Università Politecnica delle Marche

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

Progetto di un sistema per il riconoscimento di emozioni da video

Design of a system for the discovery of emotions from video

Relatore:
Domenico Potena

Laureanda:
Elisabetta Trirè

Anno Accademico 2019/20

Sommario

Capitolo 1
Introduzione

In una società sempre più basata sui sistemi tecnologici e sul digitale, le persone si servono quotidianamente di mezzi di comunicazione innovativi come i Social Network, i siti web e altri strumenti digitali.
Sono quasi 4.54 miliardi le persone che oggi sono connesse ad Internet e circa la metà della popolazione mondiale utilizza regolarmente i Social Network**[1]**.
Molte delle più grandi piattaforme di social media al mondo, oltre a consentire la pubblicazione di messaggi di testo, consentono di caricare foto e video.
Secondo un recente studio Cisco**[2]**, nel 2020, l'82% del traffico Internet sarebbe generato dai video digitali, perchè essi rappresentano i contenuti più apprezzati, visualizzati, condivisi e commentati sui principali Social.
Se gli utenti preferiscono fare i video, sia essi in live o in differita, le aziende possono sfruttare tali contenuti per catalogare informazioni afferenti il gradimento dei loro prodotti o delle loro performance attraverso il rilevamento delle emozioni provate dalle persone in video.
Giornalmente vengono però generate una grande mole di dati e il rilevamento risulterebbe molto difficoltoso da realizzare manualmente.
Per questo scopo occorre predisporre un algoritmo che riesca ad individuare le emozioni a partire dalle espressioni dei volti delle persone in video, senza un aiuto esterno.
Questo *processo* è presente in letteratura ed è chiamato *emotion recognition* e nel caso in esame si applica alle espressioni facciali, per le quali si vogliono distinguere sette diverse emozioni di base, come previste dal Modello delle Emozioni di Ekman**[3]**: gioia, stupore, tristezza, disgusto, disprezzo, rabbia, paura.
Vi sono due possibili approcci principali per l'analisi di video: considerando che un video è composto da più *frames,* si possono analizzare i singoli fotogrammi ognuno a sé stante oppure considerare una sequenza di frames.
A questo scopo, in letteratura si *propone di utilizzare* le reti neurali artificiali, degli algoritmi di machine learning in grado di simulare il cervello umano e quindi di apprendere attraverso una serie di campioni o *samples* in input delle caratteristiche o *features* che si vogliono individuare.
Una volta addestrate, le reti neurali sono poi in grado di predire tali caratteristiche su samples completamente nuovi.
Nel caso preso in esame in questo lavoro di tesi, le features che si vogliono individuare corrispondono alle espressioni dei volti rappresentanti le emozioni.

Si è perciò scelto il Cohn-Kanade Database, un dataset di sequenze video in cui diversi soggetti sono stati ripresi mentre rappresentano una delle emozioni di base, in modo da avere un dataset di partenza da poter sfruttare per addestrare delle reti neurali a classificare le emozioni seguendo tre approcci: per singolo frame, per sequenza video e un approccio misto.
Per classificare le emozioni su singoli samples si sono scelte come architetture le Convolutional Neural Networks (CNN) e in particolare un modello particolarmente adatta allo scopo, la VGG16.
Per fare predizione su sequenze video si è invece scelto un approccio misto, ovvero l'utilizzo della VGG16 con in cascata una Recurrent Neural Network (RNN), l'LSTM. Ciò è stato fatto in modo da analizzare e fare predizione con la LSTM su sequenze estratte dalla VGG16 a monte, sequenze di emozioni come prima sperimentazione e di features come seconda.
Nel primo capitolo verranno introdotte le tecnologie utilizzate e le metriche di valutazione dei modelli.
Nel secondo capitolo si parlerà in dettaglio delle architetture Vgg16 ed LSTM e verrà presentato il dataset Cohn-Kanade.
Nel terzo capitolo si tratteranno gli esperimenti svolti con la VGG, della quale si presenterà il modello con le migliori performance e quelli svolti con VGG a monte ed LSTM a valle, utilizzando sia sequenze di emozioni che sequenze di features.
Si confronteranno poi i migliori risultati ottenuti tramite i tre approcci.
Nel quarto capitolo si trarranno le conclusioni e si parlerà dei possibili sviluppi futuri.

Capitolo 2
Le reti neurali artificiali

Una rete neurale artificiale o Artificial Neural Network (ANN) è un algoritmo di apprendimento automatico (*Machine learning)* progettato per essere simile al cervello umano sia nella struttura che nelle funzionalità.
Una rete simula l'apprendimento umano: impara effettuando elaborazioni sui dati che le vengono forniti in input.
All'interno del sistema nervoso umano, le cellule più importanti sono i neuroni, i quali si scambiano informazioni tramite connessioni chiamate sinapsi.
Attraverso le sinapsi, i neuroni raccolgono informazioni grazie a proprie ramificazioni chiamate dendriti e propagano l'output della propria elaborazione attraverso una fibra, l'assone, il quale può raggiungere neuroni anche distanti.
Una ANN è anch'essa costituita da neuroni artificiali, i quali hanno capacità di elaborazione.
Essi ricevono in ingresso una combinazione di input e ne effettuano una trasformazione tramite una funzione, tipicamente non lineare, detta funzione di attivazione.
Come le reti neurali biologiche sono composte da strati di neuroni, anche una rete neurale artificiale è composta da tali strati, i quali vengono chiamati *layers*.
I layers sono connessi tra loro e ogni rete neurale presenta un input layer, lo strato d'ingresso dei dati forniti in input, uno o più hidden layers che effettuano le computazioni e un ultimo strato chiamato output layer, il quale restituisce i risultati dell'elaborazione.
Se la rete ha più di un hidden layer, essa viene chiamata rete neurale artificiale profonda (deep neural network).
Le deep neural netwokrs utilizzano infatti layers multipli per estrarre delle caratteristiche o features dai dati, ogni layer corrisponde ad un diverso livello di astrazione delle features, i quali insieme formano una gerarchia di concetti.
Le caratteristiche di più alto livello vengono derivate da quelle di livello più basso.

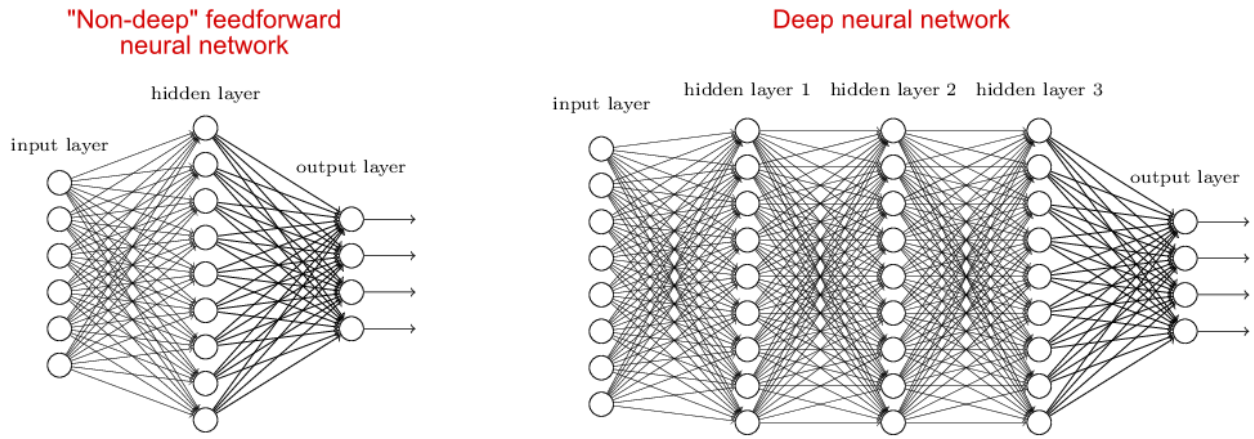

Fig. 1 - tipologie di reti neurali

In questo lavoro di tesi si sono utilizzate due particolare classi di reti neurali, le Convolutional Neural Networks e le Recurrent Neural Networks, di cui si descrivono di seguito le caratteristiche principali.

2.1 Convolutional Neural Networks

Una Convolutional Neural Networks o CNN è una rete neurale artificiale di tipo feed-forward.
Le reti feed-forward sono organizzate su più livelli (anche detti strati o layers) in cui ogni neurone di un livello riceve input solo dai neuroni del livello precedente e propaga gli output solamente verso i neuroni dei livelli successivi: il flusso di informazioni si propaga perciò in una unica direzione, dall'input all'output, senza possibilità di formare cicli.
Le reti feed-forward non hanno memoria di input avvenuti a tempi precedenti, per cui l'output è determinato solamente dall'attuale input.
Una CNN è composta da una gerarchia di livelli, il livello di input è direttamente collegato ai pixel dell'immagine in input, seguono alcuni hidden layers convoluzionali seguiti da layers di pooling ed infine gli ultimi livelli sono in genere fully-connected, utilizzati per classificare le immagini.
Le CNN sono infatti utilizzate per effettuare task classificazione su dataset anche molto ampi.

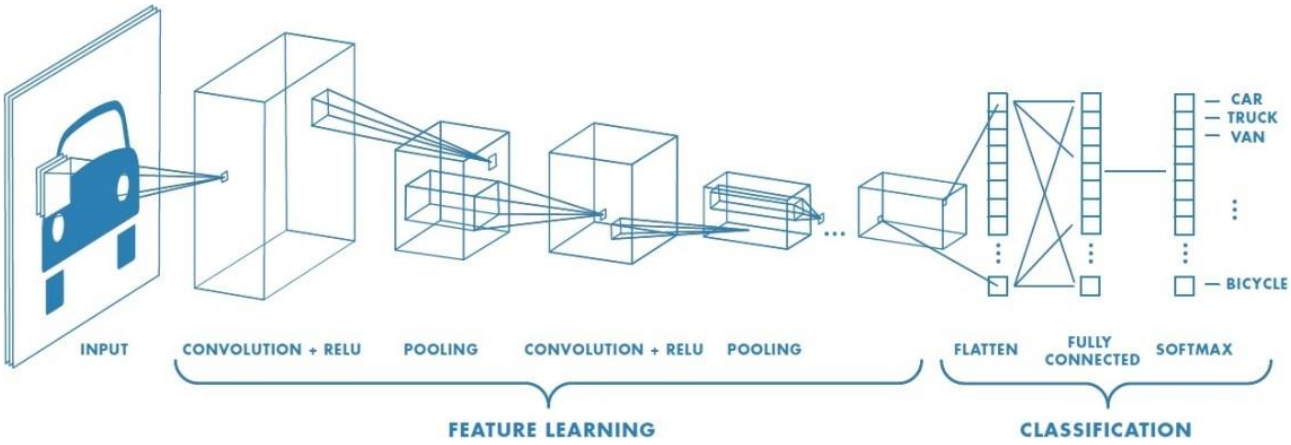

Fig. 2 – Schema di una Convolutional Neural Network

Di seguito verranno descritti i dettagli dei layers caratterizzanti una CNN e delle funzioni di attivazione

2.1.1 Il layer convoluzionale

L'obiettivo dell'operazione di convoluzione è di estrarre le caratteristiche o features dall'immagine di input e ciò è effettuato grazie all'uso di una matrice detta kernel di convoluzione.
Il kernel ha una dimensione inferiore rispetto a quella dell'immagine e viene traslato su di essa ed applicato solo su una parte dei valori dell'immagine (una regione o *patch* della stessa dimensione del kernel) e ciò equivale ad applicare un filtro sull'immagine stessa, in quanto viene eseguito il prodotto scalare tra i valori del kernel e quelli del patch.
Il risultato di tale operazione è una nuova immagine dove ogni elemento è la combinazione pesata dei valori di ogni patch dell'immagine originale, dove i valori dei pesi sono dati dal kernel di convoluzione.

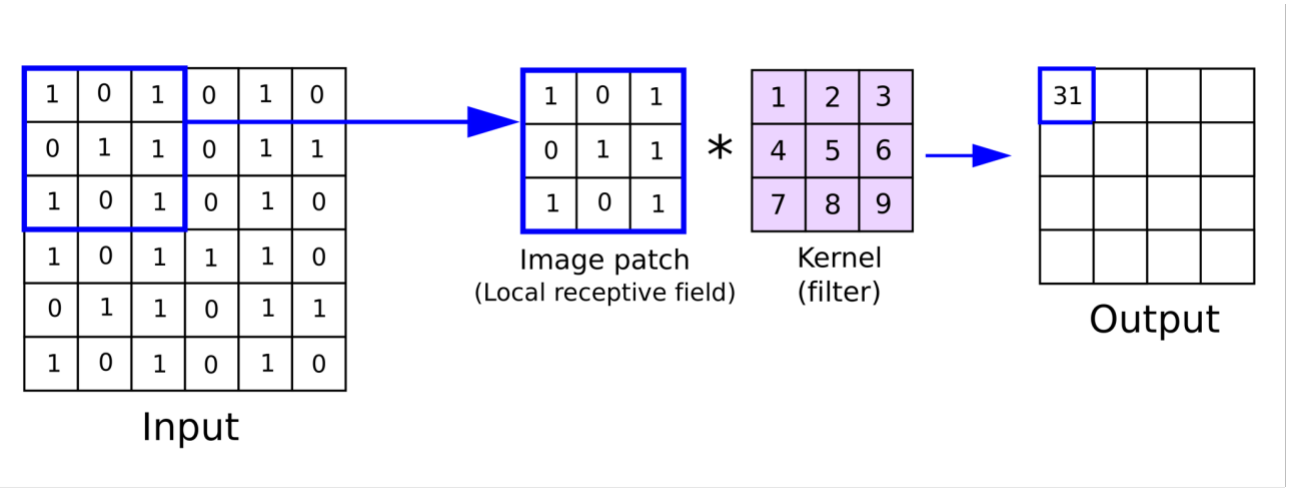

Fig. 3 – Operazione di convoluzione

La nuova immagine viene detta *feature map* o mappa delle caratteristiche, di dimensioni ridotte rispetto all'immagine originale, la quale fornisce le informazioni sulle features contenute nell'immagine originale, come ad esempio bordi o angoli presenti.
Convenzionalmente, il primo layer di convoluzione è responsabile della cattura delle caratteristiche *di basso livello* come ad esempio i bordi o il colore presenti ma con l'aggiunta di layers, l'architettura si adatta anche alle caratteristiche *di alto livello*, ad esempio riconoscendo occhi o naso di una persona.

2.1.2 Il layer di pooling

Solitamente dopo un layer di convoluzione è presente un layer di pooling (o di sub-campionamento).
Lo strato di pooling si occupa di ridurre la dimensione della feature map ottenuta dal layer di convoluzione mantenendo solo le features più importanti, in modo da snellire l'elaborazione per i layers successivi.
Ampiamente utilizzato è il layer MaxPooling, il quale applica sulla feature map una matrice che ha una funzione di filtro di dimensioni inferiori a quelle della mappa.
La matrice viene traslata su ogni parte della feature map e restituisce soltanto il valore massimo campionato.

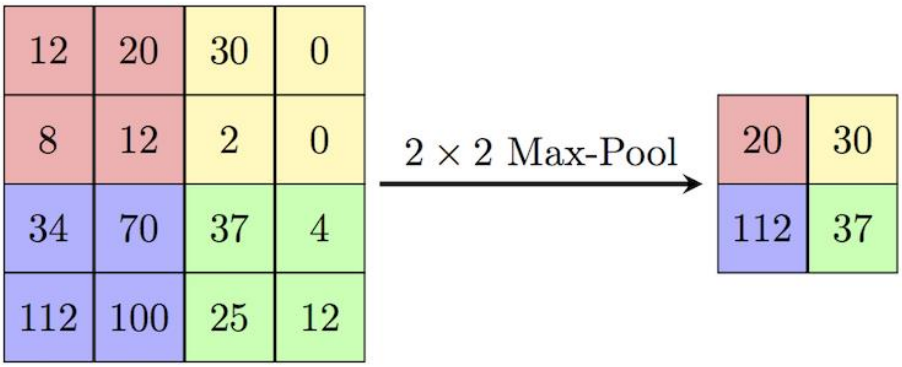

Fig. 4 – Layer MaxPooling

2.1.3 Il layer fully-connected

Il layer *fully-connected* o completamente connesso è un layer in cui ogni neurone in uno strato è connesso ad ogni neurone di un altro strato.
I layers fully-connected sono solitamente utilizzati per effettuare classificazione.

2.1.4 Le funzioni di attivazione

Le funzione di attivazione viene solitamente applicata in seguito all'operazione di convoluzione e determina il comportamento in output di un neurone in funzione del suo livello di eccitazione.
In letteratura le più note funzioni di attivazione sono quelle sigmoidali ma recentemente viene anche utilizzata la classe di funzioni rampa che, rispetto alle funzioni sigmoidali, sono funzioni non limitate.
I neuroni con la funzione attivazione rampa sono definiti Rectified Linear Units (ReLUs).
Inoltre, quando una rete neurale è utilizzata come classificatore multi-classe, nell'ultimo layer spesso si utilizza la funzione di attivazione Softmax che converte i valori in uscita dal layer in valori compresi tra 0 e 1 e la cui somma risulta pari a 1, così da poter essere interpretati come probabilità delle classi.

2.2 Recurrent Neural networks

Le reti neurali ricorrenti o RNN sono una classe di reti neurali artificiali le quali, a differenza delle CNN, prevedono collegamenti anche verso neuroni dello stesso layer o verso neuroni dello strato precedente, in modo che si possano formare all'interno della rete dei cicli o loop.
Queste interconnessioni tra livelli permettono l'utilizzo degli strati come memoria di stato in modo che, se l'input è una sequenza temporale di valori, la rete può modellarne il comportamento dinamico temporale basandosi sulle informazioni che ha ottenuto negli istanti di tempo precedenti.
Le reti neurali ricorrenti sono utilizzate in tutti quegli ambiti in cui è importante non solo la presenza di specifici elementi ma anche la loro posizione reciproca, ad esempio per la classificazione di video.

2.3 Metriche per la valutazione dei modelli

2.3.1 Matrice di confusione

La matrice di confusione o *confusion matrix* fornisce una rappresentazione dell'accuratezza della classificazione effettuata da un classificatore.
Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. L'elemento sulla riga i e sulla colonna j è il numero di casi in cui il classificatore ha classificato la classe reale i come classe j.
Tramite la matrice si può capire quanto un classificatore può confondersi nella classificazione di classi diverse.

	Classi predette			
Classi reali		A	B	C
	A	P_{AA}	P_{AB}	P_{AC}
	B	P_{BA}	P_{BB}	P_{BC}
	C	P_{CA}	P_{CB}	P_{CC}

Fig. 5 – Matrice di confusione

Ad esempio, il valore P_{AB} si riferisce a quante volte un elemento la cui classe reale era A è stato predetto come classe B.

Gli elementi sulla diagonale principale rappresentano i valori per cui il classificatore ha predetto la classe corretta (tali valori possono essere definiti come Veri Positivi, VP) : ad esempio, P_{AA} indica il numero di elementi appartenenti alla classe reale A che il modello ha correttamente predetto come A.

Dalla matrice di confusione possiamo ricavare le seguenti metriche per la valutazione dei classificatori.

2.3.2 Accuracy

L'*accuracy* o accuratezza di un modello corrisponde al rapporto tra il numero di stime corrette predette su un set di dati e il numero totale dei campioni.

Dalla matrice di confusione, l'accuracy si calcola come la sommatoria degli elementi sulla diagonale principale diviso il numero totale degli elementi.

Nel caso dell'esempio a tre classi:
Accuracy = $(P_{AA}+P_{BB}+P_{CC}) / (P_{AA}+P_{BA}+P_{CA}+P_{AB}+P_{BB}+P_{CB}+P_{AC}+P_{BC}+P_{CC})$

2.3.3 Precision

Dalla matrice di confusione, la Precision per una classe i si calcola come il rapporto tra i Veri Positivi per quella classe e il numero totale di campioni predetti come classe i, ovvero gli elementi presenti sulla colonna relativa alla classe i.

	Classi predette			
Classi reali		A	B	C
	A	P_{AA}	P_{BA}	P_{CA}
	B	P_{AB}	P_{BB}	P_{CB}
	C	P_{AC}	P_{BC}	P_{CC}

Fig. 6 – Precision

Nel caso dell'esempio a tre classi:
Precision per la classe A (in verde): Precision(A) = $P_{AA} / (P_{AA}+P_{AB}+P_{AC})$
Precision per la classe B (in azzurro): Precision(B) = $P_{BB} / (P_{BB}+P_{BA}+P_{BC})$
Precision per la classe C (in arancio): Precision(C) = $P_{CC} / (P_{CC}+P_{CA}+P_{CB})$

2.3.4 Recall

Dalla matrice di confusione, la Recall per una classe i si calcola come il rapporto tra i Veri Positivi per quella classe e il numero totale di campioni classificati erroneamente come appartenenti ad i, ovvero i valori presenti sulla riga relativa ad i.

	Classi predette			
Classi reali		A	B	C
	A	P_{AA}	P_{BA}	P_{CA}
	B	P_{AB}	P_{BB}	P_{CB}
	C	P_{AC}	P_{BC}	P_{CC}

Fig. 7 – Recall

Nel caso dell'esempio a tre classi:
Recall per la classe A (in verde): Recall(A) = P_{AA} / (P_{AA}+P_{BA}+P_{CA})
Recall per la classe B (in azzurro): Recall(B) = P_{BB} / (P_{BB}+P_{AB}+P_{CB})
Recall per la classe C (in arancio): Recall(C) = P_{CC} / (P_{CC}+P_{AC}+P_{BC})

2.3.5 F-score

L'f-score o f1-score per una classe è la media armonica di precision e recall e misura quanto un classificatore sia capace di individuare tutti e soli gli elementi di una classe. In formula, per ogni classe:

$$F - score = \frac{2\ x\ Recall\ x\ Precision}{Recall + Precision}$$

Fig. 8 – F-score

Nel caso dell'esempio a tre classi:

F-Score per la classe A = $\frac{2 * Recall(A) * Precision(A)}{Recall(A) + Precision(A)}$

Similmente per le classi B e C.

Capitolo 3
Metodologia e tecnologie utilizzate

Per la classificazione delle emozioni in video, in questo lavoro di tesi si sono utilizzate una CNN e una RNN note in letteratura per essere adatte a classificare, rispettivamente, singoli samples e sequenze video: la VGG16 e la LSTM.
In questo capitolo verranno introdotte tali architetture e i modelli utilizzati per gli esperimenti e presentato il dataset Cohn-Kanade.

3.1 Il dataset Cohn-Kanade

Per poter effettuare gli esperimenti occorreva un dataset che contenesse immagini con volti di persone riprese frontalmente, per poterne analizzare le espressioni del viso.
A questo scopo si è scelto il dataset open-source "The Extended Cohn-Kanade (CK+) database"**[4]**, contenente 593 sequenze video di 123 soggetti.
Ogni sequenza varia in durata, dai 10 fino a 60 frames, ed ogni frame è costituito da un soggetto ripreso mentre mostra l'espressione di una emozione.
In ogni sequenza il soggetto parte con un'espressione neutra ed arriva alla formazione dell'espressione facciale dell'emozione considerata, appartenente ad una delle 7 categorie di base individuate da Ekman.
Le sequenze video sono sia in RGB che in scala di grigi, in alcune sequenze nei fotogrammi oltre al soggetto sono presenti anche data e ora della ripresa.

Il dataset CK+ è suddiviso in due cartelle principali:

1. La cartella "cohn-kanade-images" contenente 593 sottocartelle, di cui ognuna è una sequenza video rappresentante un soggetto ed etichettata con il codice rappresentativo del soggetto stesso (lettera "S" seguita da un numero rappresentante l'i-esimo soggetto).
   Per ogni soggetto sono presenti una o più sequenze di fotogrammi in formato *.png*, ogni sequenza rappresenta una emozione mostrata dal soggetto (nell'esempio mostrato in fig. 9, il soggetto S005 ha una sequenza 001 in cui mostra l'emozione "Surprise").
   cohn-kanade-images/S005/001/S005_001_00000011.png

Fig. 9 – Percorso di un frame nel dataset Cohn-Kanade

2. La cartella “Emotion”, suddivisa anch’essa in sottocartelle nominate per soggetto, contenente le etichette o *labels* rappresentative dell’espressione facciale mimata dal soggetto stesso.
   Tuttavia, solo 327 delle 593 sequenze hanno la label rappresentativa, questo in quanto solo esse corrispondono esattamente alla definizione del prototipo della emozione che descrivono.

Fig. 10 – Alcuni samples appartenenti al dataset Cohn-Kanade

La distribuzione dei samples nelle classi non è equa, il dataset è sbilanciato verso la classe “Surprise” mentre della classe “Contempt” ha pochissime sequenze video.
In fig.11 si può vedere la distribuzione delle sequenze nelle varie classi.

Emozione	**Numero di sequenze presenti**
“Anger”	45
“Contempt”	18
“Disgust”	59
“Fear”	25
“Happiness”	69
“Sadness”	28
“Surprise”	83

Fig. 10 – Distribuzione delle sequenze video nelle classi del dataset Cohn-Kanade

3.1 VGG16 e modello utilizzato

Per effettuare classificazione su singoli frames si è scelto di utilizzare una Convolutional Neural Network che in letteratura è consigliata per svolgere tale task**[5]**: la VGG16.
Dovendo effettuare classificazione su di un dataset di video abbastanza piccolo (il Cohn-Kanade Database esteso o CK+ comprende 593 sequenze video per un totale di circa 2000 frames) si è scelto di fare *transfer learning* ed utilizzare un modello di rete VGG16 i cui pesi sono preallenati su ImageNet**[6]**.
ImageNet è un dataset noto in letteratura, comprendente 14 milioni di immagini di oggetti appartenenti a circa 21 mila classi. Data la grande dimensione di tale database, i pesi della rete che si vuole utilizzare per classificare i video risultano perciò abbastanza allenati a classificare correttamente immagini ed adatti allo scopo che si vuole raggiungere in questa tesi.
Il modello di base dell'architettura VGG16 è costituito dai layers rappresentati in fig.12.

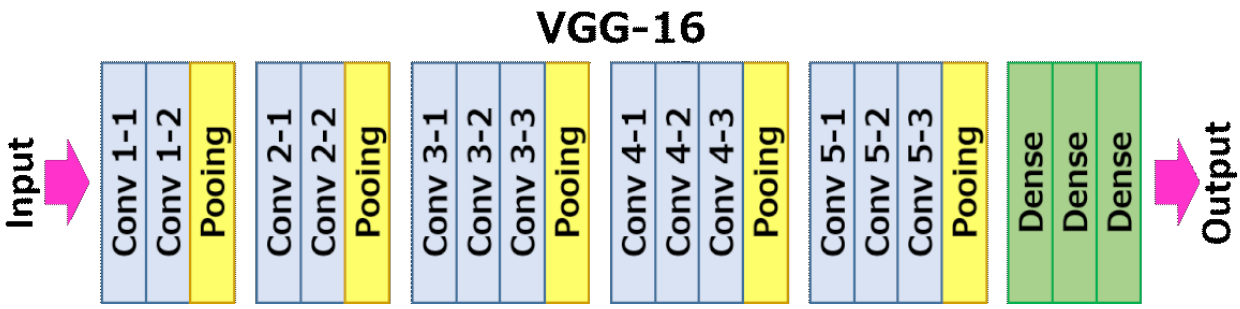

Fig. 12 – Architettura della Convolutional Neural Network VGG16

Il layer di input ha dimensione fissa, la stessa dimensione delle immagini da processare.
Ogni frame passa poi per cinque stack di layers convoluzionali con funzione di attivazione ReLU seguiti ognuno da un layer di pooling, in particolare di Max Pooling.
Vi sono infine tre layers fully-connected per la classificazione (layers *Dense*), i primi due con funzione di attivazione ReLU e l'ultimo con funzione di attivazione Softmax.
In base alle esigenze progettuali, si può scegliere se includere o meno gli ultimi tre layers Dense previsti dall'architettura (detti anche *top layers)* oppure se sostituirli con altri.
Nel caso preso in esame in questo lavoro di tesi, avendo scelto di effettuare transfer-learning, si è scelto di sostituire i top layers della VGG16 pre-trained con un layer Flatten e due layers Dense, il primo con funzione di attivazione ReLU e il secondo con

funzione di attivazione Softmax, per la classificazione di 7 classi (corrispondenti alle 7 emozioni di base da riconoscere).

Il modello utilizzato per effettuare gli esperimenti è rappresentato in figura 13.

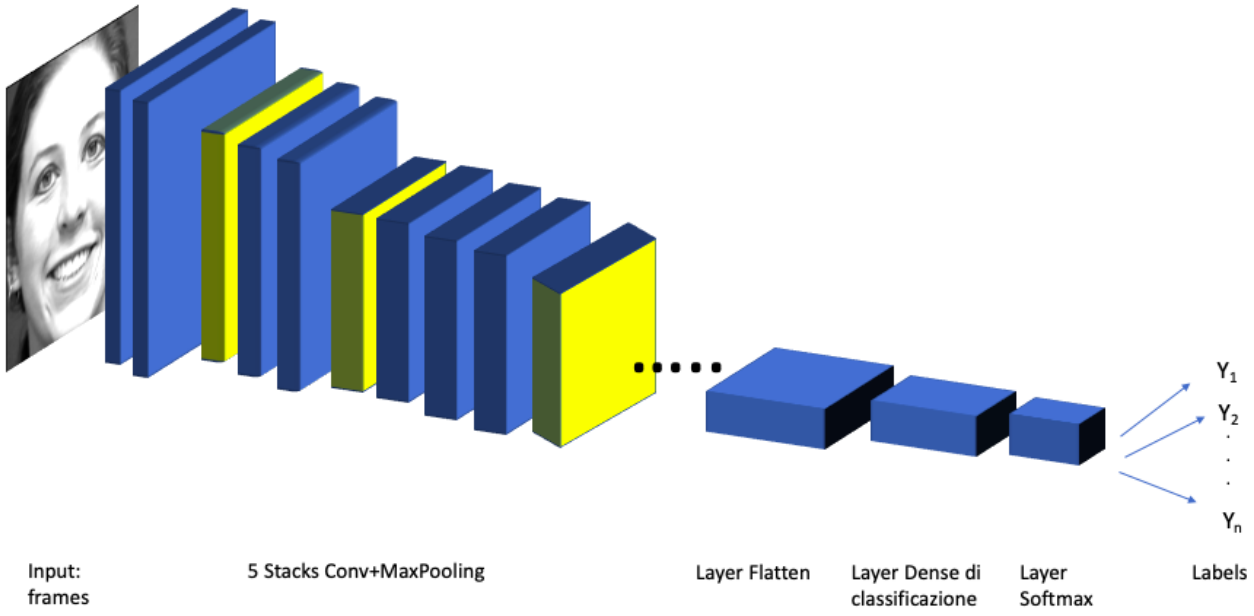

Fig. 12 – Modello di VGG16 utilizzato negli esperimenti

3.1.1 Ottimizzazione dei parametri della VGG16

Prima di effettuare classificazione su sequenze video, si è valutato quale modello di VGG utilizzare per riuscire ad ottenere una buona classificazione delle emozioni su singolo frame.
Si sono svolte molte prove per determinare quale fosse la più performante configurazione dei parametri dei layers in modo da avere una buona accuracy e una buona f-score sia sulla parte di dataset Cohn-Kanade utilizzata come train set che sulla parte usata come test set.
Si è pensato di effettuare esperimenti variando i seguenti parametri:

- dimensione delle immagini in input;
- layers completamente addestrabili o solo il penultimo layer fully-connected addestrabile (penultimo layer Dense che effettua la classificazione)
- dimensione del penultimo layer dense;

Gli esperimenti si sono concentrati soprattutto sull'ultimo punto, in quanto:

- Le immagini del CK+, dopo la fase di pre-processing, hanno una dimensione di 256x256. Si è pensato inizialmente di utilizzare tali immagini per l'addestramento dei modelli senza modificarne la dimensione.
  Si è svolto quindi un esperimento confrontando le prestazioni della VGG presa a modello variando solamente la dimensione delle immagini in input a 50x50, 150x150, 256x256.
  Dai risultati ottenuti, ridimensionare le immagini a 150x150 ha portato a prestazioni migliori ottenute in tempi abbastanza brevi.
  Si è deciso perciò di mantenere la dimensione 150x150 come dimensione fissa delle immagini nelle prove svolte in seguito.

- Si sono confrontate le prestazioni dello stesso modello (dimensione immagini in input:150x150, dimensione del penultimo layer fully-connected: 1024) lasciando modificabili, durante l'addestramento, tutti i pesi dalla rete oppure solo i pesi dei top layers.
  Dalle prove svolte, si è avuta conferma che la scarsa dimensione del dataset di train non permettesse di ottenere una buona configurazione dei pesi la rete, perciò si è deciso di lasciare addestrabili solo i pesi relativi al penultimo layer dense e svolgere *fine-tuning* dei modelli utilizzati nelle prove svolte in seguito.

- Gli esperimenti effettuati per ottenere un modello di VGG16 che classificasse al meglio si sono poi concentrati sulla variazione del numero di neuroni presenti nel penultimo layer Dense.

3.2 LSTM e architettura per la classificazione di sequenze di frames

Per la classificazione dei video come sequenze di frames si è scelto di utilizzare, a valle del modello di VGG scelto durante gli esperimenti per singolo frame, una architettura nota in letteratura per essere adatta a tale scopo: la LSTM.
La LSTM (Long Short-Term Memory) è una Recurrent Neural Network le cui celle di memoria sono più evolute rispetto alle celle di memoria base di una RNN, in quanto dotate di un *effetto memoria a lungo termine*.
Una cella è una parte della rete neurale ricorrente che preserva lo stato (o memoria) interno per ogni istante temporale.
Lo stato h al tempo t è funzione dell'input x all'istante t e dallo stato precedente h al tempo t-1 e l'output y della cella corrisponde allo stato h al tempo t.

Ogni cella è costituita da un numero prefissato di neuroni e può essere vista come un layer.
Nelle celle base vi è difficoltà a mantenere il ricordo degli input dei primi istanti temporali, ovvero la memoria di essi tende a svanire e per questo si utilizzano architetture come la LSTM, dotate di celle migliorate.
Una cella LSTM (unità utilizzata anche in questo lavoro di tesi) è composta come in fig. 13.

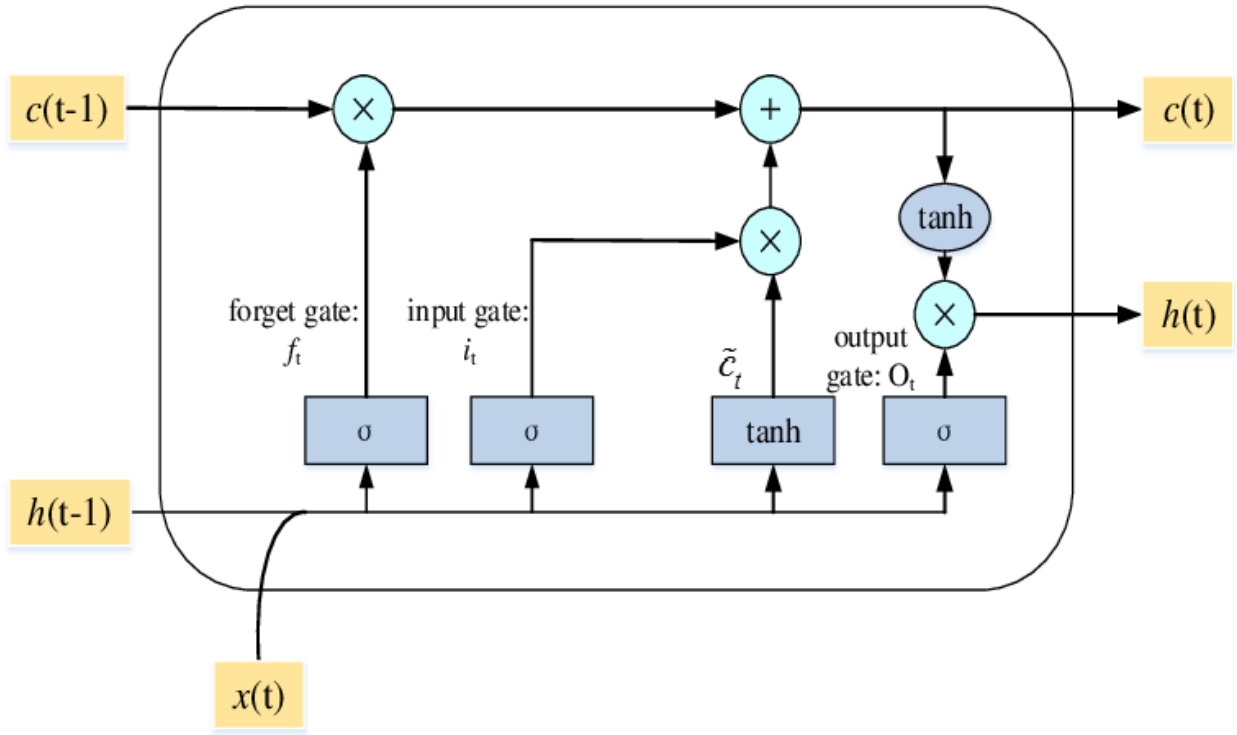

Fig. 13 – Rappresentazione di una cella LSTM

Lo stato o al tempo t, o(t), è composto da due parti: uno stato a breve termine h(t), corrispondente all'output della cella e uno stato o memoria a lungo termine c(t).
Durante la fase di training della RNN la cella apprende cosa può essere dimenticato (Forget gate, ft in fig.13) dello stato passato, c(t-1), e cosa invece deve essere mantenuto in memoria e aggiunto (Input gate, it) dall'input corrente x(t).
Le funzioni di attivazione sono sigmoidali tranne una funzione tanh per rendere i valori dello stato compresi tra -1 e 1.
Per il calcolo dell'output o(t) viene combinato l'input corrente (Output gate, Ot) con le informazioni estratte dalla memoria a lungo termine.

L'architettura scelta per classificare le sequenze di frames in questo lavoro di tesi è composta a monte dal modello di VGG16 scelto nella fase di esperimenti per singolo

sample e a valle dalla LSTM, con in cascata un layer flatten e un layer Dense con funzione di attivazione Softmax. In fig.14 una rappresentazione dell'architettura.

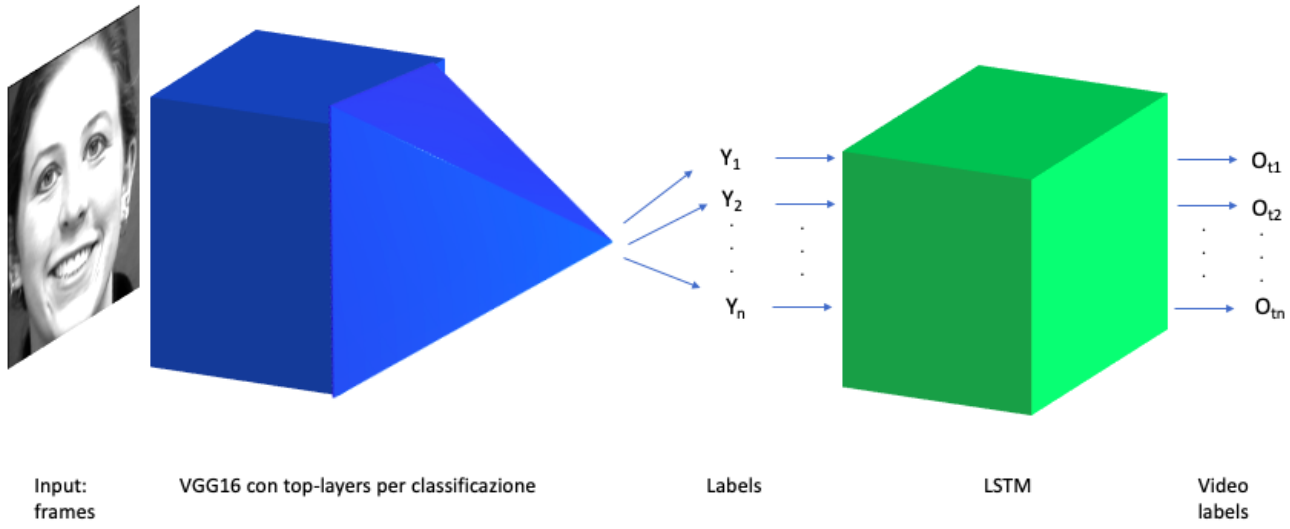

Fig. 14 – Architettura adottata per la classificazione di sequenze di frames

Il modello di VGG scelto è utilizzato per fare predizione sui frames contenuti in ogni sequenza video per ottenere sequenze di classi, predette per ogni video, da utilizzare come input per la LSTM.
Questo è stato fatto in quanto, utilizzando la singola VGG, per ottenere l'emozione di un video si sarebbe dovuto utilizzare il *majority voting.*
Tale tecnica però non tiene in considerazione che possano esistere dei *pattern* nel manifestarsi delle emozioni.
Ad esempio, se la persona in video riceve una piacevole sorpresa, la sua espressione del volto potrebbe passare da neutra a felice (posizione intermedia in cui si aprono parzialmente la bocca e gli occhi per arrivare ad esprimere la sorpresa) a sorpresa.
Se i frame contenuti nel video fossero maggiormente incentrati sulla posizione intermedia e magari solo gli ultimi sull'emozione sorpresa, il majority voting potrebbe classificare tale sequenza come "Happiness".
La LSTM potrebbe invece riconoscere quel tipo di pattern (nell'esempio, primi frames classificati come "Happiness" e solo i pochi ultimi come "Surprise") e classificare correttamente la sequenza video come "Surprise".

3.2.1 Ottimizzazione dei parametri

Anche per questa architettura si sono svolte prove per individuare la migliore configurazione dei parametri che permettesse di classificare con una buona accuracy ed f-score.
In particolare, ci si è concentrati su:

- La variazione del numero di neuroni presenti nella LSTM (50, 100, 200)
- L'aggiunta o meno di un layer Dropout subito dopo il layer LSTM, per evitare il veloce overfitting emerso dopo alcune prove.

3.3 Architettura per la classificazione di sequenze di features

Negli ultimi esperimenti si è svolta la classificazione su sequenze di features estratte dal modello di VGG utilizzato al precedente step, invece che su sequenze di classi predette dallo stesso.
Si è pensato che la LSTM potesse individuare pattern anche basandosi sulle features.
Si è scelto perciò di utilizzare lo stesso modello di VGG ma senza gli ultimi due layers fully-connected, per estrarre features di alto livello dai frames delle sequenze video ed utilizzarle come sequenze in input alla LSTM a valle.
L'architettura utilizzata negli esperimenti è rappresentata in fig.15.

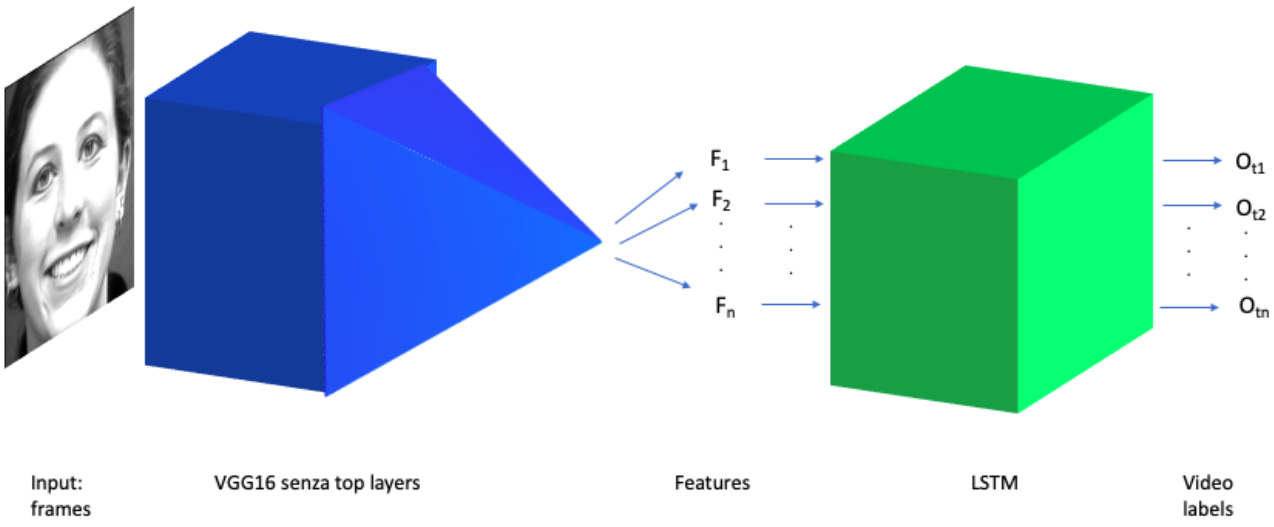

Fig. 15 – Architettura adottata per la classificazione di sequenze di features

3.3.1 Ottimizzazione dei parametri

SI sono svolte prove per individuare la migliore configurazione dei parametri che permettesse di classificare con una buona accuracy ed f-score.
In particolare, ci si è concentrati su:

- La variazione del numero di neuroni presenti nella LSTM (50,100, 200)
- L'aggiunta o meno di un layer Dropout subito dopo il layer LSTM.

Capitolo 4
Esperimenti svolti

In questo capitolo vengono presentati gli esperimenti svolti in ordine cronologico. Nel primo paragrafo viene inoltre mostrato il lavoro di pre-processing svolto sul dataset per adeguarlo alle esigenze progettuali.
In tutti gli esperimenti il modello viene addestrato per un numero di iterazioni o *epoche (epochs)* deciso dal progettista.
Inizialmente si setta un numero alto di epochs, per valutare dopo quante iterazioni è conveniente fermare l'addestramento.
Ad ogni epoca viene valutato il modello tramite una *loss function*: un metodo per valutare quanto bene un algoritmo specifico modelli i dati in input.
Se le previsioni si discostano troppo dai risultati reali, la funzione di loss durante l'addestramento aumenta invece che diminuire.
Quando la funzione di loss sul train set continua a diminuire ma sul test set inizia ad aumentare, è segno che il modello si sta adattando eccessivamente all'input e perde la capacità di generalizzare su dati completamente nuovi. Questo fenomeno è chiamato *overfitting* (fig.16).

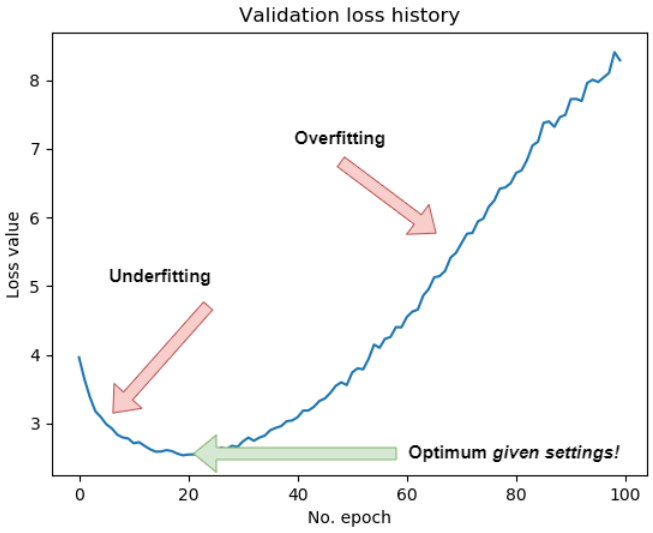

Fig. 16 – Loss function e overfitting

Negli esperimenti mostrati, per ottenere dei valori corretti riguardo alle migliori prestazioni dei modelli, si calcolano matrice di confusione e metriche all'epoca in cui il valore di loss è il più basso possibile, prima che il modello overfitti.

Tutti i passaggi mostrati sono stati svolti utilizzando la piattaforma Google Colab**[7]**, la quale mette a dispone di un ambiente di runtime virtuale.

4.1 Pre-processing del dataset Cohn-Kanade

Prima di poter essere utilizzato dalle reti neurali per effettuare classificazione, il dataset è stato sottoposto ad una fase di pre-processing.
Questa fase è stata fatta per migliorarne l'analisi da parte delle reti neurali e per adeguarlo alla struttura in termini di suddivisione in directories consigliata dalla documentazione del framework Keras**[8]**.
Si illustrano i passaggi svolti per fasi.

4.1.1 Eliminazione delle cartelle non etichettate

Come precedentemente accennato, solo 327 delle 593 sequenze disponibili nel dataset hanno l'etichetta associata alla emozione o classe di appartenenza.
Dovendo effettuare classificazione, tali sequenze sono risultate poco utili allo scopo in quanto mancanti della etichetta con la classe reale associata.
Si è perciò deciso di eliminare tali sequenze tramite codice: tutte le sequenze video senza etichetta sono state rimosse dal dataset.

4.1.2 Face detection e cropping dei volti con Haar Cascades

Si vogliono ritagliare tutti i frame di tutte le video sequenze al solo volto del soggetto, eliminando parti di immagine poco interessanti come collo e spalle o lo sfondo, in modo da alleggerire il lavoro di classificazione.
Per lo stesso motivo si vogliono riportare tutti i frames così ritagliati in scala di grigi.

È stato scritto un codice apposito, il quale:

1. Prende in input ognuno dei frame di cui è composto ogni video del dataset;
2. Tramite gli Haar Cascades**[9]**, effettua il riconoscimento del volto nell'immagine e ne disegna attorno una *finestra* per ritagliare tutti i pixel che non sono compresi in essa;

3. Ritaglia l'area contenente il volto e il risultato viene salvato in scala di grigi nella stessa posizione in termini di cartelle e con lo stesso nome del frame originale, di fatto sostituendolo.
   Tutte le immagini risultanti hanno dimensione fissa di 256x256.

4.1.3 Eliminazione di frames meno significativi

Nel dataset CK+ non sono presenti samples rappresentanti la classe "neutra", tutte le sequenze video appartengono ad una delle 7 emozioni di Ekman.
In ogni sequenza, tuttavia, il soggetto inizialmente ha un'espressione neutra per poi arrivare negli ultimi frame a rappresentare l'emozione.
Le prime immagini di ogni sequenza non dovrebbero quindi essere prese in considerazione da un classificatore, in quanto non sono caratterizzanti per l'emozione associata.
Si è pensato perciò di effettuare un *cut* dei primi frame di ogni sequenza video.
È stato scritto un codice apposito, il quale per ogni sequenza presente nel CK+, ne calcola la lunghezza, N, in termini di numero di frames ed elimina dalla sequenza i primi N/2 samples.
Tramite questo processo si ottengono sequenze video con meno frames ma meglio rappresentanti l'emozione considerata.

4.1.4 Suddivisione delle sequenze in directories

Dovendo lavorare con CNN, l'input è costituito dai samples e dalle *labels* rappresentanti la classe associata ad ognuno dei frames di ogni sequenza video del dataset.
Nel CK+ sequenze video e labels sono in due cartelle separate e ogni label è all'interno di una directory il cui nome corrisponde alla sequenza di appartenenza.
Per importare il dataset tuttavia, si utilizza un *metodo* di Keras il quale richiede che le directories abbiano la struttura in figura, con tutti i samples suddivisi per classi (fig.17).

main_directory/
...class_a/
.....a_image_1.jpg
.....a_image_2.jpg
...class_b/
.....b_image_1.jpg
.....b_image_2.jpg

Fig. 17 – Loss function e overfitting

Si è perciò riorganizzata tramite codice la struttura del CK+, in modo da suddividere i video nelle classi di appartenenza (fig. 18).

CK_images/
..1_anger/
...S075/
.....S075_008_00000007.png
.....S075_008_00000008.png
...
... S087/
.....S087_007_00000009.png
.....S087_007_00000010.png
...
..2_contempt/
... S138/
.....S138_008_00000005.png
...
..3_disgust/...
..4_fear/...
..5_happy/...
..6_sadness/...
..7_surprise/...

Fig. 18 – Loss function e overfitting

In questo modo, all'importazione del dataset, ad ognuno dei samples viene corrisposta automaticamente la label associata alla classe della directory corrispondente.

4.1.5 Suddivisione in train e test set

Per poter fare classificazione, una rete neurale necessita dei dataset di *train*, su cui addestrare la rete e di *test*, per valutare l'apprendimento.
È stato perciò necessario suddividere il CK+ in train e test set.
Si è scelto di farlo seguendo la tecnica dell'*hold-out*: 80% dei samples nel train e il restante 20% nel test.
Durante questo passaggio si è tenuto conto del fatto che per alcuni soggetti, lo stesso soggetto è presente in più sequenze video rappresentanti emozioni diverse: ad esempio, il soggetto 10 è presente in una sequenza per l'emozione "Anger" e in un'altra per l'emozione "Surprise".
Per una classificazione più accurata, in letteratura si consiglia di non inserire lo stesso soggetto, anche se in due sequenze video di emozioni diverse, nello stesso dataset: idealmente, i samples relativi ad un soggetto andrebbero tutti nel train o nel test set.
Si è tuttavia scelto di rilassare questo vincolo e di separare le sequenze del dataset in modo tale da evitare che la stessa sequenza dello stesso soggetto riguardante una emozione non compaia contemporaneamente nel train e nel test.
Lo stesso soggetto può comparire in entrambi i dataset, in video riguardanti emozioni diverse.

4.2 Classificazione per singolo frame

In questo paragrafo si riportanto gli esperimenti svolti per individuare la migliore configurazione dei parametri della VGG-16 per la classificazione dei singoli frames.
Ad ogni test effettuato si è valutata l'accuracy del modello per train e test set ad ogni epoca.
Finito l'addestramento, si è calcolata la matrice di confusione e le metriche ad essa associate tramite il metodo *classification_report* di Keras.
In Keras, con il termine *accuracy* si intende l'accuracy calcolata sul train, con *val_accuracy* quella calcolata sul test.

4.2.1 Dimensione delle immagini in input

L'architettura VGG16 ammette in input immagini con una dimensione (*shape*) di default pari a 224x224, tuttavia la dimensione può essere cambiata in base a quella delle immagini del proprio dataset.
Questo parametro è stato il primo oggetto di esperimenti svolti.
Come prima prova si è scelto un modello di VGG con dimensione del layer di classificazione pari a 4096 (per comodità definito "Modello 4096") e si sono svolti tre addestramenti in parallelo, il primo utilizzando le immagini del CK+ senzaridimensionarle, il secondo effettuando un *reshape* delle immagini a 150x150, il terzo con immagini a 50x50.
Si è lasciato solo il penultimo layer Dense addestrabile.
I risultati ottenuti in termini di accuracy e f1-score dopo l'addestramento sono riportati in fig.19 e fig.20.

Modello 4096: Accuracy		
	Train	**Test**
1. 256x256	82%	61%
2. 150x150	88%	66%
3. 50x50	70%	48%

Fig. 19 – Accuracy ottenuta nei tre esperimenti con layer Dense 4096

	Modello 4096: F1-score					
	256x256		150x150		50x50	
	Train	Test	Train	Test	Train	Test
Anger	0.90	0.54	0.92	0.54	0.76	0.45
Contempt	0.00	0.00	0.71	0.45	0.46	0.33
Disgust	0.67	0.50	0.86	0.75	0.67	0.42
Fear	0.81	0.10	0.56	0.00	0.51	0.08
Happiness	0.89	0.71	0.90	0.70	0.80	0.65
Sadness	0.67	0.43	0.90	0.40	0.08	0.00
Surprise	0.92	0.87	0.94	0.85	0.81	0.62

Fig. 20 – F1-score raggiunte nei tre esperimenti con layer Dense 4096

Confrontando i valori nelle due tabelle, si può notare come i risultati raggiunti con un ridimensionamento delle immagini a 150x150 siano migliori rispetto a quelli raggiunti negli altri due casi.
Inoltre i tempi di addestramento erano circa la metà passando da 256x256 a 150x150 (4 ore e mezza nel primo caso, un'ora e mezza nel secondo).
Tra 150x150 e 50x50, la differenza di tempi era di circa mezz'ora.

Si è svolto poi un secondo test: variando solamente la dimensione del layer di classificazione a 1024 ("Modello 1024"), unico layer addestrabile, si sono fatti ripartire gli addestramenti per le tre diverse dimensioni delle immagini in parallelo.
I risultati ottenuti sono riportati in fig.21 e fig.22.

Modello 1024: Accuracy		
	Train	Test
1. 256x256	58%	44%
2. 150x150	82%	65%
3. 50x50	65%	53%

Fig. 22 – Accuracy ottenuta nei tre esperimenti con layer Dense 1024

	Modello 1024: F1-score					
	256x256		150x150		50x50	
	Train	Test	Train	Test	Train	Test
Anger	0.63	0.52	0.80	0.65	0.65	0.48
Contempt	0.00	0.00	0.00	0.00	0.05	0.00
Disgust	0.50	0.37	0.84	0.85	0.54	0.38
Fear	0.69	0.00	0.81	0.07	0.51	0.00
Happiness	0.64	0.53	0.91	0.72	0.73	0.68
Sadness	0.01	0.00	0.48	0.00	0.46	0.00
Surprise	0.66	0.50	0.87	0.75	0.80	0.75

Fig. 23 – F1-score raggiunte nei tre esperimenti con layer Dense 1024

Per questo secondo caso le considerazioni sono analoghe: i valori di accuracy e f1-score nel caso 150x150 sono risultati essere i più performanti raggiunti in tempi ragionevoli.
Nelle prove successive, si è scelto perciò di utilizzare un metodo di Keras per caricare i samples del dataset che effettuasse automaticamente un *rescaling* delle immagini del CK+ a 150x150 e di utilizzare tale dataset ridimensionato come input per l'addestramento dei modelli.

4.2.2 Pesi completamente addestrabili

Si è svolto un esperimento per valutare se la rete potesse migliorare le proprie performance settando come modificabili tutti i suoi pesi.
I risultati ottenuti sono mostrati in fig.19 e fig.20.

	Risultati per 7 epoche			
Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1.	4.2058	0.1360	1.9261	0.2375
2.	1.9040	0.2285	1.8813	0.2375
3.	1.8619	0.2285	1.8586	0.2375
4.	1.8409	0.1600	1.8528	0.2375
5.	1.8337	0.2285	1.8516	0.2375
6.	1.8309	0.2285	1.8526	0.2375
7.	1.8306	0.1728	1.8523	0.2375

Fig. 24 – Valori di accuracy e loss per 7 epoche

F1-score		
	Train	**Test**
Anger	0.00	0.00
Contempt	0.00	0.00
Disgust	0.00	0.00
Fear	0.00	0.00
Happiness	0.00	0.00
Sadness	0.00	0.00
Surprise	0.37	0.38
Accuracy:	0.23	0.24

Fig. 25 – Valori di f1-score e accuracy dopo l'addestramento

Dai risultati ottenuti è evidente che il modello non riesce ad apprendere correttamente dal dataset di train e le performance sono minime.
Si è scelto perciò di settare come addestrabili solo i top layers in tutti gli esperimenti svolti in seguito.

4.2.3 Dimensione del penultimo layer Dense

Si è riflettuto se cambiare la dimensione del penultimo layer Dense, ovvero se il numero di neuroni presenti in esso, potesse influire sulla classificazione.
Si sono svolte 5 prove, nelle quali la dimensione delle immagini in input è stata fissata a 150x150 e si è cambiata la dimensione del penultimo layer Dense ai valori 512,1024,2048,3200,4096.
Di ogni esperimento vengono mostrati i valori delle matrici di confusione, del classification_report e dell'accuracy massima raggiunta su train e test set.
Solo per il primo esperimento svolto vengono visualizzati i valori di accuracy su train e test per tutte le epochs dell'addestramento.
La prima prova svolta è stata fatta settando come dimensione del penultimo layer Dense 1024.
Il modello di VGG16 utilizzato è visibile in fig.26.

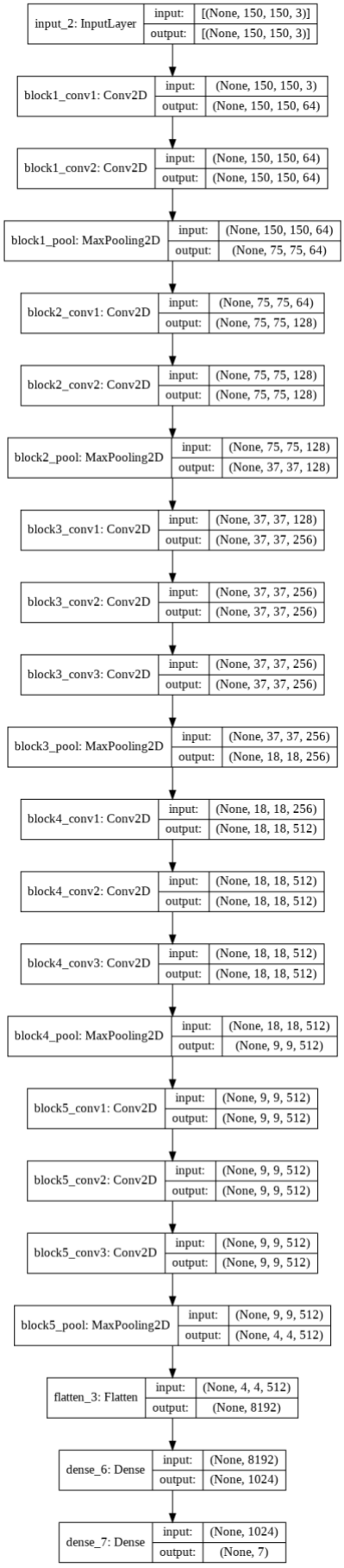

Fig. 26 – Modello utilizzato nel primo test

I valori di *accuracy* e *val_accuracy* calcolati durante l'addestramento epoch per epoch sono riportati in figura 4.
I valori più alti raggiunti sono 82% sul train e 65% sul test.

Risultati per 20 epoche				
Epoch	**Loss**	**Accuracy**	**Val_loss**	**Val_accuracy**
1.	2.1250	0.2158	1.6754	0.2995
2.	1.6365	0.4283	1.5914	0.5043
3.	1.4421	0.5147	1.4849	0.5439
4.	1.3262	0.5721	1.3992	0.5680
5.	1.2630	0.5767	1.3704	0.5422
6.	1.1362	0.6651	1.3325	0.5370
7.	1.1110	0.6228	1.2563	0.6024
8.	1.0098	0.6610	1.1943	0.6730
9.	0.9108	0.7541	1.3332	0.5508
10.	1.0019	0.6552	1.2411	0.5404
11.	0.8309	0.7599	1.2027	0.6437
12.	0.8415	0.7254	1.1138	0.6145
13.	0.8145	0.7291	1.1186	0.5697
14.	0.7661	0.7636	1.1207	0.6609
15.	0.7282	0.7761	1.0802	0.6523
16.	0.6700	0.8201	1.1652	0.5904
17.	0.6942	0.7831	1.0379	0.6472
18.	0.6517	0.7939	1.1082	0.6454
19.	0.6588	0.7935	1.0288	0.6627
20.	0.6177	0.8209	1.1150	0.6506

Fig. 27 – Valori di loss e accuracy ottenute nel primo test

I valori della matrice di confusione per il dataset di train sono mostrati in fig.28.

Classi predette								
Classi reali		An.	Con.	Dis.	Fear	Hap.	Sad.	Surp.
	Anger	418	0	11	0	0	0	16
	Contempt	51	0	0	1	19	2	12
	Disgust	21	0	278	0	10	0	43
	Fear	7	0	3	149	34	3	20
	Happiness	8	0	4	0	506	0	17
	Sadness	99	0	10	0	0	72	43
	Surprise	2	0	3	0	4	0	541

Fig. 28 – Matrice di confusione per il dataset di train del primo test

Dalla matrice di confusione di fig.28, si può notare come la classe “Contempt” è stata classificata erroneamente nella totalità dei casi, la maggior parte delle quali come classe “Anger”.
La classe “Contempt” è la classe che presenta meno samples e ci si aspettava che non venisse classificata sempre correttamente, tuttavia si è pensato di effettuare altre prove per verificare se cambiando il numero di neuroni del penultimo layer fully-connected le capacità di classificazione del modello potessero migliorare.
In fig.29 è rappresentato il classification_report per il dataset di train.

	Precision	**Recall**	**F1-score**	**Totale dei samples**
Anger	0.69	0.94	0.80	445
Contempt	0.00	0.00	0.00	85
Disgust	0.90	0.79	0.84	352
Fear	0.99	0.69	0.81	216
Happiness	0.88	0.95	0.91	535
Sadness	0.94	0.32	0.48	224
Surprise	0.78	0.98	0.87	550
		Accuracy:	82%	2407

Fig. 29 – Classification report per il dataset di train del primo test

Si può notare come il classificatore ha un’accuracy dell’82% sul train.

Per il dataset di test, la matrice di confusione è presentata in fig.30.

Classi predette								
Classi reali		An.	Con.	Dis.	Fear	Hap.	Sad.	Surp.
	Anger	62	0	2	0	0	0	12
	Contempt	16	0	0	0	14	0	2
	Disgust	4	0	78	0	0	0	11
	Fear	2	0	0	2	42	0	9
	Happiness	11	0	1	0	112	0	10
	Sadness	12	0	4	0	11	0	26
	Surprise	8	0	6	0	0	0	124

Fig. 30 – Matrice di confusione per il dataset di test del primo esperimento

Dalla matrice di confusione di figura 7, si può notare come nel test il classificatore non riesca ad individuare correttamente, oltre ai frames appartenenti alla classe "Contempt", neanche i samples appartenenti alle classi "Fear" e "Sadness", quest'ultima confusa nella totalità dei casi con altre classi.
In fig.31 è presente il classification_report per il dataset di test.
L'accuracy raggiunta dal classificatore sul test set è del 65% ma presenta una f1-score molto bassa o nulla per le tre classi "Contempt", "Fear", "Sadness".

	Precision	**Recall**	**F1-score**	**Totale dei samples**
Anger	0.54	0.82	0.65	445
Contempt	0.00	0.00	0.00	85
Disgust	0.86	0.84	0.85	352
Fear	1.00	0.04	0.07	216
Happiness	0.63	0.84	0.72	535
Sadness	0.00	0.00	0.00	224
Surprise	0.64	0.90	0.75	550
		Accuracy:	65%	2407

Fig. 31 – Classification report per il dataset di test della prima prova

Come secondo esperimento si è pensato di diminuire la dimensione del penultimo layer Dense a 512. I valori ottenuti dopo circa 20 epoche sono in fig.32.

Accuracy: 0.8575 Val_accuracy: 0.5542

Fig. 32 – Accuracy su train e test raggiunta nel secondo esperimento

Rispetto allla prima prova svolta, l'accuracy sul test è diminuita.
Per il dataset di train si è ottenuta la matrice di confusione di fig.33.

		Classi predette						
Classi reali		An.	Con.	Dis.	Fear	Hap.	Sad.	Surp.
	Anger	438	0	0	0	1	0	6
	Contempt	9	70	0	0	6	0	0
	Disgust	55	0	244	0	10	0	43
	Fear	3	0	0	179	25	1	8
	Happiness	8	0	0	0	516	0	11
	Sadness	113	0	0	0	2	71	38
	Surprise	3	0	0	0	1	0	546

Fig. 33 – Matrice di confusione per il dataset di train del secondo esperimento

Il classification_report per il dataset di train è rappresentato in fig.34.

	Precision	**Recall**	**F1-score**	**Totale dei samples**
Anger	0.70	0.98	0.82	445
Contempt	1.00	0.82	0.90	85
Disgust	1.00	0.69	0.82	352
Fear	1.00	0.83	0.91	216
Happiness	0.92	0.96	0.94	535
Sadness	0.99	0.32	0.48	224
Surprise	0.84	0.99	0.91	550
		Accuracy:	86%	2407

Fig. 34 – Classification report per il dataset di train del secondo esperimento

Dalla matrice di confusione di fig.34, si può notare come in questo caso la classe "Contempt" non venga ignorata dal classificatore ma classificata correttamente per la maggioranza dei casi.
Tuttavia, per verificare se fosse una casualità o il classificatore avesse davvero imparato correttamente durante l'addestramento, si sono visti i valori calcolati per il test (fig.35).

Classi predette								
Classi reali		An.	Con.	Dis.	Fear	Hap.	Sad.	Surp.
	Anger	13	0	24	0	8	19	12
	Contempt	5	0	0	0	18	1	8
	Disgust	17	0	68	0	8	0	0
	Fear	1	0	0	6	44	0	4
	Happiness	5	0	7	0	111	1	10
	Sadness	7	0	12	0	17	0	17
	Surprise	0	0	8	0	6	0	124

Fig. 35 – Matrice di confusione per il dataset di test del secondo esperimento

Il classification_report per il test è riportato in fig. 36.

	Precision	**Recall**	**F1-score**	**Totale dei samples**
Anger	0.27	0.17	0.21	445
Contempt	0.00	0.00	0.00	85
Disgust	0.57	0.73	0.64	352
Fear	1.00	0.11	0.20	216
Happiness	0.52	0.83	0.64	535
Sadness	0.00	0.00	0.00	224
Surprise	0.71	0.90	0.79	550
		Accuracy:	55%	2407

Fig. 36 – Classification report per il dataset di test del secondo esperimento

Dai valori di matrice e report per il test si evince che il classificatore ha un comportamento peggiore rispetto al primo test effettuato: non classifica correttamente i samples delle tre classi “Contempt”, “Fear” e “Sadness”, i valori della f1-score sono molto bassi o nulli e presenta anche una accuracy sul test inferiore.
Si è scartata l’ipotesi di effettuare prove diminuendo ulteriormente la dimensione del penultimo layer Dense.
Le prove svolte in seguito perciò si sono concentrate sull aumento di tale parametro: si è settata la dimensione del layer di classificazione a 2048, 3200 ed infine a 4096.
In fig.37 si confrontano le accuracy su train e test raggiunte, in confronto anche quelle ottenute nei primi esperimenti.

	Valori di accuracy	
	Train	**Test**
1. 512	86%	55%
2. 1024	82%	65%
3. 2048	84%	66%
4. 3200	87%	58%
5. 4096	88%	66%

Fig. 37 – Confronto fra i valori di accuracy raggiunti

Considerando solo i valori di accuracy, sembrerebbe che il modello con migliori performance sia il quinto.
Si è pensato tuttavia di valutare i valori di f1-score per i 4 modelli testati, in quanto per un dataset fortemente sbilanciato risultano dei parametri di confronto più accurati.

	Valori di f1-score									
	512		**1024**		**2048**		**3200**		**4096**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Anger	0.82	0.21	0.80	0.65	0.84	0.66	0.78	0.31	0.92	0.54
Contempt	0.90	0.00	0.00	0.00	0.71	0.34	0.81	0.40	0.71	0.45
Disgust	0.82	0.64	0.84	0.85	0.85	0.69	0.83	0.63	0.86	0.75
Fear	0.91	0.20	0.81	0.07	0.85	0.00	0.86	0.17	0.56	0.00
Happiness	0.94	0.64	0.91	0.72	0.89	0.69	0.95	0.71	0.90	0.70
Sadness	0.48	0.00	0.48	0.00	0.41	0.00	0.80	0.20	0.90	0.40
Surprise	0.91	0.79	0.87	0.75	0.91	0.87	0.93	0.84	0.94	0.85

Fig. 38 – Confronto fra i valori di f1-score raggiunti

Dalla fig. 38 si evince che il modello con dimensione del layer di classificazione a 3200 sia l'unico per il quale sia sul train che sul test non sono presenti valori nulli.
Tale modello, anche se meno accurato degli altri sul test, riesce però a classificare tutte le classi seppure con valori mediamente più bassi degli altri.
Si è deciso perciò di adottare tale modello come architettura a monte per i test effettuati con LSTM ("Modello 3200", rappresentato in fig.39).

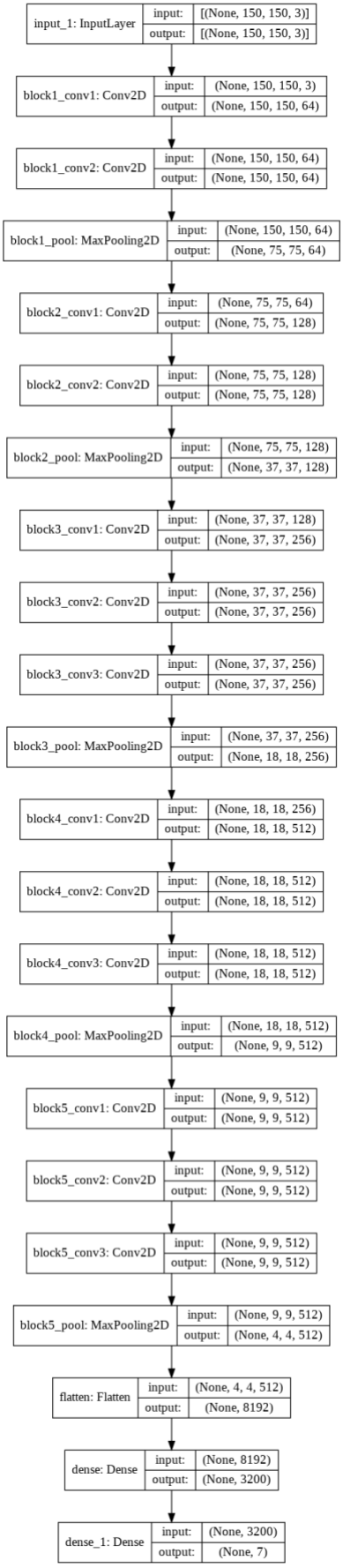

Fig. 39 – Modello 3200

4.3 Classificazione su sequenze video

Prima di effettuare classificazione su sequenze di frames e sequenze di features, è stato necessario effettuare un ulteriore step di pre-processing sul CK+.
I video contenuti in esso non hanno lunghezza fissa, ovvero ogni sequenza può contenere un numero di frames diverso.
La LSTM, tuttavia, ammette sequenze video di lunghezza fissa. È stato perciò necessario tagliare alcuni frames da ogni sequenza video in modo da ottenere video con un numero fisso di fotogrammi.

4.3.1 Creazione delle sequenze

Si è valutato quanti frames mantenere in ogni sequenza e di conseguenza quanti doverne eliminare dalle sequenze più lunghe: un taglio eccessivo avrebbe ridotto enormemente un dataset già di ridotte dimensioni.
Nel dataset di train, la maggior parte delle sequenze ha una lunghezza compresa tra 4 e 12 fotogrammi mentre nel dataset di test tra 5 e 11.
Le sequenze contenenti più di 15 immagini sono un'esigua minoranza.
Dovendo ottenere tutti video della medesima lunghezza, si è scelto di tagliare a 4 frames, in modo da mantenere quasi tutte le sequenze (solo due sequenze nel test set hanno lunghezza inferiore a 4).
Tramite codice, si è effettuato un taglio di tutti i video presenti nel CK+ mantenendo solo gli ultimi 4 fotogrammi di ogni sequenza, in quanto essi rappresentano meglio l'emozione associata.
Le sequenze da 2 e 3 frames sono state eliminate.

4.3.2 Classificazione su sequenze di frames

Per provare ad individuare dei pattern dai dati si è pensato inizialmente di utilizzare sequenze di frames.
Si è utilizzato a questo scopo il modello di VGG16 "Modello 3200", scelto negli esperimenti precedenti, a monte e la LSTM a valle.
La VGG classifica i singoli frames per ogni sequenza video, si ottengono così sequenze di labels per ogni video da dare in input alla LSTM, la quale effettua classificazione su tali sequenze.

Le classi ottenute, per poter essere utilizzate dalla LSTM, vengono inoltre codificate come vettore di probabilità *one-hot (*nel vettore vi è un 1 in corrispondenza della classe predetta, 0 altrimenti).
I risultati ottenuti si valutano tramite matrice di confusione e metriche ad essa associate.
Inizialmente si è variato il numero di celle della LSTM, per valutare dagli esperimenti se tale parametro influisse sulle prestazioni della rete.
In particolare, si è impostato il numero di celle pari a 50, 100, 200.
Per l'esperimento con 50 celle, dopo circa 10 epoche la funzione di loss sul dataset di test iniziava ad aumentare: il modello andava in overfitting. Per gli altri due modelli, ciò accadeva dopo circa 6 epoche.
In tutti e tre gli esperimenti tuttavia, il valore di accuracy sul test set oscillava tra pochi valori intorno al 66% fin dalla prima epoca.
Nelle fig.40 e 41 sono riportati i risultati degli esperimenti svolti.

	Valori di accuracy	
	Train	**Test**
1. 50	91%	68%
2. 100	91%	69%
3. 200	92%	66%

Fig. 40 – Accuracy raggiunte variando il numero di celle della LSTM

	Valori di f1-score					
	50		**100**		**200**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Anger	0.90	0.70	0.90	0.70	0.91	0.64
Contempt	0.82	0.29	0.82	0.29	0.82	0.29
Disgust	0.89	0.83	0.90	0.87	0.90	0.83
Fear	0.88	0.25	0.84	0.25	0.88	0.25
Happiness	0.93	0.69	0.92	0.69	0.93	0.62
Sadness	0.86	0.00	0.86	0.29	0.86	0.29
Surprise	0.98	0.85	0.98	0.85	0.98	0.85

Fig. 41 – F1-score raggiunte variando il numero di celle della LSTM

Dalla fig.41, si nota come il modello da 50 celle è l'unico con un valore di f1-score nullo, perciò la scelta sul modello più performante è tra gli altri due modelli.

Tra il modello con 100 unità e il modello con 200 unità, i valori di f1-score risultano simili.
Tuttavia, calcolando la f1-score media, si sono ottenuti i valori di fig. 42.

F1-score media		
	Train	**Test**
100	0.88	0.56
200	0.89	0.54

Fig. 42 - Confronto delle f1-score medie tra i modelli a 100 e 200 unità

Il modello a 100 celle ha un valore di f-score media più alto sul test rispetto al modello da 200.
Tale modello ("LSTM 100", fig. 43) è stato scelto come parametro di confronto per il test successivo, dove si è valutata l'aggiunta di un layer Dropout.

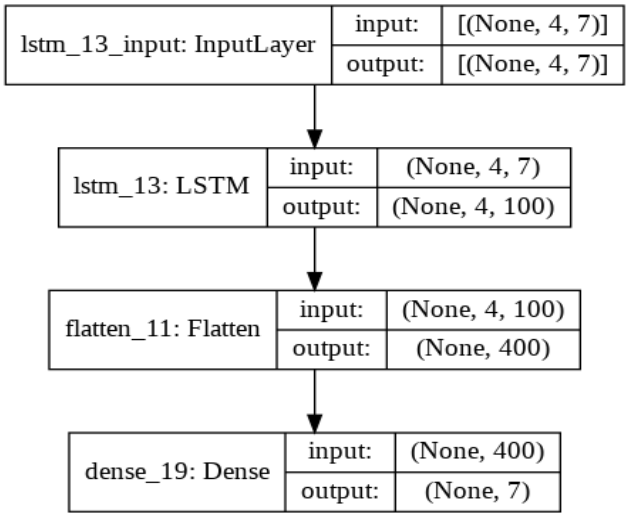

Fig. 43 – Modello "LSTM 100"

Un problema per l'LSTM riscontrato negli esperimenti è che essa può facilmente andare in overfitting sul training set, riducendo la propria abilità di generalizzazione.
Per migliorare tale situazione, si è pensato perciò di utilizzare un layer Dropout.
Il nome di tale layer deriva dal fatto che esso permette di "ignorare" alcuni neuroni durante la fase di training, scelti randomicamente, ovvero tali neuroni non vengono considerati durante l'apprendimento.
È stato perciò fatto un esperimento aggiungendo tale layer a valle della LSTM e confrontati i valori di accuracy e f1-score con quelli ottenuti tramite il modello "LSTM 100".

In fig.44 si mostra un confronto con i valori ottenuti durante il primo esperimento.

	Valori di f1-score			
	LSTM 100		**LSTM 100 con Dropout**	
	Train	**Test**	**Train**	**Test**
Anger	0.90	0.70	0.92	0.63
Contempt	0.82	0.29	0.82	0.29
Disgust	0.90	0.87	0.92	0.83
Fear	0.84	0.25	0.91	0.25
Happiness	0.92	0.69	0.93	0.65
Sadness	0.86	0.29	0.83	0.25
Surprise	0.98	0.85	0.98	0.85
Accuracy:	91%	69%	90%	69%

Fig. 44 – Confronto tra i valori di f1-score

Dalla fig. 44 si nota che i valori di f1-score sono simili, tuttavia confrontando i valori medi (fig. 45) si evince che il layer Dropout non ha portato ad un miglioramento delle prestazioni del modello, essendo più bassa la f-score media sul test set.

F1-score media		
	Train	**Test**
100	0.88	0.56
100 con Dropout	0.90	0.54

Fig. 45 – F1-score medie

4.3.3 Classificazione su sequenze di features

Si è valutato se l'architettura VGG+LSTM potesse individuare pattern nella manifestazione delle emozioni, utilizzando come input sequenze di features invece che sequenze di labels.
La VGG a monte è usata per estrarre features di alto livello dalle sequenze video di 4 frames, le quali poi sono date in input alla LSTM.

Come prima tipologia di esperimenti, si è variato il numero di celle della LSTM, per valutare se tale parametro influisse sulle prestazioni della rete e anche per verificare se l'utilizzo delle features migliorasse le prestazioni rispetto all'uso delle labels.
In particolare, si è impostato il numero di celle pari a 50, 100, 200.
Durante l'addestramento, per tutte e tre le prove, l'accuracy calcolata sul test set partiva già dalla prima epoca da un valore di circa 0.53.
Inoltre durante gli esperimenti, dopo circa 5-6 epoche la loss iniziava ad aumentare per tutti e 3 i modelli, quindi overfittavano abbastanza velocemente.
I risultati vengono riportati nelle fig. 46, 47 e 48.

	Valori di accuracy	
	Train	**Test**
1. 50	96%	69%
2. 100	95%	58%
3. 200	93%	69%

Fig. 46 – Valori di accuracy raggiunti nei tre esperimenti

	Valori di f1-score					
	50		**100**		**200**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Anger	0.92	0.70	0.91	0.43	0.88	0.80
Contempt	0.92	0.50	0.86	0.50	0.86	0.50
Disgust	0.97	0.87	0.93	0.59	0.96	0.90
Fear	0.91	0.00	0.91	0.00	0.80	0.00
Happiness	0.97	0.77	0.98	0.67	0.96	0.71
Sadness	0.95	0.00	0.91	0.00	0.93	0.00
Surprise	0.99	0.81	0.98	0.81	0.98	0.81

Fig. 47 – Valori di f1-score raggiunti nei tre esperimenti

F1-score media		
	Train	**Test**
50	0.95	0.52
200	0.91	0.53

Fig. 48 – Confronto fra le f1-score medie dei modelli

Dall'analisi dei risultati si evince che:

- Utilizzando le features, la LSTM ha mediamente prestazioni di poco inferiori sul test rispetto a quelle che si avevano utilizzando le labels (figura).
- Tra i tre modelli, quello con la f-score media più alta è l'architettura la cui LSTM ha 200 celle.

Si è utilizzato quest'ultimo modello per l'esperimento successivo, dove si è valutato se le prestazioni del modello potessero migliorare ulteriormente aggiungendo un layer Dropout in cascata.
Si riportano i risultati ottenuti nelle fig. 49 e 50.

	Valori di f1-score			
	LSTM 200		**LSTM 200 con Dropout**	
	Train	**Test**	**Train**	**Test**
Anger	0.88	0.80	0.95	0.67
Contempt	0.86	0.50	0.76	0.33
Disgust	0.96	0.90	0.99	0.74
Fear	0.80	0.00	1.00	0.20
Happiness	0.96	0.71	0.99	0.67
Sadness	0.93	0.00	0.98	0.29
Surprise	0.98	0.81	1.00	0.94
Accuracy:	93%	69%	98%	68%

Fig. 49 – Confronto fra i valori di f1-score

F1-score media		
	Train	**Test**
200	0.91	0.53
200 con Dropout	0.95	0.55

Fig. 50 – F1-score medie

In questo caso, l'aggiunta del layer Dropout ha portato ad un miglioramento delle prestazioni del modello.

4.3.4 Considerazioni sui risultati ottenuti

Gli esperimenti svolti non sono stati effettuati in un numero congruo tale da dedurre una interpretazione definita.

Tuttavia, è emerso che:

- L'accuracy ottenuta sul dataset di test per entrambi gli approcci seguiti (sequenze di frames e sequenze di features) non supera il 69%: questo potrebbe dipendere dalla scarsità di elementi presenti nel dataset, i quali sono ulteriormente diminuiti dopo aver creato le sequenze di dimensione fissa. Un possibile miglioramento potrebbe sussistere se venissero prese in considerazione sequenze più lunghe.

- L'accuracy sul test in entrambi gli approcci ha un valore superiore al 50% circa fin dalla prima epoca, poi aumenta nel range 60-69%.
  La spiegazione più plausibile è che l'architettura riesca a riconoscere abbastanza bene fin da subito alcuni pattern nella manifestazione delle emozioni ma forse, vista anche la dimensione ridotta del dataset, non ve ne sono molti.

- Utilizzare le features nel riconoscimento delle emozioni sembrerebbe più conveniente rispetto all'utilizzare le labels solo in un caso, ovvero utilizzando il layer Dropout a valle della LSTM. Questo potrebbe suggerire che l'architettura riesca ad apprendere meglio attraverso le features ma potrebbe adattarsi troppo al training set e quindi generalizzare meno.

- Tra i due approcci non vi è una differenza considerevole in termini di accuracy sul dataset di test.

- Un possibile miglioramento nelle prestazioni si potrebbe ottenere anche migliorando l'accuratezza del modello di VGG a monte che effettua classificazione o estrazione di features.

Conclusioni e sviluppi futuri

Le emozioni hanno un ruolo importante per poter interagire tra esseri umani, in quanto danno la possibilità alle persone di articolare sé stesse senza usare le parole. Il linguaggio del corpo, le espressioni del viso, permettono di poter esprimere sentimenti e sensazioni in modo più efficace ed anche di poter esporre opinioni non sempre intuibili dal solo linguaggio.
Al giorno d'oggi la maggior parte dei contenuti disponibili in rete è composta da dati non strutturati, come immagini e video, condivisi e commentati sui principali Social Network e piattaforme di streaming.
In molti dei video sono presenti persone che esprimono opinioni su svariati campi, dalla politica ai prodotti acquistati.
Per questo motivo risulta cruciale per le aziende valutare l'opinione dei clienti, per capire se i prodotti proposti sono apprezzati o meno ma anche per aprirsi a nuove fette di mercato, grazie ad una idea che può emergere dalle recensioni.
In questo lavoro di tesi si è proposto un approccio di analisi dei video volto ad individuare le 7 emozioni di base umanamente esprimibili, facendo riferimento a metodologie note in letteratura per la loro efficacia, in modo da ottenere i migliori risultati possibili.
Come punto di partenza per l'analisi è stato perciò scelto il Cohn-Kanade Database esteso, in quanto è uno dei più grandi set di video con emozioni annotate disponibili open-source.
Su di esso è stata inoltre effettuata una fase di *data cleansing* volta a migliorarne l'utilizzo da parte delle architetture scelte in seguito.
Si è poi fatta estrazione di features e labels utilizzando come rete neurale la VGG16, architettura che ha raggiunto il 92.7% di accuracy sul test nella ILSVRC competition del 2014: tale scelta è risultata vincente in quanto l'*accuracy* ottenuta nelle predizioni dopo la fase di addestramento è del 60%.
Tale risultato può considerarsi molto valido, vista anche la dimensione ridotta dei dati a disposizione.
Passando all'analisi video, l'utilizzo combinato di VGG16 con LSTM in cascata si è rivelato efficace per lo scopo prefissato: le architetture raggiungono una *accuracy* del 70%, un ottimo risultato per un problema di classificazione a 7 classi.
Tale risultato è un buon punto di partenza per migliorare ulteriormente le prestazioni dell'architettura, variando alcuni parametri della rete come ad esempio l'aggiunta di più hidden layers.
Un grande miglioramento si potrebbe ottenere acquisendo nuovi dataset annotati per testare i modelli.

In questo ambito tuttavia, il test dell'architettura è un processo non semplice, in quanto pochissimi dataset contenenti video sono accuratamente etichettati in base alle emozioni che trasmettono.
Questo perché si ha bisogno di una annotazione manuale dei video stessi, ma ciò comporta una problematica intrinseca in quanto l'annotazione potrebbe variare in base alla sensibilità, affidabilità, serietà dell'annotatore, ma anche dalla sua estrazione sociale ed etnia.
Se le micro espressioni delle emozioni in viso sono più o meno simili, le macro espressioni invece variano in base alla cultura e al paese di provenienza ed in alcuni casi le emozioni potrebbero risultare più difficili da individuare.
Nondimeno, per migliorare ulteriormente le performance dei modelli presentati si potrebbero utilizzare tecniche di *data augmentation* o estendere la metodologia sviluppata anche su altre applicazioni di machine learning come ad esempio la *voice recognition.*

Bibliografia

[1] wearesocial.com/it/blog/2020/01/report-digital-2020-i-dati-global

[2] cutt.ly/lb463II

[3] Ekman, P. (1999). *Basic emotions.* In T. Dalgleish & M. J. Power (Eds.), *Handbook of cognition and emotion* (p. 45–60). John Wiley & Sons Ltd.

[4] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, 2010, pp. 94-101, doi: 10.1109/CVPRW.2010.5543262.

[5] Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

[6] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[7] Bisong E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA.

[8] keras.io

[9] docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[10] T. -H. S. Li, P. -H. Kuo, T. -N. Tsai and P. -C. Luan, "CNN and LSTM Based Facial Expression Analysis Model for a Humanoid Robot," in *IEEE Access*, vol. 7, pp. 93998-94011, 2019, doi: 10.1109/ACCESS.2019.2928364.

[11] P. Khorrami, T. L. Paine and T. S. Huang, "Do Deep Neural Networks Learn Facial Action Units When Doing Expression Recognition?," *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2015, pp. 19-27, doi: 10.1109/ICCVW.2015.12.

[12] towardsdatascience.com

[13] neurohive.io/en/popular-networks/vgg16/

[14] scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix

[15] colah.github.io/posts/2015-08-Understanding-LSTMs/

[16] tensorflow.org/tutorials/images/classification

[17] en.wikipedia.org/wiki/Artificial_neural_network