

Progetto di Fondamenti di Intelligenza Artificiale

SignHelper: Parlare a gesti (riconoscimento del
linguaggio dei segni americano)

Vito Gerardo Piegari

Matricola: 0512110333

January 6, 2024



Indice

1	Introduzione	3
2	Definizione del problema	4
2.1	Obiettivi	4
2.2	Specifica PEAS	4
2.3	Caratteristiche dell'ambiente	4
3	Dataset	6
3.1	Struttura	6
3.2	Bilanciamento del dataset	6
3.3	Ulteriori operazioni sui dati	8
4	Soluzione proposta	9
4.1	Modello	9
4.2	Composizione CNN (Convolutional Neural Network)	9
4.3	Miglioramento del modello	11
4.3.1	Data augmentation	12
4.3.2	Preprocessing dell'immagine	13
4.4	Riepilogo	14
5	Conclusioni	15
5.1	Sviluppi futuri	15

1 Introduzione

L'atto di comunicare attraverso il linguaggio dei segni, inizialmente appreso dai bambini come un passatempo, si evolve ben presto in una competenza fondamentale che, una volta acquisite le nozioni basilari dell'alfabeto gestuale, consente una comunicazione agevole senza ricorrere alla verbalizzazione.

Tuttavia, vi sono situazioni in cui il linguaggio dei segni non costituisce un mero passatempo ludico, bensì una necessità imprescindibile, specialmente quando sorgono difficoltà nel parlare o sono presenti condizioni di disabilità.

Comprendere e comunicare con il linguaggio dei segni, diventa oltre ad un modo divertente di comunicare, anche un modo per favorire l'**inclusione** nella vita di tutti i giorni di persone che non riescono a comunicare vocalmente.

Nonostante la comprensione del linguaggio dei segni sia relativamente facile per le persone, non si può dire lo stesso per i computer, quindi, questo progetto ha come obiettivo quello di implementare una "semplice" intelligenza artificiale in grado di comprendere brevi parole comunicate con il linguaggio dei segni.

Il progetto è disponibile al seguente link: <https://github.com/veetaw/fia>

2 Definizione del problema

2.1 Obiettivi

Lo scopo del progetto consiste nel creare una intelligenza artificiale in grado di capire il linguaggio dei segni.

2.2 Specifica PEAS

La specifica Peas (Performance Environment Actuators and Sensors) è il primo e fondamentale passo per la formalizzazione del nostro problema attraverso un modello capace di descriverne velocemente le caratteristiche fondamentali. Nel caso di questo progetto, la specifica PEAS dell'ambiente operativo è la seguente:

- **Performance:** La misura di prestazione del sistema si basa sulla capacità dell'agente nella corretta identificazione della singola lettera nell'alfabeto dei segni.
- **Environment:** L'ambiente dell'agente è la foto della mano inviata al classificatore
- **Actuators:** Gli attuatori consistono in un sistema di traduzione della foto ricevuta
- **Sensors:** Il sensore è il sistema di ricezione della foto della mano.

2.3 Caratteristiche dell'ambiente

Dopo aver definito il modello PEAS del nostro problema è fondamentale definirne le caratteristiche dell'ambiente in cui opererà:

- L'ambiente è **Completamente osservabile**, poichè l'agente riceve sempre lo stato completo di esso dato che l'input è una immagine,
- L'ambiente è **Stocastico** poichè l'interpretazione dell'immagine corrente non influenza quella successiva,

- L'ambiente è **Statico** poichè l'agente analizza dei fotogrammi che non possono variare mentre l'agente sta deliberando,
- L'ambiente è **Singolo** poichè ammette un solo agente,
- L'ambiente è **discreto** poichè c'è un numero possibile di azioni finito (le lettere dell'alfabeto trattato sono 24¹).

¹Ovviamente le lettere dell'alfabeto sono 26, ma il dataset utilizzato ha evitato le lettere Z e J poichè richiedono movimenti con la mano

3 Dataset

Il dataset che è stato scelto di utilizzare si chiama sign-language-mnist, è disponibile su kaggle al seguente link.

Questo dataset è ispirato al celebre dataset MNIST che tratta di cifre e lettere scritte a mano, ne condivide alcune caratteristiche come la presenza di label e di immagini di dimensione 28x28.

3.1 Struttura

Il dataset è fornito con una serie di immagini suddivise per classi, che indicano la lettera associata al gesto.

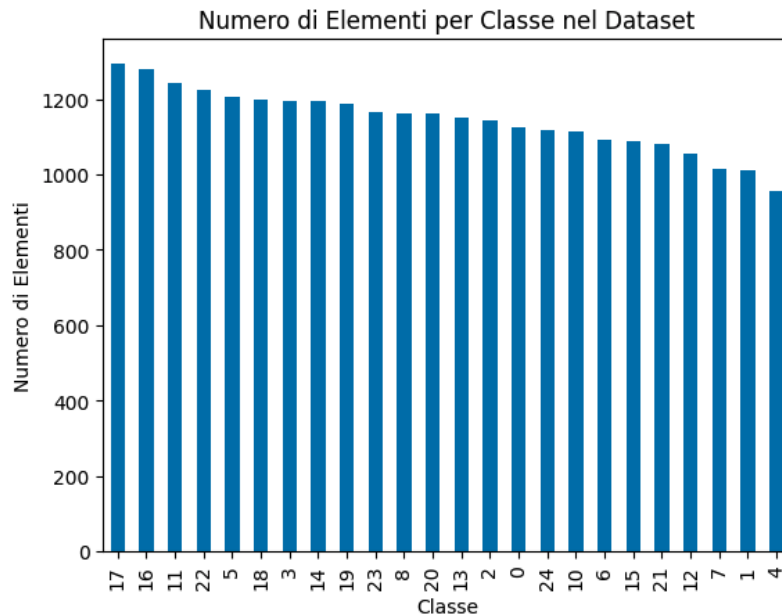
Il dataset è provvisto soltanto di 24 delle 26 lettere poichè le lettere Z e J richiedono movimenti della mano.



3.2 Bilanciamento del dataset

Da una prima analisi si evince che il dataset di allenamento è leggermente sbilanciato, quindi sono state necessarie operazioni di data cleaning per evitare

possibili problemi di overfitting causati dalla maggiore presenza di alcune classi rispetto ad altre.



In questo caso è stato scelto di prendere la classe con meno elementi nel dataset e fare undersampling sulle altre classi andando a rimuovere i dati in eccesso per livellare tutte le classi sullo stesso numero di elementi.

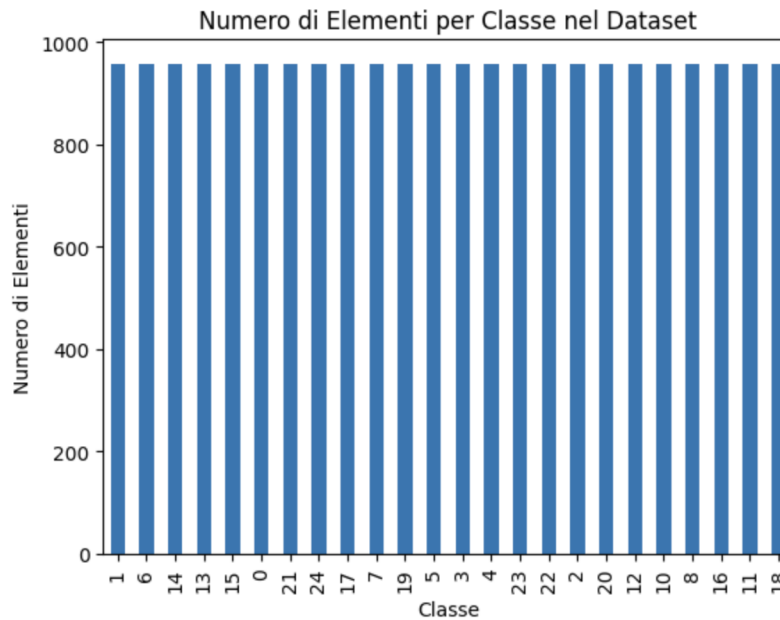
```
# Calcola il numero minimo di elementi tra tutte le classi
min_class_count = df['label'].value_counts().min()

# Esegui l'undersampling per ogni classe
undersampled_df = pd.concat([
    df[df['label'] == label].sample(min_class_count, replace=False, random_state=42)
    for label in df['label'].unique()
])

# Effettuo lo shuffle del DataFrame risultante
undersampled_df = undersampled_df.sample(frac=1, random_state=42).reset_index(drop=True)

df = undersampled_df
```

A questo punto l'operazione di bilanciamento è conclusa poichè il dataset, come si evince dal prossimo grafico, risulta bilanciato.



3.3 Ulteriori operazioni sui dati

Dato che il nostro dataset è un csv in cui ogni riga è composta da una label e 784 celle successive, è stato necessario trasformare il vettore dei pixel dell'immagine in una matrice 28x28.

```
train_features = df.iloc[:, 1:].values
train_labels = df.iloc[:, 0].values

train_original_shape = train_features.shape
print("[TRAIN] Original shape:", train_original_shape)

train_features = train_features.reshape(-1, 28, 28, 1)

# Normalizzo le features (che sono i pixel dell'immagine presi uno per uno)
train_features = train_features / 255.0

print("[TRAIN] Current shape:", train_features.shape)

[TRAIN] Original shape: (22968, 784)
[TRAIN] Current shape: (22968, 28, 28, 1)
```

Notiamo come il risultato dell'operazione di reshaping sia, appunto una matrice 28x28 con un solo canale colore (dato che le immagini sono in scala di grigi).

4 Soluzione proposta

La soluzione proposta è un modello di Machine Learning in grado di riconoscere la lettera associata al gesto fatto con la mano, passato al modello come foto.

4.1 Modello

Il modello scelto è una CNN (Convolutional Neural Network), che ha come obiettivo quello di classificare, tramite varie operazioni, l'immagine ad esso inviata e restituirne la classe di appartenenza (in questo caso la lettera).

4.2 Composizione CNN (Convolutional Neural Network)

Nella composizione della rete neurale sono stati utilizzati i seguenti layer:

- **Conv2d** : Questo tipo di layer si occupa dell'operazione di convoluzione, ossia far passare un kernel(filtro) di dimensione 2x2 nell'immagine per iniziare a mappare le caratteristiche dell'immagine. A questo layer è associata la funzione di attivazione ReLU che si occupa di introdurre non linearità
- **MaxPool2d** : Questo tipo di layer non apprende nulla, ma svolge un ruolo importante nella feature extraction, infatti verrà fatto passare un kernel 2x2 nell'immagine e verrà preso il pixel con il valore più alto. Questo tipo di operazione serve a togliere ancora di più linearità nell'immagine e di conseguenza andare a limitare situazioni di overfitting.
- **Dropout** : Questo tipo di layer ha come compito quello di disconnettere casualmente un certo numero di neuroni, impedendo loro di contribuire all'output in quel passaggio. I layer di dropout sono utili a limitare l'overfitting.
- **Flatten** : Questo layer si occupa di convertire l'output della convoluzione in un vettore monodimensionale. Questo permette di passare dalla rappresentazione spaziale delle caratteristiche estratte dai filtri convoluzionali

a una rappresentazione unidimensionale che può essere utilizzata dai layer densamente connessi.

- **Dense** : Questo tipo di layer, anche chiamato layer densamente connesso poichè ogni nodo (neurone) è connesso con tutti i nodi del layer precedente, utilizzato con funzione di attivazione softmax si occupa di creare una distribuzione di probabilità su tutte le classi di output, quindi è il responsabile della predizione del risultato.

Dopo un'analisi iterativa della rete neurale, provando più combinazioni di layer, è stato notato che il miglior risultato è ottenuto con la seguente composizione:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 27, 27, 64)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 64)	0
conv2d_1 (Conv2D)	(None, 12, 12, 64)	16448
dropout (Dropout)	(None, 12, 12, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 5, 5, 64)	16448
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204928
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 25)	3225

Total params: 241369 (942.85 KB)
Trainable params: 241369 (942.85 KB)
Non-trainable params: 0 (0.00 Byte)

Inoltre, è stato scelto di usare un kernel di dimensione 2x2 per i layer di convoluzione, poichè:

- Impostando un kernel per il layer convoluzionale 2x2 ho una accuracy del 99% sul dataset di TRAIN dopo 1 epoca, su quello di test 90

- Impostando un kernel per il layer convoluzionale 3x3 ho una accuracy del 95% sul dataset di TRAIN dopo 1 epoca, su quello di test 89

Creata la rete neurale, viene compilato il modello con le API messe a disposizione da Tensorflow, utilizzando i seguenti parametri:

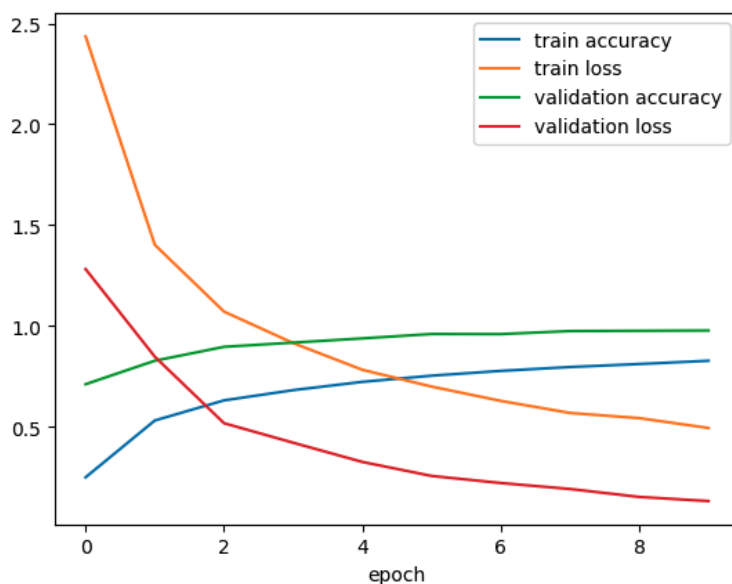
- **Optimizer:** *Adam*, un algoritmo di ottimizzazione che adatta dinamicamente i tassi di apprendimento durante l'allenamento.
- **Funzione di perdita:** *sparse_categorical_crossentropy*, una loss function utilizzata per problemi di classificazione in cui le etichette sono intere, come nel caso studiato.
- **Metriche:** è stato deciso di usare l'*accuracy* come metrica da monitorare durante l'allenamento del modello, ricordando che l'accuratezza è calcolata come $Accuracy_{modello} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$.

4.3 Miglioramento del modello

Dopo i primi test il modello risultava funzionare molto bene sui dati di testing e di training, ma dava problemi con dati nuovi, quindi vi era presenza di overfitting. Questo problema è stato risolto aggiungendo un layer di Dropout e prendendo dei dati dal dataset di training e usandoli come dati di validazione durante l'addestramento del modello.

Queste operazioni hanno portato ad una significativa diminuzione dell'accuratezza sui dati di training al pari di numero di epoche, ma ad un importante aumento dell'accuratezza sui dati di test.

Incrementando, poi, il numero di epoche a 10 è stata raggiunta una accuratezza sul dataset di training dell'80% (contro la precedente 95%), e dell'95% sul dataset di testing (contro la precedente 89%).



Il modello risultava a questo punto significativamente migliore di quello iniziale, ma continuava ad avere problemi con foto esterne, scattate in condizioni diverse da quelle che aveva visto durante il training.

Per risolvere questo problema sono state adottate due strategie parallele: effettuare data augmentation sul dataset di training e maggiore preprocessing sulle immagini per valorizzarne i dettagli utili.

4.3.1 Data augmentation

Considerando che le foto scattate da un potenziale utente non hanno sempre un perfetto sfondo bianco dietro o che la mano non si trova nella perfetta angolazione si è deciso di effettuare un'operazione di data augmentation sul dataset. Questo tipo di operazione serve a generare nuova conoscenza per il modello, basandosi sulle immagini già presenti e effettuando sopra di esse operazioni quali rotazione, zoom, shift, specchiamento, etc...

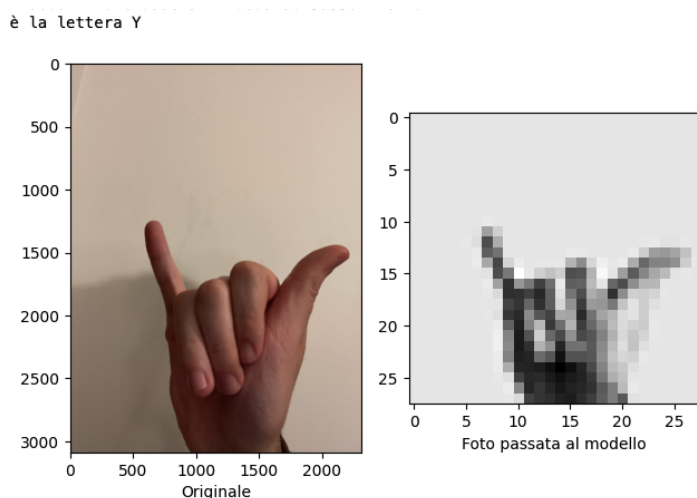
Nel caso di questo modello è stato scelto di effettuare le seguenti operazioni:

- **Rotazione** nel range di $[-10, +10]$ gradi. Questo range è stato scelto poichè una rotazione eccessiva avrebbe potuto confondere il modello,

- **Zoom** nel range di $[-0.1, +0.1]$,
- **Shift** sia in altezza che in larghezza per allenare il modello su foto in cui la mano potrebbe essere leggermente decentrata.

4.3.2 Preprocessing dell'immagine

Questo tipo di operazione è stata messa in atto poichè lo sfondo dietro la foto del gesto fatto con la mano potrebbe confondere il modello. Durante lo sviluppo di questa parte del progetto sono state testate diverse tecniche per massimizzare i dettagli della mano, come aumentare la luminosità o il contrasto dell'immagine, alcune di esse portavano a benefici, altre peggioravano le predizioni del modello. L'equilibrio è stato raggiunto semplicemente trasformando l'immagine in scala di grigi, rimuovendo lo sfondo alla mano attraverso la libreria *rembg* e incollandola su uno sfondo grigio, per poi ridimensionare l'immagine in 28x28.



A questo punto per migliorare ancora di più il modello sono state effettuate alcune prove come quella di diminuire la lunghezza del batch da 32 a 16, che ha contribuito al risultato finale del modello.

4.4 Riepilogo

I dati di accuracy e loss finali sono stati i seguenti (il modello è stato allenato per 10 epoche):

```
1149/1149 [=====] - 9s 8ms/step - loss: 0.4936 - accuracy: 0.8270 - val_loss: 0.1310 - val_accuracy: 0.9771
[TRAIN] Allenamento completato, loss: 0.4936492145061493 accuracy: 0.8269838094711304

: # Valuta il modello sui dati di test
  test_result = model.evaluate(test_dataset)

  print("[TEST] Risultati sul dataset di test:")
  print("[TEST] Loss:", test_result[0], ", Accuracy:", test_result[1], sep=" ")

449/449 [=====] - 1s 3ms/step - loss: 0.2059 - accuracy: 0.9554
[TEST] Risultati sul dataset di test:
[TEST] Loss: 0.20587514340877533 , Accuracy: 0.9553820490837097
```

5 Conclusioni

Questo progetto è stato sviluppato come approfondimento per il corso, il modello non è perfetto, ma si è cercato di ottenere il massimo sfruttando le conoscenze apprese durante il corso e su internet, oltre che effettuando una sperimentazione continua per cercare di migliorare il risultato atteso.

5.1 Sviluppi futuri

Questo modello non è facilmente utilizzabile da utenti comuni, per questo motivo una possibile modifica da fare sarebbe quella di implementare una applicazione user-friendly in modo da rendere utilizzabile il sistema a tutti.

Un'altra possibile implementazione futura sarebbe quella di avere un sistema che supporta il linguaggio dei segni "completo", quindi in grado di funzionare come una specie di t9, completare le parole e velocizzare la comunicazione. Un sistema di questo tipo potrebbe essere ancora più utile integrato in sistemi di videocall come Whatsapp, Teams, Zoom, etc...