UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Relazione del Laboratorio di Deep Learning, Emotions Recognition

STUDENTI

Zuccato Francesco 143095
Bolletta Christian 142478

Anno accademico 2021-2022

Consegna

In questo laboratorio è richiesto di realizzare una rete neurale in grado di predire le emozioni dal database "fer2013.csv" che contiene 35887 immagini di 48x48 pixel. Nello specifico, in ogni riga è presente sia un'immagine in bianco e nero della faccia di una persona, sia la relativa emozione, compresa tra le seguenti: rabbia, disgusto, paura, felicità, tristezza, stupore, neutralità.

Per l'analisi di immagini è noto come le reti più adatte siano le CNN (*Convolutional Neural Networks*). Nel testo è presentata il preprocessing e l'analisi iniziale del dataset, l'implementazione di alcuni modelli di CNN e l'analisi dei risultati finali.

Indice

1

Analisi e Preprocessing del dataset

Il dataset è formato da immagini, ciò implica che non è possibile eseguire un'analisi iniziale delle x. Non resta che analizzare le y del dataset, ovvero le emozioni. Si è contato il numero di campioni per emozione, ed è emerso che non è presente una distribuzione uniforme delle labels.

Dal grafico si può vedere come siano presenti pochissimi campioni per il disgusto, e, di conseguenza, tale emozione sara molto più difficilmente apprendibile nella fase di riconoscimento. Al contrario, la felicità sarà l'emozione su cui predire sarà più semplice e si otterrà un'accuracy migliore.

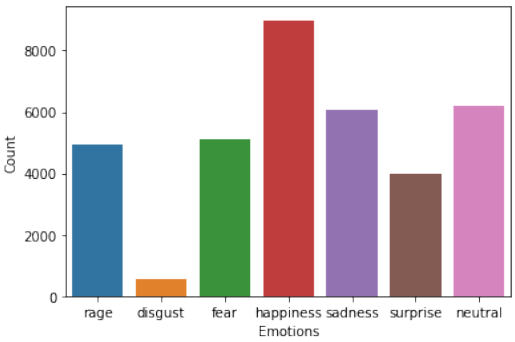

Figura 1.1: Numero di campioni per emozione

Divisione in Train, Validation, Test

Sebbene nel dataset sia già fornita una possibile suddivisione del dataset è stato deciso comunque di eseguire uno split randomico, specificando l'opzione *stratify=y*, per non alterare le distribuzioni delle y.

La dimensione del train usata è il classico 70%, ovvero 25120, che è un po' più piccola del 28709, che è quella consigliata. L'utilizzo del random split evita il rischio di focalizzarsi solo su una parte dei dati, e, memorizzando ed esportando il random seed usato, si garantisce la riproducibilità delle divisioni.

Preprocessing

Dall'analisi è risultato quindi che il dataset contiene un numero abbastanza esiguo di campioni, circa 35mila, e non è uniformemente distribuito. Si è deciso di procedere con l'*augmentation* delle immagini. Tale procedura consente di "ingrandire" un database col fine di ottenere più informazioni necessarie per l'addestramento della rete.

Per il caso delle immagini si possono utilizzare varie tecniche, modificando sia le coordinate spaziali dei pixels, tramite rotazioni, inversioni, ritagli ecc., sia i colori, facendo uso di appositi filtri. Tra questi si annoverano l'*edge-detector*, per la rilevazione le righe verticali e/o orizzontali, il *sobel* per i contorni, ed altri ancora per modificare contrasto, luminosità ed intensità dei colori.

Per non appesantire troppo la rete, è stato deciso di fare uso di solamente due filtri: il sobel e uno per l'aumento del contrasto, entrambi aggiunti al dataset chiamato 'fer2021.csv', usato poi nella rete.

2

Rete CNN

La rete è stata implementata in modo parametrico, così da agevolare la modifica dei parametri quali il numero di layers, le dimensioni dei pesi ed il numero dei canali. Per fare ciò sono state realizzate due reti diverse, entrambe parametriche. Le reti sono generate dinamicamente a tempo di compilazione, in quanto la grandezza delle liste e degli output sono passati alla rete stessa come parametri.

1) una lista di (n liste di *CDrop-Block*, *MaxPool2D*)
2) un *DropOut*
3) una lista di *Linear*
4) una *SoftMax*

1) una lista di (n liste di *C-Block*, *MaxPool2D*)
2) un *DropOut*
3) una lista di *Inception-Block*
4) un *DropOut*
5) una lista di *Linear*
6) una *SoftMax*

2.1 Convolutional Part

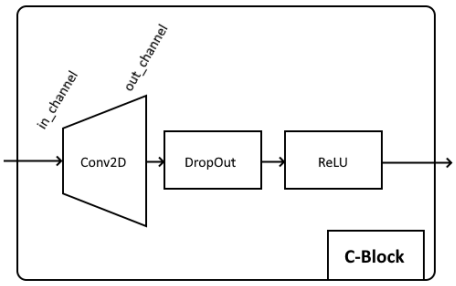

Figura 2.1: ConvDrop-Block

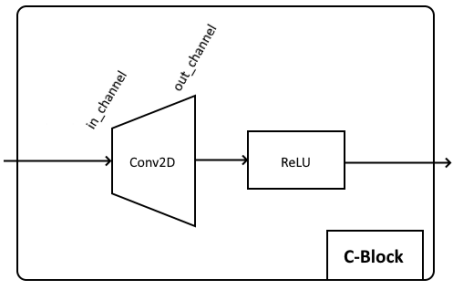

Figura 2.2: Conv-Block

La parte convoluzionale è composta da una sequenza di blocchi (Fig. 2.3) che vengono ripetuti un a_0 volte, poi sono filtrati da un MaxPool, poi si ripetono di nuovo a_1 volte, poi filtrati da un MaxPool ecc. Solitamente il numero dei blocchi esterni usato è 3-5, la ripetizione di ciascun blocco interno è 1-4 volte.

I blocchi esterni sono così implementati: il primo blocco interno riceve $in\_chan = 3 = N_0$ e produce $out\_chan = N_1$, i seguenti blocchi interni continuano a produrre canali grandi N_1. Poi c'è il MaxPool e quindi il secondo blocco esterno, in cui il primo interno incrementerà i canali da N_1 a N_2 ecc.

Si noti come sia possibile (ma non avrebbe molto senso) diminuire le dimensioni dei canali. Ciò che invece dimuinuisce dopo ogni blocco esterno è la dimensione dell'input, che viene dimezzato dal MaxPool2D (ma che preserva la dimensione dei canali). Ogni blocco interno (Fig. 2.1, 2.2) è composto da un Conv2D, un DropOut (nel caso 1) e da una ReLU.

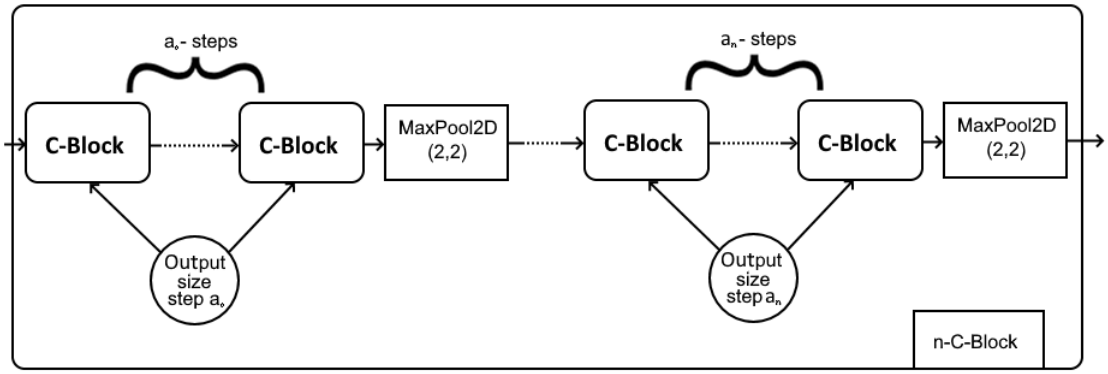

Figura 2.3: Lista di (lista di C-Block, MaxPool)

2.1.1 Inception Part

Per migliorare la stabilità e non ottenere un alto livello di overfit si è deciso di implementare una sequenza di Inception. Sono implementati come da specifiche Google, con una generalizzazione. La versione originale prevede infatti 256 canali di output: 64 da 1x1, 128 da 3x3, 32 da 5x5, 32 da pool. Mentre il codice prevede un parametro *incep__multiplier* che consente di ritornare $256 * mul$ canali.

Solitamente *mul* è stato usato con valori pari a 1 (ininfluente), ma soprattutto con 2 ($out\_ch = 512$) o 3 ($out\_ch = 769$). Si è evidenziato come l'aggiunta degli inception modules in gran numero (da 15 a 30) garantisca molta stabilità alla rete ed una crescita lenta ma abbastanza costante, al contrario un numero ridotto è praticamente inutile, soprattutto in un dataset piccolo come quello in questione.

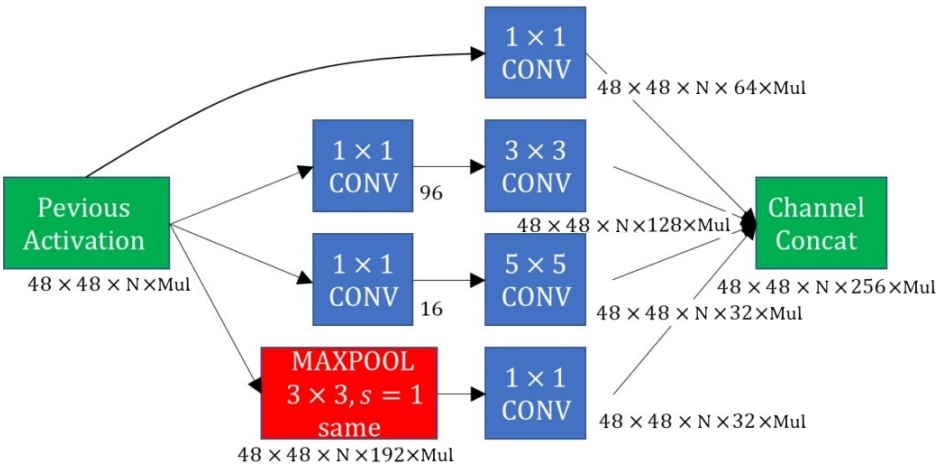

Figura 2.4: Inception Module

2.1.2 Classification Part

La parte di classificazione (Fig. 2.5), anch'essa parametrica, consiste in un susseguirsi di Linear layers di dimensione decrescente, fino all'ultimo che avrà dimensione pari a 7, che è il numero delle label. Questo diventa poi l'input per la SoftMax che finalmente restituirà la predizione dell'emozione dell'immagine. Solitamente in classificazione sono usati 4-5 layers, il primo con 500-1000 valori in input e i seguenti con un fattore decrescente esponenziale (1/2, 1/4, 1/8 del precedente).

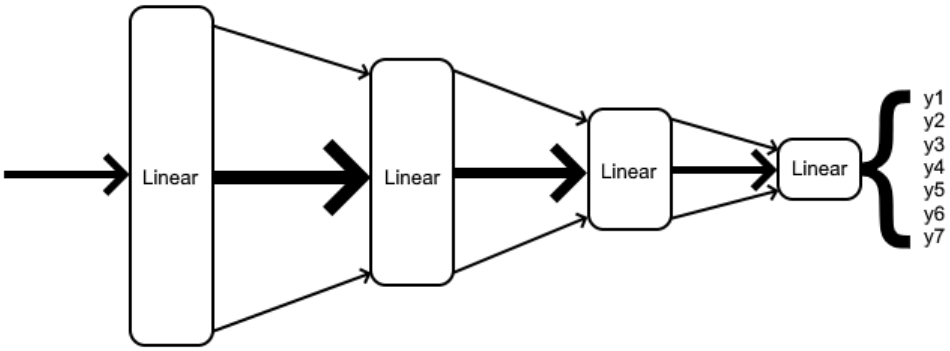

Figura 2.5: Linear Layers

3

Analisi complessiva

In questi giorni sono state provate molte reti diverse. Si è giunti alla conclusione che le migliori fossero quelle spiegate prima, senza moltissimi layer (dai 3 ai 8 solitamente) ma con canali di output molto grandi (ad es. 50,100,250,500), fino ad un massimo di 800, note come 'fat'. Entrambe le reti raggiungono il 50-60% di accuracy in testing. In ogni rete però il problema maggiore è quello di riuscire a riconoscere il disgusto, molte di quelle provate ottengono tutte lo 0%.

3.1 R1: Rete senza inceptions

La rete senza gli inceptions ottiene un'accuracy in training decisamente più alta, il 75-85%, rispetto a quella nel test, il 60%. Ovviamente la rete è andata in overfit. Tuttavia, questa è la migliore realizzata per l'accuracy nel testing accuracy. Inoltre, come si vede dalla Confusion Matrix (Fig. 3.1) è in grado di riconoscere **tutte** le labels, compreso il disgusto. Dalle osservazioni fatte usando parametri diversi si è riscontrato che le performance migliori si ottengano usando reti molto profonde e con pochi layers, mentre le reti "snelle" non producono un'accuracy accettabile (sopra il 50%). Il difetto principale di queste reti è sicuramente l'overfitting, che in certe osservazioni ha prodotto un'accuracy anche del 100% (!) in training (mentre in testing del 55-60%). Siccome il DropOut2D utilizzato è del 35% per contrastare l'overfitting è stato provato un dropout maggiore (40-50%) ma il risultato è che l'accuracy si riduceva drasticamente. A causa del trade-off e dei tempi di consegna ristretti abbiamo accettato un overfitting elevato che ci consente un buonissimo 60% di accuracy in testing.

3.2 R2: Rete con inceptions

La rete con gli inceptions migliore ha ottenuto un'accuracy del 57%. Si è riscontrato essere più lenta nell'apprendimento ma ad avere un overfit minore. Sicuramente sarebbe la rete ideale se il dataset fosse maggiore e se ci fosse stato il tempo di testarla con 100 epoche più volte, magari usando un learning rate minore (1^{e-5} al posto di 1^{e-4}) e con un dropout (forse) ancora maggiore del 62%, attualmente usato. Le prestazioni attualmente migliori le si sono ottenute concatenando ben 30 (!) inceptions modules diversi, con fattore moltiplicativo dei canali di output pari a 3. Purtroppo nessuno dei vari tentativi con questa rete è stato in grado di riuscire a predirre il disgusto, sempre fermo allo 0% di accuracy in testing.

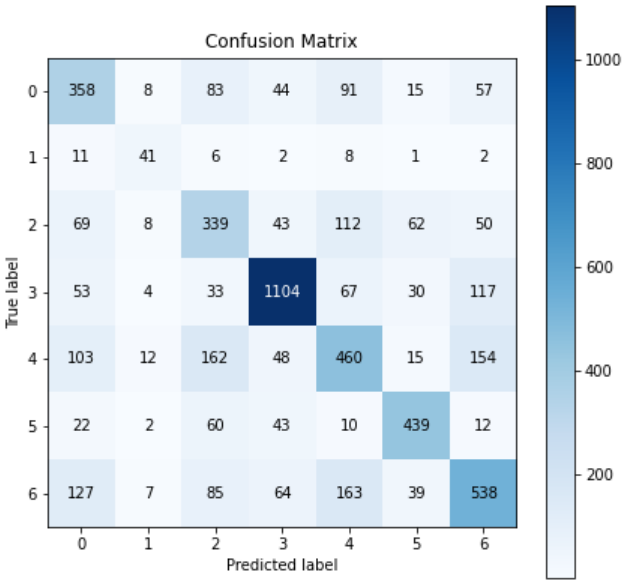

Figura 3.1: R1 confusion matrix

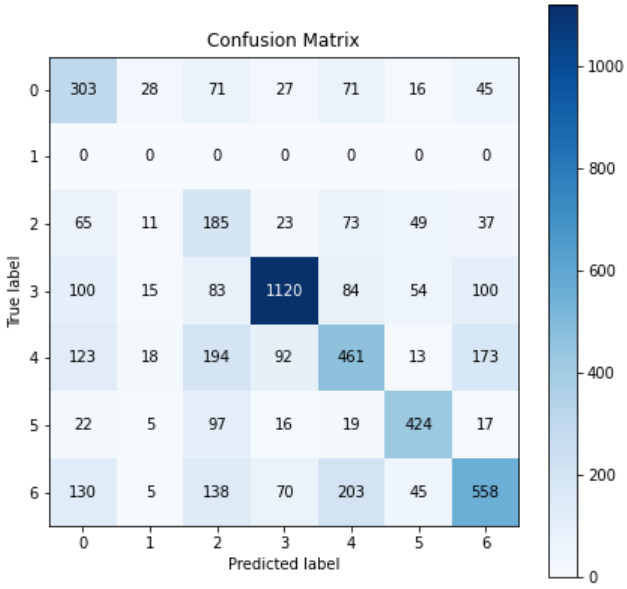

Figura 3.2: R2 confusion matrix

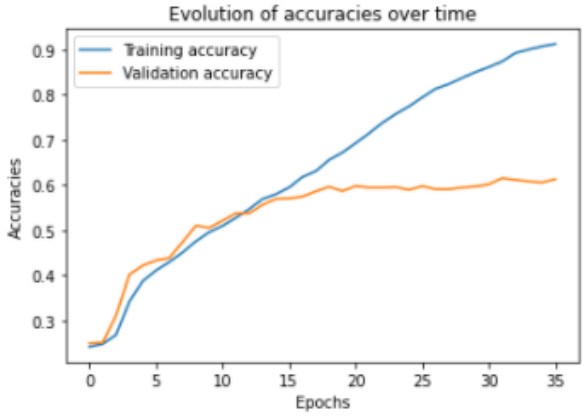

Figura 3.3: R1 accuracy

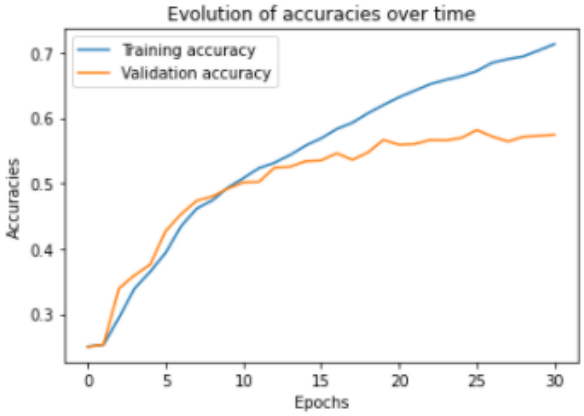

Figura 3.4: R2 accuracy

0.57944 mean of accuracy of each label
0.60914 accuracy of all test set

	emotions	true_preds	false_preds	accuracy
0	rage	358	385	0.481830
1	disgust	41	41	0.500000
2	fear	339	429	0.441406
3	happiness	1104	244	0.818991
4	sadness	460	451	0.504940
5	surprise	439	162	0.730449
6	neutral	538	392	0.578495

Figura 3.5: R1 summary

0.47015 mean of accuracy of each label
0.56678 accuracy of all test set

	emotions	true_preds	false_preds	accuracy
0	rage	303	440	0.407806
1	disgust	0	82	0.000000
2	fear	185	583	0.240885
3	happiness	1120	228	0.830861
4	sadness	461	450	0.506037
5	surprise	424	177	0.705491
6	neutral	558	372	0.600000

Figura 3.6: R2 summary

4

Conclusione

Dai i dati da noi analizzati, e dalle varie reti testate; sopraggiungiamo alla conclusione che le reti che fanno uso di Inception Modules sarebbero da usare su reti neurali molto più profonde e su dataset molto più popolati, così da consentirne una maggiore efficenza; mentre nel caso si volesse usufruire di reti meno profonde, dai nostri dati abbiamo concluso che le Pure Convolutional danno il meglio di loro nel caso siano molto larghe con un numero di ripetizioni decrescente alla sua profondità nella rete.