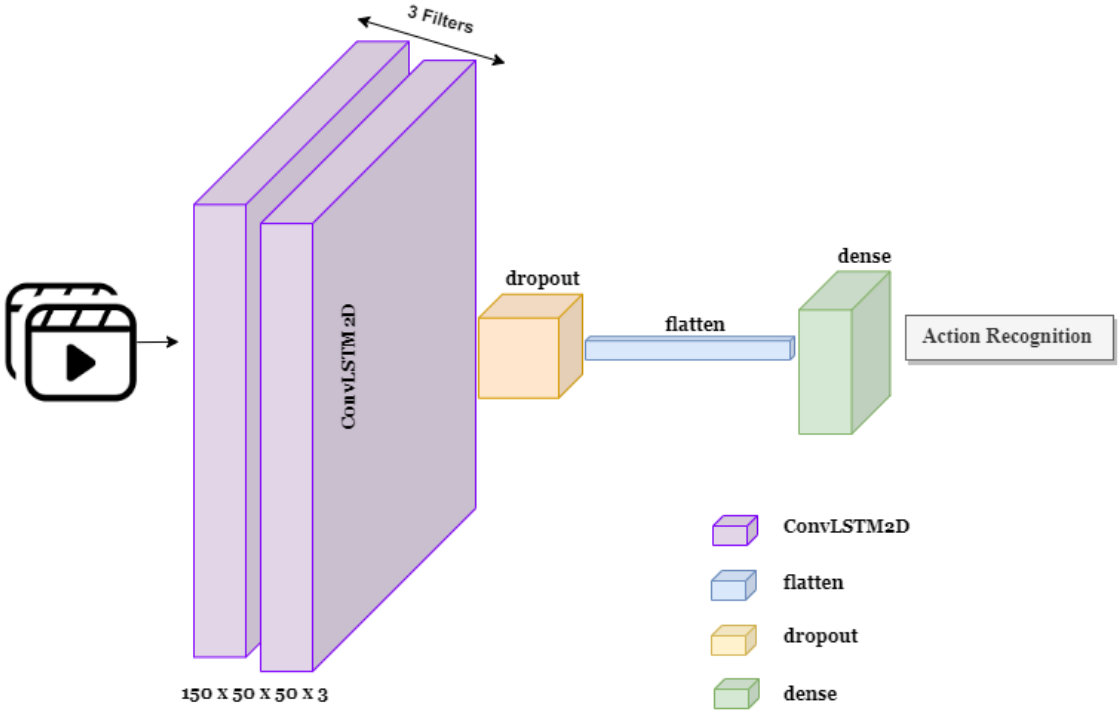


Figura 10: Rete neurale

Nella tabella in basso sono elencati tutti layers utilizzati e il numero di parametri del modello:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_lstm2d (ConvLSTM2D)    (None, 50, 50, 3)         660

 dropout (Dropout)           (None, 50, 50, 3)         0

 flatten (Flatten)           (None, 7500)              0

 dense (Dense)               (None, 8)                 60008

=================================================================
Total params: 60,668
Trainable params: 60,668
Non-trainable params: 0
_________________________________________________________________
```

Figura 11: Rete neurale

2.4 Misure di accuratezza

Uno dei modi più completi per rappresentare i risultati di un modello di classificazione è utilizzando la confusion matrix, l'idea generale della matrice di confusione è quella di contare il numero di volte in cui le istanze di classe A sono classificate come classe B, o come classe A e viceversa. Dunque confronta le classi previste con le classi reali calcolando la frequenza delle classi che sono state correttamente previste (True positive), la frequenza delle classi che sono state previste in modo sbagliato (True negative) e il conteggio delle classi che sono state classificate come false ma che in realtà sono vere (False negative) e quelle che sono state classificate come vere ma che in realtà sono false (False positive). Nel caso descritto dopo aver stimato le classi previste esse sono state confrontate con le classi reali del test set generando la seguente confusione matrix:

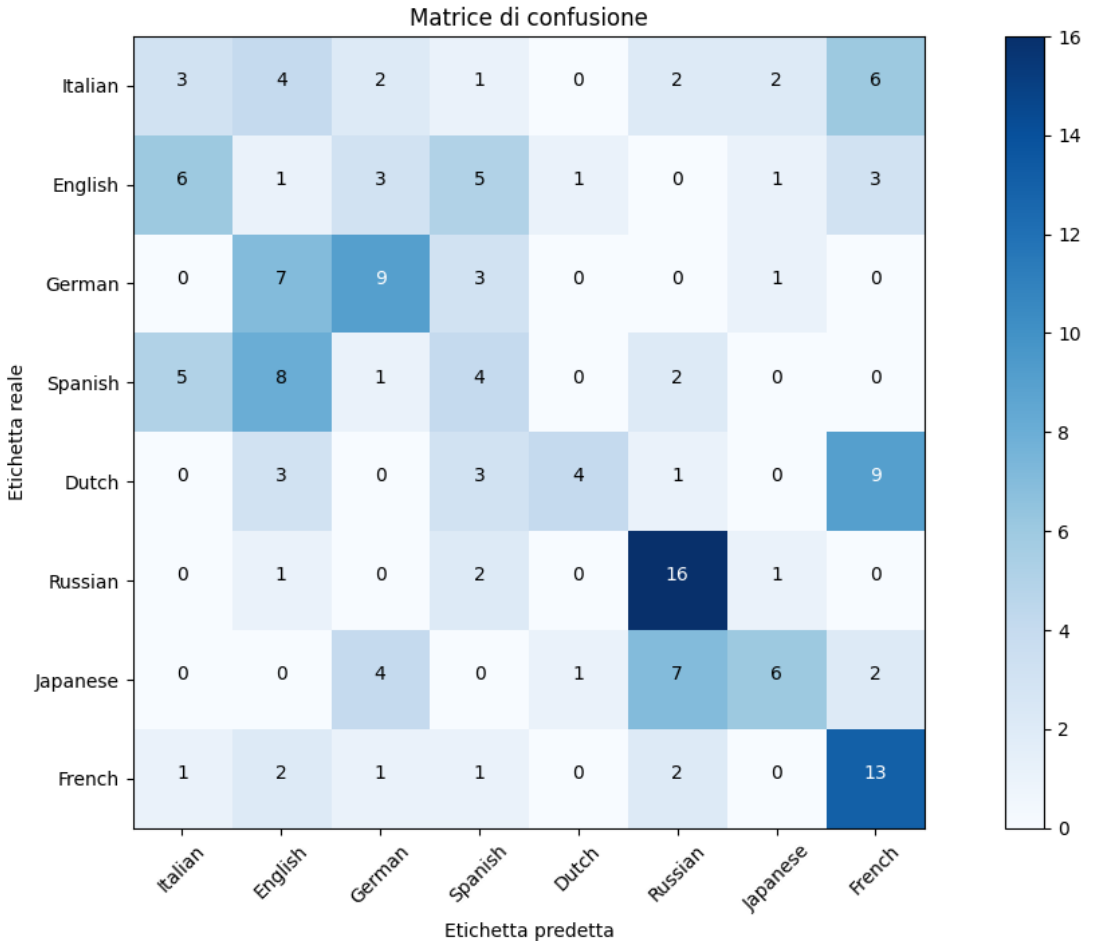


Figura 12: Confusion matrix di tutte le lingue

Le voci sulla diagonale principale della matrice di confusione corrispondono a classificazioni corrette, le altre voci ci dicono quanti campioni di una classe sono stati erroneamente classificati come un'altra classe. Più nello specifico nel grafico in alto è possibile notare che le lingue che vengono classificate meglio sono il russo e il francese a seguire il tedesco, il giapponese, l'olandese e lo spagnolo, ed infine è possibile notare che le lingue che vengono più erroneamente classificate sono l'italiano e l'inglese. Inoltre analizzando le altre celle della matrice è possibile notare che il francese viene talvolta confuso con l'olandese, l'inglese con lo spagnolo e il russo con il giapponese. A seguire sono illustrate le misure di valutazione comuni utilizzate per valutare le prestazioni di un modello di classificazione per ogni singola lingua presente nel set di dati:

Classe	Precision	Recall	F1-score	Accuracy
Italian	0.20	0.15	0.17	0.15
English	0.04	0.05	0.04	0.05
German	0.45	0.45	0.45	0.45
Spanish	0.21	0.20	0.21	0.20
Dutch	0.67	0.20	0.31	0.20
Russian	0.53	0.80	0.64	0.80
Japanese	0.55	0.30	0.39	0.30
French	0.39	0.65	0.49	0.65

Figura 13

Le misure in alto confermano quanto visto nella confusion matrix, infatti partendo dalle accuratezze per ogni singola classe è possibile notare che le lingue classificate meglio sono il russo e il francese rispettivamente l'80% e il 65% a seguire in ordine decrescente tutte le altre lingue: il tedesco: 45%, il giapponese: 30%, lo spagnolo e l'olandese: 20% ed infine l'italiano e l'inglese rispettivamente 15% e 5%. Nelle altre colonne è possibile osservare i risultati delle altre metriche: la Precision, la Recall e l'F1 Score il quale è la media armonica delle due misure precedenti, in particolare bilancia la capacità del modello di identificare correttamente le istanze positive (Recall) e la sua accuratezza nel farlo (Precisione), più è alta più la F1 Score e più il modello è in grado di classificare le istanze in modo corretto per quella relativa classe, nel caso in esame i valori più alti di F1-score corrispondono alle lingue meglio classificate ovvero il russo e il francese a seguire il resto delle lingue.

3 Feature level fusion

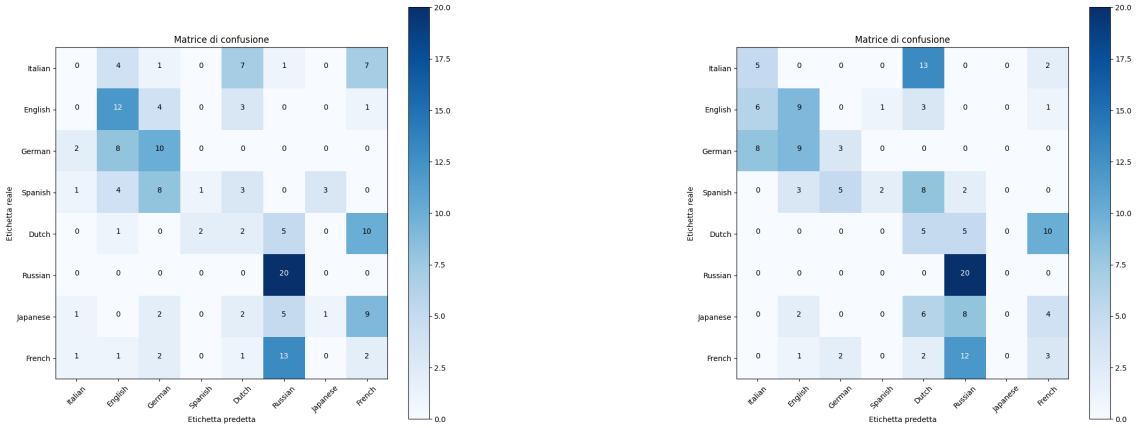
La feature level fusion [RKBT07] consiste nel realizzare un dataset formato dalla concatenazione di diverse previsioni ottenute da fonti differenti. In questo modo spesso si riesce ad ottenere una migliore discriminazione rispetto all'utilizzo di un singolo predittore, come nel caso in questione la rete neurale. Dunque, sono stati utilizzati i dati provenienti da uno studio condotto dal gruppo 24 dell'anno 2023, che si concentrava sulla previsione utilizzando una PNN (Probabilistic Neural Network) che apprendeva sulla base delle distanze tra landmark posizionati sulle labbra delle persone. Il dataset utilizzato, denominato "babele_cityblock_random6", è stato appositamente creato da tale gruppo e ogni variabile rappresenta la posizione di un landmark specifico sulle labbra. Questo dataset è stato poi dato ad una batteria di classificatori di cui seguito la lista:

1. KNeighborsClassifier;
2. SVC;
3. SVC(kernel="rbf");
4. GaussianProcessClassifier;
5. DecisionTreeClassifier;
6. RandomForestClassifier;
7. MLPClassifier;
8. AdaBoostClassifier;
9. GaussianNB;
10. QuadraticDiscriminantAnalysis

Al fine di ottenere, a seguito di una fase di addestramento, delle previsioni più accurate possibili. Tra la lista dei 10 classificatori a spiccare sono stati due, GaussianProcessClassifier e GaussianNB, con un accuratezza rispettivamente del 30% e del 29.7%. Una volta compresi quali fossero i migliori

classificatori, sono stati impiegati per generare previsioni utilizzando i dati del test set. La scelta di utilizzare il test set non è stata casuale. Se si fosse deciso di utilizzare i dati di addestramento per effettuare previsioni, si sarebbero ottenute accuratezze superiori. Tuttavia, tale approccio avrebbe presentato delle limitazioni in quanto i classificatori erano stati addestrati utilizzando tali dati e quindi erano ben familiarizzati con essi. Al contrario, i dati del test set rappresentavano un insieme sconosciuto per i classificatori, evitando così il problema dell'overfitting.

3.0.1 Matrici di confusione dei migliori classificatori



(a) GaussianProcessClasifier

(b) GaussianNB

Figura 14: Matrice di confusione

3.1 Il problema del dataset Subject Dependent

Una volta completate tutte le previsioni sul set di test, includendo quelle ottenute dalla rete neurale, dal classificatore GaussianProcessClassifier e dal classificatore GaussianNB, sono stati generati due dataset distinti. Il primo dataset è stato creato unendo le previsioni, ovvero le classi che il modello addestrato assegna alle osservazioni del test set, della rete neurale e del GaussianProcessClassifier, mentre il secondo dataset è stato creato combinando le previsioni della rete neurale e del GaussianNB. L'unione di questi dataset è stata effettuata utilizzando i nomi dei video come criterio di fusione. Tuttavia, tale aspetto si è rivelato critico in quanto l'unione dei dataset senza questa accortezza risultava casuale e dunque deleteria per l'ottenimento di una buona accuratezza. Dopo aver ottenuto i nuovi dataset, viene applicata la feature level fusion, ossia i dati vengono utilizzati per un'ulteriore addestramento sulla stessa batteria di classificatori al fine di ottenere delle previsioni migliori sulla lingua parlata. Nonostante il superamento di tale errore, ne è stato commesso un altro, sebbene meno banale rispetto al precedente, che ha portato le accuratezze ottenute dalla serie di classificatori al 90% cifra purtroppo fittizia. Per capire l'errore va spiegata la composizione del dataset di addestramento che segue la seguente struttura: sono presenti 160 video in totale, distribuiti equamente tra le 8 lingue considerate. Pertanto, avendo un dataset bilanciato, si hanno esattamente 20 video per ciascuna lingua. Tuttavia, tenendo conto che per ogni lingua sono presenti 4 individui distinti, si deduce che ogni individuo contribuisce con 5 dei propri video all'interno del dataset. Il problema sorge dal fatto che se viene eseguito un campionamento semplice con una proporzione dell'80% per l'addestramento e del 20% per la validazione, si ottengono accuratezze molto elevate, poiché vi è una forte probabilità che il modello impari a riconoscere l'associazione tra una lingua e un soggetto specifico, anziché imparare effettivamente quale lingua viene parlata. Dunque quello che è stato fatto è stato splittare il dataset in training e test non casualmente bensì sulla base degli individui, in particolare 3 dei 5 individui sono stati posizionati nel training set e 2 nel test set. Non è stata provata una diversa configurazione, dunque nei lavori futuri c'è la possibilità di provare a splittare in 4 e 1. Con questa correzione, il metodo della feature level fusion ha ottenuto un'accuratezza del 67% con il classificatore RandomForest, sia nel caso del dataset generato dall'unione delle previsioni della rete neurale con GaussianProcessClassifier e sia con l'unione di GaussianNB.

3.2 Misure di accuratezza della tecnica Feature level score

3.2.1 GaussianNB

La matrice di confusione in basso mostra quali sono le lingue classificate meglio e quali invece vengono erroneamente categorizzate, per quanto riguarda il classificatore GaussianNB il risultato è il seguente:

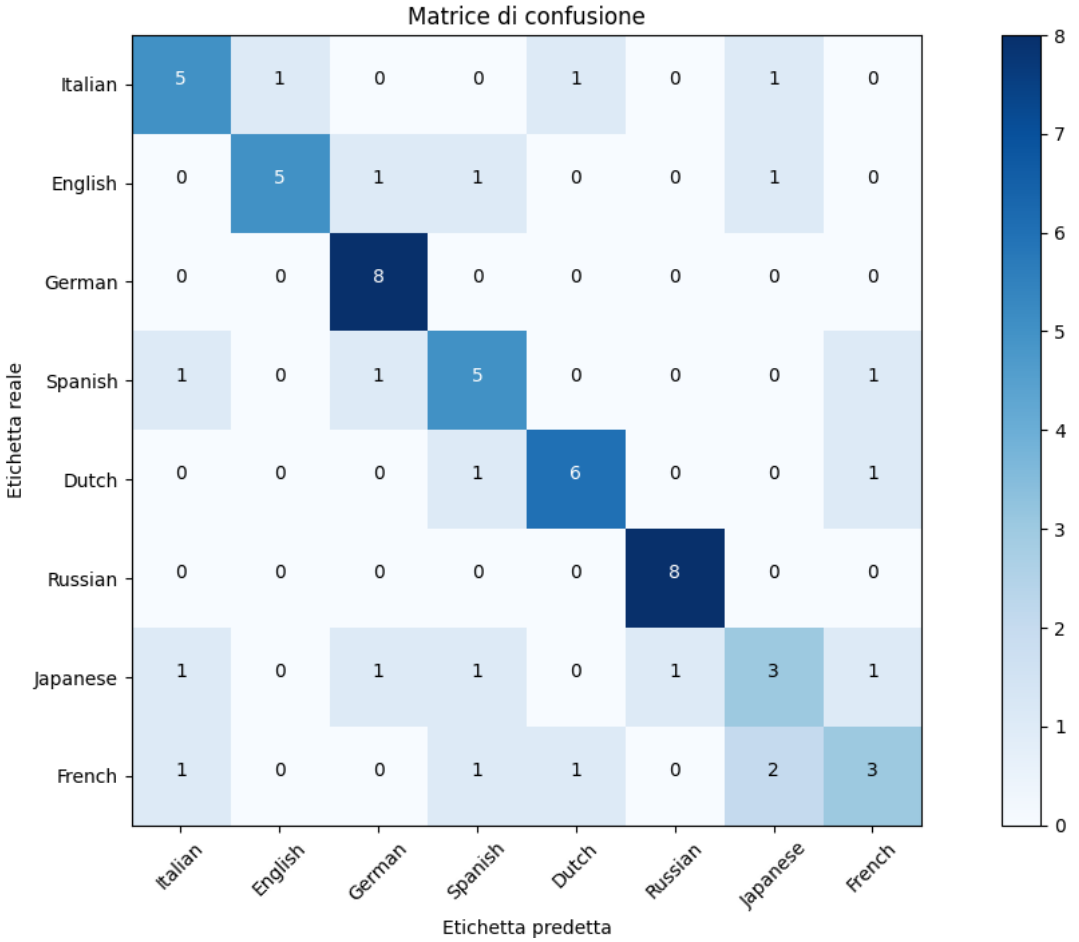


Figura 15: Confusion matrix di tutte le lingue

Come osservabile, si constata che sono poche le classi che vengono classificate in modo sbagliato, ad eccezione di due video che vengono classificati come giapponese ma che in realtà sono francesi; per il resto sono poche le osservazioni che non sono etichettate in maniera corretta, quanto detto è confermato dalla tabella in basso:

```
Accuratezza RandomForestClassifier():  0.671875
```

Classe	Precision	Recall	F1-score	Accuracy
Italian	0.62	0.62	0.62	0.62
English	0.83	0.62	0.71	0.62
German	0.73	1.00	0.84	1.00
Spanish	0.56	0.62	0.59	0.62
Dutch	0.75	0.75	0.75	0.75
Russian	0.89	1.00	0.94	1.00
Japanese	0.43	0.38	0.40	0.38
French	0.50	0.38	0.43	0.38

Figura 16

Dalla tabella sopra riportata emerge chiaramente che le classi con l'accuratezza più elevata sono il russo e il tedesco, con una percentuale del 100%. Seguono l'olandese, con il 75%, l'italiano e l'inglese, con il 62%. Infine, lo spagnolo registra un'accuratezza del 75%.

3.2.2 GaussianProcessClassifier

Per quanto concerne il classificatore GaussianProcessClassifier, si nota che le accuratezze più elevate sono registrate dal classificatore RandomForestClassifier(). La matrice di confusione sottostante illustra l'accuratezza relativa a ciascuna lingua.

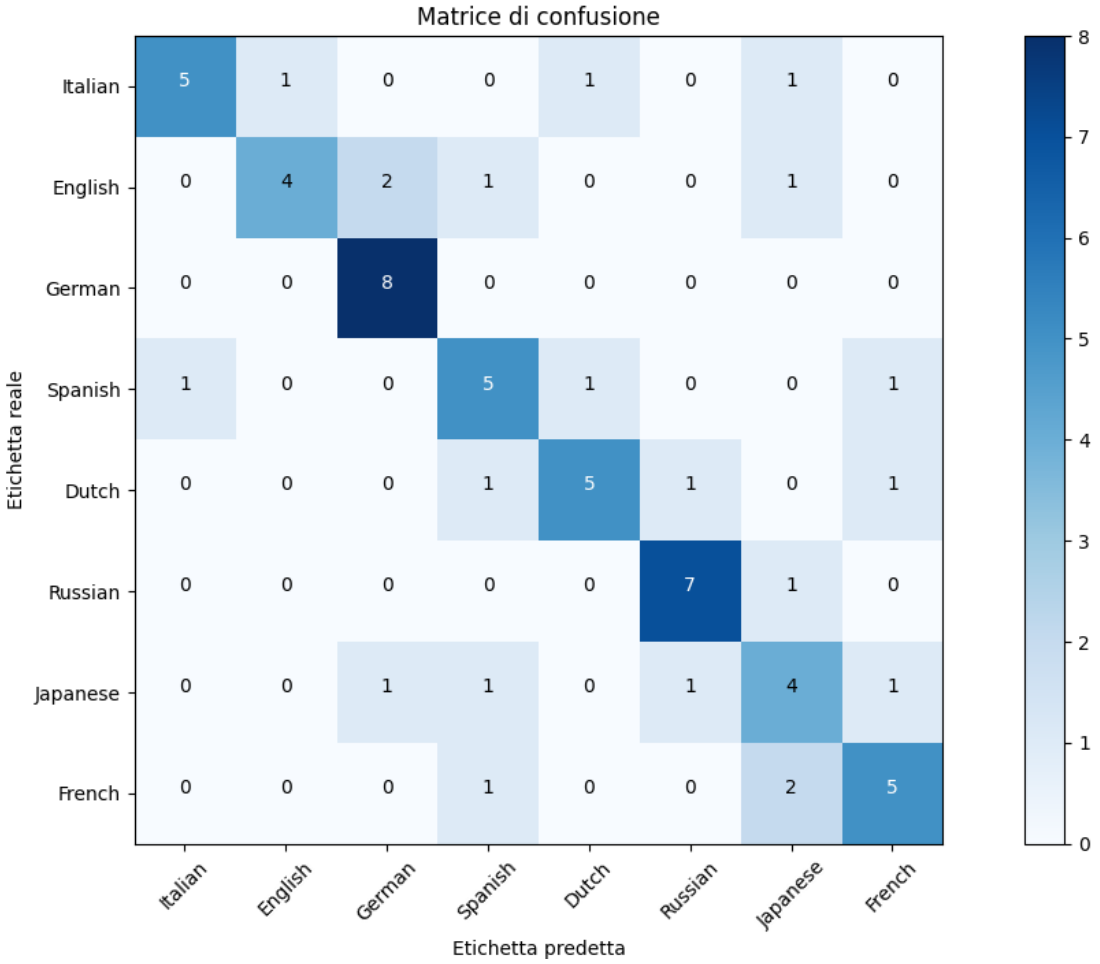


Figura 17: Confusion matrix di tutte le lingue

Dalla matrice in alto si possono dedurre le medesime conclusioni del classificatore precedente, poiché anche in questo caso i video vengono etichettati correttamente rispetto alle etichette reali. Tale constatazione è confermata dalle accuratezze relative ad ogni classe, le quali risultano essere molto elevate, come evidenziato nella figura in basso.

```
Accuratezza RandomForestClassifier():  0.671875
```

Classe	Precision	Recall	F1-score	Accuracy
Italian	0.83	0.62	0.71	0.62
English	0.80	0.50	0.62	0.50
German	0.73	1.00	0.84	1.00
Spanish	0.56	0.62	0.59	0.62
Dutch	0.71	0.62	0.67	0.62
Russian	0.78	0.88	0.82	0.88
Japanese	0.44	0.50	0.47	0.50
French	0.62	0.62	0.62	0.62

Figura 18

La lingua tedesca registra un'accuratezza del 100%, seguita dal russo con un'accuratezza dell'88%. Successivamente, il francese, l'italiano, lo spagnolo e l'olandese le quali raggiungono il 62%.

4 Dataset pre-processato in gray scale

4.1 Preprocessing

Quanto segue rappresenta la stessa procedura descritta in precedenza, con la differenza che, in questa occasione, durante la fase di preprocessing, le immagini sono state convertite in scala di grigi al fine di ridurre il carico computazionale. Tuttavia, come sarà evidenziato successivamente, l'eliminazione del colore ha comportato un peggioramento delle performance del modello.

4.2 Addestramento del modello

Una volta ottenuti i dati preprocessati, sono stati utilizzati come input nella stessa rete neurale utilizzata per addestrare il dataset BABELE nella versione RGB. Il risultato ottenuto è stato un'accuratezza del 28% e un evidente caso di overfitting da parte del modello. Di seguito sono riportati i grafici corrispondenti:

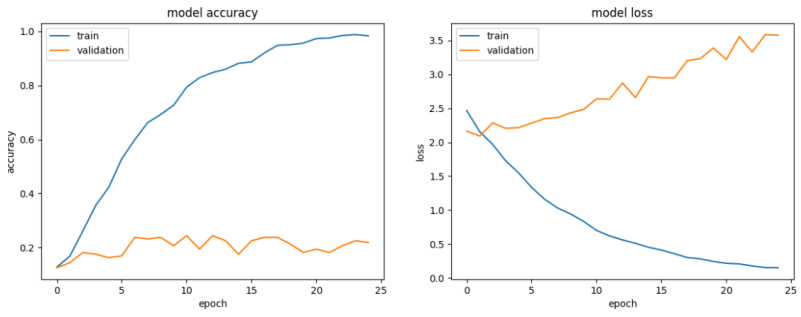


Figura 19: Accuratezza e Loss generati dalla rete neurale

Nel tentativo di mitigare l'overfitting, si è provveduto ad aumentare la dimensione del batch size da 64 a 120. Questa modifica ha comportato un miglioramento dell'accuratezza, che è passata dal 28% al 31%. Tuttavia, nonostante ciò, il problema dell'overfitting persisteva ancora.

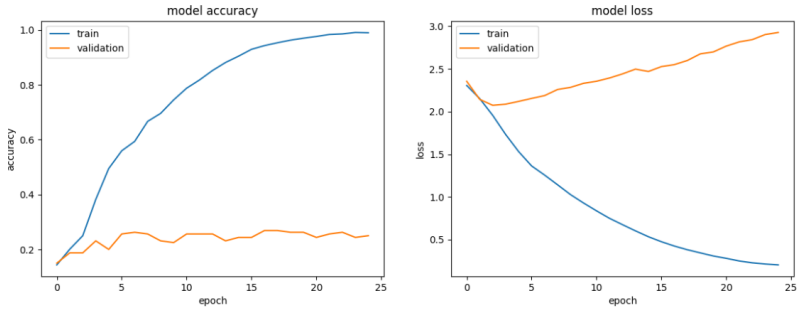


Figura 20: Accuratezza e Loss generati dalla rete neurale

Successivamente, si è proceduto ad ulteriori modifiche, aumentando la batch size e riducendo il numero di filtri da 3 a 1. Purtroppo, questa modifica ha comportato una diminuzione dell'accuratezza e l'overfitting continuava a essere un problema persistente. Nonostante ulteriori tentativi di aumentare la batch size, la situazione è rimasta sostanzialmente invariata. A seguire, sono riportati i risultati ottenuti:

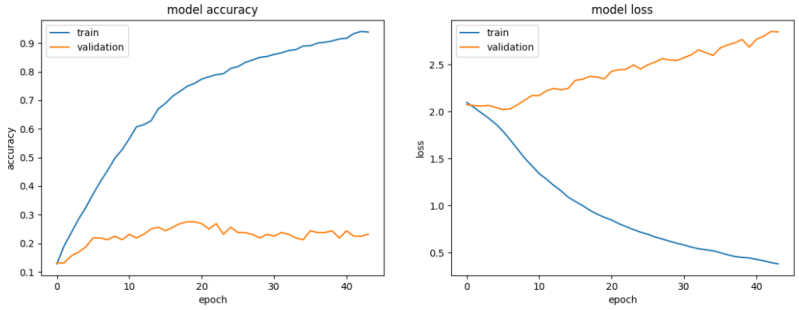


Figura 21: Accuratezza e Loss generati dalla rete neurale

Come ulteriore tentativo, sono stati introdotti due strati di "Dropout", uno prima dello strato "Flatten" e uno dopo. Tuttavia, questa modifica ha portato a una drastica diminuzione dell'accuratezza del modello, mentre l'overfitting sembrava diminuire nella fase iniziale della curva di addestramento, ma aumentare nelle epoche successive.

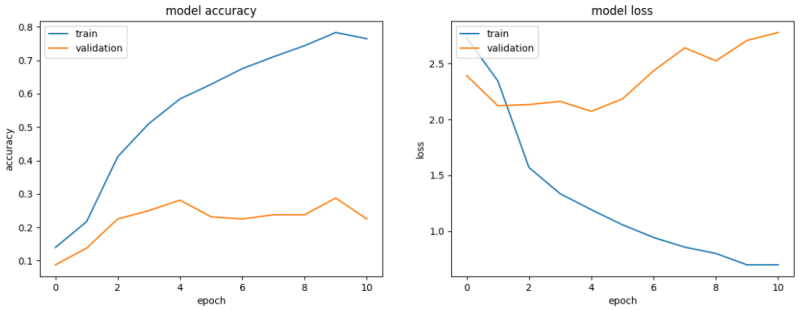


Figura 22: Accuratezza e Loss generati dalla rete neurale

pertanto dopo aver variato numerose volte gli iperparametri e fatto altrettanti tentativi per ridurre l'overfitting, è stato deciso di aggiungere un'ulteriore layer di *Dropout* prima del layer *Flatten*, per un totale di tre layers *Dropout*; infine è stato rimosso il layer *ConvLSTM2D* e sostituito con una *Conv2D*, questo cambiamento è stato effettuato perchè la *Conv2D* è una rete non ricorrente e meno complessa rispetto alla *ConvLSTM2D* tale proprietà garantisce un minor rischio di overfitting da parte del modello, i grafici ottenuti sono i seguenti:

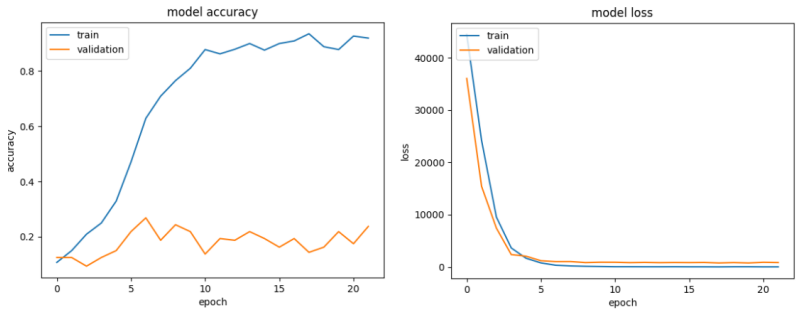


Figura 23: Accuratezza e Loss generati dalla rete neurale

come è possibile notare l'overfitting è totalmente scomparso ed è stata anche raggiunta un'ottima accuratezza seppur inferiore rispetto all'accuratezza ottenuta con l'addestramento del set di dati avente le immagini a colori.

Ecco una tabella riassuntiva di tutte le configurazioni:

	Numero di filtri	Batch size	Accuratezza	Livelli della rete
1° Tentativo	3	64	28.12%	ConvLSTM2D + Flatten + Dense
2° Tentativo	3	120	31.87%	ConvLSTM2D + Flatten + Dense
3° Tentativo	1	126	30.00%	ConvLSTM2D + Flatten + Dense
4° Tentativo	6	32	30.63%	ConvLSTM2D + Dropout + Flatten + Dropout + Dense
5° Tentativo	4	80	33.75%	Conv2D + Dropout + Dropout + Flatten + Dropout + Dense

Figura 24: Rete neurale

Di seguito è illustrata la rete finale utilizzata:

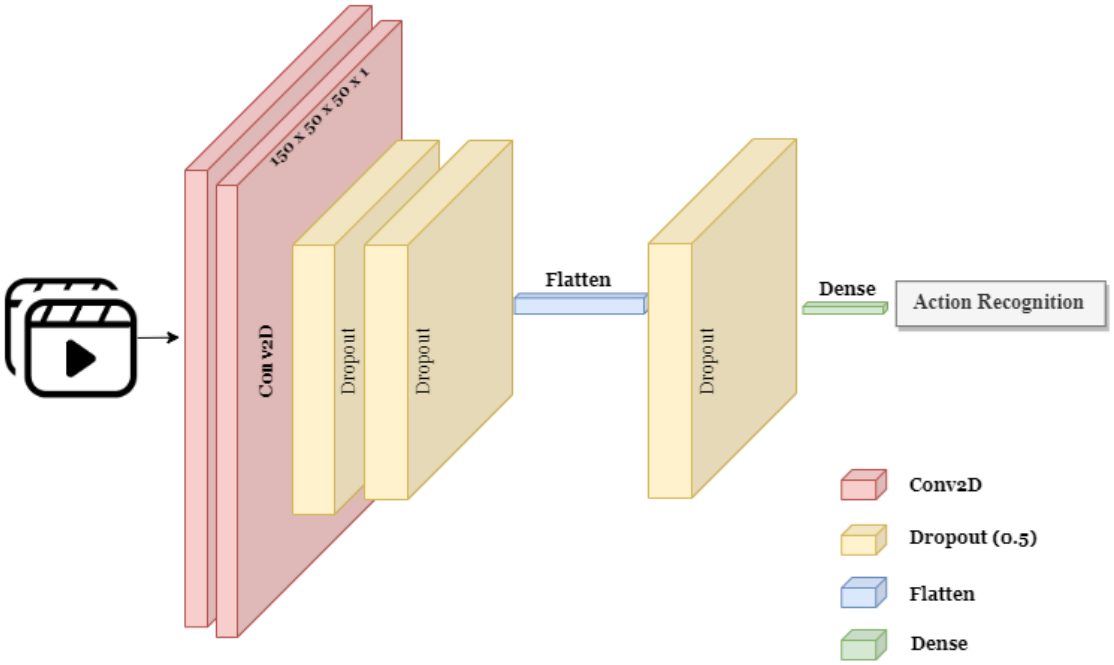


Figura 25: Rete neurale

nella tabella successiva è mostrato un riassunto del modello, ovvero una panoramica dettagliata dei diversi strati, di tutti i parametri utilizzati e il numero di output per ogni strato:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 50, 50, 4)    40

 dropout (Dropout)           (None, 150, 50, 50, 4)    0

 dropout_1 (Dropout)         (None, 150, 50, 50, 4)    0

 flatten (Flatten)           (None, 1500000)           0

 dropout_2 (Dropout)         (None, 1500000)           0

 dense (Dense)               (None, 8)                 12000008

=================================================================
Total params: 12,000,048
Trainable params: 12,000,048
Non-trainable params: 0
_________________________________________________________________
```

Figura 26: Rete neurale

4.3 Misure di accuratezza

Dopo aver effettuato l'addestramento del set di dati in esame si calcolano le previsioni che il modello è in grado di fare sul test set, poi confrontando queste ultime con le etichette reali si ottiene l'accuratezza effettiva del modello che nel caso appena analizzato è pari al 33.75%. Di seguito è illustrata la matrice di confusione che mette in relazione le etichette reali con le etichette previste dal modello:

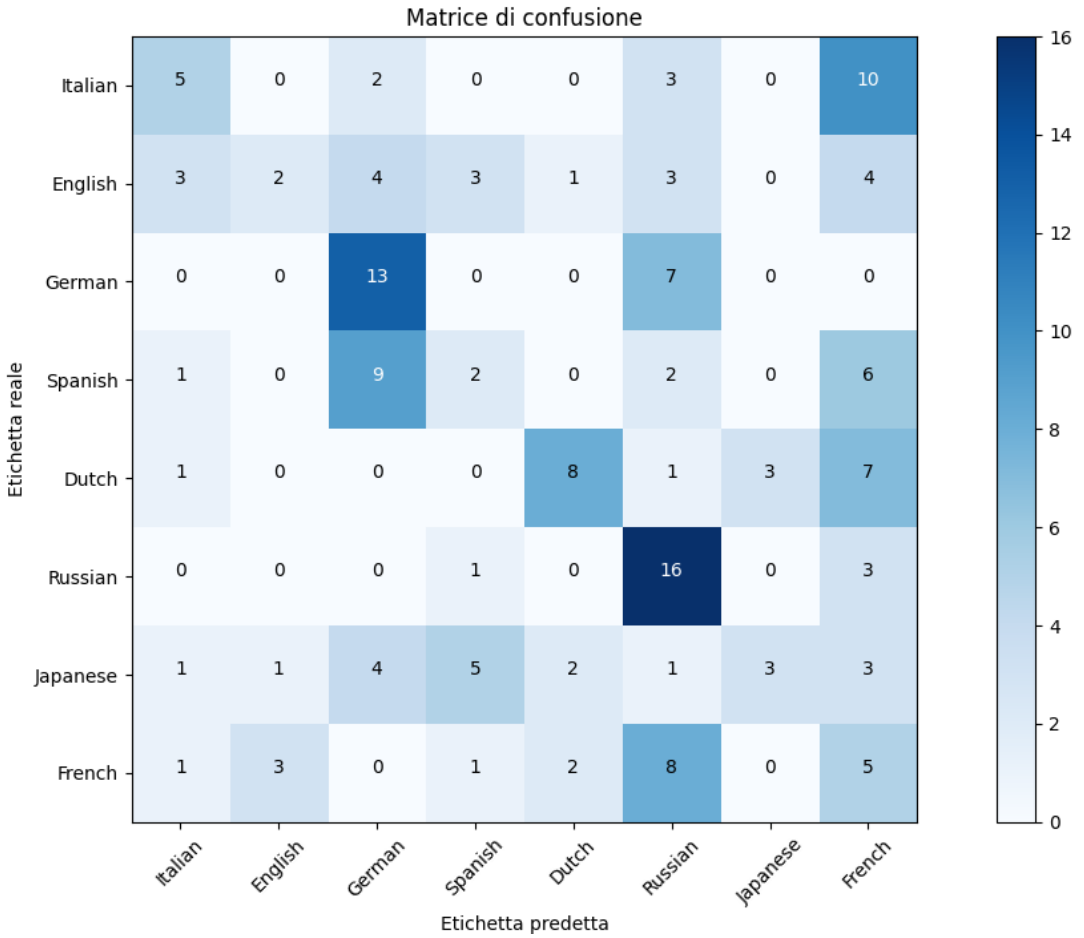


Figura 27: Confusion matrix di tutte le lingue

Dalla matrice di confusione emerge che la lingua con la classificazione più accurata è il russo, seguita dal tedesco e dall'olandese. Analizzando le altre celle della matrice, si nota che il francese viene spesso confuso con l'italiano, il tedesco con lo spagnolo e infine il russo con il francese. Nella parte inferiore sono riportate le altre metriche di valutazione che confermano quanto evidenziato nella matrice di confusione:

Classe	Precision	Recall	F1-score	Accuracy
Italian	0.42	0.25	0.31	0.25
English	0.33	0.10	0.15	0.10
German	0.41	0.65	0.50	0.65
Spanish	0.17	0.10	0.12	0.10
Dutch	0.62	0.40	0.48	0.40
Russian	0.39	0.80	0.52	0.80
Japanese	0.50	0.15	0.23	0.15
French	0.13	0.25	0.17	0.25

Figura 28: Confusion matrix di tutte le lingue

nello specifico dalla tabella in alto è possibile notare che l'accuratezza della classe russo è la più alta, pari all'80%, a seguire in ordine decrescente il tedesco con il 65%, l'olandese con il 40%, le lingue italiano e francese con il 25% , lo giapponese con il 15% e per finire l'inglese e lo spagnolo con il 10%, la misura F1 score conferma quanto visto in precedenza ovvero che il russo è la classe classificata meglio dal modello utilizzato.

5 Riconoscimento di Inglese e Cinese

Il dataset denominato "CH-SIMS_CMU-MOSEI" consiste in video di individui che parlano sia inglese che cinese. È stato creato con l'obiettivo di affrontare una classificazione binaria tra due classi. I video presenti in questo dataset sono più brevi rispetto a quelli presenti nel dataset BABELE e il numero complessivo di video è significativamente inferiore. Inoltre, va sottolineato che il dataset non include un validation set. L'obiettivo principale era ottenere un risultato migliore rispetto ad un semplice lancio di una monetina, tuttavia ciò è stato possibile solo dopo aver risolto alcune problematiche avute durante la fase di pre-processing.

5.1 Preprocessing

Il dataset *CH-SIMS_CMU-MOSEI* contiene video relativi solo a due lingue:

- Inglese
- Cinese

ed è suddiviso in train set e test set nel seguente ordine:

- il train set del dataset MOSEI è costituito da 174 video
- il train set del dataset SIMS è costituito da 150 video
- il test set del dataset MOSEI è costituito da 36 video
- il test set del dataset SIMS è costituito da 36 video

Nella fase di pre-processing, si è iniziato estraendo i video dalle sottocartelle e successivamente adattandoli ad una dimensione di 50x50, utilizzando la scala di colori RGB, il numero di frame è stato impostato su 100, invece di 150 come avviene per BABELE, perché alcuni video hanno un numero di frame minore. Queste decisioni sono state prese tenendo conto del fatto che, nel dataset BABELE, l'utilizzo del canale RGB ha prodotto risultati più accurati rispetto alla scala di grigi, mentre la dimensione di 50x50 è stata scelta per ridurre il carico computazionale durante la fase di addestramento. Tuttavia, nel caso in cui vi fossero disponibilità di risorse computazionali più elevate, potrebbe essere opportuno esplorare l'utilizzo di dimensioni maggiori. Al termine di ciascuna operazione di trasformazione, il video è stato salvato in una lista composta da tuple, in cui ogni tupla rappresenta il video insieme alla lingua parlata all'interno del video stesso. In particolare le etichette sono:

- 0 ai video in inglese dunque relativi al dataset MOSEI
- 1 ai video in cinese dunque relativi al dataset SIMS

Dopo aver completato il pre-processing, la lista contenente tutti i video è stata salvata in formato np.array. Questa scelta è considerata una buona pratica, innanzitutto perché l'utilizzo di array numpy consente una rappresentazione efficiente e compatta dei dati video, facilitando così la manipolazione e l'elaborazione successiva. Inoltre, l'uso di array numpy garantisce una maggiore velocità di accesso e operazioni vettoriali ottimizzate, contribuendo così a migliorare le prestazioni complessive del sistema di classificazione. Nella seconda parte del pre-processing i dati in formato np.array sono stati importati, partendo dal Training set di MOSEI e il Training set di SIMS i due sono stati concatenati creando un unico dataset di training set. La stessa cosa è avvenuta per il dataset di test, partendo dal test set di MOSEI e dal test set di SIMS è stato ottenuto un singolo dataset di test attraverso la concatenazione dei due. La sfida che ha limitato l'ottenimento di accuratezze significative, con risultati che oscillavano tra il 25% e il 53% massimo, risiedeva nel fatto che unendo i dataset in questo modo si creavano delle istanze ordinate, dunque dalla riga 0 alla riga 174 vi erano tutti i video di train di MOSEI, e dalla riga 174 alla riga 324 i video di SIMS, questa stessa situazione si verificava per il test set. Di conseguenza, durante le previsioni, la rete non riusciva ad ottenere una sequenza consecutiva di valori, in particolare non riusciva ad identificare 36 classi 0 consecutive seguite da 36 classi 1 consecutive. Per risolvere questa problematica, è stato sufficiente mischiare casualmente le righe all'interno del set di addestramento e del set di test, che erano stati precedentemente combinati. Infine i due dataset sono stati salvati in un np.array e importati nel modello per l'addestramento.

5.2 Addestramento del modello

La prima rete utilizzata per addestrare il set di dati di riferimento è stata quella usata per l'addestramento del dataset BABELE ma in questo caso essendoci solo due categorie da classificare e non 8 il risultato che ha prodotto è stato l'overfitting. Dunque il passaggio successivo è stato quello di modificarla, sostituendo la rete *ConvLSTM2D* con una rete non ricorsiva nota come *Conv1D*, la quale è un tipo di strato di rete neurale artificiale utilizzata per l'elaborazione di dati tridimensionali come immagini o video, questo strato è in grado di riconoscere e analizzare le caratteristiche tridimensionali dell'input e di produrre un output tridimensionale, alla rete è stato aggiunto poi un layer *MaxPooling3D*, il cui scopo è ridurre la dimensionalità delle feature map generate dalle convoluzioni tridimensionali, esso oltre a ridurre la complessità computazionale diminuisce anche il rischio di overfitting, a seguire un layer *Flatten* e per finire un layer *dropout* e un layer *Dense* nel quale sono state inserite due unità come input perchè 2 sono le classi da classificare mentre la funzione di attivazione è la *activation="sigmoid"* la quale viene utilizzata nei problemi di classificazione binaria; l'ultimo step è stato quello di configurare il processo di addestramento utilizzando *model.compile* nel quale sono stati inseriti: *loss='binary_crossentropy'* perchè il problema che si sta affrontando è di tipo binario e *l'optimizer="adam"*, con questo modello dopo aver eseguito l'addestramento è stato ottenuto il 59.72% di accuratezza, molto bassa per un problema di classificazione binaria, di seguito i grafici ottenuti:

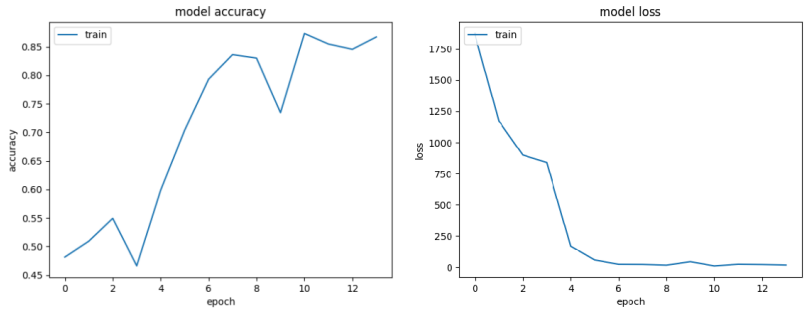


Figura 29: Accuratezza e Loss generati dalla rete neurale

Pertanto il passo successivo è stato quello di variare gli iperparametri: mentre nel tentativo precedente erano stati inseriti 16 filtri nel primo layer con una batch size di 64, in questo tentativo i filtri sono diventati 32 mentre la batch size è diminuita a 32, con queste modifiche l'accuratezza ottenuta è stata del 77.78% con un leggero overfitting da parte del modello.

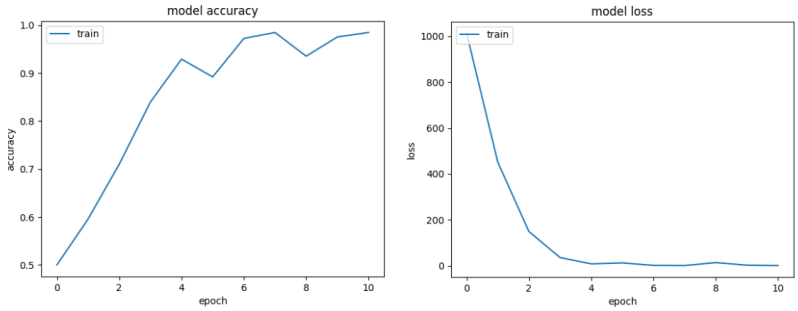


Figura 30: Accuratezza e Loss generati dalla rete neurale

Nell'addestramento successivo per ridurre l'overfitting è stata ridotta da 3 a 1 la soglia di pazienza dell'early stopping, successivamente è stato rimosso il layer *dropout*, mentre i valori di batch size e il numero di filtri nella rete sono rimasti gli stessi; con le variazioni appena illustrate l'accuratezza è salita a 80.56%, con un leggero overfitting.

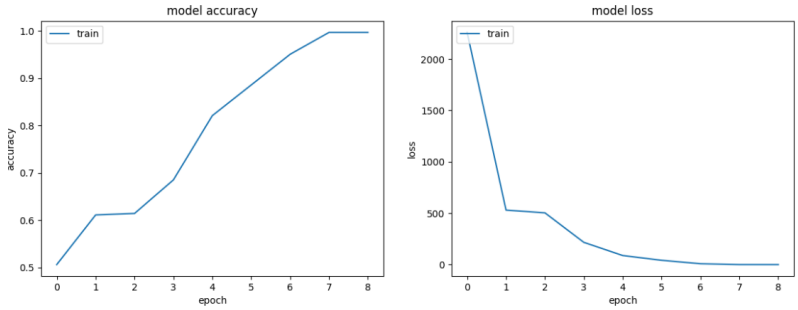


Figura 31: Accuratezza e Loss generati dalla rete neurale

Infine è stata fatta un'ulteriore modifica, i layers da 4 sono diventati 3 perchè è stato rimosso anche il layer *MaxPooling3D* e dopo aver modificato varie volte gli iperparametri per trovare la migliore configurazione è stata raggiunta una buona accuratezza con i seguenti valori; la pazienza nell'early stopping è tornata a 3, i filtri sono scesi a 16 e la batch size è salita a 64, ecco i grafici ottenuti:

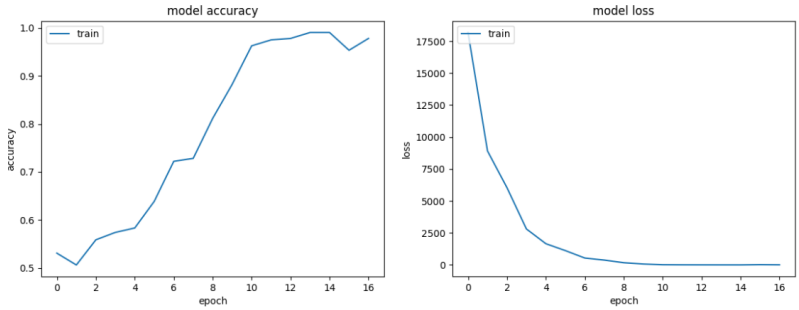


Figura 32: Accuratezza e Loss generati dalla rete neurale

Ecco una tabella riassuntiva di tutte le configurazioni:

	Numero di filtri	Batch size	Accuratezza	Livelli della rete
1° Tentativo	16	64	59.72%	Conv1D + MaxPooling3D + Flatten + Dropout + Dense
2° Tentativo	32	32	77.78%	Conv1D + MaxPooling3D + Flatten + Dropout + Dense
3° Tentativo	32	32	80.56%	Conv1D + MaxPooling3D + Flatten + Dense
4° Tentativo	16	64	87.50%	Conv1D + Flatten + Dense

Figura 33: Rete neurale

Di seguito è illustrata l'intera struttura della rete finale:

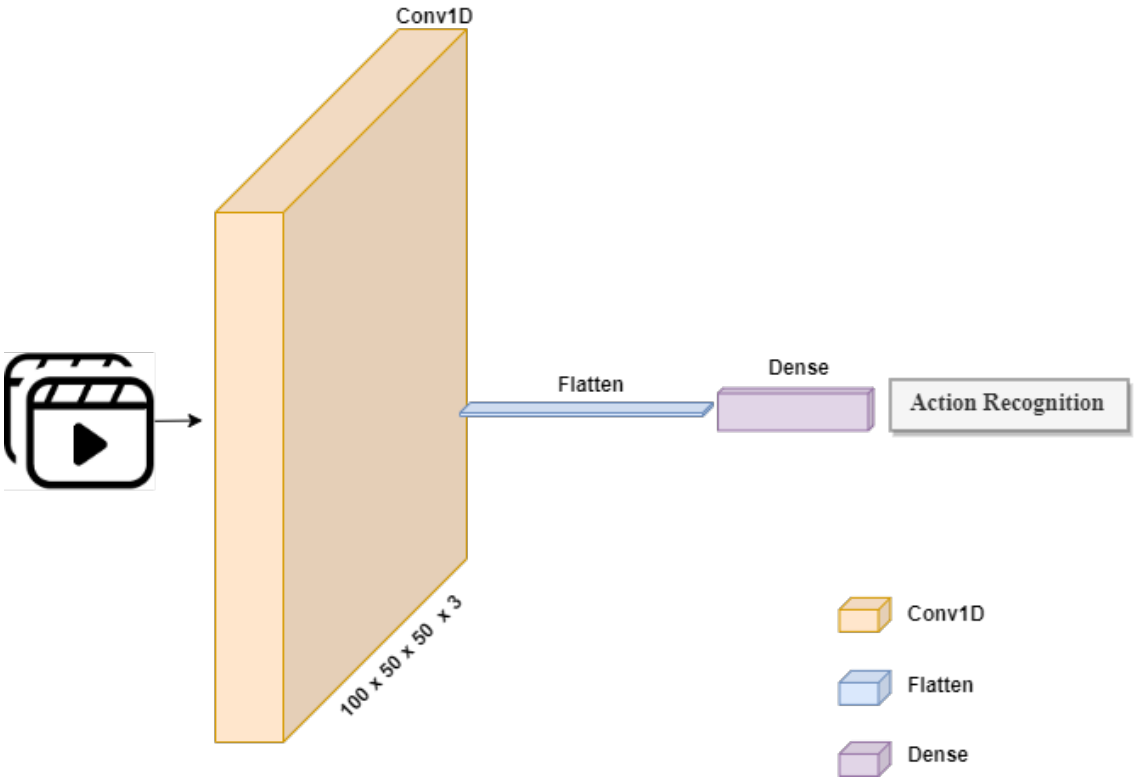


Figura 34: Rete neurale

La tabella successiva riassume tutti i layers e i parametri della rete:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 100, 50, 46, 16)   256

 flatten (Flatten)           (None, 3680000)           0

 dense (Dense)               (None, 1)                 3680001

=================================================================
Total params: 3,680,257
Trainable params: 3,680,257
Non-trainable params: 0
_________________________________________________________________
```

Figura 35: Report di tutti i parametri della rete

5.3 Misure di accuratezza

L'accuratezza raggiunta dal modello utilizzato è del 87.50%. Saranno ora analizzate in dettaglio le performance della rete per ciascuna classe mediante l'esame della matrice di confusione. Questa rappresentazione fornisce un'utile valutazione delle prestazioni del modello di classificazione implementato.

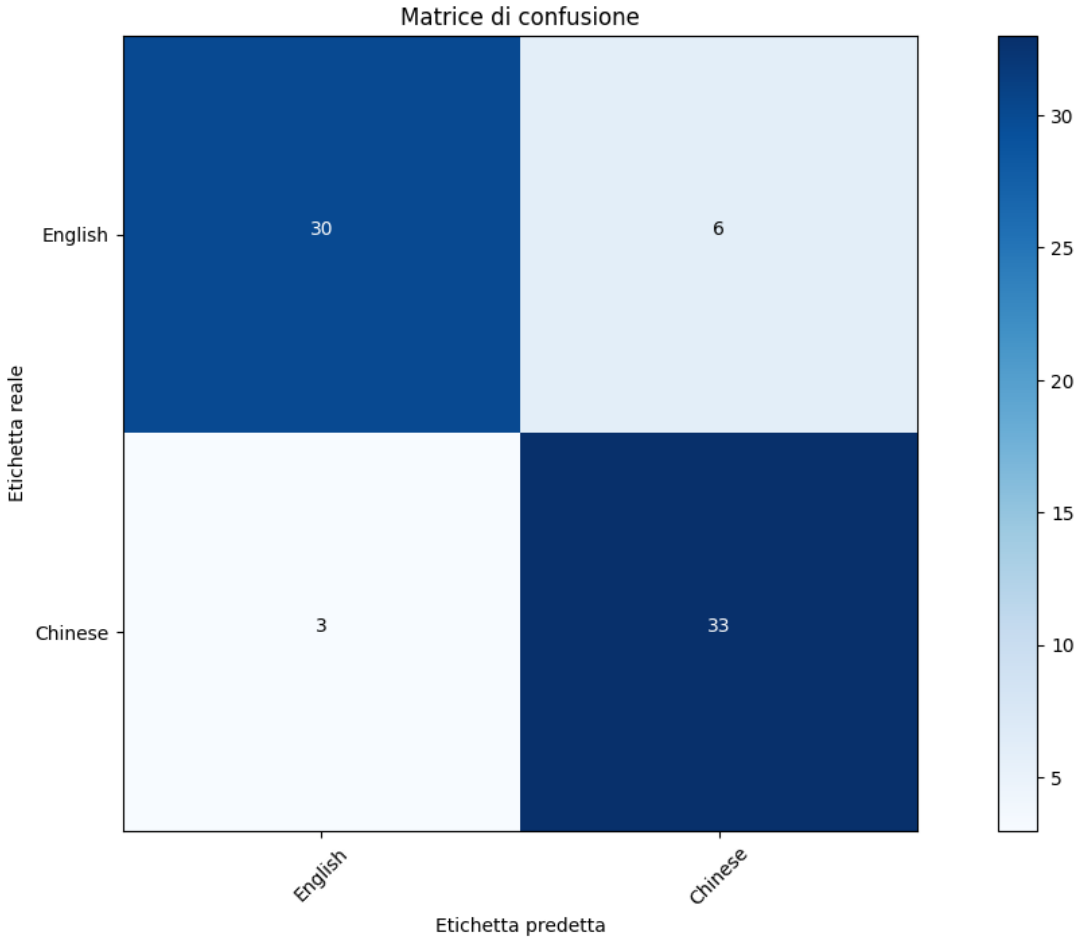


Figura 36: Confusion matrix

È possibile notare che:

- 30 sono i video classificati lingua inglese in cui si parla davvero la lingua inglese
- 3 sono i video classificati inglesi ma che in realtà si parla cinese
- 6 sono i video classificati cinese che nella realtà sono inglese
- infine 33 video sono stati classificati cinesi e sono realmente cinese

Classe	Precision	Recall	F1-score	Accuracy
English	0.91	0.83	0.87	0.83
Chinese	0.85	0.92	0.88	0.92

Figura 37: Confusion matrix

Per la classe inglese il modello fa previsioni con un livello di accuratezza pari all'83% mentre per la classe cinese l'accuratezza è addirittura più alta ovvero pari al 92%.

6 Conclusioni e Lavori futuri

In risposta alla domanda posta inizialmente; È possibile classificare la lingua dal solo movimento delle labbra, si constata che al momento non è fattibile fornire una risposta positiva in quanto i livelli di affidabilità sono limitati. Il miglior risultato che è stato ottenuto dallo studio è del 67% sul test set utilizzando dati a colori e con la tecnica della feature level fusion, che fonde le previsioni di una rete neurale costruita con un layer ConvLSTM2D e le previsioni dei classificatori GaussianProcessClassifier e GaussianNB addestrati su dati estratti dalle distanze di landmark presenti sulle labbra. Attraverso la feature level fusion si è dunque constatata la possibilità di migliorare notevolmente le misure di accuratezza di classificatori che altrimenti avrebbero fornito risultati insoddisfacenti. Ciò implica che migliorando i singoli predittori probabilmente cresceranno anche le accuratezze delle previsioni fornite dall'unione di questi ultimi. Tra le sfide che sono state incontrate sicuramente va citato il trade-off tra accuratezza, overfitting e capacità computazionali, non da meno è stata la fase di pre processing dei vari modelli la quale è stata poco esplorata in termini di combinazioni tra numero totale di frame, frame saltati e risoluzione dei video, a causa del tempo di elaborazione. In contemporanea, il secondo progetto di rilevazione della lingua inglese e cinese ha raggiunto livelli di precisione al di sopra delle aspettative, suscitando orgoglio nel team. Per ulteriori e future ricerche si propone l'utilizzo della tecnica di Mean shift sui dati al fine di ridurre la complessità, la normalizzazione durante la fase di pre processing, in modo da non avere valori che variano tra 0 e 255, bensì in un range tra 0 e 1, una tecnica che viene molto utilizzata nella computer vision, ma che non è stato possibile applicare in questo studio. Oltre a questo in futuro la tecnica della feature level fusion deve essere utilizzata in combinazione con modelli predittivi migliori. Proprio per quanto concerne il miglioramento dei modelli si consiglia l'aumento del numero dei filtri in contemporanea con un aumento della batch size oltre che l'aggiunta di nuovi layer di Dropout, che come visto nella fase di addestramento è la tecnica migliore per ottenere risultati adeguati in termine di accuratezza e in linea con il problema dell'overfitting. Per finire il presente studio rappresenta un significativo passo avanti considerando lo stato dell'arte. Grazie all'implementazione di approcci innovativi e alla feature level fusion, siamo riusciti ad ottenere risultati degni di nota, aprendo nuove prospettive future per la rilevazione della lingua attraverso il solo movimento delle labbra. Siamo entusiasti dei risultati ottenuti grazie alla costante dedizione per il miglioramento che ci ha portati all'ottenimento di soluzioni sempre migliori ed avanzate.

Riferimenti bibliografici

[Gé19] Aurélien Géron. Chapter 14: Deep computer vision using convolutional neural networks. In *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, pages 592–663. O'Reilly Media, Sebastopol, CA, third edition, 2019.

[RKBT07] Ajita Rattani, Dakshina Ranjan Kisku, Manuele Bicego, and Massimo Tistarelli. Feature level fusion of face and fingerprint biometrics. In *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–6, Crystal City, VA, USA, 2007. https://ieeexplore.ieee.org/abstract/document/4401919.

[SCW+15] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network. *arXiv preprint arXiv:1506.04214*, 2015. https://arxiv.org/abs/1506.04214.