



UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

FACOLTÀ DI SCIENZE MM. FF. NN.

DIPARTIMENTO DI INFORMATICA
CORSO DI LAUREA IN INFORMATICA

ESAME DI
SISTEMI ED AGENTI

ANALISI, APPRENDIMENTO E TESTING PER IL RICONOSCIMENTO DELLE EMOZIONI COGNITIVE SECONDARIE DAL VOLTO

SUPERVISIONE

CHIAR.MA PROF. BERARDINA DE CAROLIS

STUDENTI

FRANCESCO FARINOLA
GAETANO DE GENNARO
TIZIANA DI PIERRO

INDICE

1. Introduzione	2
2. Analisi del software FEAtuREs	2
2.1 Metodi e strumenti utilizzati	3
2.1.1 Visual Studio	3
2.1.2 Computer Vision	3
2.1.3 Librerie C++ utilizzate	3
2.2 Individuazione della problematica	4
3. Soluzione proposta	5
3.1 Creazione del nuovo dataset	5
3.1.1 Dataset validati utilizzati	5
3.1.2 Criteri per la scelta di immagini prese dal web	6
3.2 Apprendimento supervisionato	7
3.2.1 Strumenti utilizzati	7
3.2.1.1 WEKA	7
3.2.1.2 OpenFace	8
3.2.2 Estrazione delle features	8
3.2.3 Preparazione dei dati e sperimentazione	9
3.2.4 Risultati ottenuti	10
3.3 Possibili implementazioni della soluzione	15
3.3.1 Modifiche al software FEAtuREs	15
3.3.2 Altre soluzioni	17
4. Testing	17
4.1 Raccolta di nuove immagini dal web	17
4.2 Creazione di un Google Form per la validazione	18
4.3 Risultati della validazione	18
4.4 Test delle immagini sui modelli creati	21
5. Conclusioni	22

1. Introduzione

Il riconoscimento delle emozioni dalle espressioni del volto umano da parte di una macchina è una problematica interessante e attuale. La Computer Vision si occupa del problema del riconoscimento delle espressioni facciali, e ha messo a punto tecniche per l'individuazione e la codifica di queste, basate essenzialmente sull'elaborazione delle immagini (image processing). L'image-processing trova applicazione in diversi ambiti tra cui: ambito medico, lavorativo, nel mondo virtuale (social network), e-learning, Problem Solving e Decision Making.

Con il seguente caso di studio, sono state affrontate diverse problematiche. In una prima fase, è stata effettuata l'analisi di un software per il riconoscimento di emozioni secondarie creato da uno studente dell'Università degli Studi di Bari, Giuseppe Anzivino, per verificarne l'effettiva affidabilità. Dopo aver individuato le problematiche, è stata proposta una soluzione e delle valide alternative per l'implementazione. A supporto di questo lavoro, è stato creato un nuovo Dataset di volti e sono stati addestrati dei modelli con algoritmi di Machine Learning. Per concludere, si è cercato sul web un set di immagini che rappresentassero alcune delle emozioni secondarie prese in considerazione. Queste, poi, sono state analizzate da diverse tipologie di utenti (studenti, professori, psicologi), classificate definitivamente al fine di poterle utilizzare per progetti futuri e utilizzate per testare la soluzione precedentemente proposta.

2. Analisi del software FEAtuREs

La prima fase del caso di studio ha previsto l'analisi del software FEAtuREs col fine di individuare particolari problematiche nel suo funzionamento. Il sistema FEAtuREs (Facial Expression Analysis for Recognition of Emotions) ha lo scopo di analizzare il volto dell'utente acquisito in real-time tramite le HOG features e di riconoscerne l'espressione facciale. Le emozioni che il software è in grado di individuare sono 11 e sono: entusiasmo, interesse, sorpresa, curiosità, concentrazione, attenzione, delusione, noia, perplessità, fastidio e frustrazione.

2.1 Metodi e strumenti utilizzati

2.1.1 Visual Studio

Per poter sviluppare il sistema FEAtuREs (Facial Expression Analysis for Recognition of Emotions) è stato utilizzato **Visual Studio**, un ambiente di sviluppo integrato (Integrated Development Environment o IDE) sviluppato da Microsoft, che supporta attualmente tantissimi tipi di linguaggi di programmazione, quali C, C++, C#, F#, Python, Visual Basic .Net, Html, JavaScript (e molti altri), e che permette la realizzazione di applicazioni, siti web, applicazioni web, servizi web e altro ancora. Per il software corrente è stato utilizzato il linguaggio di programmazione C++ poiché supporta meglio di altri librerie per la Computer Vision come OpenCV.

2.1.2 Computer Vision

La **Computer Vision** è la disciplina che studia come abilitare i computer alla comprensione e alla interpretazione delle informazioni visuali presenti in immagini o video. La Visione è forse il senso più importante che l'uomo possiede. Essa permette di inferire il mondo tridimensionale, di riconoscere e localizzare gli oggetti presenti in una scena, di percepire i rapidi cambiamenti dell'ambiente. Se volessimo dare una definizione di Computer Vision, potremmo dire che l'insieme dei processi che mirano a creare un modello approssimato del mondo reale (3D) partendo da immagini bidimensionali (2D). L'analisi, però, è finalizzata a scoprire cosa è presente nella scena e dove.

2.1.3 Librerie C++ utilizzate

Per lo sviluppo del software FEAtuREs sono state utilizzate diverse librerie:

- **OpenCV:** è una libreria open-source multiplatforma per la Computer Vision in real-time. È una libreria sviluppata da Intel ed è principalmente sviluppata per C++, ma è possibile interfacciarsi anche attraverso C, Python e Java. La libreria permette una semplice gestione di immagini trattandole come 'matrici di pixel', alle quali è possibile accedervi in maniera molto semplice e rapida. Viene utilizzata per l'elaborazione e l'analisi delle immagini, la calibrazione e il

tracking. Contiene inoltre molte funzioni per il Machine Learning e Pattern Recognition, e per la gestione di interfacce per webcam.

- **Dlib:** è una moderna libreria scritta in C++ composta da algoritmi di machine learning e strumenti per la creazione di software complessi, utili alla risoluzione di problemi del mondo reale. Il suo impiego è presente in moltissimi domini applicativi, tra cui la robotica, i dispositivi embedded, smartphone e gli ambienti di calcolo ad alte prestazioni. Dlib ha una licenza open-source e può essere utilizzata in qualsiasi applicazione. Viene usata principalmente per rilevare il volto di una persona tramite il meccanismo dei landmark facciali
- **LibSVM:** per classificare l'emozione del volto. LIBSVM è utilizzata per l'attività di apprendimento, per la classificazione vettoriale (C-SVC, nu-SVC, multi C-SVC), la regressione (epsilon-SVR, nu-SVR) e la stima della distribuzione (SVM di una classe). In questo lavoro è stato utilizzato un approccio multiclass in modo da predire una percentuale per tutte le classi di emozioni secondarie prese in considerazione.
- **Qt:** è una libreria multiplatforma per lo sviluppo di programmi con interfaccia grafica tramite l'uso di widget. Permette lo sviluppo di applicazioni per Windows, Mac OS, Linux, Android e iOS adoperando linguaggi di programmazione come C++ e Python. La potenza di Qt risiede nel fatto che il codice scritto può essere compilato ed eseguito sulle diverse piattaforme target senza alcuna modifica.

2.2 Individuazione della problematica

Dopo aver effettuato alcuni test del software in modalità webcam è emerso che le percentuali delle emozioni secondarie individuate non superavano la soglia del 30-35%. È stato analizzato il codice, ma non è stata individuata la presenza di particolari errori in riferimento al calcolo delle percentuali. Nonostante i risultati molto positivi ottenuti da G. Anzivino nel training del suo modello, è stato ipotizzato che il non superamento di una certa soglia nella stima dell'espressione sia dovuto all'alta numerosità di emozioni analizzate e che, alcune di queste, in certi casi possono rivelarsi ambigue. Es. Interesse, attenzione e curiosità. Infatti, è stato notato, durante la fase di test, che alcune emozioni vengono mal interpretate o confuse con altre. Ad esempio, più volte, un soggetto che si

mostra con espressione di “interesse” in modalità real-time viene classificato come “frustrato”.

3. Soluzione proposta

Al fine di verificare questa nostra ipotesi sono stati creati due nuovi dataset in cui le emozioni secondarie analizzate sono solo 7, ovvero noia, sorpresa, interesse, neutrale, frustrazione, perplessità e entusiasmo. Il primo dataset è composto esclusivamente da immagini prese da altri dataset validati, mentre il secondo è composto dalle precedenti con l’aggiunta di altre immagini prese dal web. Successivamente, da queste immagini sono state estratte le HOG features, le Action Unit e feature sullo sguardo (Gaze) utilizzando il software OpenFace. Successivamente, con WEKA, sono stati creati dei modelli di apprendimento automatico e analizzate le performance per ciascuno di questi. In relazione ai risultati ottenuti sono state, infine, proposte delle possibili nuove implementazioni e modifiche al software FEAtuREs.

3.1 Creazione del nuovo dataset

Innanzitutto, sono stati creati due nuovi dataset di immagini classificate per l’emozione rappresentata. Sono state prese in considerazione solo 7 delle 11 emozioni secondarie analizzate da FEAtuREs in modo da avere 3 categorie di emozioni positive, 3 negative e una neutra. Le emozioni scelte sono quindi: sorpresa, entusiasmo, interesse, neutrale, noia, frustrazione e perplessità. Inoltre, è di notevole importanza che il dataset abbia per ciascun soggetto, una sola immagine associata alla emozione rispettiva in quanto la presenza di immagini o frame di stessi soggetti potrebbe provocare ridondanza nel dataset e quindi ottenere un dataset “subject dependent”.

3.1.1 Dataset validati utilizzati

Per la creazione del primo dataset (Dataset A), sono state utilizzate esclusivamente immagini di dataset con emozioni già validate da esperti.

I dataset utilizzati sono:

1. **EU-Emotion Stimulus Set** dell'University of Cambridge
2. **Yale Face** dataset dell'UC San Diego
3. **Japanese Female Facial Expression (JAFFE)** dell'University of Kyushu
4. **Senthil IRTT** database

Il Dataset A creato dall'unione di questi ha la seguente numerosità:

Boredom	16
Surprise	16
Interest	14
Neutral	16
Frustration	9
Perplexity	12
Enthusiasm	17

3.1.2 Criteri per la scelta di immagini prese dal web

Per la creazione del Dataset B, sono state utilizzate le immagini del Dataset A con l'aggiunta di immagini prese dal web tenendo conto di alcuni accorgimenti. È stato necessario cercare immagini non sfocate, il più frontali possibili ed escludendo soggetti anziani o con barba poiché questo tipo di immagini potrebbero interferire con la corretta estrazione delle features. Inoltre, il numero di immagini individuate sul web possiede un numero tale da poter bilanciare il Dataset, e avere così cardinalità simile per ciascuna classe. È stato creato il Dataset B con la seguente numerosità:

Boredom	29
Surprise	28
Interest	33
Neutral	27
Frustration	29
Perplexity	29
Enthusiasm	30

3.2 Apprendimento supervisionato

L'**apprendimento automatico** (noto come Machine Learning) rappresenta una delle aree fondamentali dell'Intelligenza Artificiale e si occupa della realizzazione di sistemi che si basano su osservazioni o esempi come dati per la sintesi di una nuova conoscenza (classificazioni, generalizzazioni, riformulazioni). Gli algoritmi di apprendimento automatico sono divisi in due principali tipologie:

- **Apprendimento supervisionato:** un istruttore fornisce esempi (e controesempi) di quello che si deve apprendere.
- **Apprendimento non supervisionato:** parte da osservazioni non pre-classificate.

Data la presenza di esempi, in questo caso di studio verrà utilizzato l'apprendimento supervisionato.

3.2.1 Strumenti utilizzati

3.2.1.1 WEKA

WEKA, acronimo di “Waikato Environment for Knowledge Analysis”, è un software sviluppato dall'università di Waikato in Nuova Zelanda. È open source ed è un ambiente interamente scritto in Java. Nell'ambito del machine learning, il workbench WEKA è una collezione di algoritmi e tecniche di pre-processamento allo stato dell'arte. È stato sviluppato per testare velocemente metodologie esistenti su insiemi di dati diversi, in modo flessibile; prevede infatti il supporto per tutto il processo sperimentale del data mining: la preparazione dei dati di input, la valutazione statistica degli schemi di apprendimento, la visualizzazione grafica dei dati di input e del risultato dell'apprendimento. WEKA supporta un particolare formato di file: Attribute-Relation File Format ARFF. In particolare, è un tipo di file di testo ASCII che descrive una lista di istanze che condividono degli attributi. Un **file ARFF** è composto da due distinte parti: l'Header che contiene il nome della relazione, una lista di attributi e il loro tipo; Data che contiene i valori di ciascuna istanza per i vari attributi.

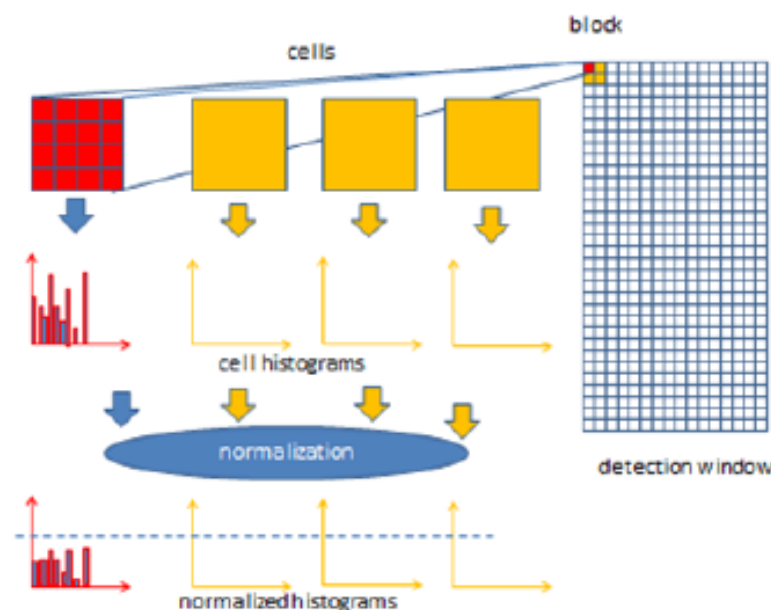
3.2.1.2 OpenFace

OpenFace è un tool destinato ai ricercatori di Computer Vision e Machine Learning. Permette di analizzare immagini e video, di estrarne i punti facciali, la posizione della testa, le Action Unit del viso, la direzione dello sguardo, i descrittori HOG e di registrare i risultati in un file CSV. È totalmente gratuito ed è sviluppato da Tadas Baltrušaitis in collaborazione con l'Università di Cambridge.

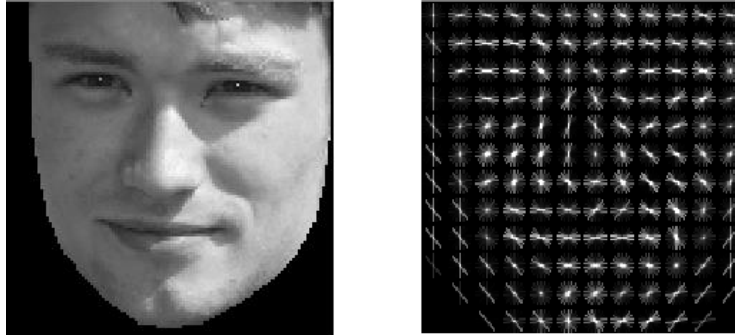
3.2.2 Estrazione delle features

Utilizzando il software OpenFace, è stato possibile estrarre in un file CSV diverse **features** da ciascuna immagine del dataset:



















- **Descrittori HOG (Histogram of Oriented Gradients):** si tratta di una potente tecnica di descrizione delle forme che conta le occorrenze dell'orientamento del gradiente in porzioni localizzate di una immagine, e che è intuitivamente utile per modellare la forma dei muscoli facciali mediante un'analisi del bordo. L'algoritmo del descrittore HOG divide l'immagine in piccole aree collegate chiamate 'celle' e per ogni cella calcola un istogramma di direzioni di gradiente per i pixel all'interno della cella. Discretizza ogni cella in contenitori angolari in base all'orientamento. I gruppi di celle adiacenti, detti blocchi, vengono normalizzati e l'insieme degli istogrammi a blocchi rappresenta il descrittore HOG.



Prima di fare questo, però, OpenFace trova e isola il volto dei soggetti nelle immagini e analizza solo le celle appartenenti a quest'ultimo. Il software è in grado di calcolare 4464 descrittori HOG per ciascun volto;



- **Sguardo (Gaze)** : esegue una stima dell'orientamento dello sguardo del soggetto
– se verso l'alto/basso e se verso destra/sinistra;
- **Action Unit (AU)**: i movimenti dei singoli muscoli facciali sono codificati da un sistema (FACS – Facial Action Coding System) tramite lievi cambiamenti istantanei nell'aspetto del viso. Usando FACS è possibile codificare praticamente qualsiasi espressione facciale anatomicamente possibile, decostruendola nelle specifiche Action Unit che hanno prodotto l'espressione. È uno standard comune per descrivere oggettivamente le espressioni facciali. OpenFace è in grado di registrare la presenza (classificazione) e l'intensità (regressione – float da 0 a 5) di un subset di AU, nello specifico:

AU01 - Inner Brow raiser		AU14 - Dimpler	
AU02 - Outer Brow raiser		AU15 - Lip corner depres	
AU04 - Brow lowerer		AU17 - Chin Raiser	
AU05 - Upper lid raiser		AU20 - Lip Stretcher	
AU06 - Cheek raiser		AU23 - Lip tightener	
AU07 - Lid tightener		AU25 - Lips part	
AU09 - Nose wrinkler		AU26 - Jaw drop	
AU10 - Upper lip raiser		AU28 - Lip suck	
AU12 - Lip corner puller (	AU45 - Blink	



Per il processo di estrazione delle features, su OpenFace è possibile utilizzare diversi modelli di Face Detection (ricerca del volto) e Landmark Detection (ricerca dei punti facciali).

I modelli proposti di **Face Detection** sono:

1. **Haar** di OpenCV: usa un approccio basato su Machine Learning dove una funzione 'a cascata' è allenata su molte immagini positive (con presenza di volti) e negative. Questo modello lavora quasi in real-time su CPU, trova volti di dimensioni differenti ma genera molte false predizioni e non funziona se il volto è ostruito (sotto occlusione) o ruotato leggermente.
2. **HOG-SVM** di dlib: è un modello su cui è stato fatto un training di 2825 immagini raccolte dall'autore di Dlib, Davis King. Si basa principalmente sulle HOG features e l'algoritmo SVM ed è stato creato da 5 filtri HOG (sguardo frontale, sguardo a destra, sguardo a sinistra, sguardo frontale ma ruotato a destra, sguardo frontale ruotato a sinistra). Questo modello è molto leggero e lavora molto bene su volti in posizione frontale o leggermente ruotate. Di contro, non riesce a riconoscere volti più piccoli di 80x80 pixel e non lavora bene con volti sotto occlusione. Inoltre, in alcuni casi, potrebbe tagliare parte della fronte e del mento.
3. **Multi-task Cascaded Convolutional Network MTCNN** di OpenFace: è un modello basato su Deep Learning. In questo modello ogni singola immagine passa attraverso tre Convolutional Neural Network CNN. Nella prima, vengono proposte delle regioni dell'immagine che possono contenere volti, nella seconda

vengono scartate le regioni che sono falsi candidati e nella terza viene fatta un'ulteriore rifinitura delle regioni. Questo modello ha un costo computazionale oneroso, ma supera di molto, in termini di performance, i precedenti due. Utilizzeremo questo, per l'analisi delle nostre feature.

Per determinare i punti facciali (**Face Landmark**) possiamo utilizzare 3 diversi modelli:

1. **Constrained Local Models CLM:** cerca ciascun punto facciale usando dei 'local detector' e utilizza un modello per eseguire un'ottimizzazione 'costretta'. Questo modello ha problemi in assenza di luce, presenza di occlusione e con dataset non validati
2. **Constrained Local Neural Field CLNF:** è una variante del CLM che implementa una Local Neural Field che permette di catturare informazioni più complesse per gestire la distanza tra i pixel. Questo modello risolve il problema della luminosità dell'immagine ma non quello dell'occlusione.
3. **Convolutional Experts Constrained Local Model CE-CLM:** anche questa è una variante del CLM che utilizza una Convolutional Experts Network CEN come local detector – è basato sul Deep Learning ed è capace di estrarre i punti facciali 3D senza dover essere allenato su dati 3D. In termini di performance risulta molto superiore alle precedenti due. Per questo motivo utilizzeremo questo modello nell'analisi delle features.

3.2.3 Preparazione dei dati e sperimentazione

Una volta estratte tutte le features necessarie, mediante il programma OpenFace, sono stati ottenuti dei file aventi estensione CSV, acronimo per Comma Separated Values: essi rappresentano tabelle in cui le righe sono linee di testo, divise in campi separati da un carattere speciale quale la virgola. Questi file vengono convertiti in file ARFF dal WEKA ARFFViewer automaticamente. Sono stati creati per ciascun Dataset 3 file ARFF: un file contenente i descrittori HOG, uno con solo le Action Unit e uno con Action Unit + Sguardo (Gaze) per tutte le immagini dei rispettivi Dataset. Tramite l'Explorer di WEKA sono stati pre-processati i dati dei file ARFF, eliminando le informazioni superflue, e, successivamente, sono stati effettuati dei test di classificazione per ciascun file creato.

Gli algoritmi di classificazione utilizzati sono:

- **Random Forest:** algoritmo di classificazione multiclass, concepito come variante del metodo di Bagging, che impiega esclusivamente alberi decisionali come classificatori di base. Una Random Forest può essere considerata come un insieme F di alberi decisionali generati casualmente, tali che ciascun elemento di esso classifichi ogni oggetto x dello spazio in modo indipendente. L'algoritmo di Bagging prevede che ciascun albero di una Random Forest sia addestrato con il metodo di ri-sostituzione su di un diverso sottoinsieme casuale di cardinalità fissata generato a partire dai dati a disposizione. Anche le caratteristiche e le soglie per ciascun nodo vengono scelte da un insieme casuale. La proprietà principale della Random Forest è che al crescere del numero di alberi le prestazioni convergono ad un preciso valore, senza generare un sovrallenamento dell'insieme (*overfitting*). Per il task di classificazione tuttavia, sono stati presi in considerazione tutti gli attributi di ciascuna istanza e sono state computeate 1000 iterazioni. Per questo tipo di algoritmo sono stati effettuati due tipi di training:

1. **K folds Cross Validation:** consiste nella suddivisione casuale del dataset in k parti (*fold*) di uguale numerosità e, ad ogni passo, la k -esima parte del dataset viene considerata come validation dataset, mentre la restante parte costituisce il training dataset. Così, per ognuna delle k parti si allena il modello, evitando *overfitting* e problemi di campionamento asimmetrico (*bias*) del training set. Questo metodo è utilizzato principalmente su dataset con alta numerosità. Durante il training sono stati utilizzati 10 folds.
2. **Leave-One-Out:** è un particolare caso di Cross Validation in cui ciascun fold contiene un solo pattern. Quindi il numero di fold corrisponde al numero di istanze del dataset. Il vantaggio principale è che ha un *bias* inferiore perché utilizza un training set con $n-1$ osservazioni ma è caratterizzato da un'elevata variabilità delle stime. È principalmente usato per dataset di piccole dimensioni in quanto ha costi di computazione molto alti e richiede molto tempo il training di un modello con questo metodo.

- **Support Vector Machine SVM Multiclass:** l'algoritmo SVM costruisce un modello che assegna nuovi esempi a una o all'altra categoria, rendendolo un classificatore lineare binario non probabilistico. Rappresenta gli esempi come punti nello spazio, mappati in modo che gli esempi delle categorie separate siano chiaramente divisi. Dato un insieme di dati di formazione etichettati (apprendimento supervisionato), l'algoritmo calcola un iperpiano ottimale (il modello addestrato) che classifica nuovi esempi nella classe giusta. È stato adottato un approccio multiclass in modo da predire una percentuale per tutte le classi prese in considerazione, piuttosto che predire una sola classe. In particolare, nella seguente sperimentazione è stata utilizzata l'attività di apprendimento per la classificazione vettoriale (multi C-SVC), implementata nella libreria LibSVM. È stato utilizzato il kernel RBF Radial Basis Function e sono stati ottimizzati i valori di '*costo*' e '*gamma*' in riferimento all'accuracy tramite la funzione fornita da WEKA di 'gridSearch'. In particolare, 'gridSearch' implementa un metodo che permette di ottimizzare due parametri di un algoritmo di classificazione scegliendone la migliore combinazione in riferimento ad una specifica metrica di valutazione (accuracy, coefficiente di correlazione, MSE, etc.). Per questa sperimentazione sono stati usati 10 fold nella cross validation.
- **MultiLayer Perceptron MLP:** è un modello di rete neurale artificiale che mappa insiemi di dati in ingresso in un insieme appropriato di dati in uscita. È costituito da strati multipli di nodi che formano un grafo diretto, con ogni strato completamente connesso al successivo, eccetto che per i nodi in ingresso. Ogni nodo è un neurone (elemento elaborante) con una funzione di attivazione non lineare. Il MultiLayer Perceptron usa una tecnica di apprendimento supervisionato chiamata Backpropagation per l'addestramento della rete. Richiede un'uscita desiderata per ogni valore in ingresso per poter calcolare il gradiente della funzione di perdita (funzione di costo), e richiede che la funzione d'attivazione usata dai neuroni artificiali (o "nodi") sia continua e differenziabile. Anche in questa sperimentazione sono stati utilizzati 10 fold nella cross validation. I parametri utilizzati sono: learning rate=0.3, momentum=0.2, numero di epoche=500.

3.2.4 Risultati ottenuti

Dopo aver effettuato le sperimentazioni degli algoritmi Random Forest (con 10 fold cross validation e leave-one-out), SVM e MLP per ciascun dataset (Dataset A e B) e per ciascun file ARFF (HOG features, Action Unit e Action Unit + Sguardo). Di seguito i risultati ottenuti:

DATASET	FEATURES	ALGORITHM	PRECISION	F1
A	HOG	SVM	0.509	0.515
		RF 10 FOLD	0.496	0.500
		RF Leave-One-Out	0.383	0.414
		MLP	-	-
	AU	SVM	0.883	0.880
		RF 10 FOLD	0.910	0.908
		RF Leave-One-Out	0.881	0.877
		MLP	0.759	0.747
	AU+GAZE	SVM	0.865	0.857
		RF 10 FOLD	0.686	0.691
		RF Leave-One-Out	0.720	0.711
		MLP	0.876	0.868
B	HOG	SVM	0.558	0.561
		RF 10 FOLD	0.485	0.491
		RF Leave-One-Out	0.466	0.471
		MLP	-	-
	AU	SVM	0.764	0.764
		RF 10 FOLD	0.867	0.864
		RF Leave-One-Out	0.877	0.874
		MLP	0.754	0.753
	AU+GAZE	SVM	0.916	0.914
		RF 10 FOLD	0.787	0.778
		Leave-One-Out	0.799	0.783
		MLP	0.885	0.884

Possiamo notare che il risultato migliore è stato ottenuto dal Dataset B con la **classificazione SVM** (cost=1000 e gamma=0.001) sul file arff con le **Action Unit + Gaze**. Risultati non accettabili sono stati ottenuti su tutti i test di classificazione con i descrittori HOG. Risultati molto positivi invece alcuni test con Random Forest su Action Unit e con MultiLayer Perceptron su AU + Sguardo hanno ottenuto una Accuracy leggermente più bassa ma possono ritenersi ugualmente validi e simili al migliore.

3.3 Possibili implementazioni della soluzione

3.3.1 Modifica del software FEAtuREs

Una possibile soluzione alla problematica esposta in precedenza, sfrutta FEAtuREs. Tale software utilizza le HOG Features per classificare le espressioni facciali in 11 classi di emozioni: entusiasmo, interesse, sorpresa, curiosità, concentrazione, attenzione, delusione, noia, perplessità, fastidio e frustrazione.

Innanzitutto, il lavoro da svolgere consiste nel ridurre le emozioni da 11 a 7, modificando l'interfaccia grafica del software presente nel file *UI.cpp*. Più dettagliatamente, andrebbero modificate le funzioni *mostra()* e *mostra2()*.

Il passo successivo consiste nel cambiare il tipo di features da estrarre. FEAtuREs, come accennato si basa sulle HOG Features. Questa modifica consiste nel rimuovere l'estrazione delle HOG e implementare l'estrazione di AU. Per implementare l'estrazione di AU ci si avvale delle API di OpenFace, documentate al seguente indirizzo:

<https://github.com/TadasBaltrusaitis/OpenFace/wiki/API-calls>.

Una volta scaricata la repository presente all'indirizzo

<https://github.com/TadasBaltrusaitis/OpenFace>, lanciare con PowerShell i file `download_libraries.ps1`, e `download_models.ps1`. Questo consentirà il download delle librerie e dei modelli.

Successivamente, le librerie da importare all'interno del progetto sono all'interno della cartella `\lib\local` e sono *CppInerop*, *FaceAnalyser*, *GazeAnalyser*, *LandmarkDetector* e *Utilities*. Quest'ultime per funzionare, necessitano delle librerie presenti in `\lib\3rdParty`, che sono *opencv*, *dlib*, *openBLAS* e *CameraEnumerator*.

Di seguito vengono riportati degli esempi di estrazione di AU, da implementare all'interno della funzione *action()* nelle classi *RealTimeAction* e *LoadAction* presenti in FEAtuREs, atte rispettivamente all'analisi di video in real-time e offline.

Esempio di estrazione di AU:

```
// Load landmark detector
LandmarkDetector::FaceModelParameters det_parameters(arguments);
LandmarkDetector::CLNF face_model(det_parameters.model_location);

// Load facial feature extractor and AU analyser
FaceAnalysis::FaceAnalyserParameters face_analysis_params(arguments);
face_analysis_params.OptimizeForImages();
FaceAnalysis::FaceAnalyser face_analyser(face_analysis_params);

bool success = LandmarkDetector::DetectLandmarksInImage( grayscale_image, face_model, det_parameters);
face_analyser.PredictStaticAUsAndComputeFeatures(captured_image, face_model.detected_landmarks);

auto aus_intensity = face_analyser.GetCurrentAUsReg();
auto aus_presence = face_analyser.GetCurrentAUsClass();
```

Per gli arguments da utilizzare negli oggetti di classe *FaceModelParameters* e *FaceAnalyserParameters* è possibile ottenere maggiori dettagli visualizzando l'implementazione dei costruttori:

FaceModelParameters(std::vector<std::string> &arguments) e

FaceAnalyserParameters(std::vector<std::string> &arguments) ai seguenti link:

- *FaceModelParameters*:

<https://github.com/TadasBaltrusaitis/OpenFace/blob/master/lib/local/LandmarkDetector/src/LandmarkDetectorParameters.cpp>,

- *FaceAnalyserParameters*:

<https://github.com/TadasBaltrusaitis/OpenFace/blob/master/lib/local/FaceAnalyser/src/FaceAnalyserParameters.cpp>

È possibile estrarre anche le features sullo sguardo con il seguente codice:

```
LandmarkDetector::FaceModelParameters det_parameters;
LandmarkDetector::CLNF face_model(det_parameters.model_location);

while(video)
{
    bool success = LandmarkDetector::DetectLandmarksInVideo( grayscale_image, face_model, det_parameters);

    cv::Point3f gazeDirection0(0, 0, -1);
    cv::Point3f gazeDirection1(0, 0, -1);
    cv::Vec2d gazeAngle(0, 0);

    if (success && det_parameters.track_gaze)
    {
        GazeAnalysis::EstimateGaze(face_model, gazeDirection0, fx, fy, cx, cy, true);
        GazeAnalysis::EstimateGaze(face_model, gazeDirection1, fx, fy, cx, cy, false);
        gazeAngle = GazeAnalysis::GetGazeAngle(gazeDirection0, gazeDirection1, pose_estimate);
    }
}
```

Infine, utilizzare la funzione `svm_predict_probability()`.

Tale funzione utilizza il modello e le features estratte dal frame che si intende classificare per ottenere un vettore contenente i valori che indicano le percentuali di appartenenza alle varie classi di emozione.

In conclusione, è opportuno cambiare il file `.model` con quello prodotto con questo caso di studio.

3.3.2 Altre soluzioni

In alternativa alla soluzione precedentemente proposta, è possibile sviluppare un software JAVA in grado di caricare una o più immagini contemporaneamente dal disco rigido. Al click di un JButton e Java Native Interface (JNI), è possibile eseguire del codice in C++ per l'estrazione le features attraverso OpenFace. Successivamente è possibile utilizzare il codice sorgente WEKA e il modello prodotto da questo studio per classificare l'emozione. Il codice sorgente WEKA è accessibile al link: <https://github.com/Waikato/weka-3.8>. Il risultato sarà poi visualizzato su una JLabel nell'interfaccia grafica. Il vantaggio principale di questa soluzione è che, utilizzando le librerie di WEKA è possibile implementare molto facilmente qualsiasi algoritmo di Machine Learning (es. RandomForest o MLP) fornito dal produttore. In questo modo, si potrebbero testare i vari modelli prodotti da questo caso di studio.

4. Testing

Come ultima fase, sono state ricercate sul Web altre immagini di soggetti che esprimono una delle espressioni facciali analizzate, al fine di poterle validare, utilizzare per lo sviluppo di un dataset in futuro, e per testare i modelli di apprendimento prodotti.

4.1 Raccolta di nuove immagini dal web

Sono state ricercate 35 immagini dal web di soggetti che rappresentassero una delle 7 classi di emozioni analizzate (noia, sorpresa, interesse, neutrale, frustrazione, perplessità, entusiasmo). Durante il processo di ricerca delle immagini, sono stati presi in considerazione esclusivamente soggetti non anziani, senza barba e in posizione frontale. Inoltre, non sono state prese in considerazione immagini sfocate.

4.2 Creazione di un Google Form per la validazione delle immagini


È stato successivamente creato un **Google Form** con lo scopo di validare le emozioni raffigurate. Per ciascuna immagine è stato creato un sondaggio dove è stato chiesto all'utente a quale classe di emozioni appartenesse l'immagine corrente. Il form viene visualizzato nel modo seguente:

Valutazione emozionale

Selezionare o riportare in forma di testo, l'emozione raffigurata

*Campo obbligatorio

1. La seguente espressione rappresenta un'emozione: *



☐ Perplexità

☐ Entusiasmo

☐ Neutrale

☐ Noia

☐ Frustrazione

☐ Sorpresa

☐ Interesse

☐ Altro: _____

È stato fatto compilare il form da diverse tipologie di utenti (studenti, professori e psicologi) e dove aver raccolto abbastanza feedback è stata assegnata definitivamente una classe di appartenenza alle immagini analizzate.

4.2 Risultati della validazione

Sono stati raccolti i dati nella tabella seguente ed è stata definita l'emozione prevalente per agreement di maggioranza:

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	Prevalente	Classif.
1	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	100 % Corretta
2	Noia	Noia	Frustrazione	Noia	Noia	Noia	Noia	Noia	Noia	Noia	89 % Corretta
3	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Noia	Perplessità	Interesse	Neutrale	67 % Corretta
4	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	100 % Errata
5	Interesse	Sorpresa	Sorpresa	Sorpresa	Entusiasmo	Altro	Sorpresa	Sorpresa	Entusiasmo	Sorpresa	56 % Corretta
6	Interesse	Frustrazione	Perplessità	Frustrazione	Altro	Perplessità	Frustrazione	Frustrazione	Perplessità	Frustrazione	44 % Corretta
7	Noia	Noia	Frustrazione	Frustrazione	Altro	Noia	Noia	Perplessità	Frustrazione	-	-
8	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Perplessità	Interesse	Perplessità	89 % Errata
9	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Altro	Sorpresa	Interesse	Sorpresa	Sorpresa	Sorpresa	78 % Errata
10	Perplessità	Neutrale	Neutrale	Interesse	Neutrale	Interesse	Perplessità	Perplessità	Neutrale	-	-
11	Altro	Noia	Entusiasmo	Noia	Noia	Altro	Noia	Sorpresa	Noia	Noia	55 % Corretta
12	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Interesse	Neutrale	Neutrale	Neutrale	89 % Errata
13	Neutrale	Interesse	Interesse	Interesse	Interesse	Altro	Interesse	Interesse	Perplessità	Interesse	67 % Errata
14	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	100 % Corretta
15	Perplessità	Frustrazione	Frustrazione	Frustrazione	Perplessità	Perplessità	Frustrazione	Frustrazione	Altro	Frustrazione	56 % Corretta
16	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	100 % Corretta
17	Neutrale	Frustrazione	Frustrazione	Frustrazione	Frustrazione	Frustrazione	Frustrazione	Frustrazione	Altro	Frustrazione	78 % Errata
18	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Neutrale	Entusiasmo	100 % Corretta
19	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	100 % Errata
20	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Neutrale	Neutrale	Entusiasmo	78 % Errata
21	Perplessità	Frustrazione	Noia	Frustrazione	Altro	Frustrazione	Frustrazione	Frustrazione	Perplessità	Frustrazione	56 % Corretta
22	Frustrazione	Frustrazione	Frustrazione	Frustrazione	Altro	Perplessità	Perplessità	Frustrazione	Perplessità	Frustrazione	56 % Corretta
23	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Neutrale	Entusiasmo	89 % Corretta
24	Perplessità	Frustrazione	Perplessità	Frustrazione	Altro	Perplessità	Frustrazione	Perplessità	Perplessità	Perplessità	56 % Errata
25	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	100 % Corretta
26	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	Neutrale	100 % Errata
27	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Entusiasmo	Neutrale	Entusiasmo	89 % Corretta
28	Interesse	Interesse	Perplessità	Noia	Perplessità	Altro	Interesse	Perplessità	Perplessità	-	-
29	Noia	Noia	Noia	Noia	Noia	Noia	Noia	Noia	Noia	Noia	100 % Errata
30	Interesse	Interesse	Perplessità	Interesse	Perplessità	Altro	Interesse	Perplessità	Altro	-	-
31	Sorpresa	Interesse	Frustrazione	Perplessità	Interesse	Interesse	Interesse	Perplessità	Sorpresa	-	-
32	Sorpresa	Interesse	Entusiasmo	Perplessità	Entusiasmo	Entusiasmo	Interesse	Perplessità	Sorpresa	-	-
33	Perplessità	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	Sorpresa	89 % Corretta
34	Neutrale	Neutrale	Neutrale	Noia	Neutrale	Neutrale	Neutrale	Neutrale	Altro	Neutrale	78 % Errata
35	Noia	Noia	Noia	Noia	Noia	Altro	Noia	Sorpresa	Noia	Noia	78 % Corretta

In particolare, si è deciso di prendere in considerazione solo le immagini che hanno superato una soglia di maggioranza e di scartare, quindi, le immagini che hanno ricevuto giudizi ambigui in quanto non rappresentano esplicitamente una specifica emozione. Per poter validare le immagini, è necessario effettuare un'ulteriore analisi. È stato calcolato il coefficiente Kappa (Fleiss' Kappa), un coefficiente statistico in grado di valutare l'accordo tra le classificazioni espresse da più esaminatori. Si considerino N soggetti, ciascuno dei quali viene classificato mediante M categorie esaustive e mutualmente esclusive da un gruppo di n esaminatori. Viene creata una matrice di confusione dove x_{ij} indica il numero di esaminatori che hanno assegnato l'i-esimo soggetto (i..N) alla j-esima categoria (1..M). Si è ottenuto così la seguente matrice di confusione:

a = Noia

b = Sorpresa

c = Interesse

d = Neutrale

e = Frustrazione

f = Perplexità

g = Entusiasmo

h = Altro

	a	b	c	d	e	f	g	h
1	0	9	0	0	0	0	0	0
2	8	0	0	0	1	0	0	0
3	1	0	1	6	0	1	0	0
4	0	0	0	0	0	9	0	0
5	0	5	1	0	0	0	2	1
6	0	0	1	0	4	3	0	1
7	4	0	0	0	3	1	0	1
8	0	0	1	0	0	8	0	0
9	0	7	1	0	0	0	0	1
10	0	0	2	4	0	3	0	0
11	5	1	0	0	0	0	1	2
12	0	0	1	8	0	0	0	0
13	0	0	6	1	0	1	0	1
14	0	9	0	0	0	0	0	0
15	0	0	0	0	5	3	0	1
16	0	0	0	0	0	0	9	0
17	0	0	0	1	7	0	0	1
18	0	0	0	1	0	0	8	0
19	0	0	0	9	0	0	0	0
20	0	0	0	2	0	0	7	0
21	1	0	0	0	5	2	0	1
22	0	0	0	0	5	3	0	1
23	0	0	0	1	0	0	8	0
24	0	0	0	0	3	5	0	1
25	0	9	0	0	0	0	0	0
26	0	0	0	9	0	0	0	0
27	0	0	0	1	0	0	8	0
28	1	0	3	0	0	4	0	1
29	9	0	0	0	0	0	0	0
30	0	0	4	0	0	3	0	2
31	0	2	4	0	1	2	0	0
32	0	2	2	0	0	2	3	0
33	0	8	0	0	0	1	0	0
34	1	0	0	7	0	0	0	1
35	7	1	0	0	0	0	0	1
TOT	37	53	27	50	34	51	46	17

È possibile definire la proporzione delle coppie di esaminatori che hanno assegnato l'immagine i alla emozione j come:

$$P_{ij} = \frac{\binom{x_{ij}}{2}}{\binom{n}{2}} = \frac{x_{ij}(x_{ij}-1)}{n(n-1)},$$

Utilizzando la precedente formula si può calcolare facilmente la proporzione delle coppie di assegnazioni concordanti relative al soggetto i:

$$P_i = \sum_{j=1}^M P_{ij} = \frac{1}{n-1} \left(\frac{1}{n} \sum_{j=1}^M x_{ij}^2 - 1 \right)$$

e misurarne poi la media per determinare l'accordo osservato (1):

$$\bar{P} = \frac{1}{N} \sum_{i=1}^N P_i = \frac{1}{n-1} \left(\frac{1}{Nn} \sum_{i,j} x_{ij}^2 - 1 \right)$$

Dopo aver effettuato una stima della probabilità di assegnazione casuale alla categoria j (p_j), secondo Scott e Fleiss, l'accordo atteso per effetto del caso (2) è dato dal quadrato della stima p_j :

$$p_j = \frac{x_{.j}}{Nn} = \frac{1}{Nn} \sum_{i=1}^N x_{ij} \quad \bar{P}_e = \sum_{j=1}^M p_j^2$$

Infine, sottraendo dall'accordo osservato (1) l'accordo atteso casuale (2) e normalizzando si ottiene la Kappa statistica:

$$K = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

$$\bar{P} = 0,59761905 \quad \bar{P}_e = 0,13674981 \quad K = 0,53387679$$

Questa misura ha un range che va da -1 a 1 e quando possiede valore uguale a 0 indica indipendenza statistica. Se il valore di Kappa è uguale a 1, vuol dire che c'è 'perfect agreement' tra gli osservatori. Se invece il valore è negativo, significa che non c'è assoluta concordanza tra gli osservatori. Quindi quanto più è alto il valore della statistica, più sarà valido l'agreement. Con il valore di kappa ottenuto, è possibile affermare che il livello di agreement è moderato.

4.4 Test delle immagini sui modelli creati

Sono stati creati due file ARFF con Action Unit + Gaze e uno con solo Action Unit per queste immagini. Per poter effettuare la classificazione è necessario assegnare nel file ARFF all'attributo di classe di ciascuna istanza il carattere speciale '?'. Successivamente è stato effettuato un test su WEKA utilizzando i modelli prodotti precedentemente e classificando, tramite questi, le immagini per testarne le performance. È stato notato che, nonostante i risultati molto alti ottenuti nel modello allenato con SVM sulle Action Unit

+ Gaze del Dataset B, effettuando un confronto con l'effettiva classificazione data dagli utenti, i risultati non sono per nulla soddisfacenti (29% di Accuracy). Dopo svariati tentativi, è stato notato che, nel processo di classificazione di questo test set, si incorre sempre nella stessa problematica dove alcune emozioni vengono riconosciute dai modelli con altre: molto spesso un volto neutrale viene classificato come perplesso o interessato. Si può pensare che questi 3 tipi di emozioni condividano alcune Action Unit. Tuttavia, il modello che ha presentato i risultati migliori è stato quello allenato con Random Forest sulle Action Unit utilizzando 10 fold nella cross validation. I risultati della classificazione sono quelli riportati nell'ultima colonna della tabella precedente e hanno prodotto un Accuracy del 58% (17/29).

5. Conclusioni

In conclusione, dopo l'analisi effettuata con questo caso di studio, si ipotizza che il basso livello di Accuracy nella fase di test sia dovuto alle dimensioni ridotte del dataset utilizzato.

La scarsa presenza sul web di dataset con immagini di emozioni secondarie, hanno impedito la realizzazione di un dataset corposo (non subject-dependent) dal quale produrre un modello sufficientemente addestrato.