Università degli studi di Catania

Dipartimento di Matematica e Informatica

FACIAL EXPRESSION RECOGNITION

(Riconoscimento delle Espressioni Facciali)

Corso di

Machine Learning

Realizzato da

Andrea Pricoco

Simone Spina

Docenti

Giovanni M. Farinella

Rosario Leonardi

Anno Accademico

2024/2025

Indice

1. Introduzione

Il presente progetto affronta il problema del *Facial Expression Recognition* (FER), ovvero il riconoscimento automatico delle emozioni umane a partire dall'analisi delle espressioni facciali. Mentre per l'essere umano questo processo avviene in modo naturale ed immediato, per un sistema computazionale si tratta di una sfida complessa che richiede l'impiego di tecniche avanzate di machine learning e deep learning.

L'obiettivo è classificare l'emozione espressa in un'immagine del volto di un soggetto, assegnandola a una delle seguenti sette categorie comunemente utilizzate nella letteratura scientifica:

1. Rabbia
2. Disgusto
3. Paura
4. Felicità
5. Neutralità
6. Tristezza
7. Stupore

Il riconoscimento delle espressioni facciali trova applicazione in numerosi contesti, quali l'interazione uomo-macchina, la sorveglianza intelligente, il monitoraggio del benessere emotivo, l'educazione personalizzata e l'assistenza sanitaria. Tuttavia, il compito è reso difficile da fattori come la variabilità fisionomica tra individui, le condizioni di illuminazione, le occlusioni parziali del viso e le diverse intensità con cui le emozioni vengono espresse.

In questo progetto, tale task viene affrontato come un problema di classificazione supervisionata, in cui un modello viene addestrato a riconoscere le emozioni in input visivi a partire da un dataset annotato.

2. Dataset e Preprocessing

Per l'addestramento e la valutazione del modello è stato utilizzato un sottoinsieme del dataset FER2013 (Facial Expression Recognition 2013), uno dei benchmark più diffusi nella letteratura scientifica per il riconoscimento automatico delle espressioni facciali. Il dataset completo è stato originariamente rilasciato nell'ambito della competizione Kaggle "Challenges in Representation Learning: Facial Expression Recognition Challenge", e contiene in totale 35.887 immagini in scala di grigi con dimensioni 48×48 pixel, suddivise in:

- 28.709 esempi per l'addestramento (training set)
- 3.589 esempi per il test pubblico (test set)
- 3.589 esempi per il test privato (non utilizzati nel presente progetto)

Ogni immagine ritrae un volto umano ripreso in condizioni naturali, con variazioni dovute a soggetti differenti, orientamenti del volto, condizioni di luce e intensità emotive. Le immagini sono state automaticamente allineate, in modo da centrare approssimativamente il volto e garantire che esso occupi una porzione coerente dell'immagine.

Ciascun esempio è etichettato con una delle sette emozioni di base, così codificate:

- 0 = Rabbia
- 1 = Disgusto

- 2 = Paura
- 3 = Felicità
- 4 = Tristezza
- 5 = Sorpresa
- 6 = Neutralità

Per motivi computazionali, è stato selezionato un sottoinsieme di 7.879 immagini, estratto casualmente dal training set. Questa scelta è stata dettata dalla necessità di ridurre significativamente i tempi di addestramento e di test, rendendo l'elaborazione più gestibile con risorse hardware limitate. Nello specifico, si è osservato un abbattimento dei tempi di training di circa un ordine di grandezza rispetto all'utilizzo dell'intero dataset, senza compromettere in modo sostanziale la validità dei risultati sperimentali.

Il dataset originale è diviso in:

- **Training set**: 80% delle immagini
- **Test set**: 20% delle immagini

Nel nostro caso invece è stato suddiviso nel seguente modo:

- **Training set**: 60% delle immagini
- **Validation set:** 10% delle immagini
- **Test set**: 30% delle immagini

Durante la fase di preprocessing, le immagini sono state trasformate per renderle compatibili con l'input atteso dal modello convoluzionale pre-addestrato utilizzato nel progetto (tramite tecniche di *transfer learning*). In particolare, è stata applicata la seguente sequenza di trasformazioni:

- **Ridimensionamento**: ogni immagine è stata ridimensionata in modo che il lato più corto fosse pari a 256 pixel, mantenendo il rapporto d'aspetto originale;

- **Ritaglio casuale (Random Crop)**: dal ridimensionamento si ottiene una regione casuale di dimensioni **224×224 pixel**, selezionata per introdurre variabilità spaziale e migliorare la capacità di generalizzazione del modello;

- **Specchiamento orizzontale (Random Horizontal Flip)**: con probabilità 0.5, le immagini vengono riflesse orizzontalmente per simulare pose differenti del volto;

- **Conversione a tensore**: le immagini, originariamente in formato PIL, vengono convertite in tensori PyTorch normalizzati;

- **Normalizzazione**: i valori dei pixel vengono normalizzati utilizzando media e deviazione standard dei canali RGB di ImageNet: **media = [0.485, 0.456, 0.406]**, **deviazione standard = [0.229, 0.224, 0.225]**.

Quest'ultimo passaggio è fondamentale quando si adotta un modello pre-addestrato su ImageNet, poiché consente di allineare le statistiche dei dati in input a quelle del dataset originale, preservando la validità dei pesi già appresi e migliorando la convergenza durante il fine-tuning.

La stessa pipeline, con l'unica differenza dell'uso del Center Crop al posto del Random Crop, è stata applicata anche al test set per garantire coerenza e replicabilità nella fase di valutazione.

3. Scelta del modello e training

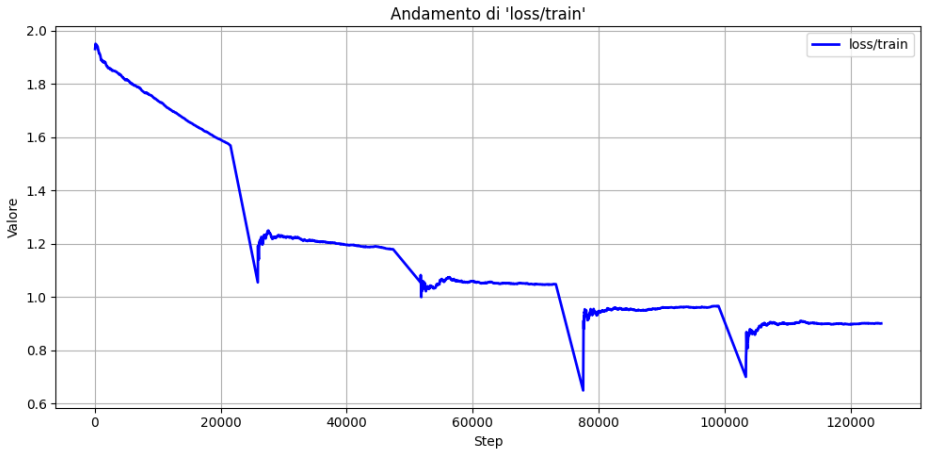
Per la scelta del modello si è optato di fare training su più modelli e fare benchmarking per vedere quale desse i risultati migliori. I modelli utilizzati sono:

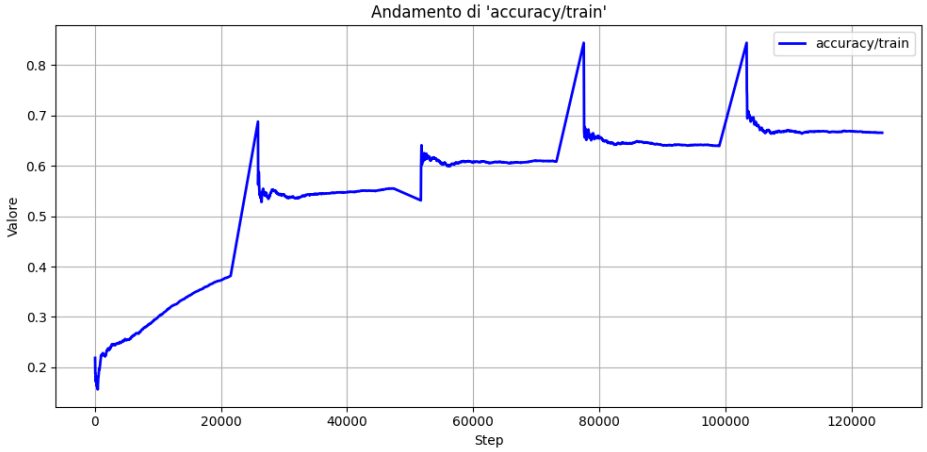
- **EfficientNet_b1**: questo modello presenta un numero di parametri non troppo elevato nonostante riesca a ottenere performance superiori a modelli molto più elaborati.
- **SqueezeNet1_0**: ha un numero di parametri estremamente basso, nonostante ciò, ottiene un accuracy buona. Dunque, considerando i tempi che ha impiegato efficientNet per il training, squeezenet è stata un buon compromesso.
- **ResNet18**: la famiglia dei modelli ResNet è molto efficiente in termini di accuratezza. Inoltre, la versione ResNet18 è molto compatta rispetto ai suoi simili, rendendola perfetta considerando i tempi di training e i risultati.

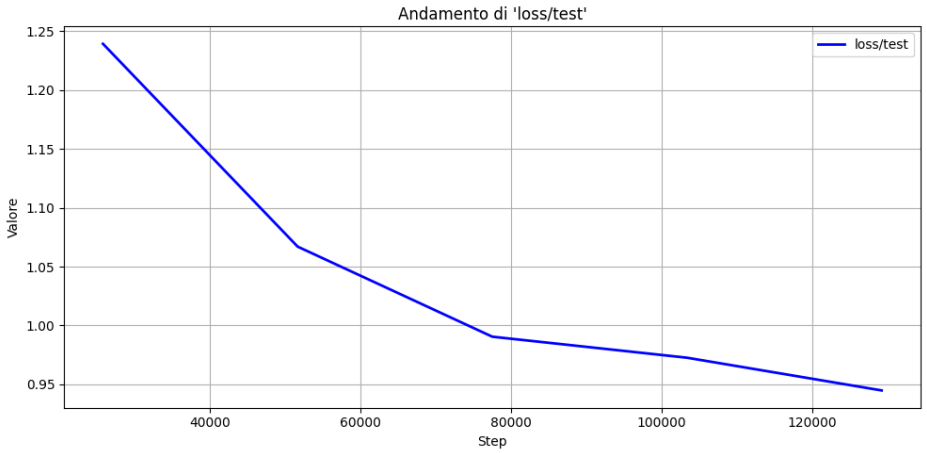
Il training ha richiesto tempo notevole per tutti e tre i modelli, questo aumentato dal fatto che quando la macchina su cui è stato fatto l'addestramento entrava in stand by il training veniva momentaneamente sospeso.

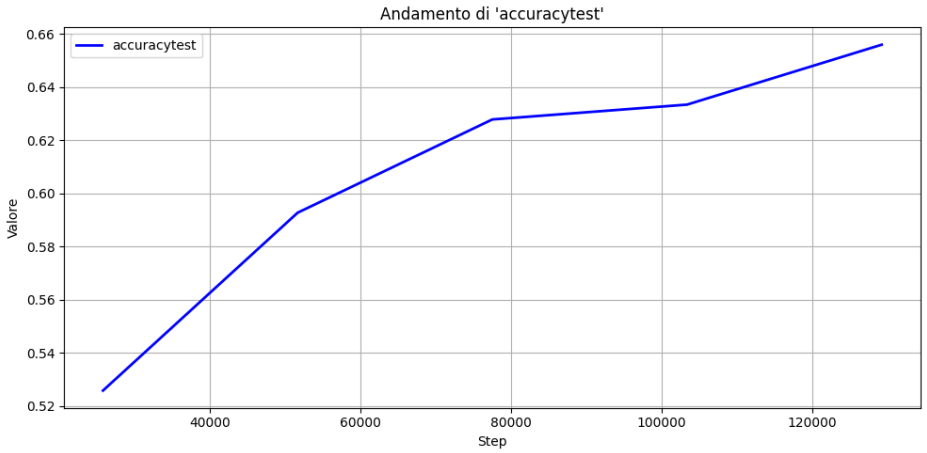
Di seguito sono presentati i grafici relativi a loss e accuracy ottenuti dopo il training di ogni modello:

EfficientNet

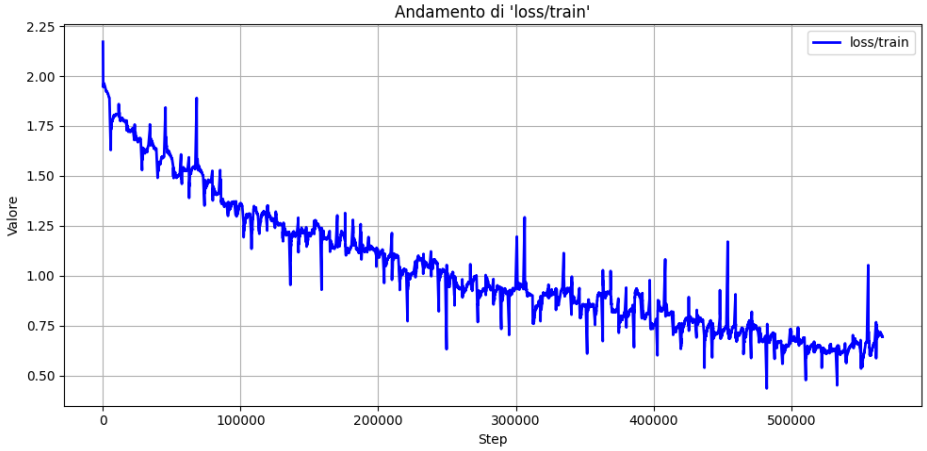


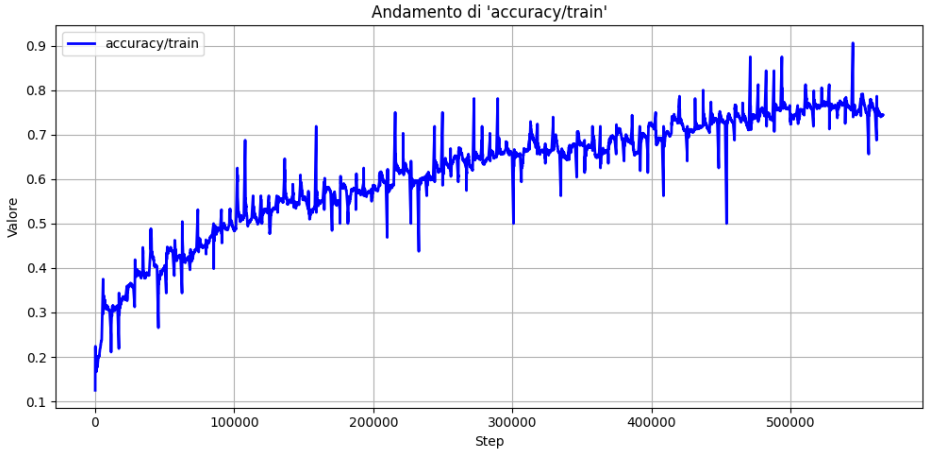

Andamento di 'accuracy/train'
accuracy/train
Valore
Step
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
20000
40000
60000
80000
100000
120000

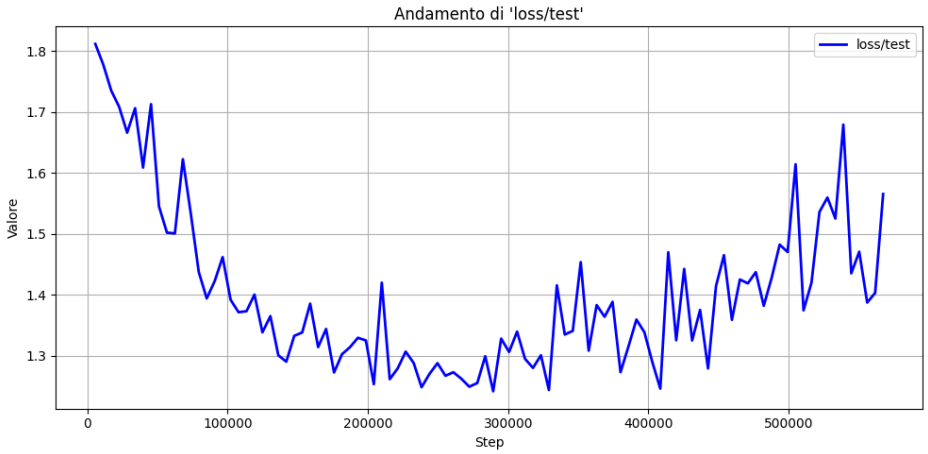

Andamento di 'loss/test'
loss/test
Valore
Step
0.95
1.00
1.05
1.10
1.15
1.20
1.25
40000
60000
80000
100000
120000

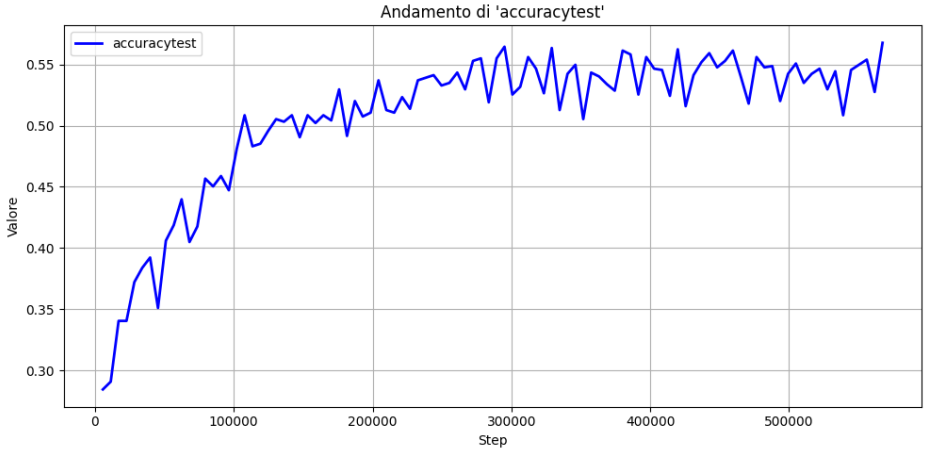

Andamento di 'accuracytest'
accuracytest
Valore
Step
0.52
0.54
0.56
0.58
0.60
0.62
0.64
0.66
40000
60000
80000
100000
120000

SqueezeNet

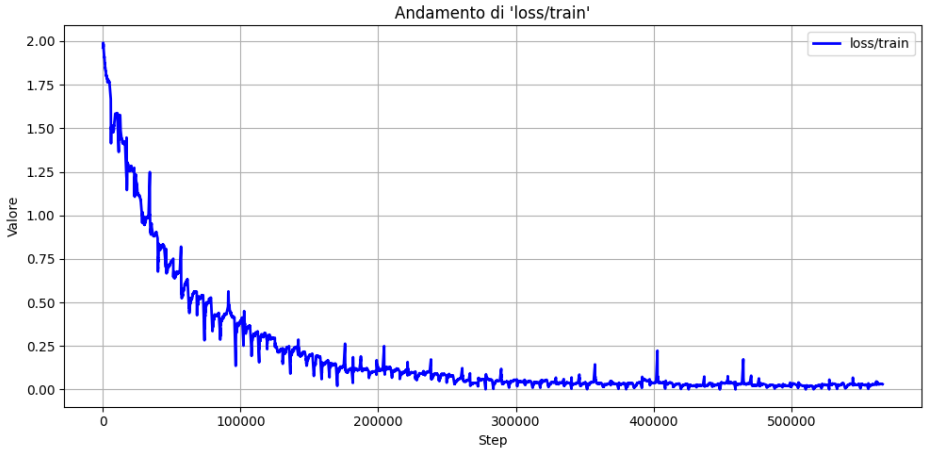

Andamento di 'loss/train'
loss/train
Valore
Step
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
100000
200000
300000
400000
500000

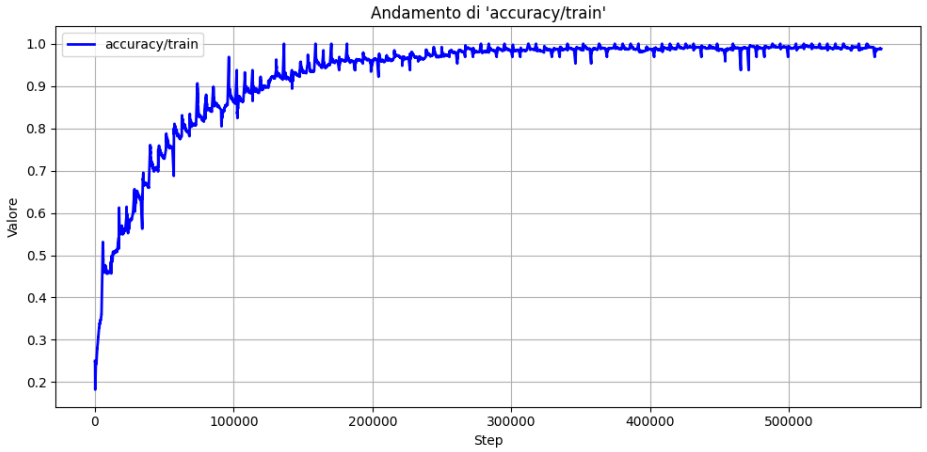

Andamento di 'accuracy/train'
accuracy/train
Valore
Step
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100000
200000
300000
400000
500000

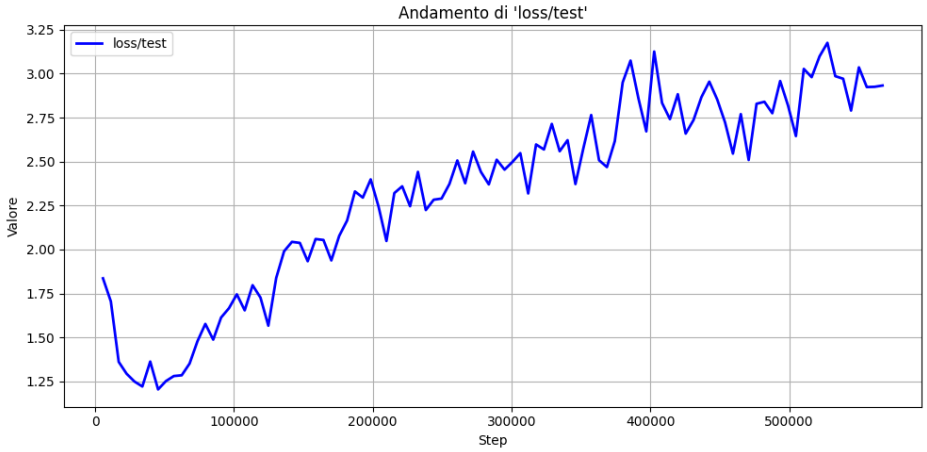

Andamento di 'loss/test'
loss/test
Valore
Step
1.8
1.7
1.6
1.5
1.4
1.3
0
100000
200000
300000
400000
500000

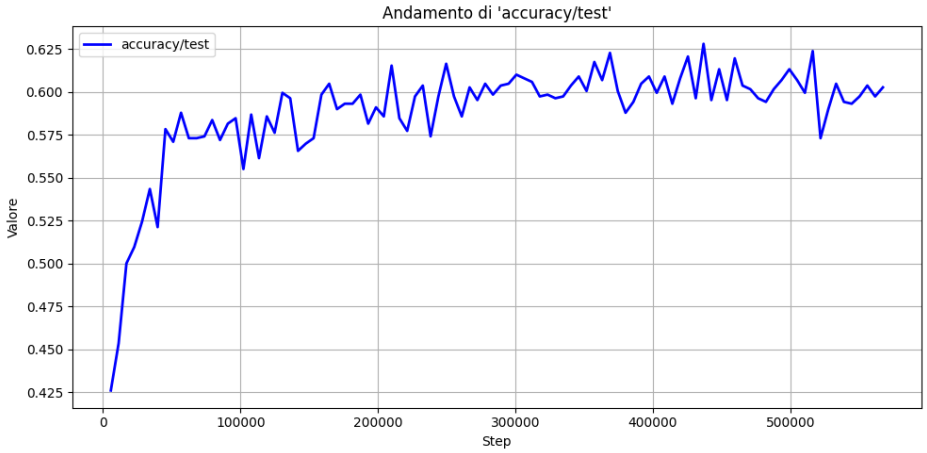

Andamento di 'accuracytest'
accuracytest
Valore
Step
0.55
0.50
0.45
0.40
0.35
0.30
0
100000
200000
300000
400000
500000

ResNet


Andamento di 'loss/train'
loss/train
Valore
Step
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
100000
200000
300000
400000
500000


Andamento di 'accuracy/train'
accuracy/train
Valore
Step
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
100000
200000
300000
400000
500000





Una volta fatto testing su tutti e tre i modelli è stato possibile confrontarli per i seguenti parametri:

- Accuracy nel training

  È la percentuale di esempi correttamente classificati dal modello sui dati di training. Indica quanto bene il modello ha appreso i dati con cui è stato addestrato

- Accuracy nel testing

  È la percentuale di esempi correttamente classificati dal modello sui dati di test, cioè dati mai visti durante l'addestramento. È un buon indicatore della capacità del modello di generalizzare a nuovi dati

- Loss di training

È una misura dell'**errore** del modello sui dati di training

- Precision

Indica la proporzione di true positive tra tutte le predizioni positive del modello

- Recall

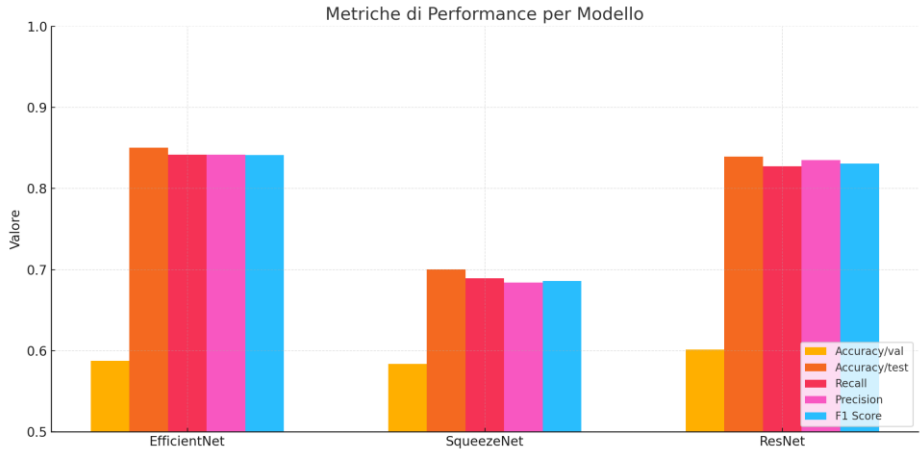
Indica la proporzione di true positive identificati correttamente dal modello rispetto a tutti i veri casi positivi esistenti

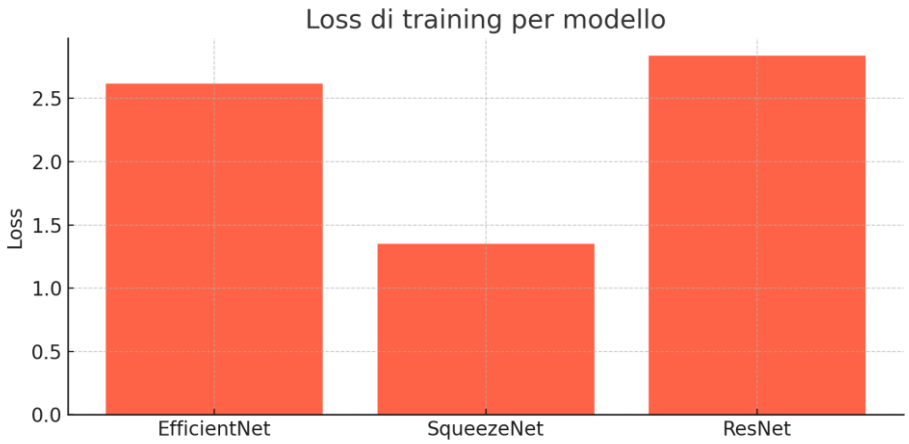
- F1_score

è l'unione della recall e della precision in un unico valore tramite una media armonica

Risultati:

I risultati ottenuti dal confronto sono riportati su questa tabella:





Da questo grafo si può dedurre il seguente per ogni modello:

EfficientNet

- Ha la miglior accuracy in test (0.843)
- F1 score (0.8355) e precision (0.8338) molto alte, ne segue che le predizioni sono bilanciate e coerenti
- Non è il migliore in validazione (0.5877), ma generalizza molto bene sul test set

SqueezeNet

- Ha la miglior accuracy in test (0.843)
- F1 score (0.8355) e precision (0.8338) molto alte; anche qui le predizioni sono bilanciate e coerenti
- Non è il migliore in validazione (0.5877), ma generalizza molto bene sul test set
- Quindi è il modello più affidabile in test

ResNet

- Molto vicino a EfficientNet su tutte le metriche in test: accuracy 0.8359, F1 score 0.8303, precision 0.834
- Ha la miglior accuracy in validazione (0.6015)
- Possiamo dedurne che è molto competitivo, con buona coerenza tra validazione e test

In conclusione, il modello con risultati migliori è EfficientNet, quindi quello più gettonato nell'uso per la demo.

4. Demo

Come già detto il modello utilizzato per la demo è efficientNet dato che ha riportato i risultati migliori.

La demo consiste in un programma python al cui avvio aprirà un'interfaccia con un menù a tendina, all'interno del quale sarà possibile scegliere tra diverse opzioni, tra cui: demo interattiva o una delle sette emozioni classificabili dal modello.

Se si sceglierà demo interattiva, si aprirà la videocamera del dispositivo grazie a una libreria python. Nel momento in cui si è pronti per scattare la foto, premendo il tasto invio, verrà catturata l'immagine e questa etichettata col nome di una possibile classe.

Nel caso in cui si scegliesse una qualsiasi delle altre sette opzioni disponibili, verrà presa una specifica immagine da una cartella, "immaginiDemo" ad esempio, per poi venire classificata.

In entrambi i casi l'output apparirà nel prompt dal quale verrà eseguito il codice python.

Purtroppo per una mancanza di risorse il training è stato fatto su pochi dati, ciò comporta che la demo non funzioni nel 100% dei casi, soprattutto per quel che riguarda la sua versione interattiva.

5. Codice

La parte di training e testing è stata scritta in un notebook, questo è suddiviso nel seguente modo:

- Dataset e preprocessing

Viene caricato il dataset FER2013 da CSV. È presente una classe CSVImageDataSet che legge le immagini codificate in formato CSV

Alle immagini del dataset vengono applicate delle trasformazioni per poter essere adattate al training come normalizzazione, ridimensionamento, ecc.

Infine, viene fatto benchmarking per verificare quale batch size può dare risultati migliori.

- Definizione dei modelli

Vengono definiti e caricati i modelli sui quali verrà fatto transfer learning: EfficientNet, SqueezeNet e ResNet.

- Funzione di training

Questa parte gestisce:

-Cicli di training e validation

-Calcolo di loss e accuratezza

-Logging dei risultati su TensorBoard

-Salvataggio dei pesi a ogni epoca (`weights/`)

Usa la CrossEntropyLoss come funzione di Loss.

Viene salvato a ogni epoch il tempo impiegato a compiere tutte le iterazioni, tuttavia questo potrebbe essere alterato per le ragioni menzionate precedentemente.

- Testing

Valuta il modello su dati mai visti. Restituisce predizioni e etichette reali e suklla base di queste calcola le diverse metriche.

- Analisi dei risultati

Vengono raccolte le metriche ottenute dai vari modelli e inseriti in tabella per confrontare quest'ultimi.

Per far funzionare il codice è necessario salvare il dataset in una cartella FER-2013, all'interno della quale le immagini saranno divise a loro volta in delle cartelle che rappresentano ognuna una classe.

Appena è tutto pronto basterà eseguire le celle in ordine e i dati verranno salvati nei file di log, inoltre i parametri dei modelli a ogni epoch verranno salvati sul file "weights", tramite i quali sarà possibile utilizzare i modelli addestrati in seguito.

La demo invece è stata scritta in un file a parte suddiviso in quattro macro-sezioni:

- Interfaccia utene

Viene creata una semplice GUI tramite la libreria tkinter. In questa interfaccia l'utente può scegliere in un menù a tendina tra diverse opzioni per poi premere sul tasto conferma.

- Caricamento del modello

Viene caricato il modello che verrà utilizzato per il funzionamento della demo, nel nostro caso EfficientNet_b1, caricando i pesi dal file ottenuto dopo il training. Dopodiché vengono applicate le trasformazioni che sono state utilizzate per il testing all'immagine.

- Definizione delle funzionalità

Qui vengono definite le funzionalità della demo e come agiscono. É un po' il cuore pulsante del codice.

- Gestione della scelta utente

Piccolo pezzo di codice in cui viene gestita la scelta fatta nell'interfaccia.

In questo caso per far funzionare correttamente il codice è necessario, oltre ad avere una webcam funzionante, la cartella dei pesi, nuovamente la cartella del dataset ma solo per ricavarne le classi e la cartella immaginiDemo dove sono presenti le immagini predefinite della demo.

6. Conclusioni

Il presente progetto ha affrontato il problema del riconoscimento delle espressioni facciali (Facial Expression Recognition) utilizzando tecniche di machine learning supervisionato e, in particolare, modelli di deep learning basati su reti neurali convoluzionali. L'obiettivo principale era sviluppare un sistema in grado di associare, a partire da un'immagine del volto di un soggetto, un'etichetta

corrispondente a una tra sette emozioni fondamentali: rabbia, disgusto, paura, felicità, tristezza, sorpresa e neutralità.

Durante lo sviluppo del progetto è stato utilizzato un sottoinsieme del dataset pubblico FER2013, composto da 2500 immagini in scala di grigi, selezionate per ridurre i tempi di addestramento e adattare il sistema a risorse computazionali limitate. Il dataset è stato preprocessato attraverso una pipeline di trasformazioni standard nel contesto del *transfer learning*, tra cui ridimensionamento, ritaglio, normalizzazione e data augmentation. È stato inoltre implementato un sistema completo per l'addestramento, la valutazione e la dimostrazione del modello, includendo una **demo interattiva** e una **relazione dettagliata** sulle scelte progettuali effettuate.

7. Lezioni apprese

Il progetto ha rappresentato un'importante occasione per mettere in pratica le conoscenze teoriche e pratiche acquisite durante il corso. Tra i concetti affrontati e concretamente applicati si possono evidenziare:

- La rappresentazione dei dati come vettori numerici e immagini normalizzate, in linea con quanto discusso nelle prime lezioni teoriche.
- L'overfitting e le tecniche di regolarizzazione, comprese attraverso l'analisi delle performance del modello su training e test set.
- Il concetto di transfer learning, messo in atto normalizzando i dati con i parametri di ImageNet e impiegando architetture pre-addestrate.
- L'importanza della valutazione del classificatore, mediante metriche come l'accuratezza e la matrice di confusione, e la gestione bilanciata del dataset.

Questa esperienza ha permesso di comprendere in maniera approfondita le dinamiche reali dello sviluppo di modelli di machine learning, la necessità di

adottare soluzioni computazionalmente sostenibili, e l'importanza di un'attenta fase di preprocessing e tuning dei parametri.

8. Possibili miglioramenti futuri

Sebbene i risultati ottenuti siano soddisfacenti, il progetto presenta margini di miglioramento. In particolare, si potrebbe:

1. **Utilizzare l'intero dataset FER2013**, o una sua porzione più ampia, per migliorare la capacità generalizzativa del modello e ridurre il rischio di underfitting.
2. **Effettuare un'analisi più approfondita della distribuzione delle classi**, bilanciando eventualmente il dataset tramite tecniche di oversampling o weighted loss.
3. **Ottimizzare ulteriormente i parametri del modello**

In conclusione, il progetto ha rappresentato un'esperienza altamente formativa, utile non solo per consolidare le competenze tecniche acquisite, ma anche per sviluppare un approccio critico nella progettazione di soluzioni basate su machine learning. L'integrazione di teoria e pratica proposta dal corso si è rivelata estremamente efficace nel guidare tutte le fasi dello sviluppo, dalla definizione del problema fino alla realizzazione di una demo funzionante.