Documentazione caso di studio "Emopain Challenge"

Sistemi ad Agenti
a.a. 2022/2023
Prof. ssa Berardina De Carolis

de Cillis Nicolò 736575
De Tullio Roberta 737821
Miranda Caterina 736546
Rubini Giuseppe 739073

INTRODUZIONE

Nel nostro caso di studio abbiamo scelto di progettare un'applicazione in grado di rilevare in tempo reale l'intensità del dolore di un determinato soggetto impiegando tecniche di Computer Vision, in particolare tecniche di riconoscimento facciale e tecniche di analisi ed estrazione delle caratteristiche del volto applicate in ambito Affective Computing con l'obiettivo di analizzare le emozioni provate.

L'Affective Computing è una branca dell'intelligenza artificiale che si occupa dello studio e dello sviluppo di sistemi e dispositivi software in grado di riconoscere, interpretare, simulare ed elaborare le emozioni umane.

Per la realizzazione del nostro caso di studio abbiamo preso come riferimento il dataset, contenente diverse features di espressioni facciali e diversi movimenti con relative etichette di intensità del dolore, della EmoPain Challenge tenutasi nel 2020, ossia la prima competizione internazionale che ha come obiettivo il riconoscimento automatico del dolore per valutare le prestazioni dei metodi di apprendimento automatico utilizzati per riconoscere o quantificare il dolore cronico tramite l'analisi del viso o del corpo o anche comportamenti di movimento correlati al dolore.

STRUTTURA DEL DATASET

Il dataset di EmoPain è costituito sia da file csv contenenti set di features estratte dai volti di ciascun partecipante ed elencate per ogni riga frame-by-frame che da file csv contenenti le etichette per il livello di dolore percepito in ciascun frame.

I partecipanti sono suddivisi in 14 soggetti che presentano dolore lombare cronico (CLBP) e da 22 soggetti che non lo presentano (HP).

La seguente tabella presenta la suddivisione del dataset in training, validation e test set:

Partitions	Face Tasks
Train	8 CLBP and 11 HP
Validation	3 CLBP and 6 HP
Test	3 CLBP and 5 HP

Fra i dati condivisi al pubblico non sono stati forniti dagli organizzatori della challenge quelli del test set così da rendere la competizione equa per ogni concorrente; non è stato quindi possibile utilizzarlo nel nostro caso di studio.

La seguente tabella rappresenta la distribuzione dei frame rispetto a ciascuna label fornita dalla challenge:

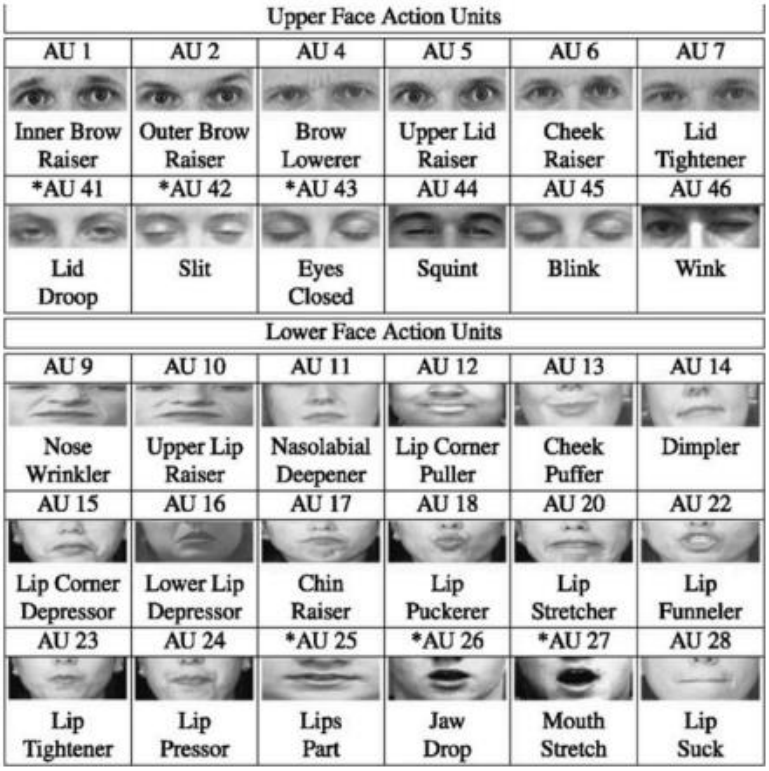
Label value	0	1	2	3	4	5	6	7	8	9	10
Training	646634	39694	31032	61148	41286	17122	16958	9140	3734	626	2078
Development	475717	20731	31697	25613	20765	15416	7425	9972	198	176	218

I dati forniti dal dataset si possono suddividere in features geometriche estratte con l'utilizzo del toolkit OpenFace 2.0, oltre che in Hog feature (Histogram of Oriented Gradients) ed in due tipi di feature orientate alle emozioni estratte da modelli di deep-learning che tuttavia non abbiamo preso in considerazione per il nostro caso di studio.

Tra le features fornite dal dataset ci siamo focalizzati sulle Action Units (AU): queste features sono un sottoinsieme del Facial Action Coding System. Il FACS è un sistema di tassonomia che rileva i micromovimenti dei muscoli facciali durante un cambio di espressione e li codifica in base a espressioni emotive umane.

OpenFace registra sia la presenza di tali tratti in una classificazione binaria che l'intensità che questi hanno tramite regressione.

Di seguito è mostrato il sottoinsieme di AU fornito da OpenFace:



Gli organizzatori della challenge hanno addestrato una Artificial Neural Network per ogni feature estratta e ne hanno valutato le prestazioni in maniera tale da fornire una baseline da dover superare.

Le baseline sono mostrate nella seguente tabella:

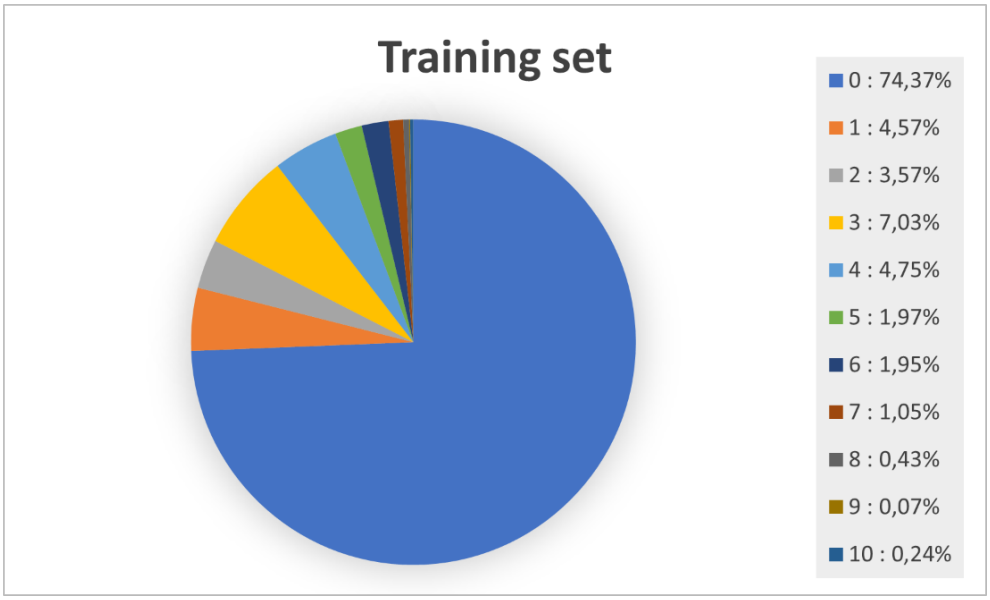
Modality	Partition	MAE	RMSE	PCC	CCC
FL+GAZE	Valid.	1.51	1.74	0.04	0.003
FL+GAZE	Test.	1.37	1.56	**0.10**	0.003
HOG	Valid.	**1.24**	1.91	0.05	0.04
HOG	Test.	0.93	1.61	0.03	0.02
VGG-16	Valid.	1.34	1.82	0.24	**0.18**
VGG-16	Test.	0.92	1.43	0.02	0.004
ResNet-50	Valid.	1.42	2.08	-0.08	-0.04
ResNet-50	Test.	1.14	1.74	-0.09	-0.06
Fusion	Valid.	1.26	**1.69**	**0.25**	**0.18**
Fusion	Test.	**0.91**	**1.41**	**0.10**	**0.06**

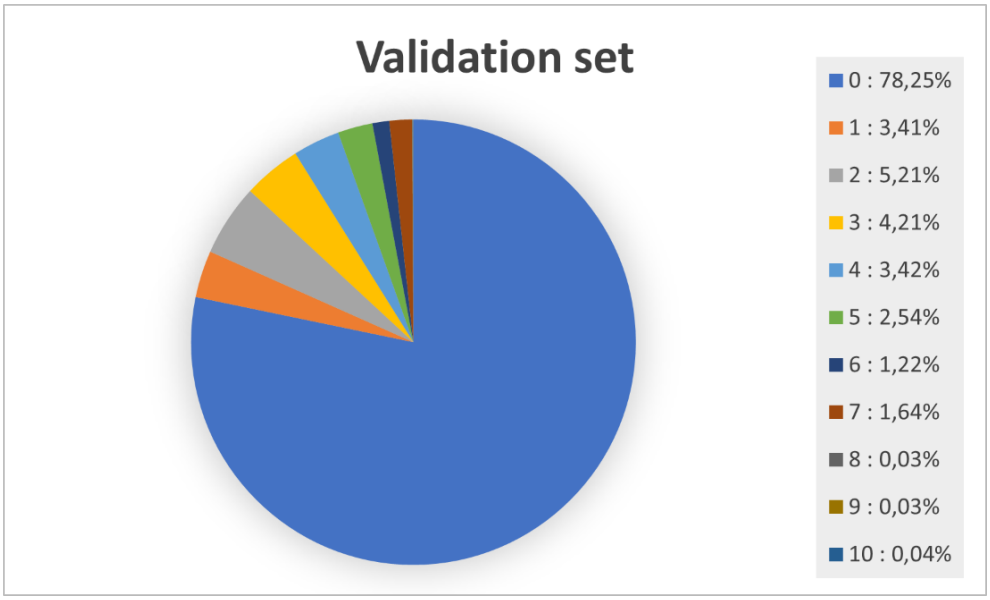
ANALISI DEL DATASET

Per il caso di studio è stato preso in esame il set di features geometriche, nello specifico le Action Units.

Per ogni frame sono state indicate le pain label, ovvero i nostri valori target che indicano l'intensità del dolore provato. Queste hanno un range di 11 valori da 0 a 10 dove zero indica l'assenza di dolore e dieci il dolore estremo. Le pain label sono distribuite nel seguente modo:

Label	0	1	2	3	4	5	6	7	8	9	10
Train set	646634	39694	31032	61148	41286	17122	16958	9140	3734	626	2078
Valid set	475717	20731	31697	25613	20765	15416	7425	9972	198	176	218

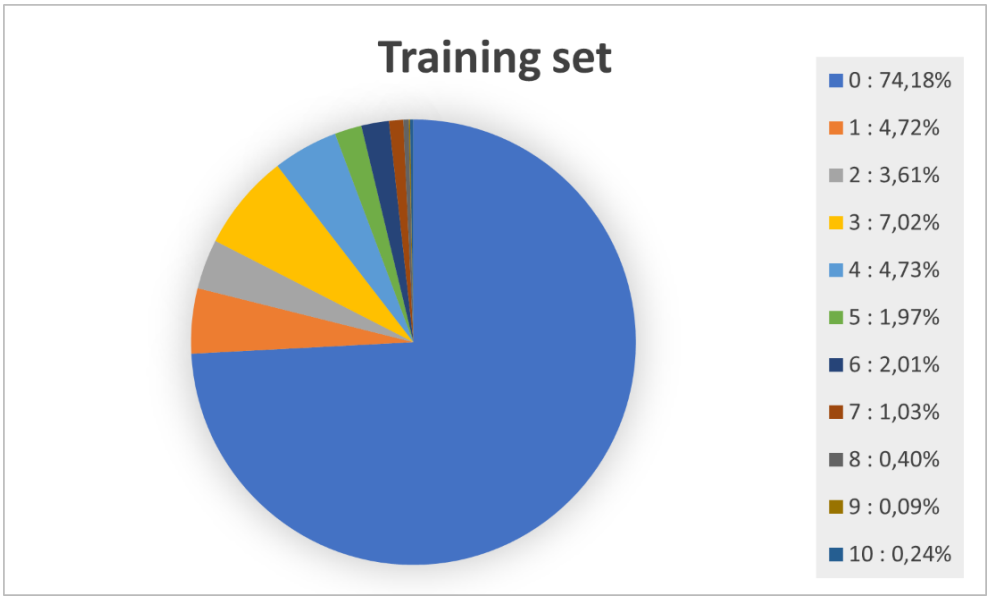


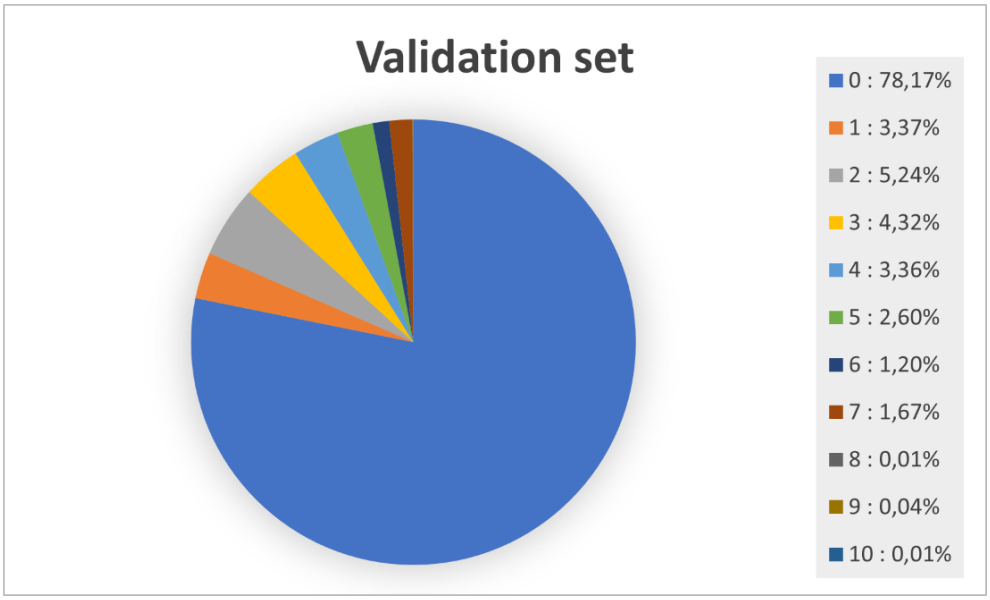


Mantenendo la medesima organizzazione del dataset, abbiamo come primo step realizzato delle finestre temporali costituite ognuna da un numero fisso di frame pari a 90 con un overlapping del 50%.

I dati ottenuti sono i seguenti:

Label	0	1	2	3	4	5	6	7	8	9	10
Train set	14258	908	694	1350	910	378	386	198	76	18	46
Valid set	10504	453	704	580	452	349	161	224	2	6	2





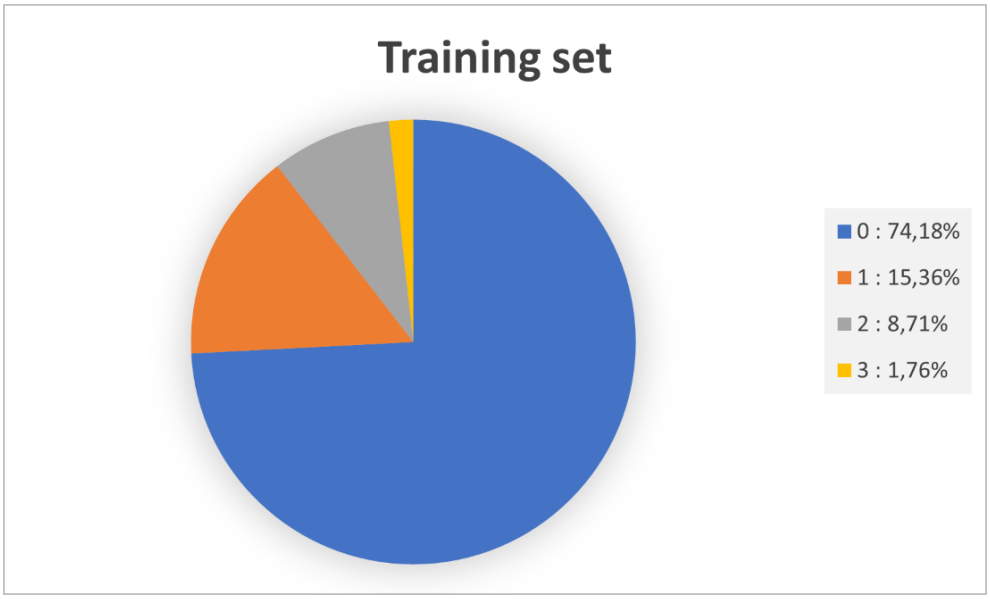
Il precedente grafico ci mostra come il dataset è fortemente sbilanciato, in quanto la maggior parte dei dati sono contrassegnati con una label pari a 0. Come primo passo per la risoluzione del problema abbiamo escluso dal dataset di tutti i dati relativi al campione delle persone che non presentano dolore cronico, in quanto i set di dati non registrano alcuna espressione di dolore e quindi nessuna loro label è diversa da zero.

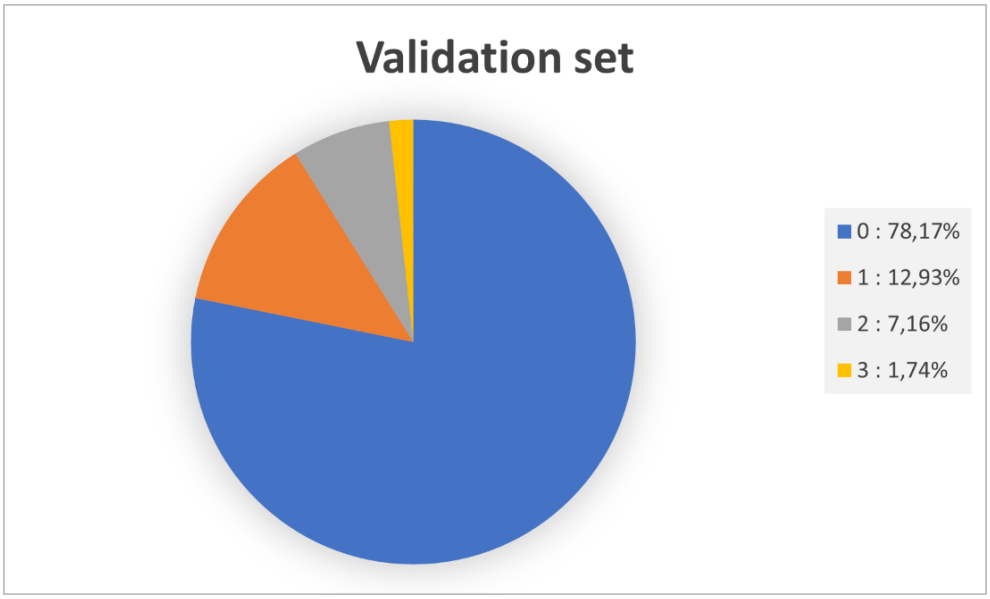
Successivamente, abbiamo pensato di aggregare le 11 pain label in 4 categorie principali, suddivise come segue:

- Pain Label 0: assenza di dolore, costituita dai frame con pain label pari a 0
- Pain Label 1: dolore minimo, costituita dai frame con pain label da 1 a 3
- Pain Label 2: dolore medio, costituita dai frame con pain label da 4 a 6
- Pain Label 3: dolore massimo, costituita dai frame con pain label da 7 a 10

e, mantenendo la suddetta organizzazione in finestre temporali di 90 frame con un 50% di sovrapposizione, abbiamo ottenuto i seguenti dati:

Label	0	1	2	3
Train set	14258	2952	1674	338
Valid set	10504	1737	962	234

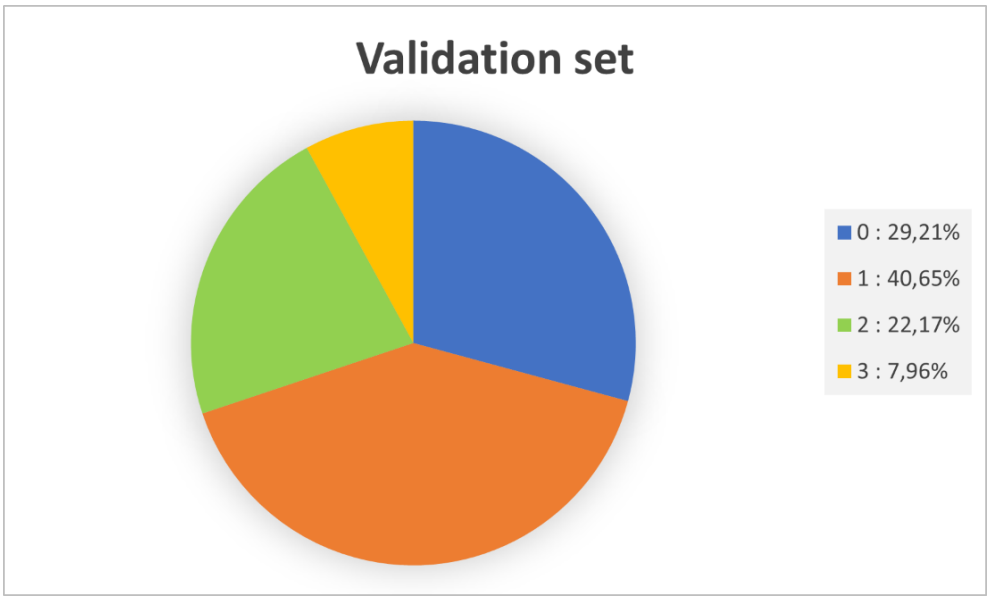




Come ultimo step abbiamo preso in considerazione unicamente quei file costituiti da più del 50% di frame con relativa pain label diversa da 0:

Label	0	1	2	3
Train set	1696	1650	1308	334
Valid set	822	1144	624	224


Training set
0 : 34,00%
1 : 33,08%
2 : 26,22%
3 : 6,70%


Validation set
0 : 29,21%
1 : 40,65%
2 : 22,17%
3 : 7,96%

REALIZZAZIONE DEL MODELLO

Il modello che abbiamo realizzato utilizza una LSTM (Long Short-Term Memory), è un tipo di rete neurale ricorrente (RNN) che è stato progettato per affrontare il problema delle scomparse a lungo termine nelle tradizionali RNN.

Le reti neurali ricorrenti sono utilizzate per elaborare dati sequenziali, come frasi, audio, serie temporali e altro ancora.

Una LSTM è composta da unità di memoria chiamate "celle LSTM" e ognuna di queste celle ha tre porte interne che aiutano la rete a decidere quali informazioni mantenere, quali scartare e quali aggiungere alle informazioni correnti durante l'elaborazione dei dati sequenziali. Queste sono:

- Porta di Input (Input Gate): la porta di input determina quali nuove informazioni verranno aggiunte alla cella LSTM dal passaggio corrente dell'input.
- Porta di Dimenticanza (Forget Gate): la porta di dimenticanza determina quali informazioni presenti nello stato precedente della cella LSTM devono essere dimenticate o ignorate.
- Porta di Output (Output Gate): la porta di output determina quali informazioni dello stato corrente della cella LSTM verranno utilizzate come output della cella.

Qui di seguito riportiamo il codice sorgente del modello e la sua relativa descrizione:

```
model = Sequential()
model.add(LSTM(64, dropout=0.5, recurrent_dropout=0.2,
input_shape=(time_step, num_features), return_sequences=True,
               kernel_regularizer=l2(0.1)))
model.add(LSTM(64, dropout=0.5, recurrent_dropout=0.2,
kernel_regularizer=l2(0.1)))
model.add(Dense(64, activation="relu"))
model.add(Dense(32, activation="relu"))
model.add(Dense(1, activation="relu"))

model.compile(optimizer="adam", loss="mse",
              metrics=['accuracy',
tf.keras.metrics.RootMeanSquaredError(),
tf.keras.metrics.MeanAbsoluteError()])
model.fit(x_train, y_train, epochs=15, verbose=1)
```

Il modello è un tipo di rete neurale sequenziale, che viene definito e addestrato utilizzando la libreria Keras con il backend TensorFlow.

Il modello inizia con un layer LSTM con 64 unità. Questo layer è progettato per lavorare con sequenze di dati, mantenendo una memoria a lungo termine delle informazioni passate.

È presente un dropout del 50%, che viene applicato alle connessioni tra le unità del layer LSTM. Il dropout aiuta a ridurre l'overfitting del modello, disattivando casualmente alcune connessioni durante l'addestramento.

È presente anche un dropout ricorrente del 20%, che viene applicato alle connessioni ricorrenti del layer LSTM. Questo aiuta a regolare il flusso delle informazioni attraverso le sequenze, migliorando la generalizzazione del modello.
L'input del layer LSTM è definito da "input_shape", che specifica il numero di passaggi temporali (time_step) e il numero di caratteristiche (num_features) per ogni passaggio temporale.
Dopodiché, viene aggiunto un secondo layer LSTM con 64 unità. Anche in questo caso, sono presenti il dropout del 50% e il dropout ricorrente del 20%.
Poiché il parametro "return_sequences" non è impostato, il secondo layer LSTM restituisce un singolo valore di output invece di una sequenza.
Dopo i due layer LSTM, ci sono tre layer densi (fully connected) che lavorano sui dati di output dei layer precedenti.
Il primo layer denso ha 64 unità con attivazione "relu", che significa che viene applicata una funzione di attivazione rettificata lineare (ReLU) alle uscite delle unità.
Il secondo layer denso ha 32 unità con attivazione "relu".
Infine, c'è un ultimo layer denso con un singolo neurone e attivazione "relu", che rappresenta l'output finale del modello.
Il modello viene compilato utilizzando l'ottimizzatore "adam", che è un algoritmo di ottimizzazione molto comune e efficiente.
La funzione di loss utilizzata è l'errore quadratico medio (MSE), che è appropriato per problemi di regressione.
Vengono anche specificate alcune metriche di valutazione per monitorare le prestazioni del modello durante l'addestramento, tra cui l'accuratezza, l'errore quadratico medio (RMSE) e l'errore assoluto medio (MAE).
Il modello viene addestrato utilizzando i dati di addestramento "x_train" e "y_train" per un totale di 15 epoche.
Questo modello LSTM con layer densi successivi è comunemente utilizzato per problemi di regressione in cui le sequenze di dati sono importanti per fare previsioni accurate.

METRICHE DI VALUTAZIONE

Le metriche di valutazione adoperate sono le stesse proposte dagli organizzatori della challenge come baseline. Queste comprendono:

- RMSE (Root Mean Squared Error): è una metrica di valutazione dell'errore di previsione per i modelli di regressione. Misura la radice quadrata della media dei quadrati delle differenze tra i valori previsti e i valori effettivi. Un valore più basso indica una migliore accuratezza delle previsioni. La formula è la seguente:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- MAE (Mean Absolute Error): è una metrica di valutazione dell'errore di previsione per i modelli di regressione. Misura la media delle differenze assolute tra le previsioni e i valori effettivi. Un MAE più basso indica una migliore precisione del modello. La formula è la seguente:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

- PCC (Pearson Correlation Coefficient): è una misura di correlazione tra due variabili che rappresenta la correlazione di Pearson tra le previsioni del modello e i valori reali. Un PCC compreso tra -1 e 1, dove 1 indica una forte correlazione positiva, -1 una forte correlazione negativa e 0 indica assenza di correlazione. Si definisce come il rapporto tra la covarianza delle variabili e il prodotto delle deviazioni standard di ciascuna di esse:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}$$

- CCC (Concordance Correlation Coefficient): è una misura di correlazione, spesso utilizzata per valutare la concordanza tra le previsioni e i valori reali in un contesto di regressione. Come il PCC, il CCC varia tra -1 e 1. Un CCC più vicino a 1 indica una concordanza più forte tra le previsioni e i valori reali. Esso è il rapporto tra il prodotto delle deviazioni standard di ciascuna variabile, per due volte il coefficiente di correlazione e la somma della varianza di ciascuna variabile, più il quadrato della differenza delle loro medie:

$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}$$

I risultati ottenuti dal nostro modello sono i seguenti:

	MAE	RMSE	PCC	CCC
BASELINE	1.51	1.74	0.04	0.003
LSTM	0.69	0.90	0.38	0.31

Confrontando i risultati, notiamo che il modello LSTM ha prestazioni generalmente migliori rispetto alla baseline su tutte le metriche considerate, dimostrando di essere più efficace nel riconoscimento dei dati oggetto di studio.

Per quanto riguarda il MAE, il modello LSTM ha un errore medio assoluto di 0.69, che è inferiore rispetto alla baseline, che ha un MAE di 1.51. Un valore inferiore di MAE indica che il modello LSTM ha una maggiore precisione nelle previsioni rispetto al baseline.

Considerando l'RMSE, il modello LSTM ha un errore quadratico medio di 0.90, mentre la baseline ha un RMSE di 1.74. Anche in questo caso, il valore inferiore di RMSE del modello LSTM indica che le sue previsioni sono più vicine ai valori reali rispetto alla baseline.

Il PCC per il modello LSTM è invece pari a 0.38, mentre per la baseline è solo 0.04. Un valore di PCC più alto indica una correlazione più forte tra le previsioni e i valori reali; quindi, il modello LSTM ha una correlazione più significativa rispetto alla baseline.

Infine, considerando il CCC per il modello LSTM è 0.31, mentre per la baseline è solo 0.003. Un valore più alto di CCC indica una maggiore concordanza tra le previsioni e i valori reali. Anche in questo caso, il modello LSTM supera la baseline.

REALIZZAZIONE DELL'APPLICAZIONE

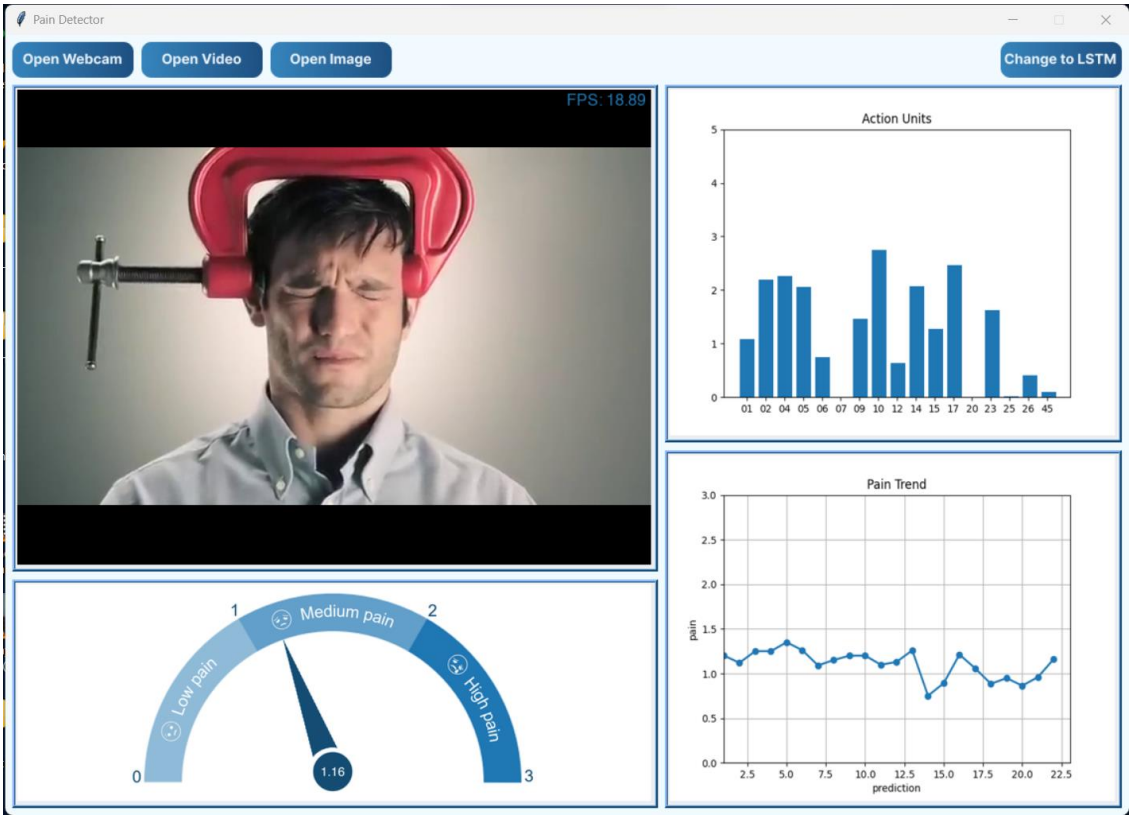
L'applicazione realizzata tramite la libreria Python Tkinter prevede una prima selezione del modello che si intende utilizzare tra LSTM e SVR:

Una volta effettuata la propria scelta, l'applicazione cattura un flusso di dati a partire da una sorgente video o da un'immagine data in input (quest'ultima opzione disponibile solo per SVR in quanto l'LSTM necessita di una sequenza di frame). Per ogni frame, tramite la libreria Py-Feat, il programma identifica il volto, se presente; successivamente vengono estratti i landmark facciali ossia i punti predefiniti sul volto utilizzati per identificare e tracciare le caratteristiche facciali e infine vengono estratte le action unit a partire dai landmark facciali ottenuti.

Le intensità delle action unit vengono poi moltiplicate per cinque in modo da renderle compatibili con quelle fornite dal dataset.

I risultati vengono dati in input al modello scelto e nel caso dell'SVR viene effettuata un riconoscimento frame-by-frame; invece, nel caso dell'LSTM la prima stima viene effettuata dopo aver raccolto i primi novanta frame, mentre le successive vengono effettuate mantenendo un overlapping del 50%.

La valutazione viene mostrata a schermo con l'ausilio di grafici. Nel caso in cui l'utente abbia selezionato la webcam o il video, il programma mostra un istogramma relativo alle intensità delle action unit, un indicatore del livello del dolore e un grafico che indica l'andamento dell'intensità del dolore nel tempo:

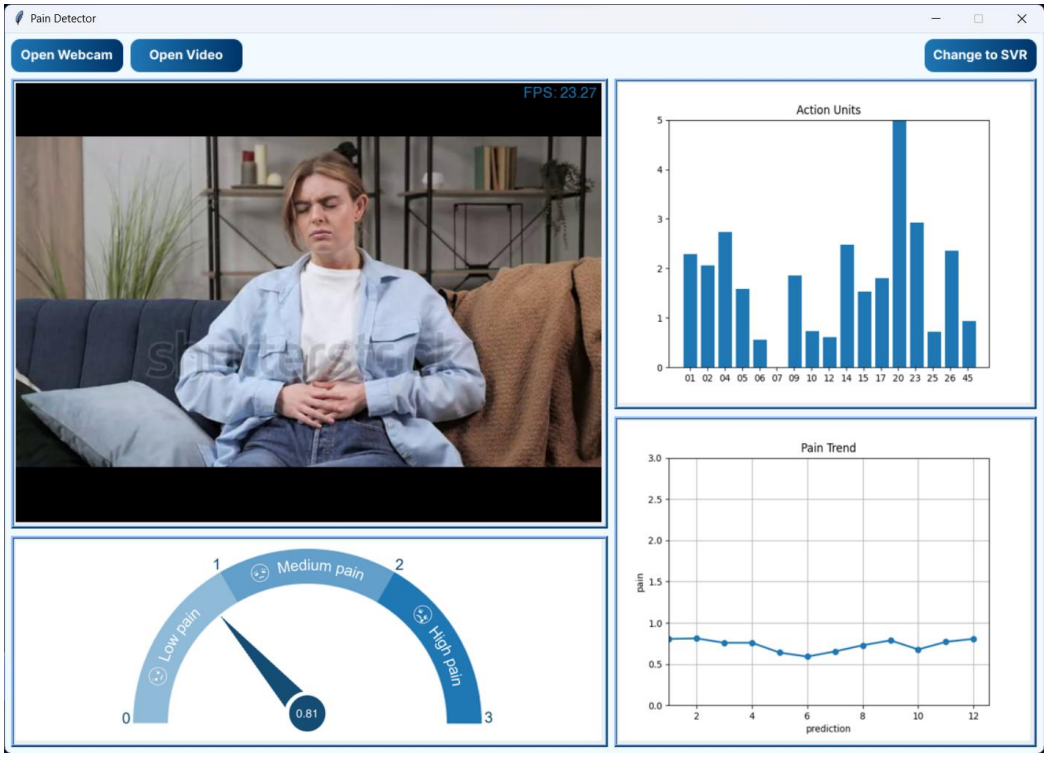


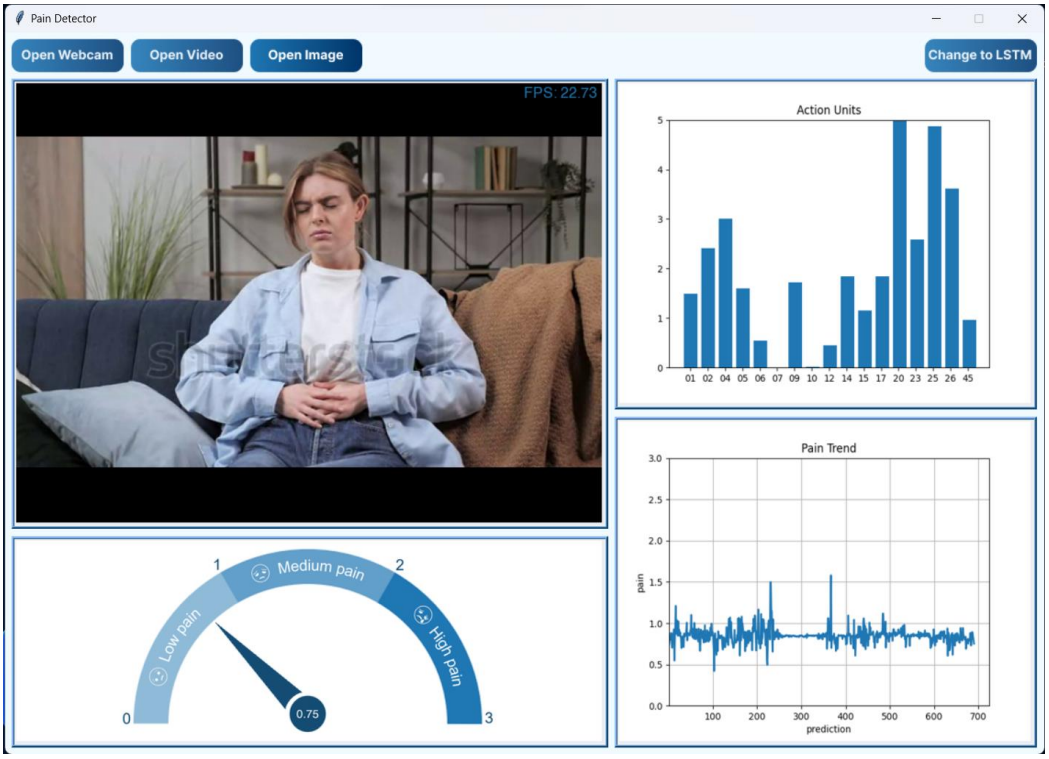
Nel caso in cui l'utente abbia selezionato un'immagine, il programma mostra invece solo l'istogramma delle intensità delle action unit e l'indicatore del livello del dolore:

CONCLUSIONI

L'obiettivo del nostro caso era quello di creare una rete deep, in particolare una LSTM confrontando i risultati con il modello di regressione SVR fornitoci.

A livello teorico i risultati ottenuti dall'LSTM sul validation set si sono dimostrati migliori rispetto a quelli ottenuti dall'SVR. Tuttavia, a livello pratico è possibile notare che i risultati sono molto simili fra di loro come mostrano le seguenti immagini:





Concludendo, riteniamo che per sfruttare l'applicazione in tempo reale sia preferibile utilizzare come modello l'SVR in quanto l'accuratezza delle predizioni effettuate dai due modelli è molto simile e le prestazioni, in termini di tempo, sono migliori.
Tuttavia, a prescindere dal modello utilizzato, crediamo che disponendo di un dataset migliore l'applicazione possa essere più accurata nel riconoscimento del dolore in quanto potrebbe risultare molto utile in diversi contesti, tra cui:

- Assistenza sanitaria: potrebbe essere utilizzato in strutture mediche e ospedali per valutare e monitorare il dolore dei pazienti in modo oggettivo e non invasivo, ad esempio per pazienti che non possono comunicare verbalmente il loro livello di dolore, come i neonati o pazienti con disabilità cognitive.
- Ricerca clinica: per valutare l'efficacia di trattamenti farmacologici o terapie non farmacologiche per il dolore consentendo una misurazione più accurata e oggettiva dei risultati delle terapie così da permettere lo sviluppo di nuovi approcci per il sollievo del dolore.
- Terapie mediche e dentali: durante procedure mediche o dentali, il riconoscimento del dolore potrebbe essere utilizzato per controllare la risposta del paziente e garantire un'esperienza più confortevole e sicura.