# BABELE
# Riconoscimento della lingua parlata

**SUMMARY**

Il riconoscimento della lingua di un soggetto è un'area di ricerca molto interessante e innovativa. In questo paper, presentiamo un approccio basato su reti neurali convoluzionali, sperimentando sia con livelli 2D che 3D, per riconoscere la lingua parlata analizzando movimenti delle labbra di una persona durante la produzione del parlato.
Questa tecnica può essere utile in diverse situazioni, ad esempio, quando l'audio non è disponibile o di scarsa qualità, o quando è necessario interpretare il parlato in ambienti rumorosi.

## Il problema

Il riconoscimento delle labbra per identificare la lingua parlata è un problema complesso nel campo dell'elaborazione del linguaggio naturale e del riconoscimento del parlato. Questo problema presenta diverse sfide significative. Una delle sfide principali è la variabilità tra le labbra dei diversi individui. Le labbra possono differire in forma, colore, dimensioni e movimenti, rendendo difficile estrarre caratteristiche discriminanti che possano essere utilizzate per identificare in modo affidabile la lingua parlata. Inoltre, le labbra possono assumere diverse posizioni durante la produzione dei suoni, aggiungendo ulteriore complessità al problema.

Inoltre, la presenza di baffi, barba o altre caratteristiche facciali può ostacolare ulteriormente la corretta interpretazione dei movimenti delle labbra. L'illuminazione e l'angolazione della telecamera possono influenzare la visibilità dei movimenti, e la qualità delle informazioni visive disponibili.

## Una possibile soluzione

Per affrontare le sfide del riconoscimento delle labbra per identificare la lingua parlata, una possibile soluzione è l'utilizzo di reti neurali convoluzionali 3D (3D CNN). Ciò consente di sfruttare le informazioni spazio-temporali presenti nelle sequenze di immagini delle labbra. Gli strati convoluzionali 3D analizzano l'informazione visiva su più dimensioni, considerando sia le correlazioni spaziali all'interno di ogni immagine delle labbra, sia le correlazioni temporali tra diverse immagini. Ciò permette alla rete di estrarre automaticamente le caratteristiche rilevanti per il riconoscimento della lingua parlata dalle sequenze di immagini delle labbra.

Per superare queste sfide, il riconoscimento del parlato da labbra fa spesso affidamento su approcci di intelligenza artificiale, come l'apprendimento automatico e le reti neurali, per analizzare i movimenti delle labbra e associarli a parole o frasi specifiche. L'utilizzo di grandi dataset di video labiali annotati con trascrizioni audio corrispondenti è fondamentale per addestrare e sviluppare modelli di riconoscimento del parlato da labbra accurati.

# Setup del sistema

## Passo 1: Creazione dell'ambiente virtuale

Per creare un ambiente virtuale utilizzando venv con Python 3.9, segui questi passaggi:

1. Assicurati di avere Python 3.9 installato sul tuo sistema. Puoi scaricarlo dal sito ufficiale di Python e seguire le istruzioni per l'installazione corretta.
2. Apri il terminale o il prompt dei comandi sul tuo sistema operativo.
3. Crea una nuova directory in cui desideri posizionare il tuo ambiente virtuale. Ad esempio, puoi creare una cartella chiamata "myenv" nella tua directory di lavoro corrente.
4. Passa alla directory appena creata utilizzando il comando `cd` seguito dal percorso della cartella. Ad esempio, se la tua cartella è "myenv" e si trova nella tua directory home, puoi utilizzare il comando `cd myenv` per passare a quella directory.
5. Una volta all'interno della directory "myenv", puoi creare il tuo ambiente virtuale utilizzando il modulo `venv` di Python. Esegui il seguente comando nel terminale o nel prompt dei comandi:

   ```
   python3.9 -m venv myenv
   ```

   Questo comando utilizzerà la versione specifica di Python 3.9 per creare un ambiente virtuale chiamato "myenv".
6. Dopo aver eseguito il comando, verrà creata una nuova directory chiamata "myenv" che conterrà tutti i file necessari per il tuo ambiente virtuale.
7. Attiva l'ambiente virtuale eseguendo uno dei seguenti comandi, a seconda del tuo sistema operativo:
   - Su macOS e Linux:

     ```
     source myenv/bin/activate
     ```

   - Su Windows:

     ```
     myenv\Scripts\activate
     ```

## Passo 2: Estrazione dell'archivio

Prima di tutto, assicurati di avere scaricato il file archivio zip chiamato *progetto_babele.zip*. Per avviare il progetto, dovrai estrarre tutto il contenuto di questo archivio.

## Passo 3: Installazione dei requisiti

Una volta estratto l'archivio, troverai un file chiamato *requirements.txt*. Questo file contiene una lista di librerie di cui il progetto dipende. Per installare correttamente le librerie richieste, segui i seguenti passaggi:

1. Apri il tuo terminale o prompt dei comandi.
2. Naviga nella directory principale del progetto (dove si trova il file *requirements.txt*).
3. Esegui il seguente comando per installare le librerie tramite pip:

```
pip install -r requirements.txt
```

Assicurati di avere una versione aggiornata di pip installata sul tuo sistema.

## Passo 4: Cartella "videos"

Una volta completata l'installazione delle librerie, troverai una cartella chiamata *videos* nella directory principale del progetto. Questa cartella contiene 8 sottocartelle numerate da 1 a 8. Ogni sottocartella contiene video relativi a una lingua specifica che sono rappresentate nella tabella sottostante.

**Table 1: Suddivisione lingue**

| Lingua | Classe |
|---|---|
| Italiano | 1 |
| Inglese | 2 |
| Tedesco | 3 |
| Spagnolo | 4 |
| Olandese | 5 |
| Russo | 6 |
| Giapponese | 7 |
| Francese | 8 |

## Passo 5: Avvio del progetto

Ora sei pronto per avviare il progetto. Assicurati di avere Jupyter Notebook installato sul tuo sistema. Segui i passaggi seguenti:

1. Apri Jupyter Notebook.
2. Naviga nella directory principale del progetto e scegli il modello da eseguire.
3. Scegliere se si vuole addestrare (nel nome del file c'è "Train") o caricare il modello (nel nome del file c'è "Load")
4. Cerca e apri il file Jupyter Notebook (con estensione *.ipynb*).
5. Verifica che tutte le librerie siano state installate correttamente eseguendo le prime celle di codice del notebook.
6. Cambia i path delle directory in cui sono contenuti i video.
7. Avvia il caricamento o l'addestramento del modello eseguendo le celle successive nel notebook.

# Fase di Pre-processing

Le immagini a colori contengono una vasta quantità di informazioni, ma spesso possono presentare sfide nella fase di elaborazione, come la complessità computazionale e la presenza di rumore. Per superare queste difficoltà, la trasformazione delle immagini a colori in bianco e nero è diventata una pratica comune nel pre-processing delle immagini. Questo processo semplifica la rappresentazione dei dati, riduce il rumore e facilita l'analisi delle caratteristiche visive.

Nel nostro studio, abbiamo implementato un algoritmo di pre-processing per la trasformazione di immagini a colori in bianco e nero. Siamo partiti da un dataset di video a colori da cui abbiamo estratto dei frame in bianco e nero. Utilizzando tecniche di conversione basate su modelli di colore come la scala di grigi, siamo stati in grado di preservare le caratteristiche visive chiave delle immagini originali, mentre eliminavamo le informazioni di colore superflue.

## Risultati e Considerazioni

Attraverso l'applicazione della nostra metodologia di pre-processing, abbiamo osservato diversi vantaggi significativi. In primo luogo, la trasformazione in bianco e nero ha ridotto la complessità computazionale, consentendo un'elaborazione più rapida dei frames. Questo ha portato a un significativo miglioramento delle prestazioni in termini di velocità di computazione rispetto all'elaborazione di immagini a colori. Tuttavia, abbiamo notato che l'utilizzo di frames in bianco e nero ha influenzato i risultati delle reti neurali addestrate su tali dati.

Le reti neurali addestrate utilizzando i frames in bianco e nero hanno mostrato risultati più scarsi rispetto a quelle addestrate su immagini a colori. Questo può essere attribuito alla perdita di informazioni dettagliate presenti nel colore delle immagini originali. I dettagli cromatici, come le diverse tonalità di rosso, rosa o marrone sulle labbra, possono fornire informazioni importanti per il riconoscimento dei movimenti delle labbra. Di conseguenza, nonostante i benefici di velocità di computazione, è importante considerare il trade-off tra prestazioni e complessità computazionale quando si utilizzano frames in bianco e nero.
Come possiamo notare dai risultati delle sperimentazioni in basso, i risultati sono stati discreti, ma hanno comunque dimostrato di peggiorare i risultati delle reti da noi utilizzate. Pertanto abbiamo deciso di non proseguire in questa direzione.

Di seguito riportiamo durante il capitolo dedicato alle sperimentazioni riportiamo le 2 configurazioni provate (una Rete 2D e una Rete 3D)

# Metodologia e Analisi dei Risultati

Abbiamo valutato le prestazioni del nostro sistema utilizzando metriche come l'accuratezza e la loss. Inoltre per valutare l'efficacia di una rete neurale convoluzionale (CNN) nella classificazione di dati, vengono utilizzate le matrici di confusione. Questa matrice organizza le previsioni del modello rispetto alle classi reali, consentendo di identificare le predizioni corrette (veri positivi e veri negativi) e gli errori di classificazione (falsi positivi e falsi negativi).

Le matrici di confusione consentono di calcolare diverse metriche di valutazione delle prestazioni del modello, come l'accuratezza, la precisione (**precision**), e il richiamo (**recall**). Queste metriche forniscono informazioni sulle performance del modello in termini di capacità di identificare correttamente le diverse classi o categorie di dati.

Durante tutto il progetto abbiamo portato avanti in parallelo due approcci, quello delle reti convoluzionali **2D** (non tengono conto della sequenza temporale tra i frames) e quello delle reti convoluzionali **3D** (tengono conto della sequenza temporale tra i frames).

Per ogni sperimentazione riportata su questa guida si avranno i seguenti dati:

- **Tabella dei parametri** della rete neurale
- **Tabella dei valori** di **precision** e **recall**
- **Grafico** della **accuracy** e **loss** durate la fase di **Train**
- **Matrice di confusione** basata sul dataset di **Train**
- **Matrice di confusione** basata sul dataset di **Test**
- **Valori di accuracy** e **loss** della rete sul dataset di **Test**

La rete prende in input una sequenza di frames, che rappresentano un video, e che sono etichettati con la lingua associata.

## Sperimentazioni effettuate

Qui riportiamo una breve tabella che riassume le sperimentazioni che troveremo più avanti nel paper.

| Nome sperimentazione | Test accuracy |
|---|---|
| **3D CNN 150 epoche** | **0.36** |
| 3D CNN 120 epoche | 0.23 |
| 3D CNN 300 epoche | 0.26 |
| 3D CNN 100 epoche (B&W) | 0.28 |
| **2D CNN 150 epoche (con shuffle)** | **0.42** |
| 2D CNN 150 epoche | 0.41 |
| 2D CNN 100 epoche (B&W) | 0.30 |

*In **grassetto** sono riportate le reti migliori*

## Rappresentazione delle reti

Per rappresentare le reti è stato utilizzato un tool online per disegnare e rappresentare diagrammi e oggetti chiamato Drawio.com. Draw.io è un'applicazione web gratuita che consente agli utenti di creare diagrammi e disegni di vario tipo.
In particolare ci siamo riferiti a questo articolo che fornisce una guida per la rappresentazione di reti neurali su Drawio.com
Qui di seguito sono riportate le figure rappresentanti le due reti (3D e 2D) che hanno avuto risultati migliori:

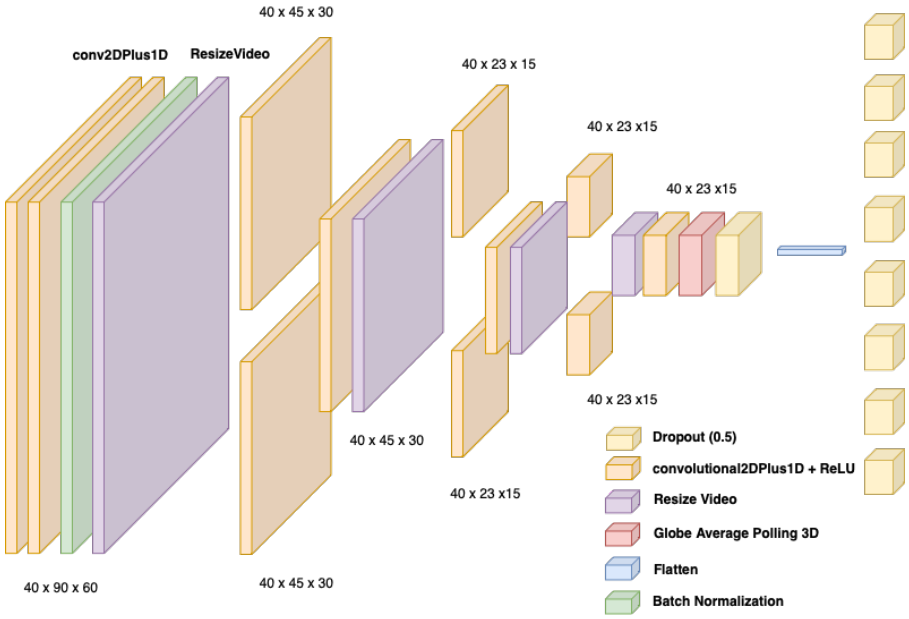



**Fig. 1:** Rappresentazione della rete 3D CNN

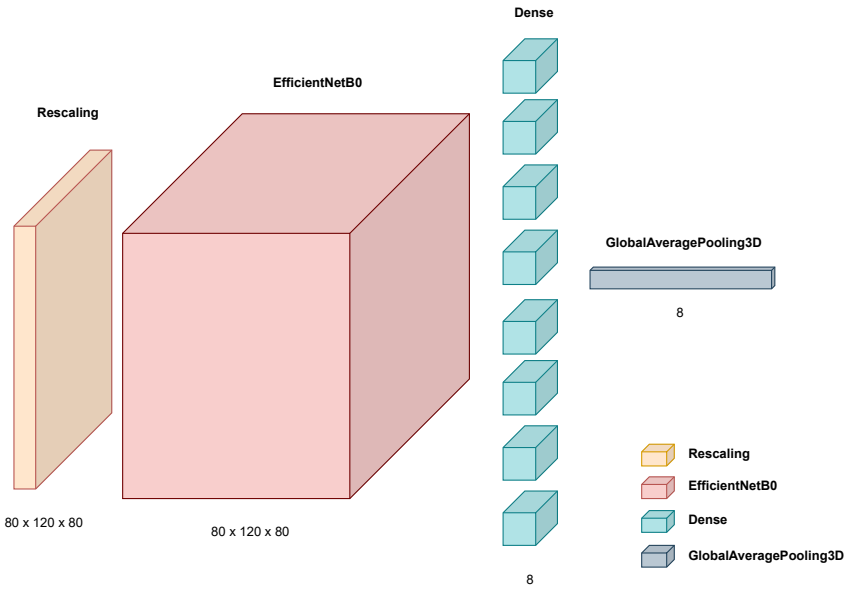
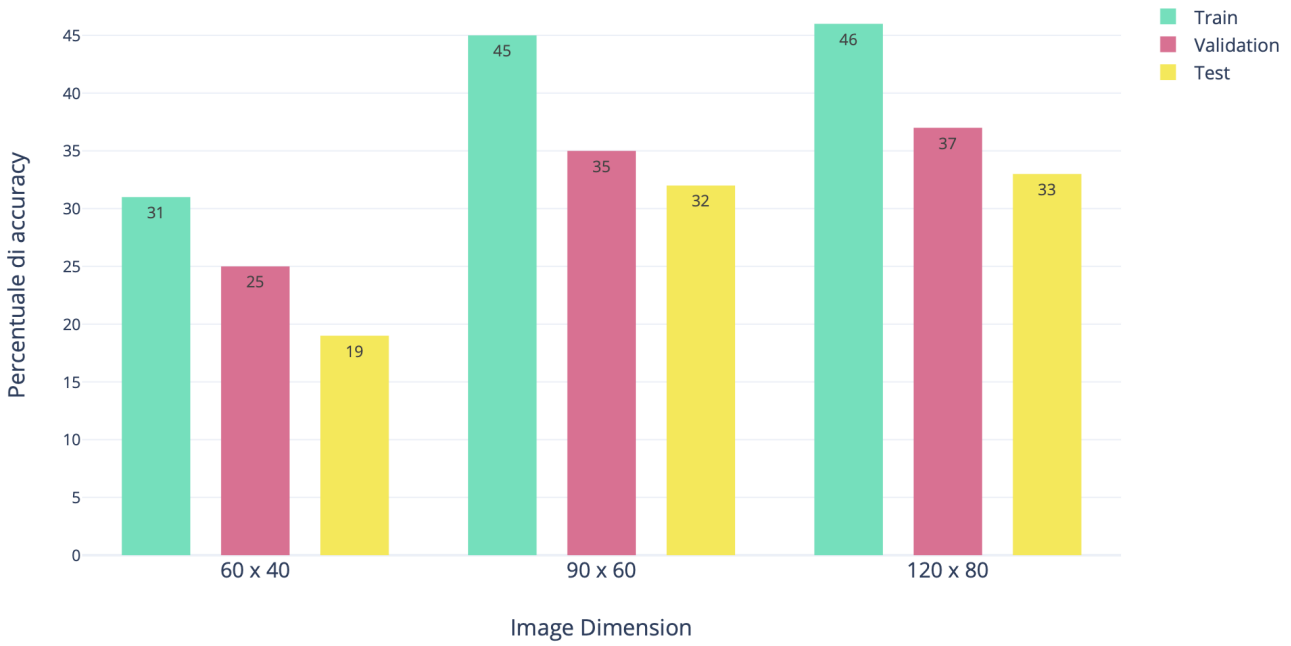



**Fig. 2:** Rappresentazione della rete 2D CNN

# Tuning parametri e risultati 3D CNN

Sono stati effettuati numerosi test, di seguito sono elencati alcuni grafici che mostrano, per ogni parametro che abbiamo ritenuto rilevante, come l'accuratezza del modello può variare se essi subiscono variazioni:

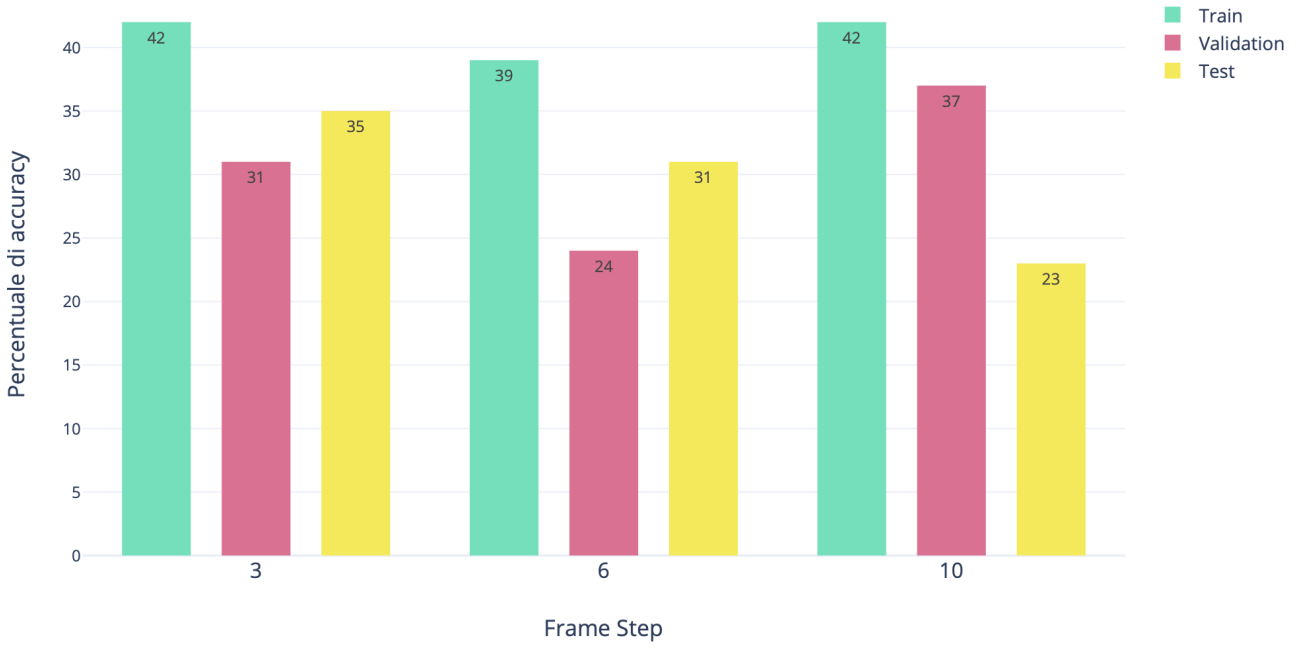
## Image dimension influence 3D CNN




| Parametro | Valore |
|---|---|
| Numero frames | 40 |
| Frame step | 3 |
| Batch Size | 32 |
| Shuffle | False |
| Learning rate | Adam (0.0001) |
| Epoche | 50 |

Per quanto riguarda la dimensione dell'immagine, abbiamo notato miglioramenti passando da 60 x 40 a 90 x 60, ma non ce ne sono stati di significativi portandola a 120 x 80, quindi abbiamo optato per una dimensione di **90 x 60**, dopo aver riscontrato anche che passando da 90 x 60 a 120 x 80, non solo registravamo gli stessi risultati, ma ci portava anche ad avere un grosso carico computazionale che rallentava molto il processo di training.

## Frame step influence 3D CNN

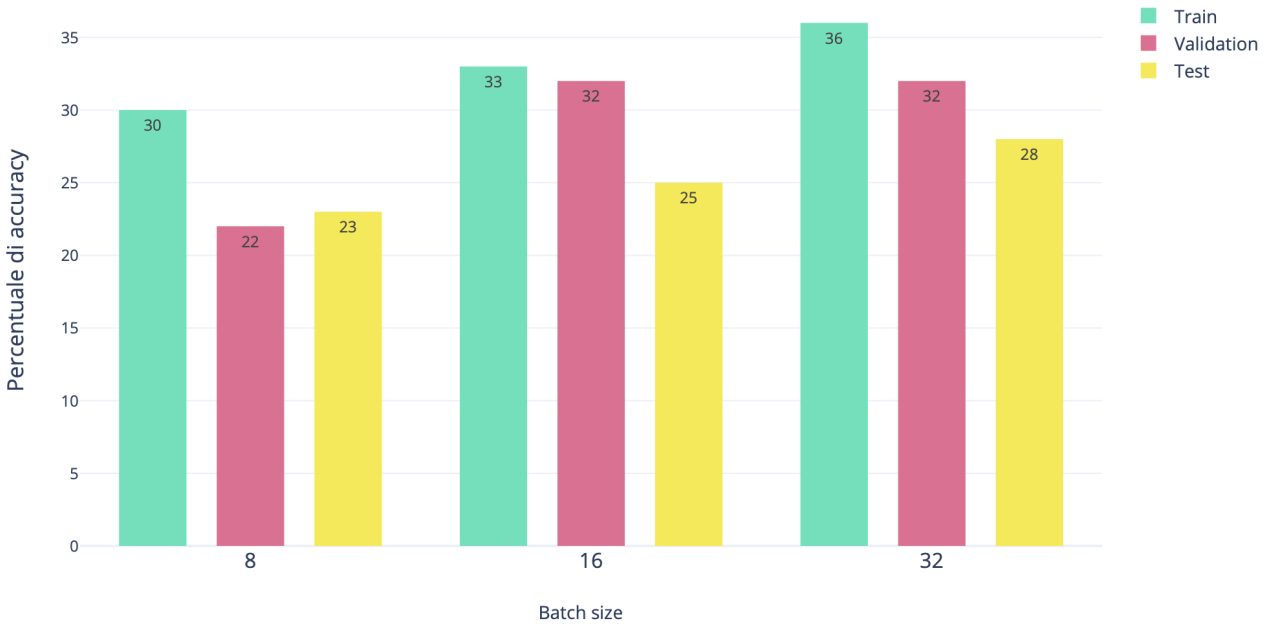



| Parametro | Valore |
|---|---|
| Image res | 90 x 60 |
| Numero frames | 40 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.0001) |
| Epoche | 50 |

Il parametro di **frame step** ci consente di scegliere ogni quanti frame dobbiamo saltare prima di estrarne uno.
Ad esempio in un video di 10 secondi a 24 fps con frame step 10 possiamo estrarre da un totale di 240 frames solo 24 frame, estratti uno ogni 10.

Si è potuto notare come impostando un frame step più basso, generalmente si ottengono risultati migliori, motivo per il quale le sperimentazioni che seguono sono state effettuate con un frame step impostato a 3 (ogni 3 frames ne prendiamo 1).

## Batch size influence 3D CNN



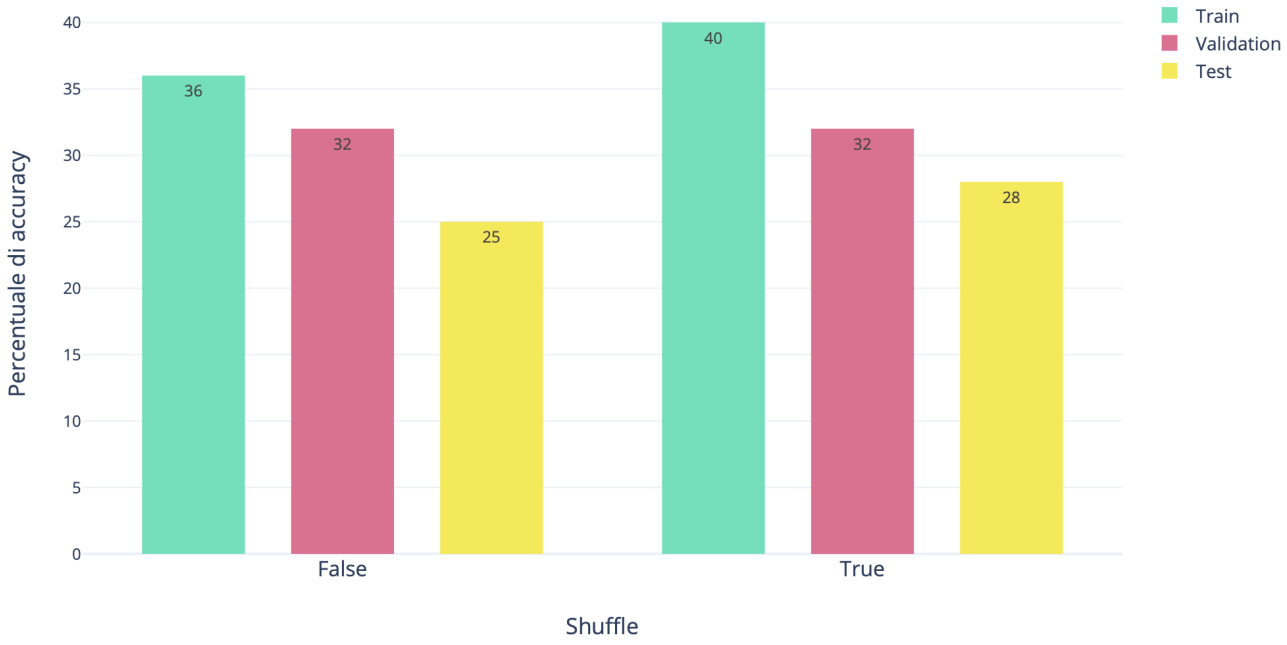

| Parametro | Valore |
|---|---|
| Image res | 90 x 60 |
| Numero frames | 40 |
| Frame step | 3 |
| Shuffle | False |
| Learning rate | Adam (0.0001) |
| Epoche | 50 |

Durante l'addestramento di una rete neurale, i dati di addestramento vengono suddivisi in batch più piccoli per gestire meglio il processo di calcolo e migliorare l'efficienza. Il batch size determina quanti esempi di addestramento vengono utilizzati contemporaneamente per calcolare i gradienti e aggiornare i pesi del modello.

Un batch size più grande comporta il calcolo di gradienti medi su un maggior numero di esempi, consentendo una maggiore stabilità nella direzione dell'aggiornamento dei pesi. Tuttavia, l'utilizzo di un batch size più grande richiede una maggiore memoria e capacità di calcolo, poiché più dati devono essere elaborati contemporaneamente.
Infatti per questo motivo non siamo potuti andare oltre a 32.

Come si può notare nel grafico sovrastante, la **batch size impostata a 32** ha portato in ogni caso a risultati migliori, difatti è stata proprio impostata con questo valore nelle sperimentazioni che seguiranno.

## Shuffle influence 3D CNN




| Parametro | Valore |
|---|---|
| Image res | 90 x 60 |
| Numero frames | 40 |
| Frame step | 3 |
| Batch Size | 32 |
| Learning rate | Adam (0.0001) |
| Epoche | 50 |

Con gli altri parametri impostati come mostrato nella tabella sovrastante, avendo eseguito sempre sperimentazioni con il parametro **Shuffle = false**, la nostra supposizione era che impostando il parametro **Shuffle = true** durante la fase di training avrebbe portato a miglioramenti sulla validation accuracy, ciò però non è accaduto sul validation set (che è peggiorato), bensì è salita la test accuracy.

## 3D CNN 150 epoche (la migliore delle 3D)

| Parametro | Valore |
|---|---|
| Image res | 90 x 60 |
| Numero frames | 40 |
| Frame step | 3 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.001) |
| Space convolution | 7x7 |
| Time convolution | 3 |

| Classe | Precision | Recall |
|---|---|---|
| Italiano | 0.38 | 0.25 |
| Inglese | 0.00 | 0.00 |
| Tedesco | 0.38 | 0.80 |
| Spagnolo | 0.13 | 0.25 |
| Olandese | 0.73 | 0.95 |
| Russo | 0.75 | 0.30 |
| Giapponese | 1.00 | 0.20 |
| Francese | 0.17 | 0.25 |

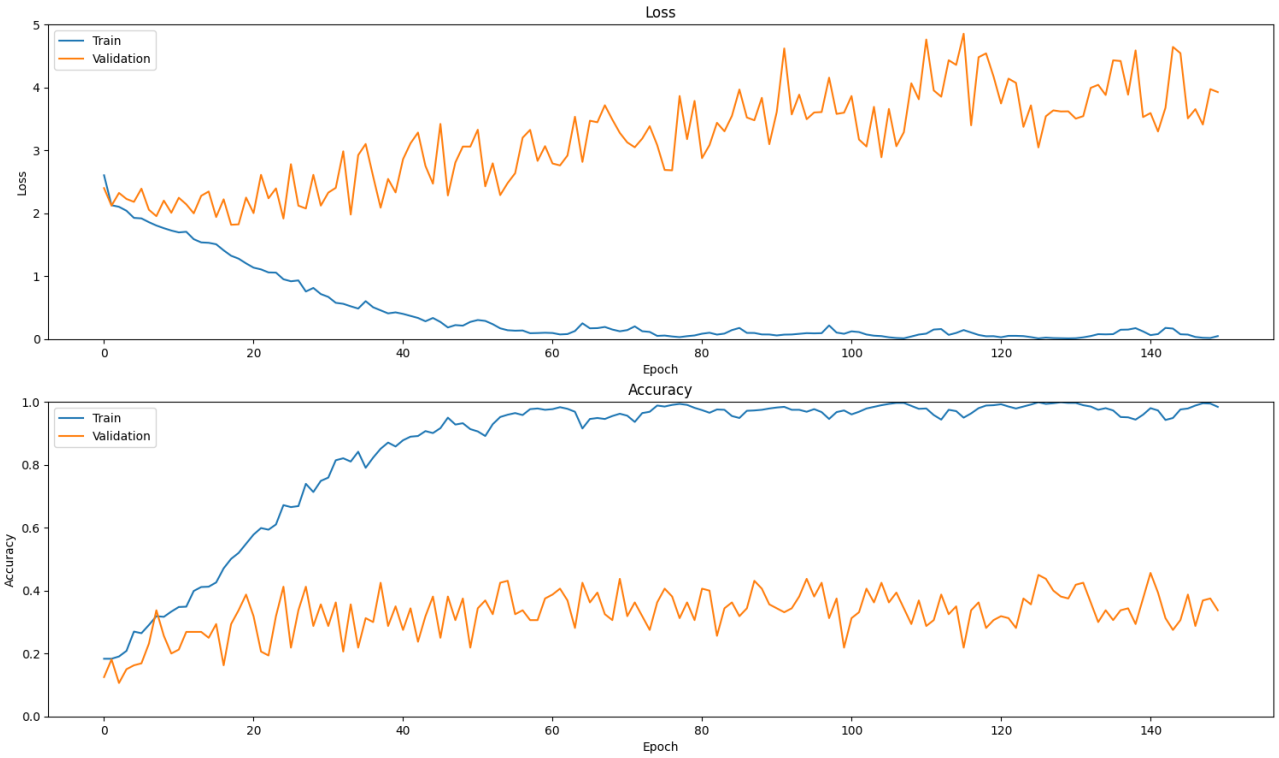



**Fig. 3:** **Grafico Accuracy/Loss**

**Test Accuracy: 0.36**

**Fig. 4:** Train

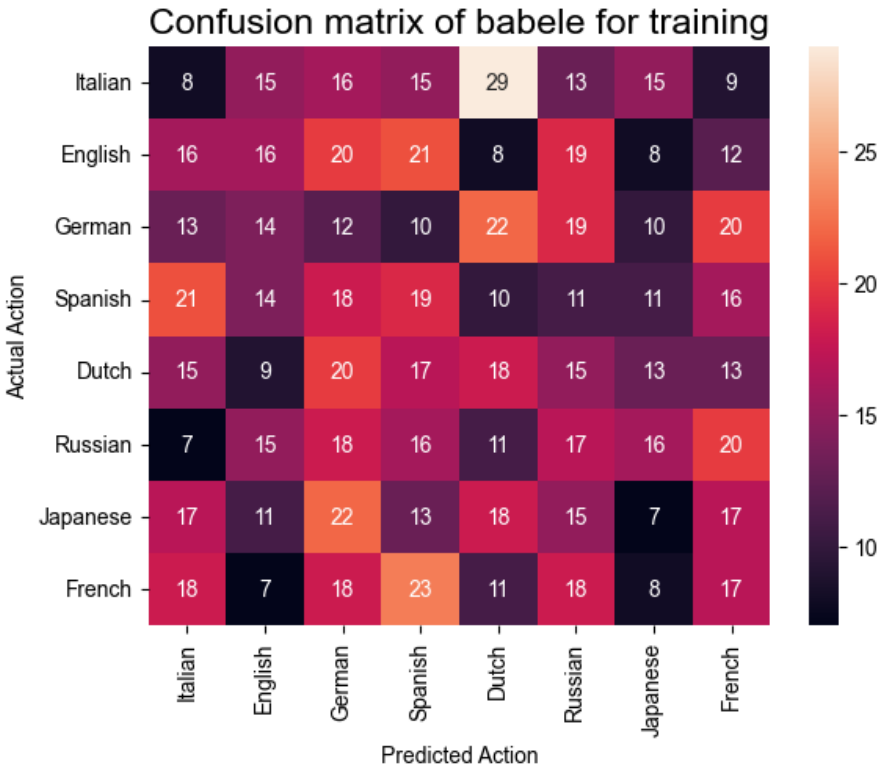
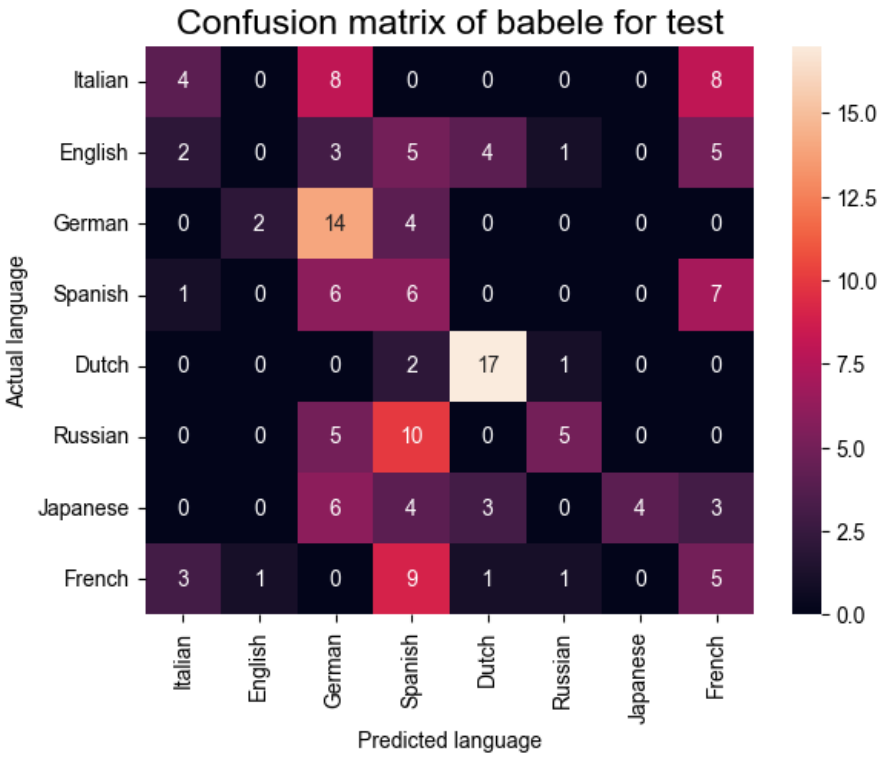



**Fig. 5:** Test



## 3D CNN 120 epoche

| Parametro | Valore |
|---|---|
| Image res | 120 x 80 |
| Numero frames | 80 |
| Frame step | 3 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.0001) |
| Space convolution | 7 x 7 |
| Time convolution | 5 |

| Classe | Precision | Recall |
|---|---|---|
| Italiano | 0.00 | 0.00 |
| Inglese | 0.00 | 0.00 |
| Tedesco | 0.71 | 0.50 |
| Spagnolo | 0.00 | 0.00 |
| Olandese | 0.71 | 0.75 |
| Russo | 0.00 | 0.00 |
| Giapponese | 0.36 | 0.35 |
| Francese | 0.09 | 0.20 |

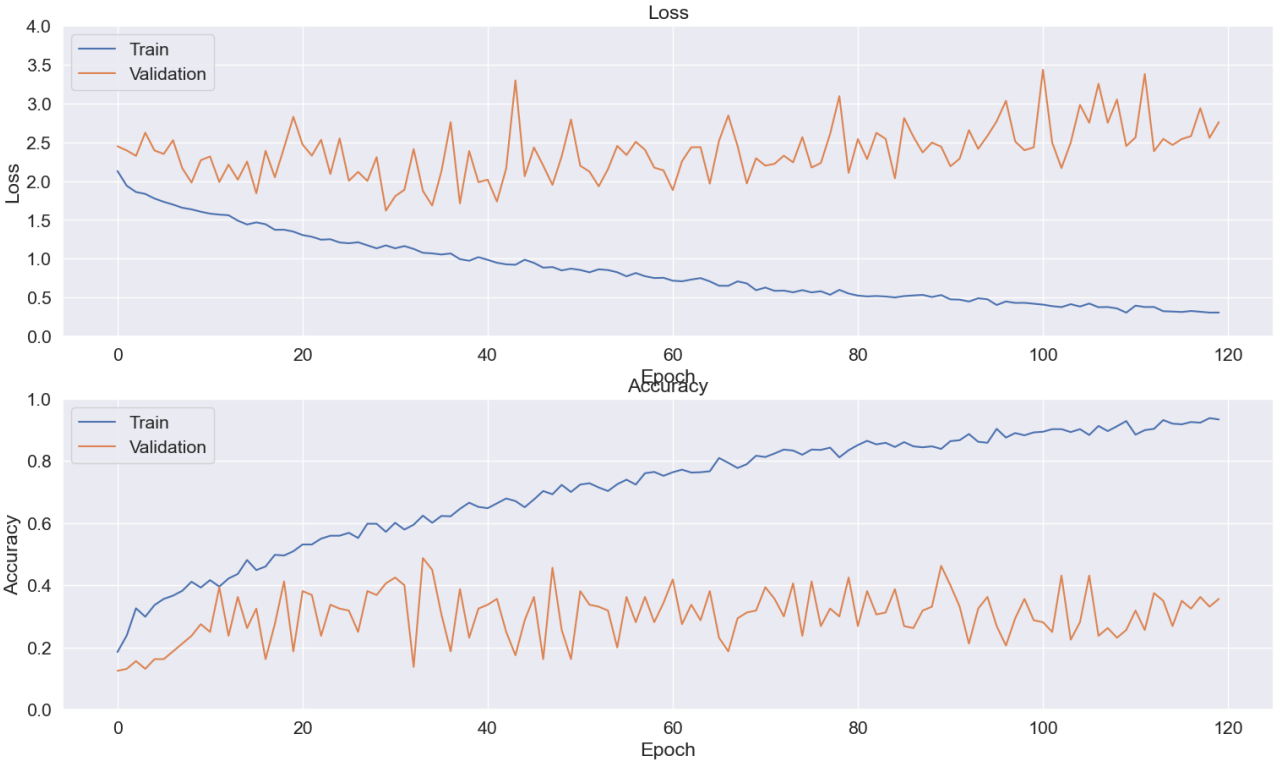



**Fig. 6: Grafico Accuracy/Loss**

**Test Accuracy: 0.23**

## Motivazione

Questa rete è stata riportata per far vedere che anche alzando la correlazione dei frame temporanei il risultato non è migliorato ma è risultato peggiore. Anche avendo più frame in input e quindi più dati da elaborare i risultati di questo approccio sono stati molto deludenti, quindi abbiamo abbandonato questo approccio.

**Fig. 7:** Train

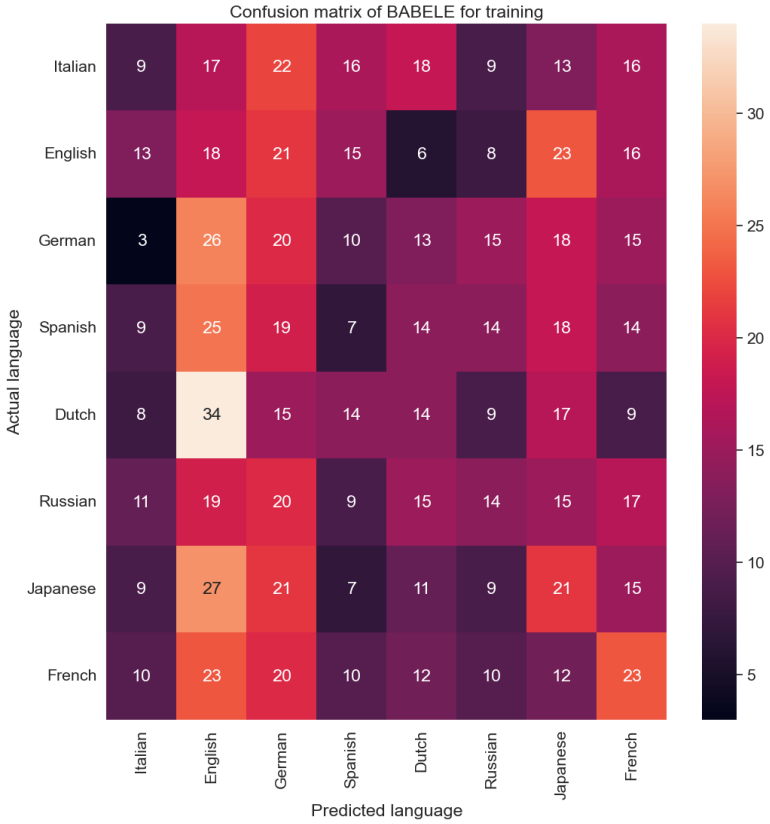
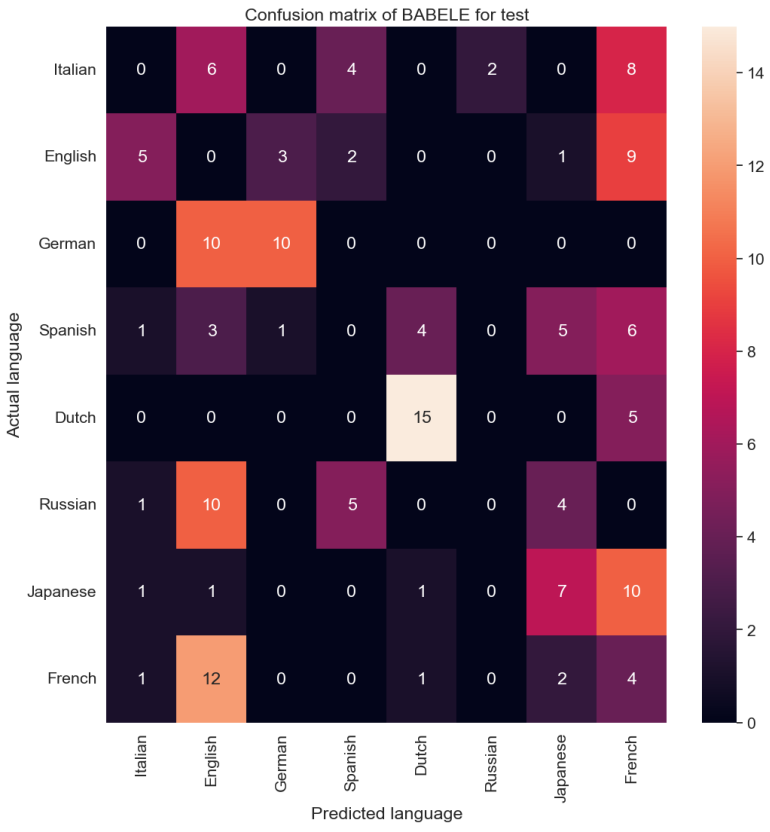



**Fig. 8:** Test



## 3D CNN 300 epoche

| Parametro | Valore |
|---|---|
| Image res | 60 x 40 |
| Numero frames | 12 |
| Frame step | 3 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.001) |
| Space convolution | 7 x 7 |
| Time convolution | 3 |

| Classe | Precision | Recall |
|---|---|---|
| Italiano | 0.23 | 0.60 |
| Inglese | 0.62 | 0.05 |
| Tedesco | 0.35 | 0.35 |
| Spagnolo | 0.00 | 0.00 |
| Olandese | 0.54 | 0.30 |
| Russo | 0.52 | 0.50 |
| Giapponese | 0.50 | 0.20 |
| Francese | 0.20 | 0.25 |

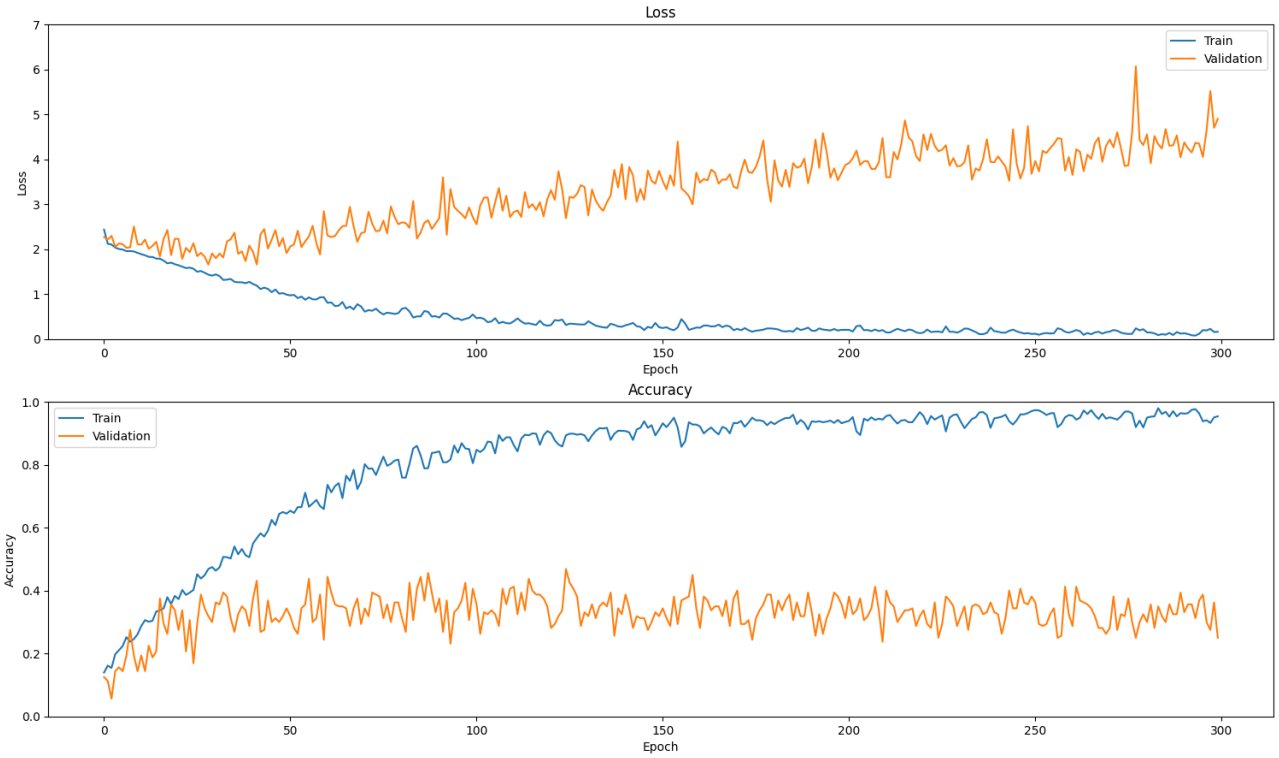



**Fig. 9:** **Grafico Accuracy/Loss**

**Test Accuracy: 0.26**

**Fig. 10:** Train

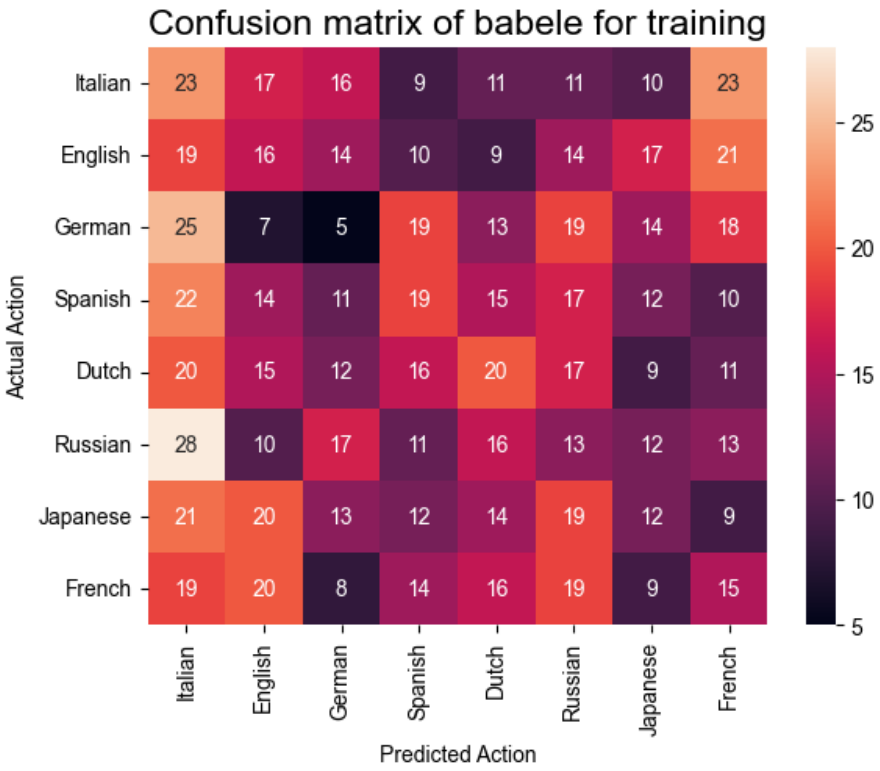
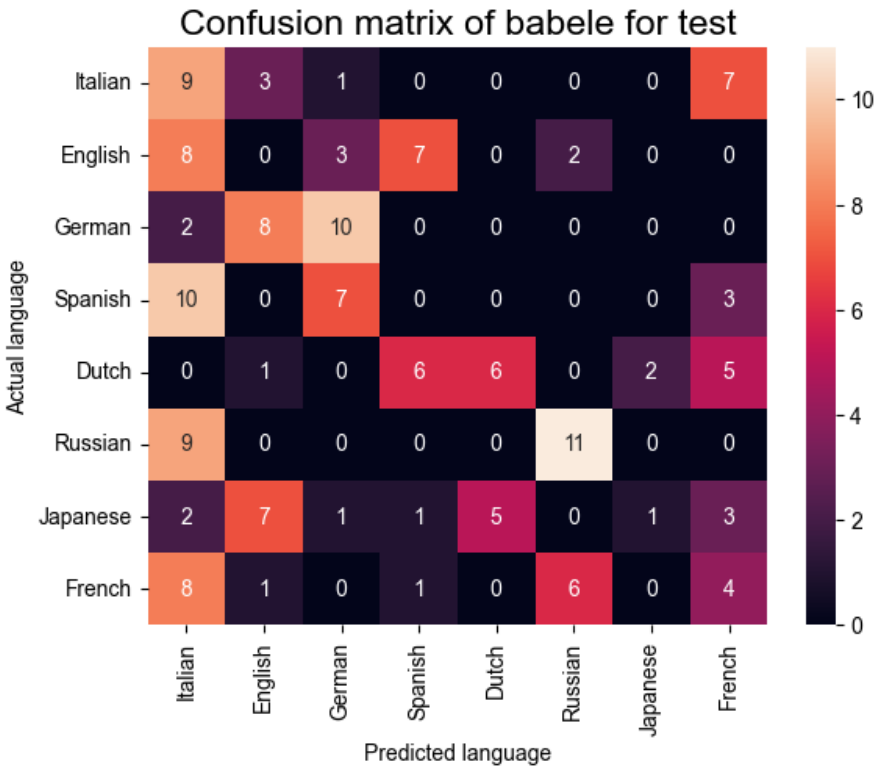



**Fig. 11:** Test



## 3D CNN 100 epoche con frames in Bianco e Nero

| Parametro | Valore |
|---|---|
| Image res | 60 x 40 |
| Numero frames | 40 |
| Frame step | 3 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.0001) |
| Space convolution | 7 x 7 |
| Time convolution | 5 |

| Classe | Precision | Recall |
|---|---|---|
| Italiano | 0.32 | 0.40 |
| Inglese | 0.11 | 0.10 |
| Tedesco | 0.53 | 0.45 |
| Spagnolo | 0.00 | 0.00 |
| Olandese | 0.63 | 0.95 |
| Russo | 0.57 | 0.20 |
| Giapponese | 0.11 | 0.20 |
| Francese | 0.22 | 0.10 |

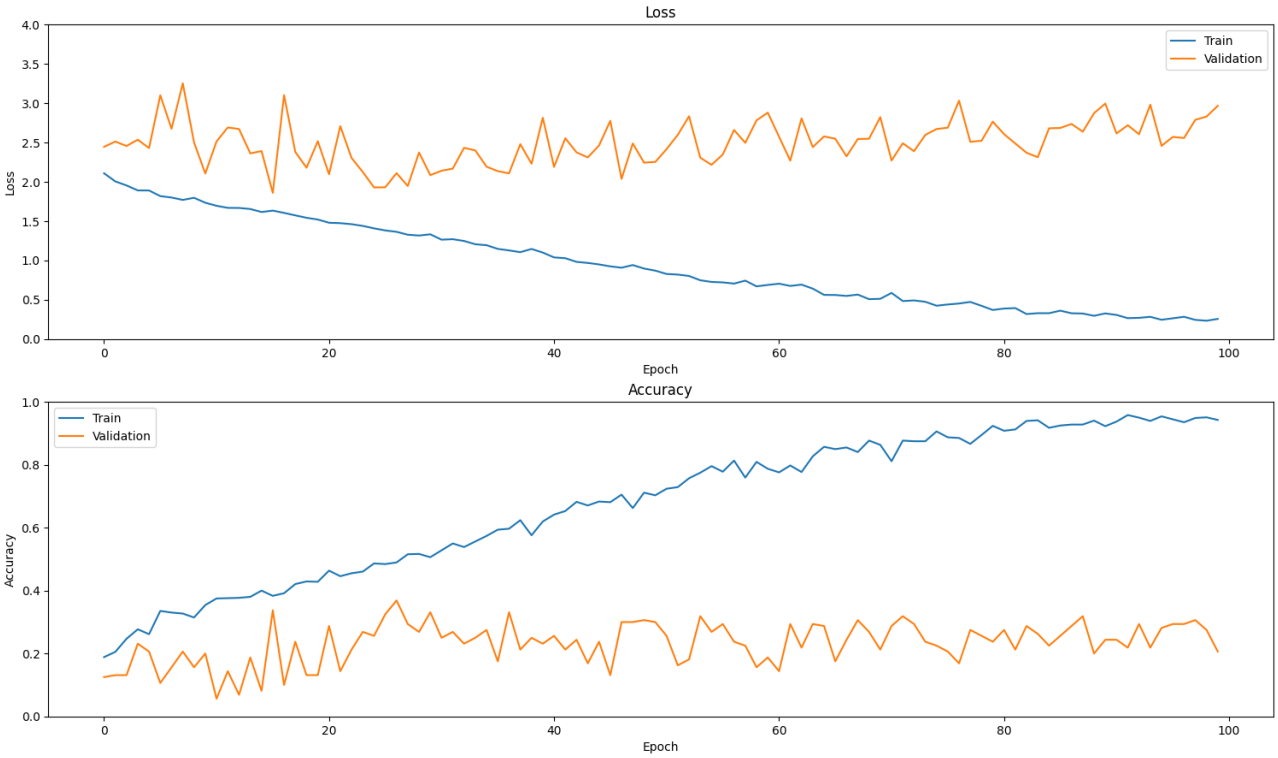



**Fig. 12: Grafico Accuracy/Loss**

**Test Accuracy: 0.28**

Fig. 13: Train

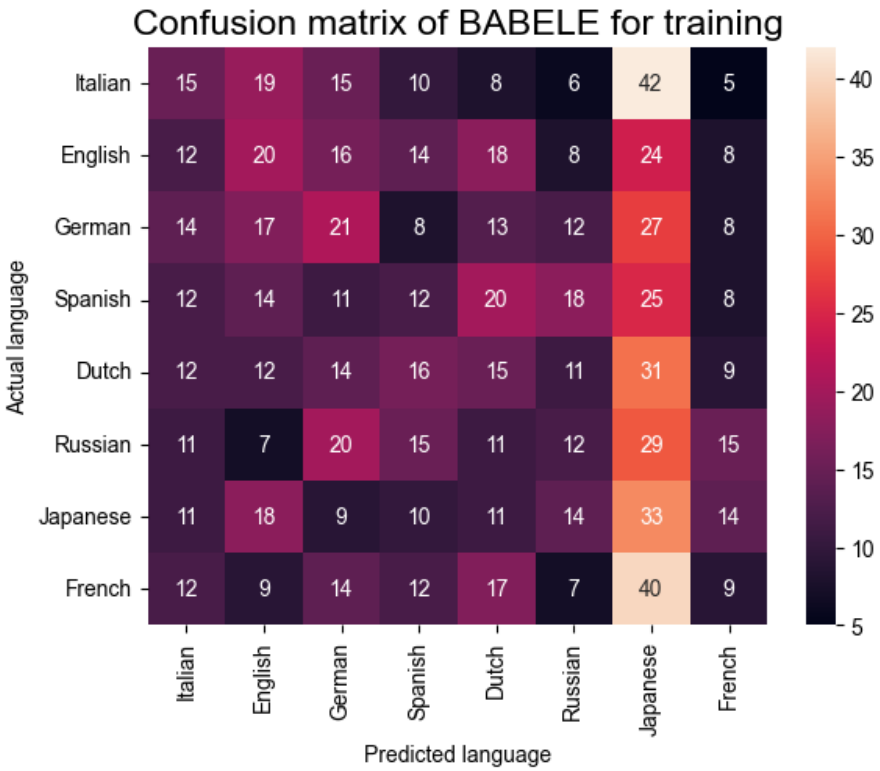
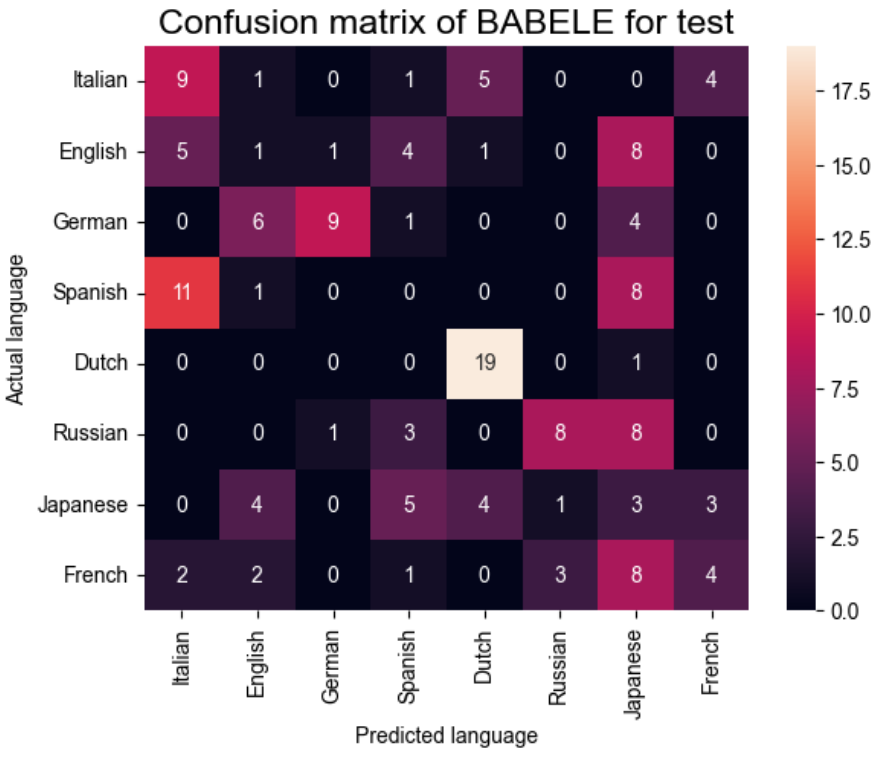
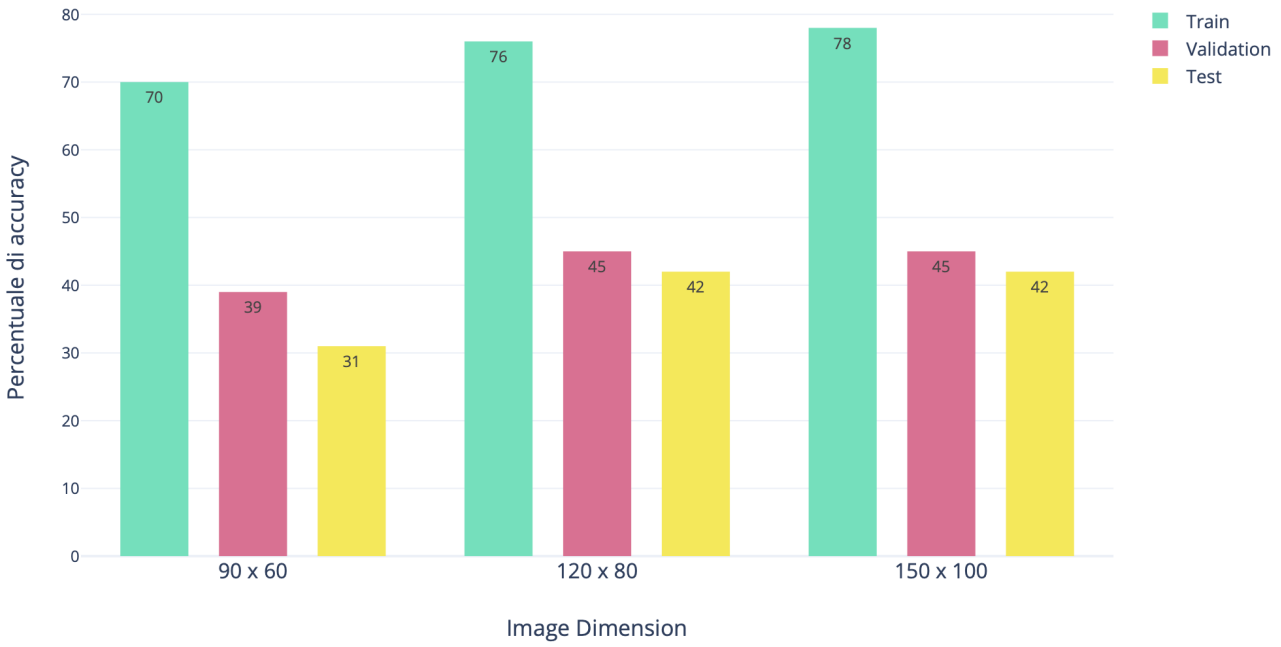



Fig. 14: Test



# Tuning parametri e risultati 2D CNN

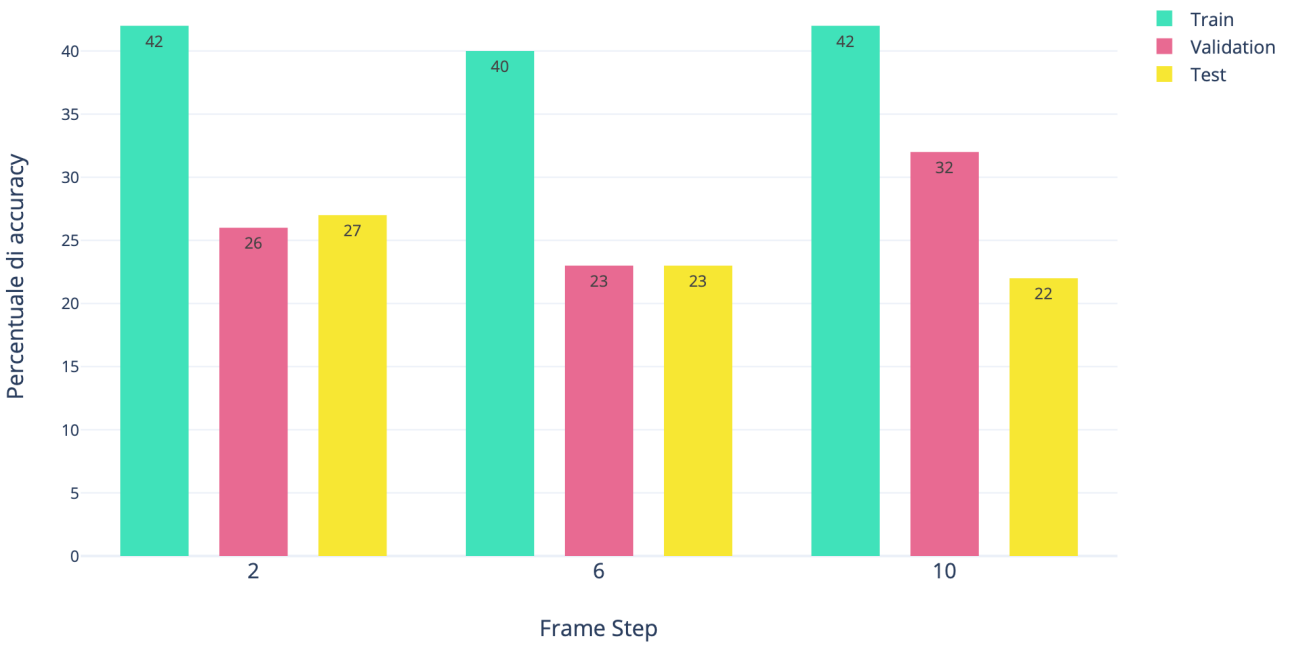
## Image Dimension influence in 2D CNN




| Parametro | Valore |
|---|---|
| Numero frames | 40 |
| Frame step | 2 |
| Batch size | 32 |
| Shuffle | False |
| Learning rate | Adam (0.0001) |
| Epoche | 70 |

Così come per la rete 3D, per quanto riguarda la dimensione dell'immagine, abbiamo notato miglioramenti passando da 90 x 60 a 120 x 80, ma non ce ne sono stati di significativi portandola a 150 x 100, quindi abbiamo optato per una dimensione di **120 x 80**, dopo aver notato anche che passando da 120 x 80 a 150 x 100, non solo ci portava ad avere gli stessi risultati, ma ci portava anche ad avere un grosso carico computazionale.

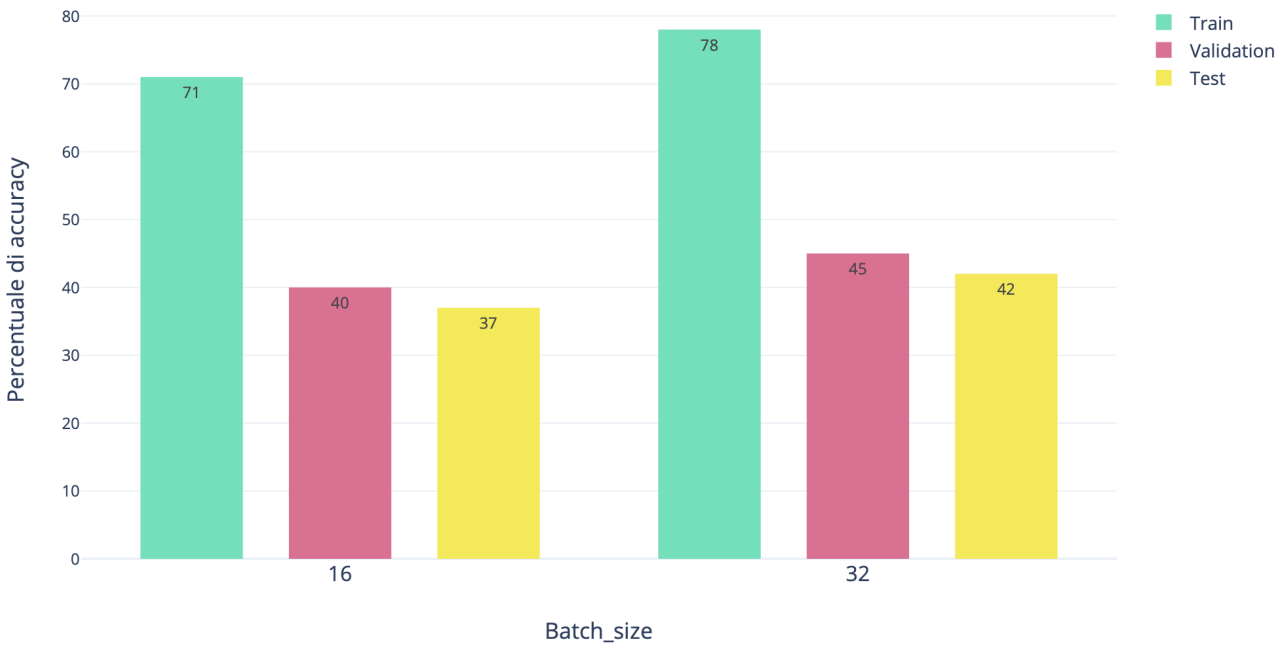
## Frame step influence in 2D CNN




| Parametro | Valore |
|---|---|
| Image res | 120 x 80 |
| Numero frames | 40 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.0001) |
| Epoche | 50 |

Si è potuto notare come impostando un frame step più basso, generalmente si ottengono risultati migliori, motivo per il quale le sperimentazioni che seguono sono state effettuate con un frame step impostato a 2 (ogni 2 frames ne prendiamo 1).

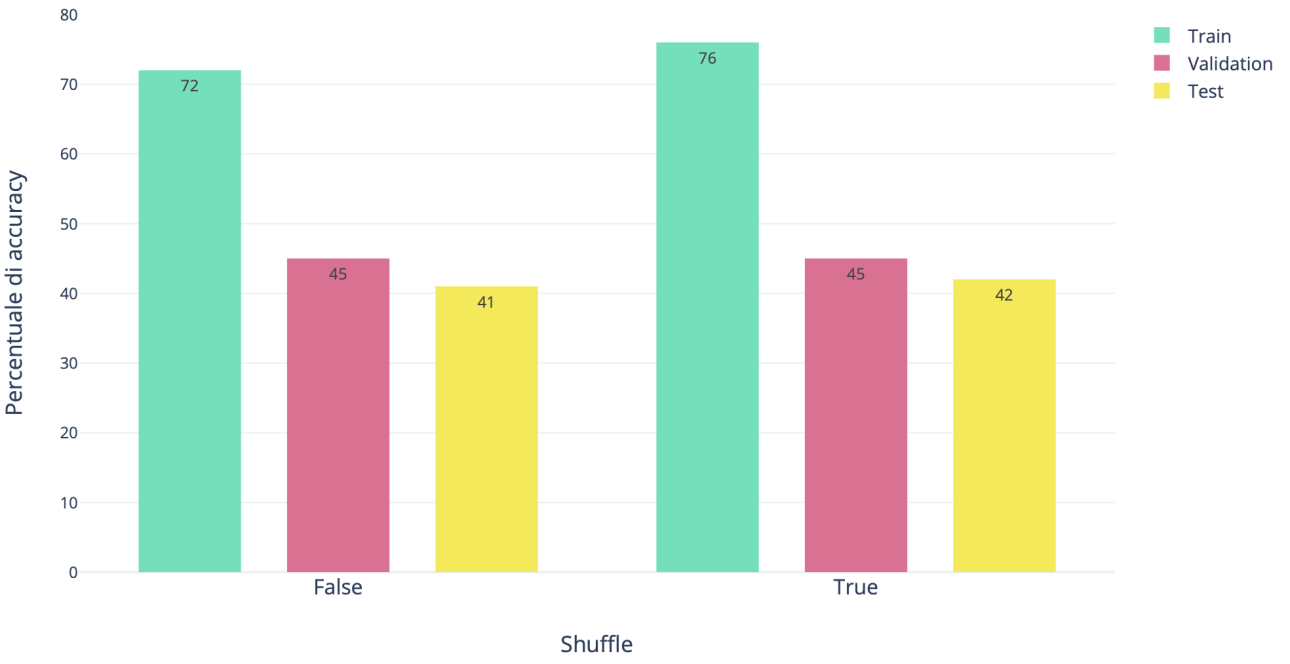
## Batch size influence in 2D CNN




| Parametro | Valore |
|---|---|
| Image res | 120 x 80 |
| Numero frames | 40 |
| Frame step | 2 |
| Shuffle | False |
| Learning rate | Adam (0.0001) |
| Epoche | 70 |

Il parametro **batch size** ha mostrato, così come nella rete 3D, risultati migliori se impostato a **32**. Difatti ogni sperimentazione che viene fornita nelle pagine successive, riportano il parametro **batch size = 32**.

## Shuffle influence in 2D CNN




| Parametro | Valore |
|---|---|
| Image res | 120 x 80 |
| Numero frames | 40 |
| Frame step | 2 |
| Batch size | 32 |
| Learning rate | Adam (0.0001) |
| Epoche | 70 |

Analizzando il parametro **shuffle**, abbiamo capito che metterlo a **True**, piuttosto che a **False**, non porta cambiamenti significativi sui risultati della rete. Difatti nei vari esperimenti abbiamo alternato l'utilizzo sia dell'uno che dell'altro valore per provare i modelli in un addestramento su più epoche.

## 2D CNN 150 epoche (shuffle = False)

| Parametro | Valore |
|---|---|
| Image res | 120 x 80 |
| Numero frames | 60 |
| Frame step | 2 |
| Batch size | 32 |
| Shuffle | False |
| Learning rate | Adam (0.0001) |

| Classe | Precision | Recall |
|---|---|---|
| Italiano | 0.36 | 0.35 |
| Inglese | 0.00 | 0.00 |
| Tedesco | 0.72 | 0.80 |
| Spagnolo | 0.07 | 0.05 |
| Olandese | 0.56 | 0.90 |
| Russo | 0.38 | 0.25 |
| Giapponese | 0.38 | 0.35 |
| Francese | 0.46 | 0.60 |

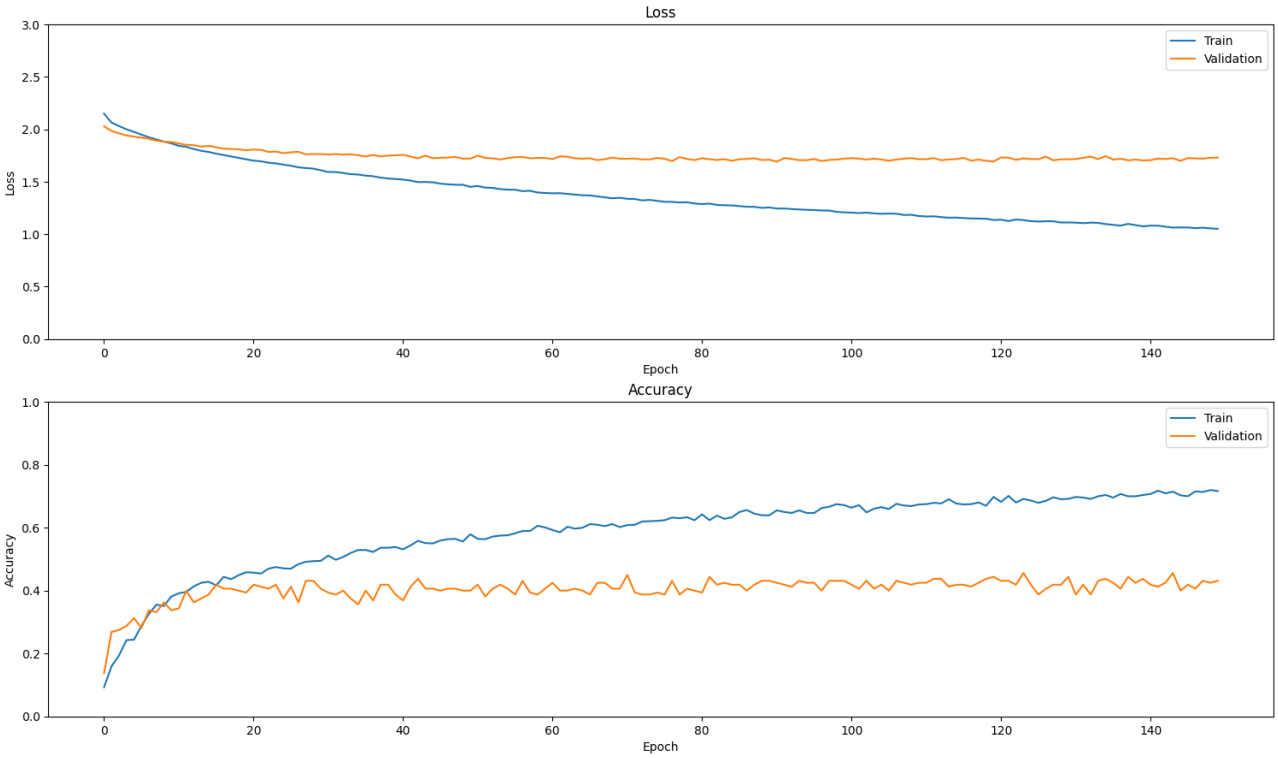



**Fig. 15: Grafico Accuracy/Loss**

**Test Accuracy: 0.41**

**Fig. 16:** Train

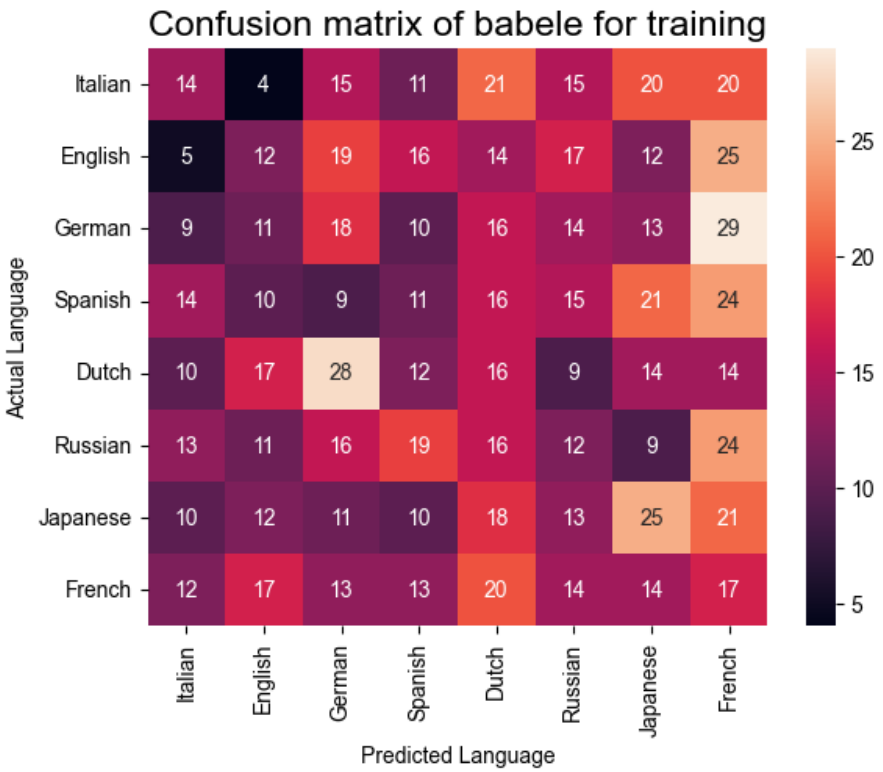
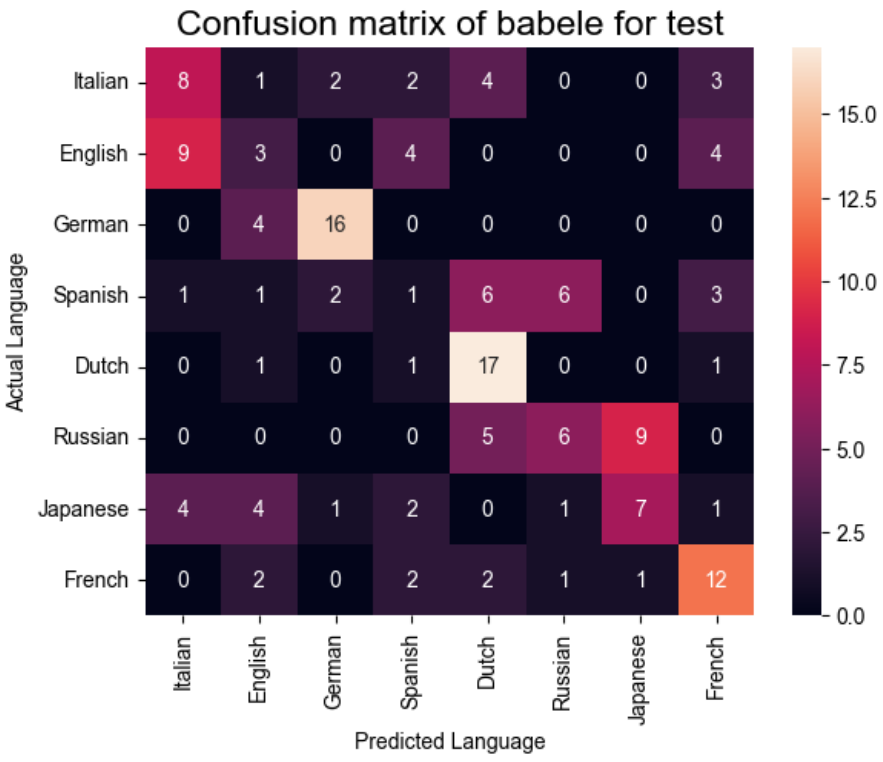



**Fig. 17:** Test



## 2D CNN 150 epoche (shuffle = True)

| Parametro | Valore |
|---|---|
| Image res | 120 x 80 |
| Numero frames | 60 |
| Frame step | 2 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.0001) |

| Classe | Precision | Recall |
|---|---|---|
| Italiano | 0.45 | 0.45 |
| Inglese | 0.11 | 0.10 |
| Tedesco | 0.78 | 0.75 |
| Spagnolo | 0.30 | 0.20 |
| Olandese | 0.53 | 0.85 |
| Russo | 0.13 | 0.05 |
| Giapponese | 0.23 | 0.25 |
| Francese | 0.38 | 0.60 |

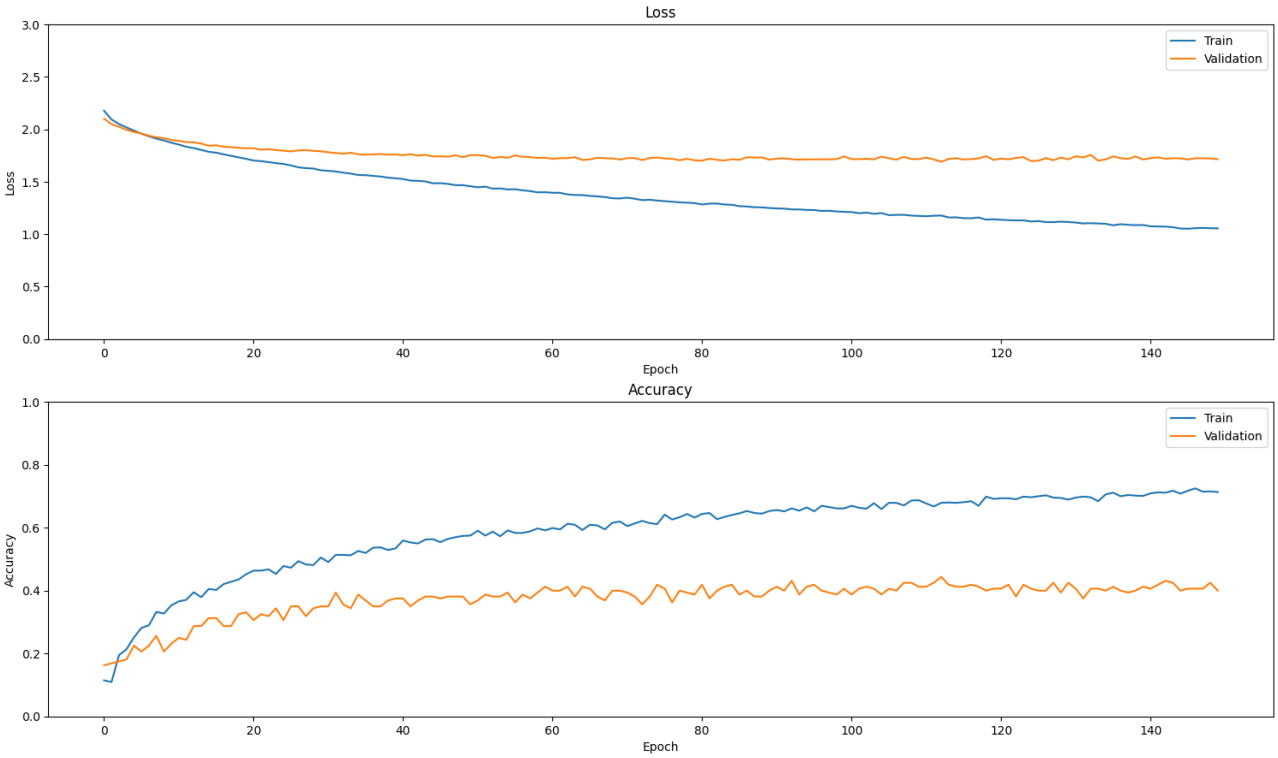



**Fig. 18:** **Grafico Accuracy/Loss**

**Test Accuracy: 0.42**

**Fig. 19:** Train

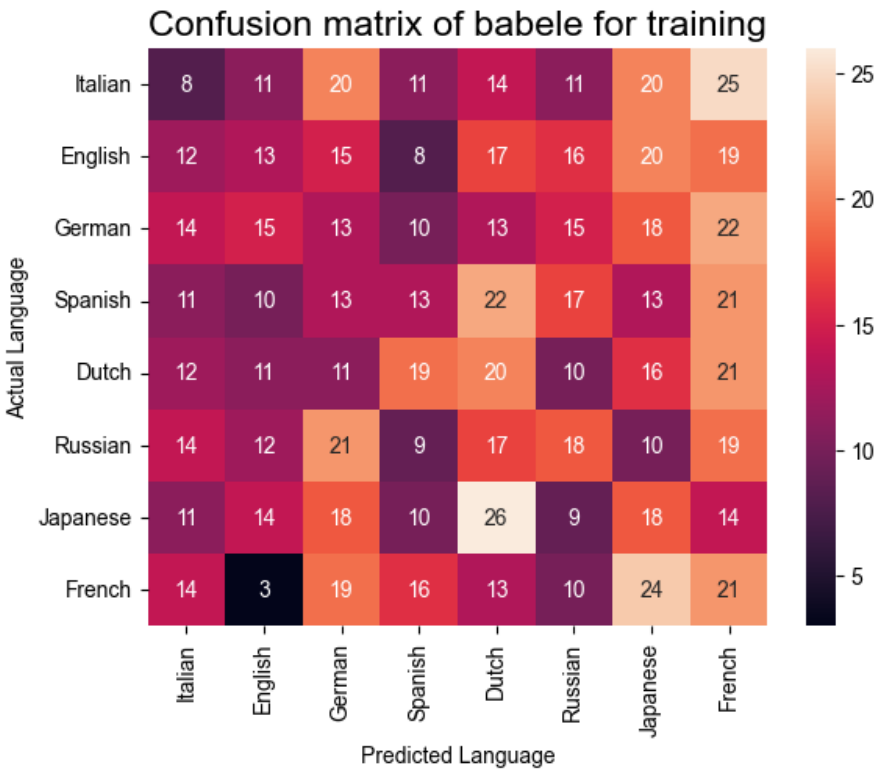
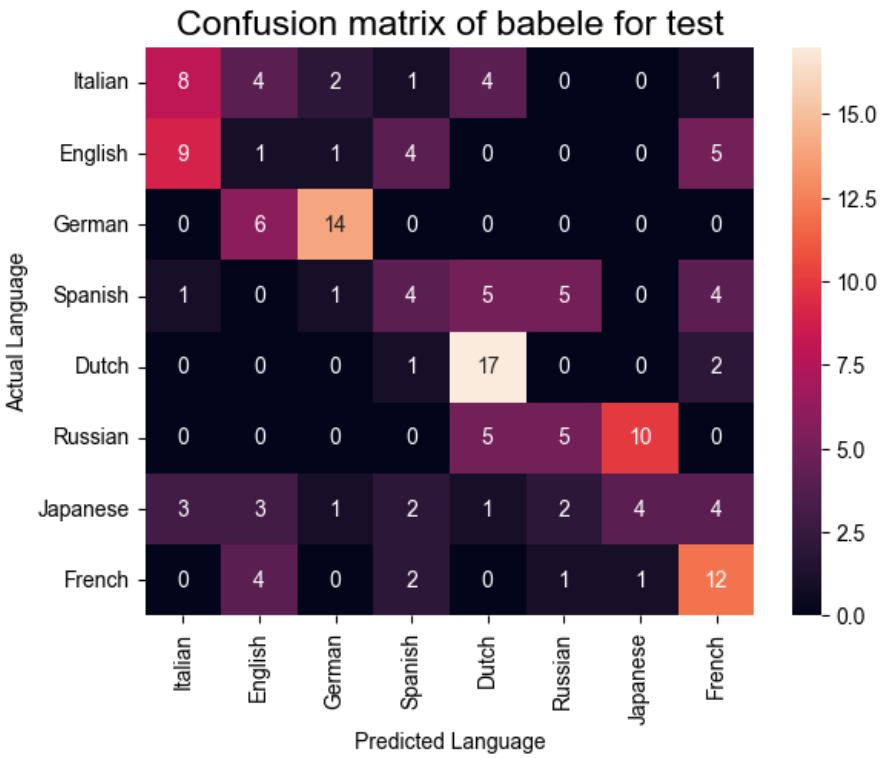



**Fig. 20:** Test



## 2D CNN 100 epoche con frames in Bianco e Nero

| Parametro | Valore |
|---|---|
| Image res | 90 x 60 |
| Numero frames | 40 |
| Frame step | 3 |
| Batch size | 32 |
| Shuffle | True |
| Learning rate | Adam (0.0001) |

| Classe | Precision | Recall |
|---|---|---|
| Italiano | 0.16 | 0.10 |
| Inglese | 0.40 | 0.05 |
| Tedesco | 0.68 | 0.75 |
| Spagnolo | 0.07 | 0.05 |
| Olandese | 0.38 | 0.70 |
| Russo | 0.33 | 0.35 |
| Giapponese | 0.07 | 0.05 |
| Francese | 0.06 | 0.05 |

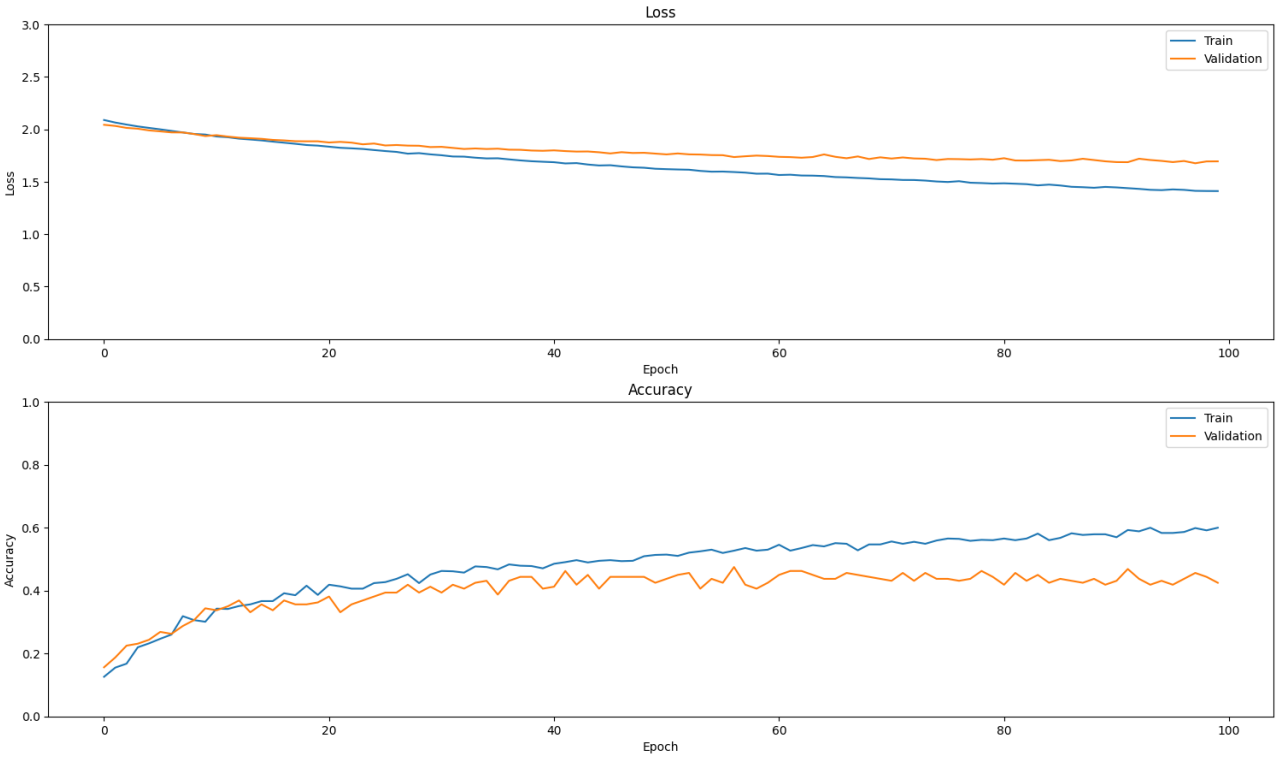



**Fig. 21: Grafico Accuracy/Loss**

**Test Accuracy: 0.30**

**Fig. 22:** Train

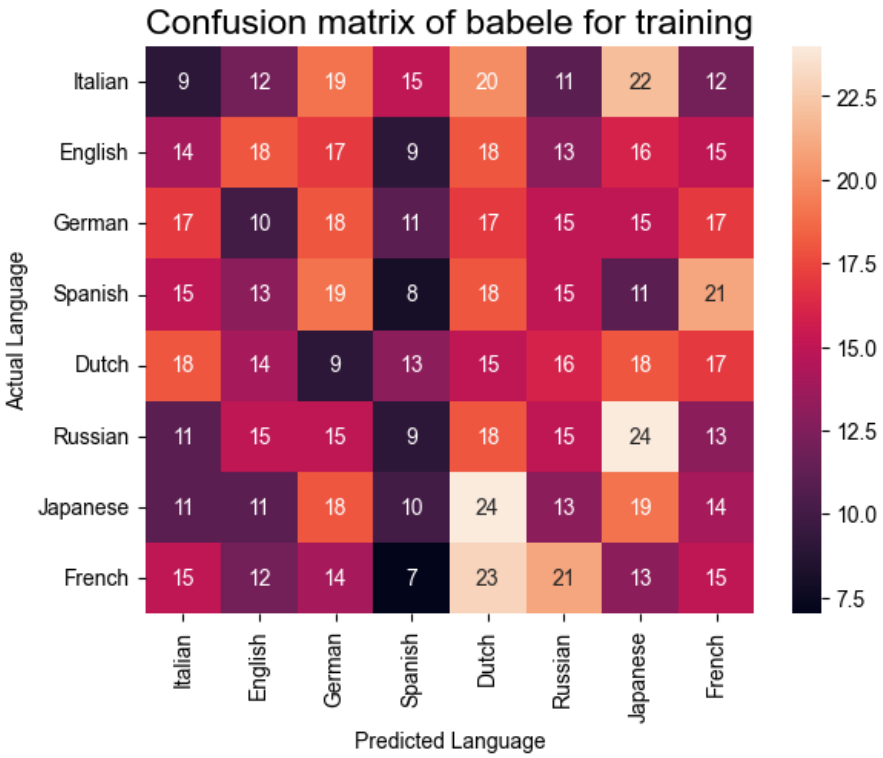
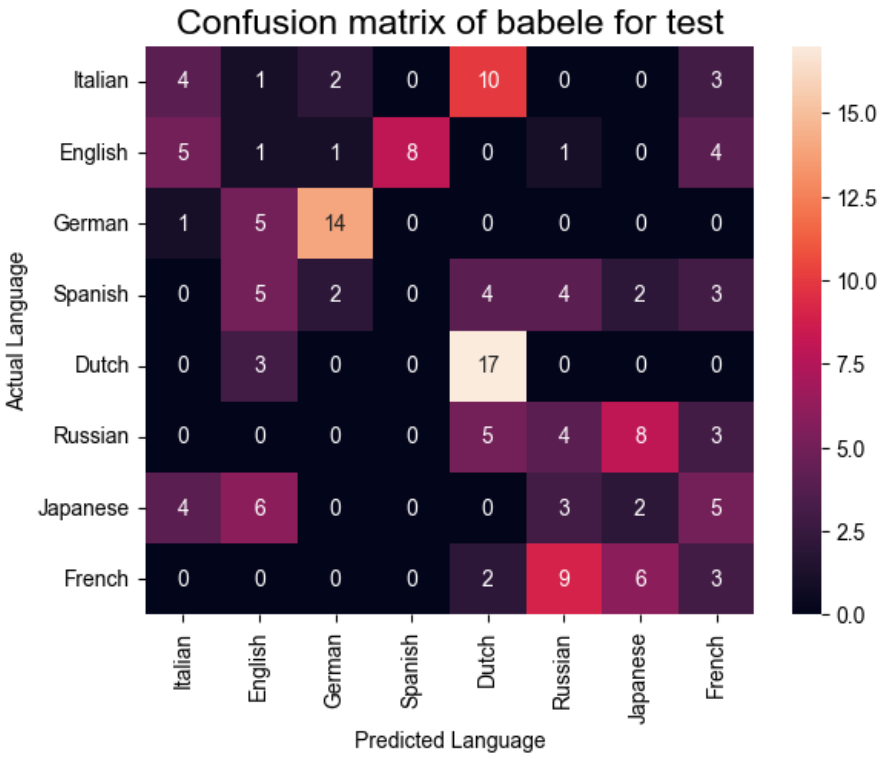



**Fig. 23:** Test



# Conclusioni

Ci teniamo a specificare che sono state effettuate molto prove e sperimentazioni per ottenere queste reti che abbiamo riportato in questa guida. Nonostante ciò abbiamo dato spazio solo alle reti di cui abbiamo salvato i modelli.

Abbiamo notato che in tutte le configurazioni di rete che abbiamo provato dopo una fase iniziale di Training i modelli non riuscivano a migliorare troppo sul Dataset di Validation. Dunque siamo caduti nel caso dell'**Overfitting**, e anche provando alcune soluzioni per migliorare questa problematica non siamo riusciti ad eliminarla del tutto.
Ad esempio abbiamo adottato le seguenti strategie:

- Aggiungere più dati (Abbiamo aumentato il numero di frame).
- Semplificato l'architettura della rete.
- Aggiunti livelli di Dropout.

Contro le nostre aspettative, la rete 2D ha performato leggermente meglio di quella 3D, questo ci ha fatto pensare che andando ad analizzare i frame in modo indipendente, invece che effettuare delle correlazioni temporali come fa la 3D CNN, possa portare a risultati migliori poiché il modello riesce ad identificare dei pattern di immagini all'interno di quest'ultime prese per ogni video, secondo i quali riesce ad estrapolare le informazioni utili relative ad identificare la lingua parlata.

Per quanto riguarda le reti 2D, le lingue che sono state riconosciute meglio sono il **Tedesco**, l'**Olandese** e il **Francese** (in misura minore).

Le 3D invece sono molto abili nel riconoscere solo seguenti lingue: **Olandese** e **Tedesco**.

Noi pensiamo che tutto questo sia dovuto al fatto che le lingue sopra elencate producano dei pattern tali da renderle abbastanza distinguibili dalle altre (che hanno molte similitudini a livello di pronuncia).

Alcune lingue possono presentare movimenti delle labbra simili, anche se le parole pronunciate sono in lingue molto diverse tra loro. Questo fenomeno è noto come coarticolazione o assimilazione.
Questo può accadere perché alcuni suoni condividono caratteristiche articolatorie simili. Ad esempio, diverse lingue possono condividere suoni che richiedono una posizione simile delle labbra, come i suoni ”p” o ”m.
Alcune lingue possono avere movimenti delle labbra simili, nonostante siano pronunciate parole molto diverse. I movimenti delle labbra possono essere influenzati da suoni simili tra diverse lingue o da variazioni di pronuncia. Questa coarticolazione può rendere difficile per le reti neurali distinguere tra lingue simili basandosi solo sui movimenti delle labbra.

# Sviluppi futuri

Sarebbe interessante andare ad effettuare degli esperimenti avendo a disposizione un dataset molto più grande, probabilmente il modello potrebbe riuscire a generalizzare meglio su tutte le lingue. Inoltre sarebbe anche interessante in futuro provare ad utilizzare la tecnica MeanShift, che consiste nel ridurre il numero di colori utilizzati per rappresentare un'immagine / video, ciò potrebbe portare ad una riduzione della potenza di calcolo richiesta in fase di training, anche se d'altro canto bisognerebbe avere molta potenza per trasformare inizialmente tutte le immagini / video nella loro versione MeanShift (motivo per il quale non abbiamo potuto effettuare questo esperimento). Anche la fase di pre-processing in futuro potrebbe essere migliorata, difatti abbiamo notato che forse nel nostro caso si poteva fare meglio da quel punto di vista, provando anche ad alleggerire i dati in input così da rendere le fasi addestrament molto più rapide.

Inoltre abbiamo notato che alcuni parametri come il batch size e il numero di frame determinano in modo profondo le prestazioni di questa implementazione delle reti neurali (sia 3D che 2D CNN). Per questo motivo nella fase iniziale del progetto abbiamo avuto delle problematiche in termini di efficienza, infatti in locale non riuscivamo a far girare le nostre sperimentazioni con le configurazioni ottimali.

# Autori

Francesco Paciello
Luca Boffa
Vincenzo Di Leo