WLASL classification & Signer classification

Visione Artificiale e Riconoscimento 23/24

Giulia Nardicchia, Simone Ceredi

Alma Mater Studiorum - Universitá di Bologna

24 giugno 2025

Indice

1. Introduzione
2. Pre-elaborazione dei dati
   - Analisi del dataset
   - Elaborazione dei dati
3. Hand-crafted feature
   - Descrittori classici
   - Estrazione Keypoint con MediaPipe
4. Classificazione WLASL
5. Classificazione signer
6. Conclusioni

Introduzione

Il progetto nasce con l'obiettivo di **classificare i segni della Lingua dei Segni Americana** (ASL) **in parole**, utilizzando il dataset WLASL (Word-Level American Sign Language) trovato su *Kaggle*.

Data la complessità del problema e i risultati insoddisfacenti ottenuti inizialmente, è stato aggiunto un ulteriore problema di classificazione: il **riconoscimento del volto del signer** (persona che esegue il segno).

Introduzione

Sono stati messi a confronto diversi approcci:

- Metodi tradizionali (con feature hand-crafted)
- Reti neurali

È stato inoltre effettuato un confronto diretto con un modello proposto su Kaggle per valutare la qualità dei risultati (per il primo problema).

Indice

1. Introduzione
2. Pre-elaborazione dei dati
   - Analisi del dataset
   - Elaborazione dei dati
3. Hand-crafted feature
   - Descrittori classici
   - Estrazione Keypoint con MediaPipe
4. Classificazione WLASL
5. Classificazione signer
6. Conclusioni

Analisi del dataset

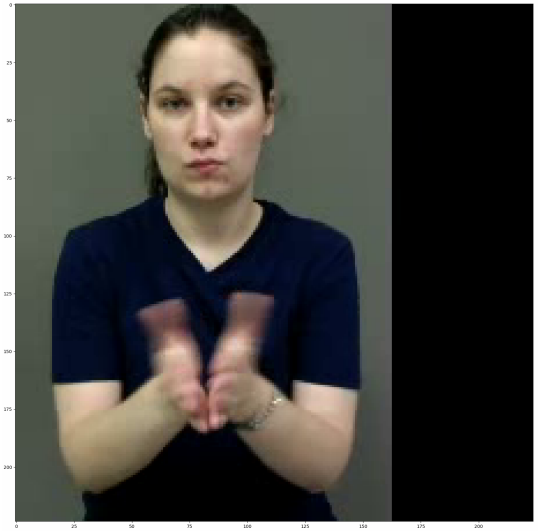
Informazioni sui video estratte dal file `WLASL_v0.3.json`:


```
"gloss": "book",
"instances": [
  {
    "bbox": [
      385,
      37,
      885,
      720
    ],
    "fps": 25,
    "frame_end": 75,
    "frame_start": 1,
    "instance_id": 0,
    "signer_id": 118,
    "source": "aslbrick",
    "split": "train",
    "url": "http://aslbricks.org/New/ASL-Videos/book.mp4",
    "variation_id": 0,
    "video_id": "69241"
  }
]
```

Elaborazione dei dati

- Calcolo del parametro `frame_end`;
- Estrazione della *region of interest* (ROI);
- Ridimensionamento (224x224) e aggiunta *padding*.

(a) Frame originale



(b) Frame pre-processato

Indice

1. Introduzione
2. Pre-elaborazione dei dati
   - Analisi del dataset
   - Elaborazione dei dati
3. Hand-crafted feature
   - Descrittori classici
   - Estrazione Keypoint con MediaPipe
4. Classificazione WLASL
5. Classificazione signer
6. Conclusioni

Edge extractor

Pipeline di elaborazione per evidenziare i contorni:

1. Grayscale
2. Gaussian Blur
3. Equalizzazione dell'istogramma
4. Gradiente con operatore Sobel
5. Binarizzazione
6. Chiusura morfologica (dilatazione + erosione)

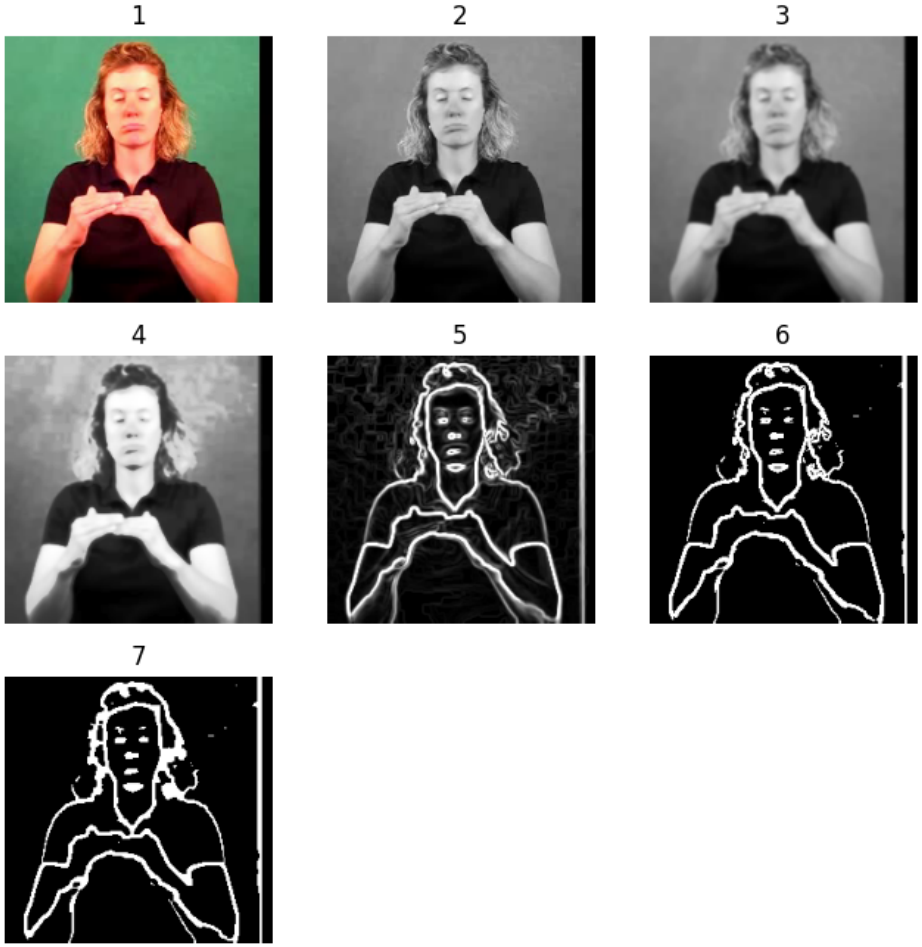


Figura: Edge extractor

Histogram of Oriented Gradients extractor

1. Grayscale
2. Suddivizione in celle di dimensione 8×8 pixel
3. Calcolo dei gradienti
4. Creazione degli istogrammi delle orientazioni: per ogni cella, si costruisce un istogramma delle direzioni dei gradienti, suddividendo le orientazioni in 9 bin
5. Normalizzazione per blocchi: le celle sono raggruppate in blocchi di 2×2 celle e gli istogrammi all'interno di ciascun blocco sono normalizzati

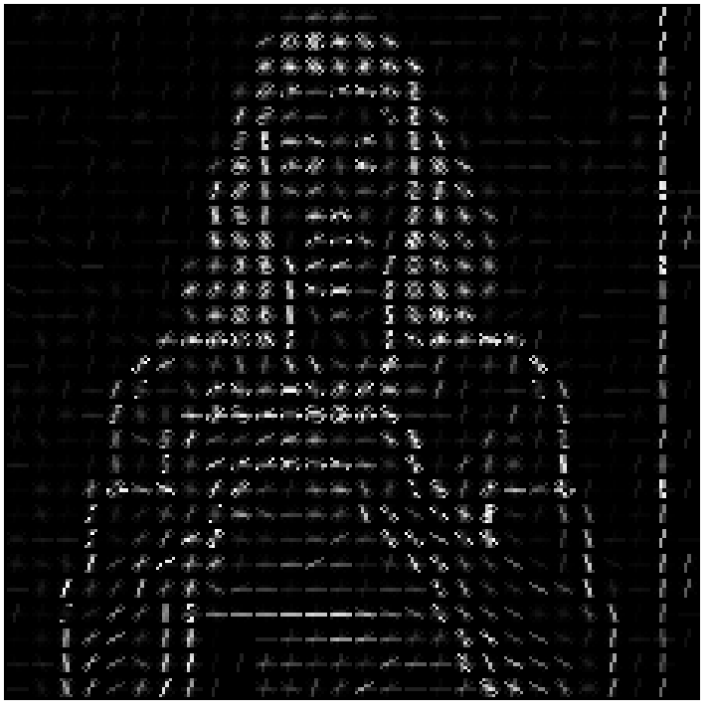
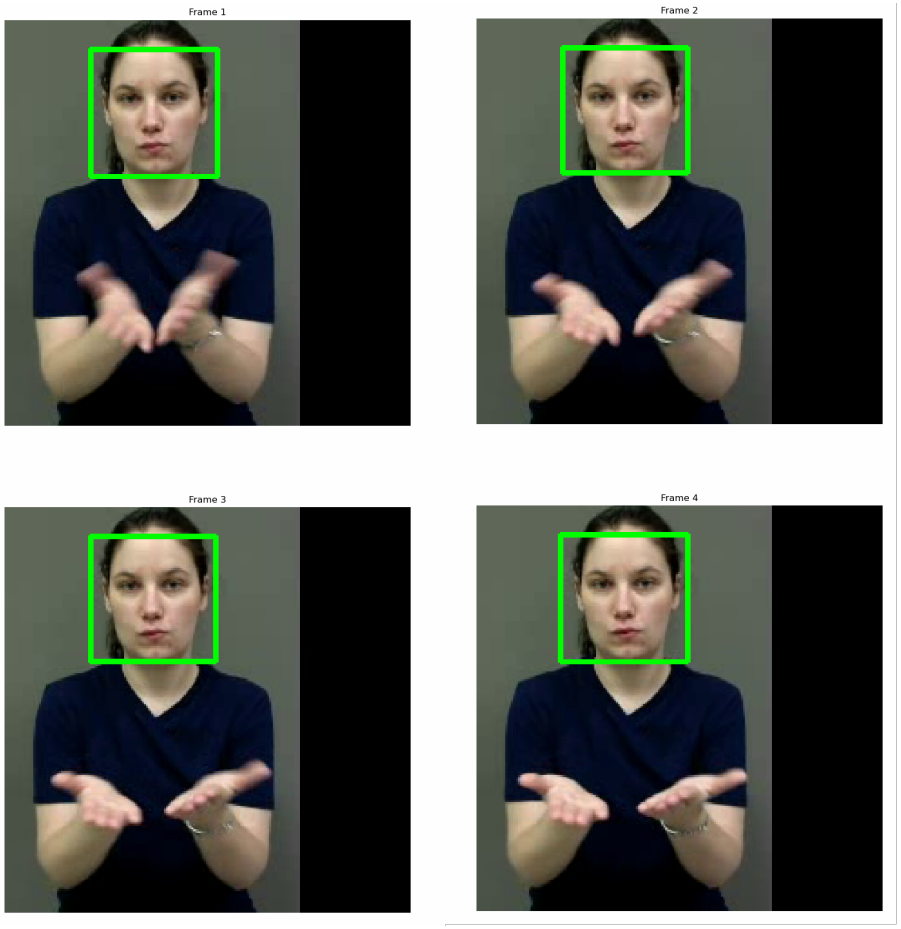


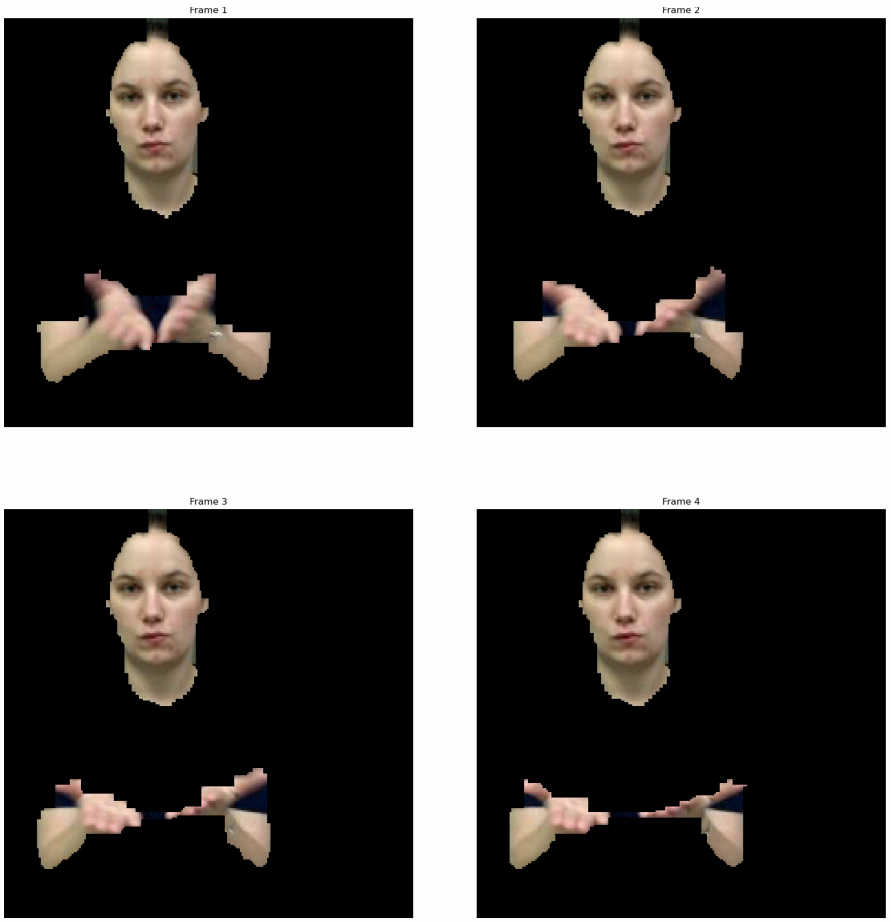
Figura: HoG extractor

Feature di texture

- Haar-like features



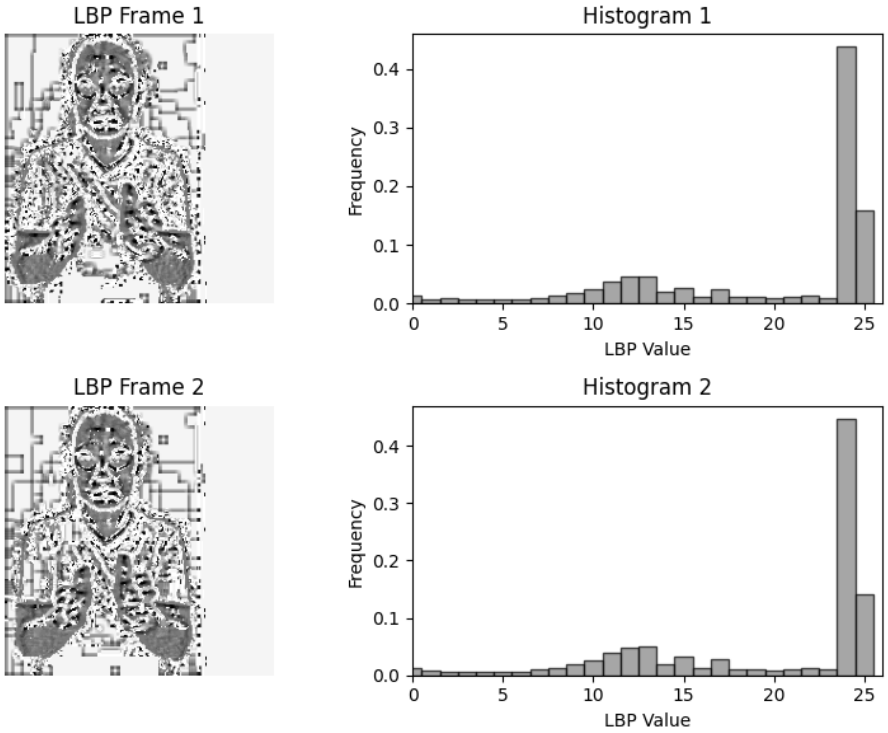
(a) Localizzatore di Viola e Jones



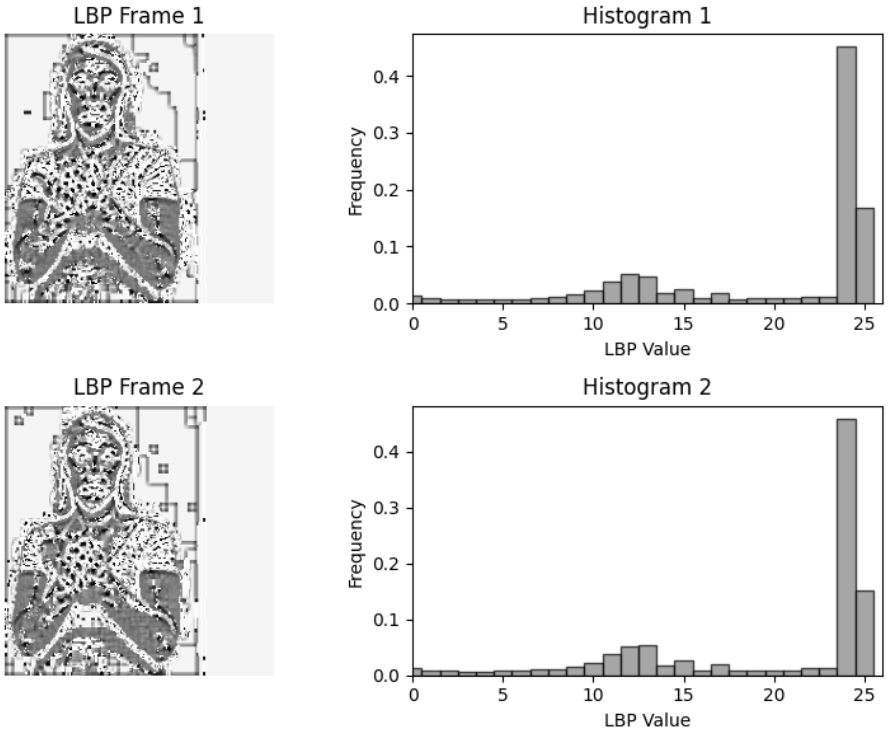
(b) Segmentazione di volto ed arti

Feature di texture

- Local Binary Pattern (LBP)



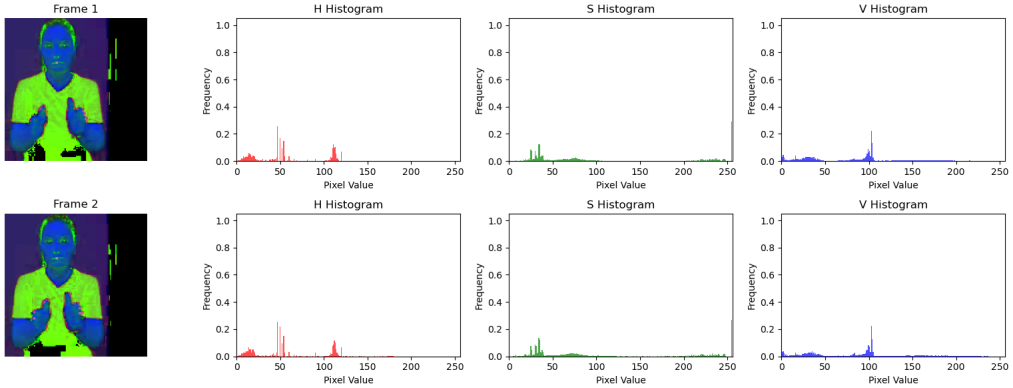
(a) Soggetto 1



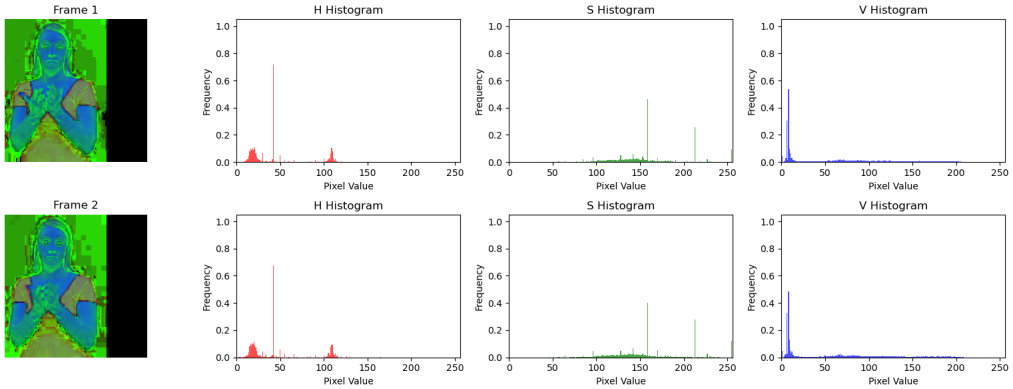
(b) Soggetto 2

Feature di colore

- Color histogram (spazio colore HSV)



(a) Soggtto 1



(b) Soggetto 2

Feature di movimento

- Optical flow (Farneback)

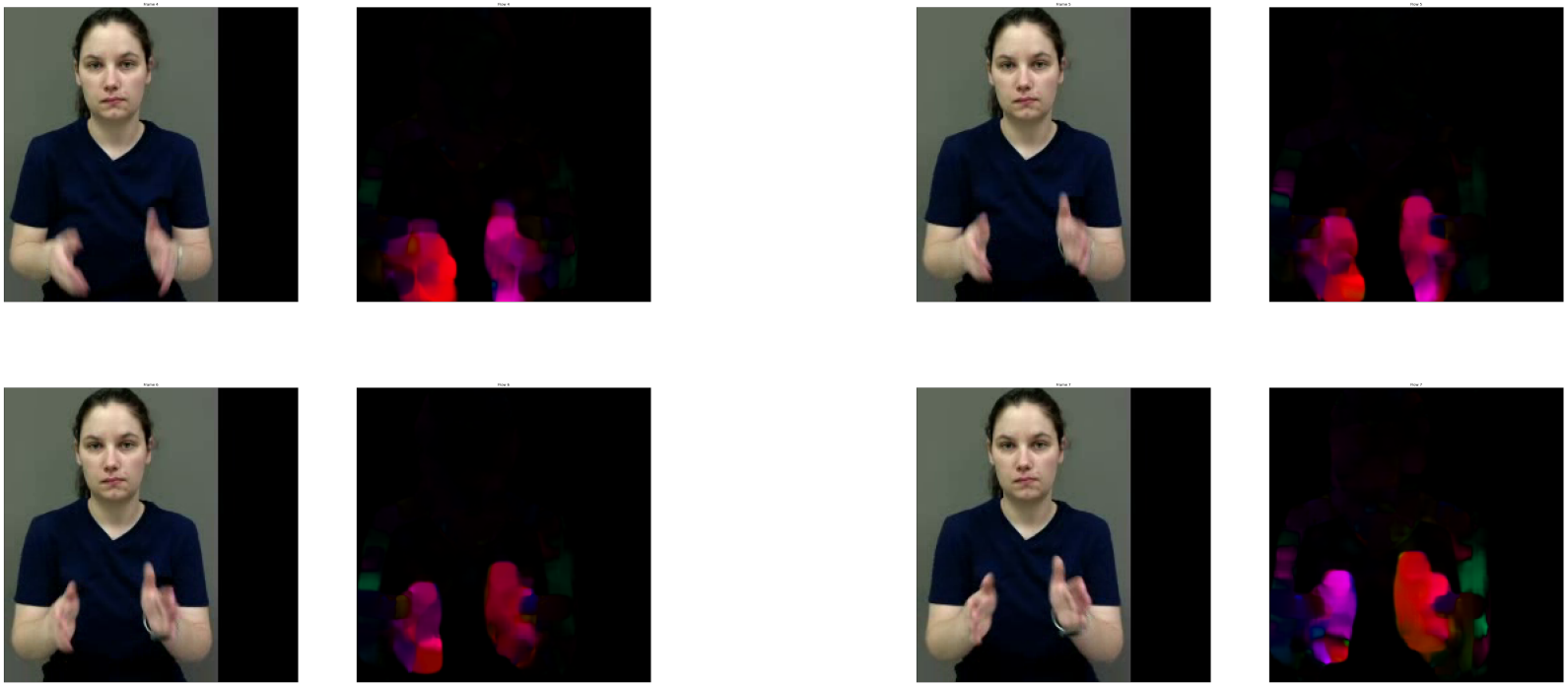


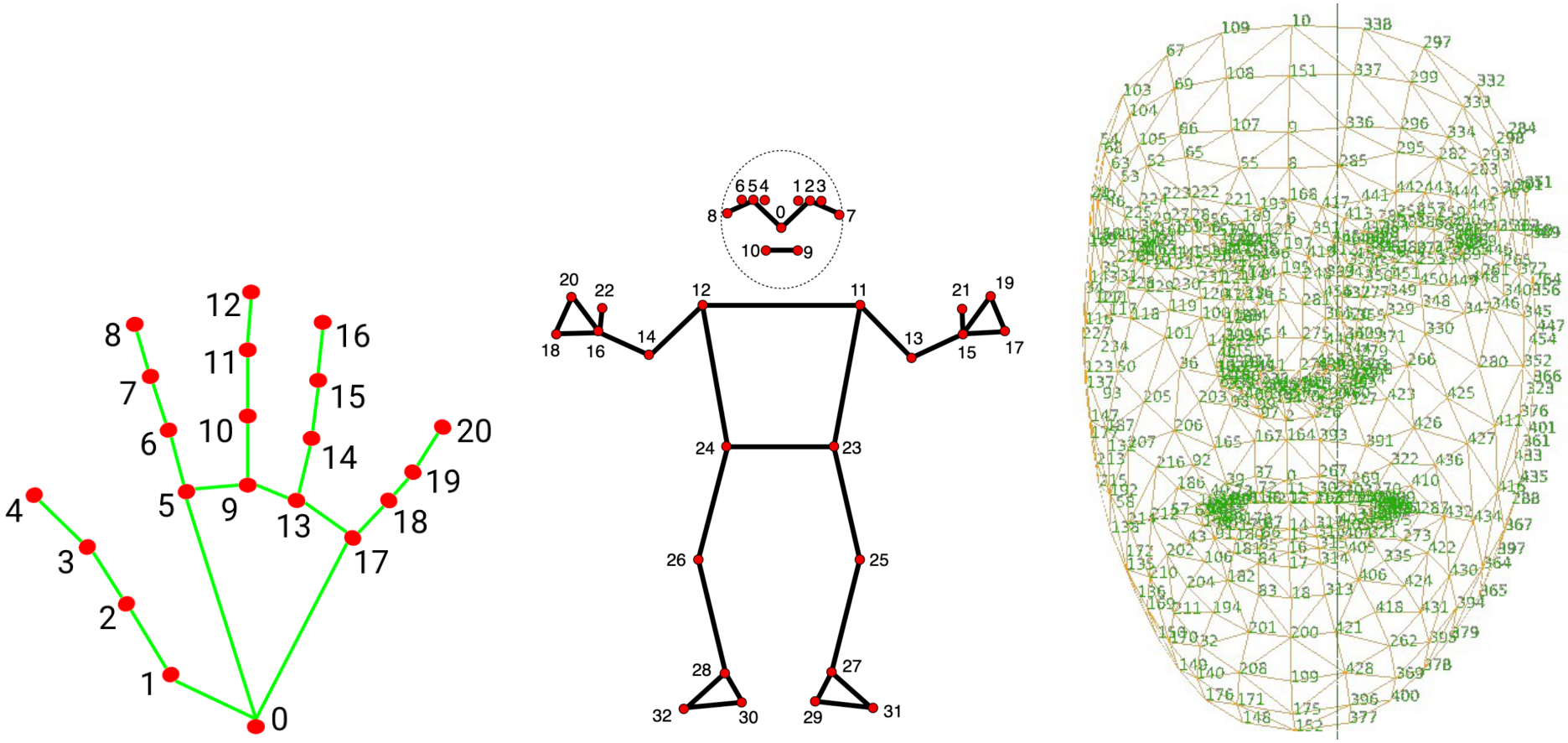
Figura: Risultati del calcolo dell'optical flow su 4 frames consecutivi.

Feature hand-crafted utilizzate

Nella pratica sono state utilizzate solo una parte di tutte le feature estratte, che, sulla base di alcuni test empirici, si sono rivelate più efficaci rispetto alle altre:

- **Classificazione dei video**: HoG, LBP
- **Riconoscimento del signer**: HoG, LBP, Color histograms

MediaPipe



(a) Hand (b) Pose (c) Face

Figura: Keypoints: mani, posa e volto

Estrazione Keypoint con MediaPipe

I keypoint estratti tramite MediaPipe seguono il seguente formato:

```
landmark {
  x: 0.5392837524414062
  y: 0.7506672739982605
  z: -0.06435433775186539
}
```

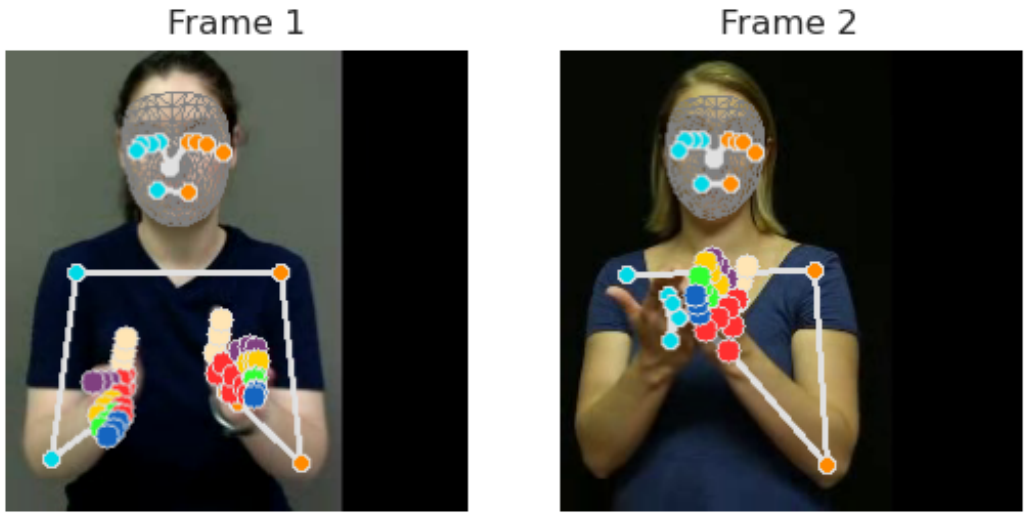


Figura: Visualizzazione dei keypoint estratti tramite MediaPipe.

Indice

1. Introduzione
2. Pre-elaborazione dei dati
   - Analisi del dataset
   - Elaborazione dei dati
3. Hand-crafted feature
   - Descrittori classici
   - Estrazione Keypoint con MediaPipe
4. Classificazione WLASL
5. Classificazione signer
6. Conclusioni

Dynamic Time Warping (DTW)

È stato utilizzato `FastDTW`, un algoritmo più rapido rispetto alle implementazioni tradizionali di DTW grazie a un'approssimazione multi-risoluzione del percorso ottimale.

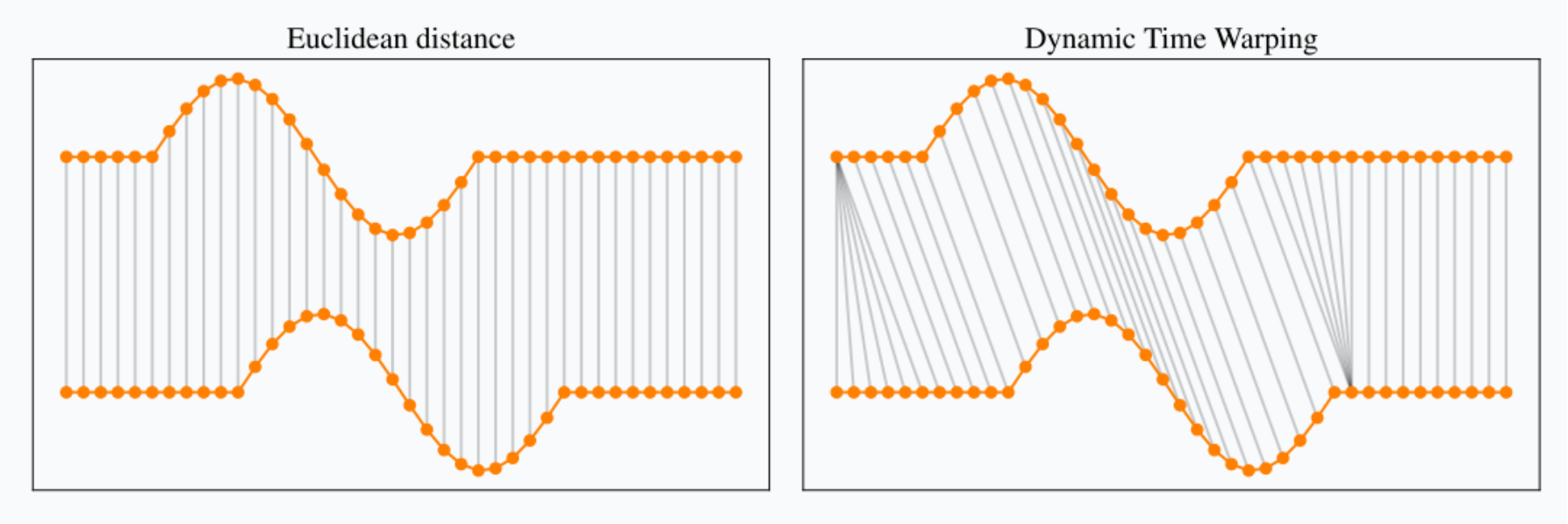


Figura: Distanza Euclidea VS DTW

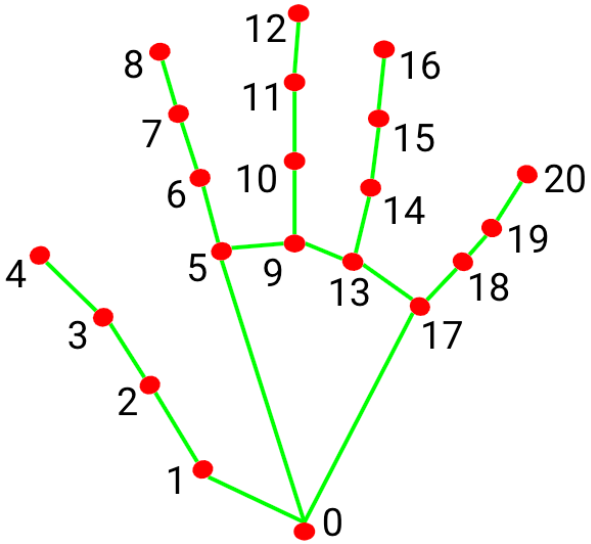
Classificazione tradizionale

- Unione di training e validation set;
- FastDTW;
- Feature: HOG e LBP concatenate;
- Viene confrontato ogni video di test con tutti i video di training;
- Dalla distanza ottenuta con DTW si calcola la similarità che risulta più interpretabile, avendo valori nel range [0, 1].

Numero Parole	**Accuratezza**
10	32.4%
25	20.3%
50	15.7%
100	10%

Classificazione con MediaPipe: preparazione dati

- Unione training e validation set;
- Utilizzo solo landmark relativi alle mani;
- Data augmentation effettuando rotazioni ad entrambe le mani attorno assi x, y, z;
- Calcolo dell'angolo compreso tra due landmark connessi.



Classificazione con MediaPipe: valutazione del modello

- Caricamento dell'intero dataset di test;
- Confronto di ogni video di test con tutti i video relativi ad una parola di training per volta (elevata occupazione RAM);
- Valutazione accuracy top-1 e top-5.

#Parole	**Accuracy (top-1) no aug**	**Accuracy (top-1) con aug**
10	68.6%	68.6%
25	47.5%	48.8%
50	39.9%	41.3%
100	33.3%	33.7%

Classificazione: soluzione di Kaggle

- Progetto utilizzato come punto di partenza;
- Utilizzo di landmark estratti con MediaPipe (volto, posa e mani filtrate);
- Utilizzo di `fasttext` per encoding (300 valori per parola) e calcolo similarità tra parole;
- "Classificazione" tramite MLP;
- 44.6% accuratezza sul test set.

Problemi riscontrati (troppo tardi) nella soluzione:

- Modello cerca di stimare i 300 valori estratti tramite `fasttext`;
- Ultimo layer ha attivazione `linear`;
- Performance del modello valutate tramite `accuracy`.

Kaggle: performance reali

Il quinto notebook realizzato mostra il 42% di `accuracy` valutando la prediction della stessa parola sull'intero glossario filtrato.

Valutazione del modello facendo regressione:

- Modifica della metrica di valutazione del modello da `accuracy` a `MSE`;
- `MSE` sul test set pari a 75.2;
- I valori predetti sono 300 nel range [-1, 1].

Valutazione del modello facendo classificazione:

- Modifica dell'ultimo layer, attivazione da `linear` a `softmax`, da 300 neuroni a 351;
- Modifica della `loss` da `cosine_similarity` a `categorical_crossentropy`;
- Modifica del dataset da valori calcolati con `fasttext` a parole codificate con `one hot encoding`;
- Accuracy del modello pari al 2.61%.

Confronto dei risultati

- Tradizionale vs MediaPipe (top-1 acc):

#Parole	Accuratezza (%)		
	Tradizionale	**MP no aug**	**MP + aug**
10	32.4%	68.6%	68.6%
25	20.3%	47.5%	48.8%
50	15.7%	39.9%	41.3%
100	10%	33.3%	33.7%

- Mediapipe (top-5 acc):

Numero parole	Accuratezza (%)	
	MediaPipe no aug	**MediaPipe + aug**
10	100%	100%
25	88.8%	90%
50	72%	73.4%
100	64.7%	63.6%

Indice

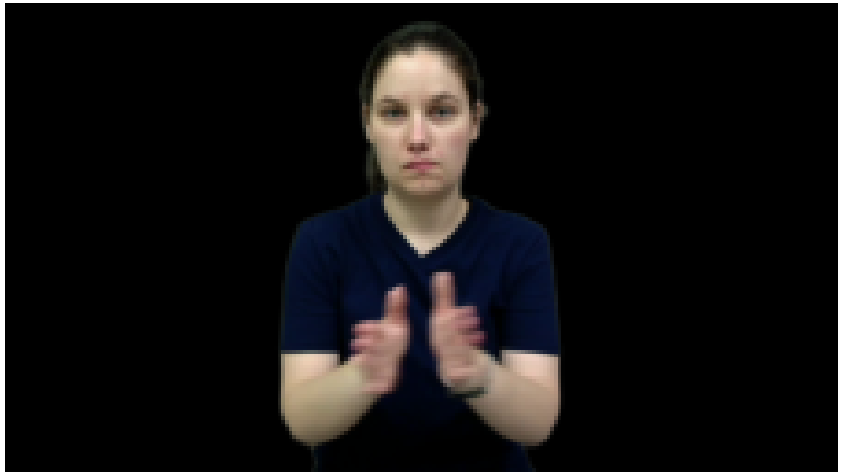
1. Introduzione
2. Pre-elaborazione dei dati
   - Analisi del dataset
   - Elaborazione dei dati
3. Hand-crafted feature
   - Descrittori classici
   - Estrazione Keypoint con MediaPipe
4. Classificazione WLASL
5. Classificazione signer
6. Conclusioni

Pre-elaborazione dei dati

Caratteristiche osservate:

- i *signer* hanno lo stesso sfondo in ogni video in cui appaiono;
- i video sono stati registrati in momenti diversi, infatti i *signer* hanno capigliature e vestiti diversi nei video in cui appaiono.

Per questi motivi si è deciso di rimuovere lo sfondo da tutti i video.



Struttura del dataset

Nel rimuovere il background, si è scelto di non salvare i video originali, ma solo i frame estratti, ricostruendo la struttura del dataset. Questa scelta consente un'elaborazione più rapida e meno onerosa in termini di risorse utilizzate.

```
frames_no_bg/
├── 0/
│   ├── 05076/
│   │   ├── 0.png
│   │   ├── 1.png
│   │   └── ..
│   └── ..
├── 1/
│   └── ..
└── ..
```

Suddivisione del dataset e gestione dello sbilanciamento

- I video sono stati suddivisi in *training*, *validation* e *test set*, assegnando ciascun signer (se presente in ≥ 3 video) a un solo insieme.
- Altrimenti, si crea un unico video a partire dai (max) 2 video disponibili, si effettua uno shuffle e si suddivide in tre sottoporzioni che vengono assegnate agli insiemi.
- Da ogni video sono stati estratti casualmente dei frame per comporre i set finali.
- Il dataset è fortemente sbilanciato.
  Es. signer 12 $\rightarrow$ 117.374 frame, signer 23 $\rightarrow$ solo 49 frame.
- Per bilanciare i dati, è stata applicata data augmentation solo ai signer meno rappresentati, replicando i frame in modo mirato.

Data Augmentation

Trasformazioni applicate:

- Rotazione all'interno di un range;
- Traslazione orizzontale e verticale;
- Modifica della luminosità;
- Flip orizzontale;
- Applicazione di un filtro di sfocatura gaussiana;
- Addizione di rumore gaussiano;
- Trasformazione elastica;
- Zoom in/out.

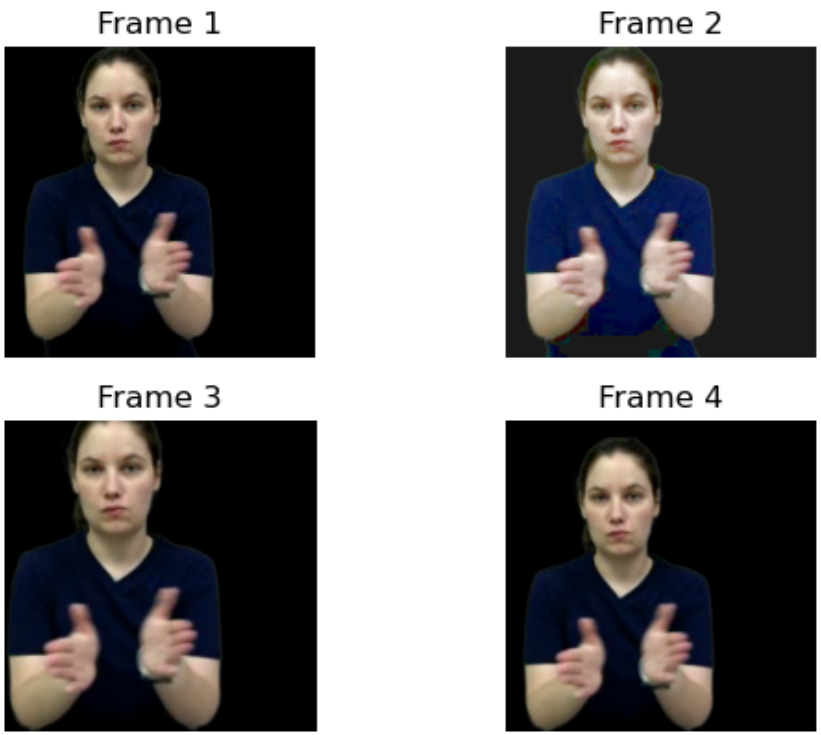


Figura: Frame 1 è originale, mentre agli altri sono stati applicate le trasformazioni.

Classificazione dei signer con XGBoost

Caricamento del dataset
Il dataset viene caricato da disco sfruttando la struttura delle cartelle per identificare i frame e i relativi label (signer ID).

Suddivisione in training e test set
Viene suddiviso il dataset in training e test (escludendo il validation set) e si estraggono un numero definito di frame per video. Viene anche calcolato il numero di *augmentations* necessari per bilanciare le classi.

Preparazione dei dati
Le label dei signer vengono codificate con `LabelEncoder`.
I dati vengono organizzati in `MiniBatch` per caricare in memoria solo piccoli blocchi di dati alla volta.

Classificazione dei signer con XGBoost

Definizione del modello XGBoost
Il modello viene configurato per la classificazione multiclasse con `multi:softmax`, specificando il numero di classi e parametri ottimizzati per efficienza.

Addestramento incrementale
Il training avviene in modo incrementale: ogni `MiniBatch` viene caricato, convertito in `DMatrix`, e usato per aggiornare il modello tramite `xgb.train`.

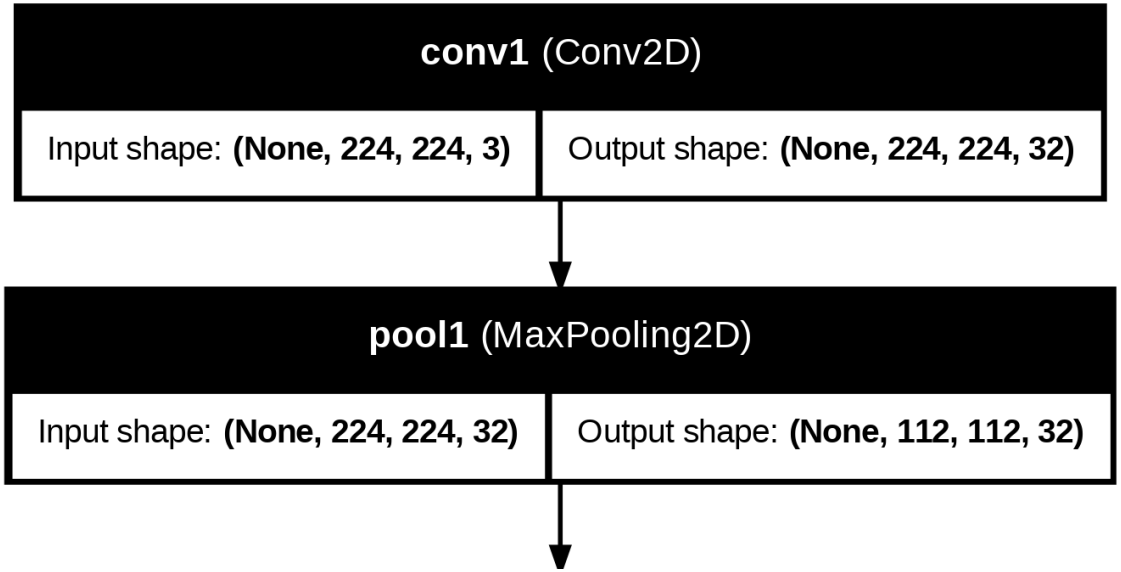
Valutazione sul test set
I dati di test vengono caricati batch per batch. Le predizioni del modello vengono confrontate con le label reali per calcolare le metriche.

Classificazione dei signer con CNN

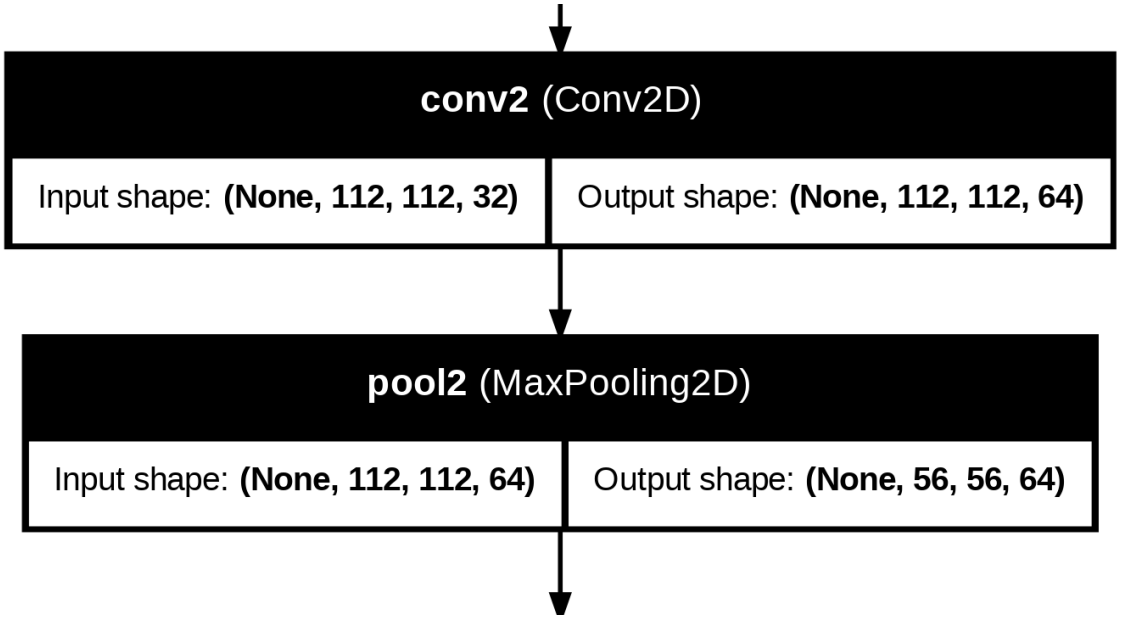
Caricamento e suddivisione del dataset
Il caricamento dei dati avviene riutilizzando il codice già usato per i classificatori tradizionali.
Tuttavia, la suddivisione del dataset prevede ora tre insiemi: **training**, **validation** e **test**.
Le augmentation vengono applicate solo al training set. I dati sono mantenuti come percorsi su disco per non occupare memoria.

Preparazione delle label e creazione del dataset
Le label sono codificate tramite `LabelEncoder` e convertite in `one-hot encoding` con `to_categorical`.
A differenza della pipeline con XGBoost, qui viene utilizzato `tf.data.Dataset`, che gestisce in modo efficiente il caricamento e le augmentation durante l'addestramento.

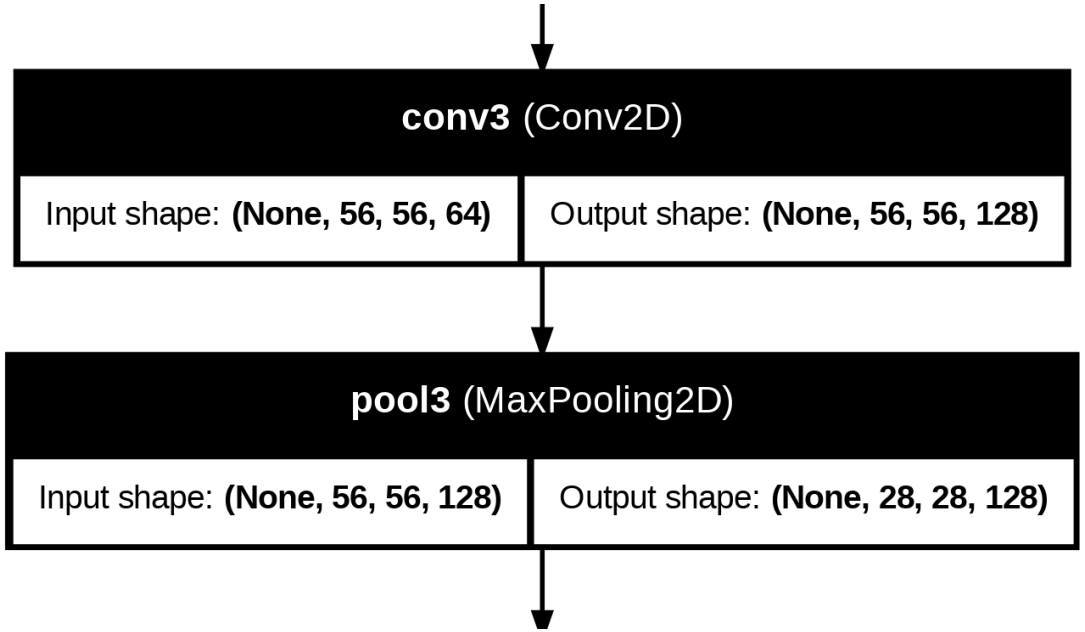
Definizione del modello (1)



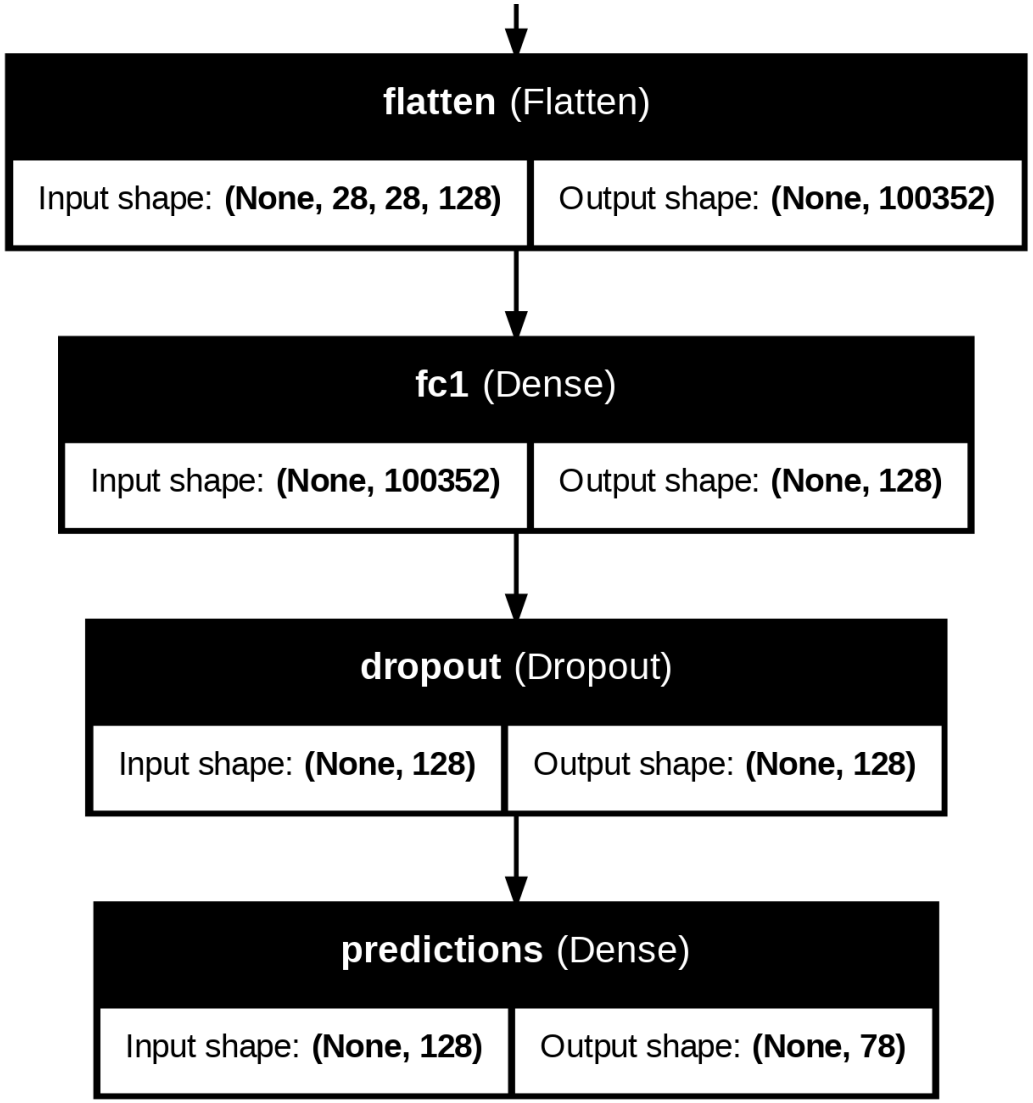
Definizione del modello (2)



Definizione del modello (3)



Definizione del modello (4)

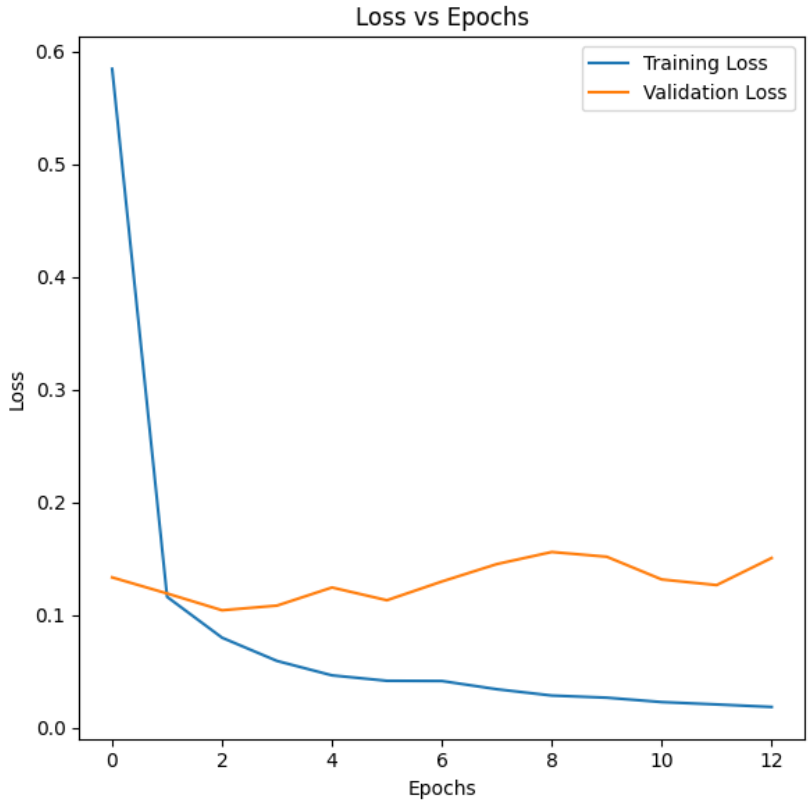


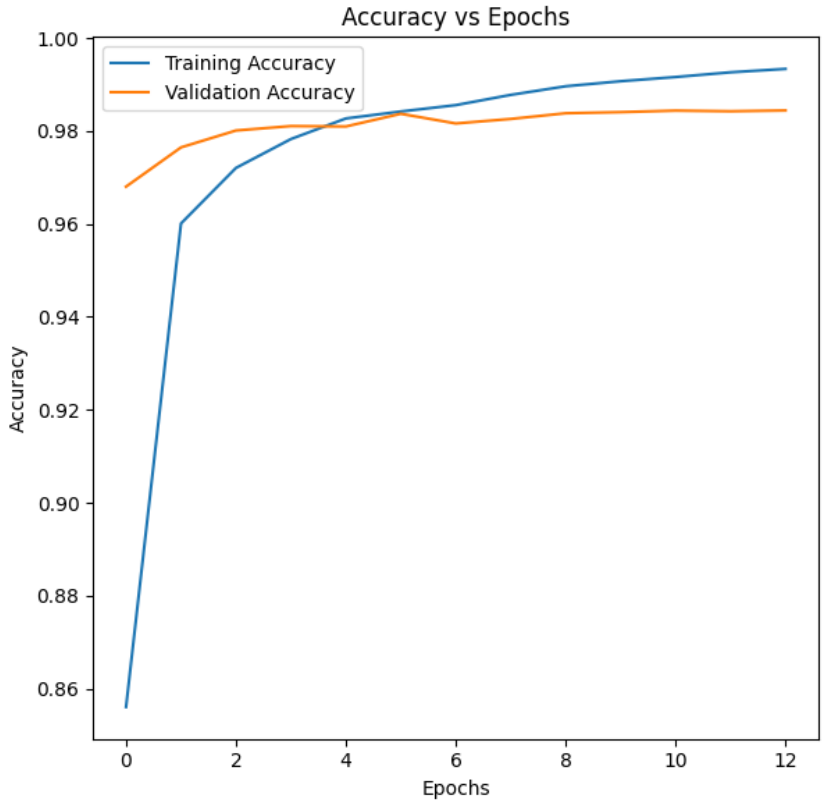
Classificazione dei signer con CNN

Training del modello
Il modello è addestrato con `model.fit` in batch, sfruttando `early stopping` per evitare overfitting.
È utilizzata la strategia `MirroredStrategy` per sfruttare più GPU se disponibili.

Valutazione e analisi dei risultati
Il modello è valutato sul test set con `model.evaluate`.
Le predizioni vengono decodificate e confrontate con le label originali, e abbiamo analizzato la matrice di confusione per analizzare l'efficacia della classificazione.

Loss e Accuracy





Risultati a confronto

Gli addestramenti dei classificatori sono stati effettuati sull'intero dataset, l'unica differenza tra i due è il numero di frames che sono stati estratti per ogni video.

Soluzione	**Numero immagini di training**	**Accuratezza (%)**
Tradizionale	33292	94.67%
CNN	185635	97.75%

Tabella: Confronto tra i risultati utilizzando metodi tradizionali e rete neurale.

Indice

1. Introduzione
2. Pre-elaborazione dei dati
   - Analisi del dataset
   - Elaborazione dei dati
3. Hand-crafted feature
   - Descrittori classici
   - Estrazione Keypoint con MediaPipe
4. Classificazione WLASL
5. Classificazione signer
6. Conclusioni

Conclusioni

- Lavorare con video si è rivelato molto complesso;
- La classificazione dei video richiede molto tempo;
- In generale ci possiamo considerare soddisfatti dei risultati ottenuti;
- In futuro si potrebbero migliorare le prestazioni per permettere la traduzione real-time di video nelle relative parole.