Visual Assistant for Image Editing

Federico Maresca, Mattia Manfredini, Francesco Giacalone

February 2022

Indice

1 Introduzione e Obiettivi

In questo progetto usiamo una rete neurale Tensorflow [1] allenata su un dataset da noi generato e OpenCV per modificare immagini attraverso tecniche di image processing. Le modifiche alle immagini vengono fatte tramite il riconoscimento di gesti da webcam attraverso la suddetta rete neurale. Il progetto è disponibile sul nostro github assieme a istruzioni di utilizzo.

Gesti Disponibili:

- Più
- Meno
- Conferma
- Rotazione
- Saturazione
- Luminosità
- Contrasto
- Visualizza
- Filtri
- Esci

Le modifiche alle immagini sono descritte nella sezione 5

1.1 File di Progetto

I file **requirements.txt** e **setup.py** sono file usati per installare i package necessari per python per far funzionare il programma.
GuiImages contiene le immagini usate per l'interfaccia utente, come immagini di feedback per gesture riconosciute.
Images è la cartella default dove il programma ,in caso non venga definito un path da linea di comando, cerca le immagini da modificare.
SSD_Network contiene i file della rete neurale in formato tflite abbiamo deciso di usare questo formato per la sua minore dimensione e maggiore velocità di inferenza.
Sample videos sono gif usate del readme.md su github per dare esempi di utilizzo.
Modules contiene i 4 moduli principali oltre il main.

2 Overview dei moduli del programma

Il programma si divide in 5 moduli principali:

- main
- function
- menu
- capture
- object_detector

2.1 Main

Il ruolo del main è la raccolta degli argomenti di linea di comando e la gestione generale dei thread del processo. Inizializza la coda di gesti, la classe oggetto menu, fa partire il thread secondario di capture (a cui passa l'oggetto menu per poter ottenere le immagini con le quali crea l'interfaccia) e infine inizializza la classe oggetto menu. Alla fine del quando l'utente chiude chiama capture.stop() per fermare il thread capture.

2.2 Functions

Questo modulo contiene le funzioni di modifica delle immagini e la classe Gesture Queue che è una coda thread safe usata come mezzo di comunicazione dei gesti tra il thread del detector e il thread del menù.

2.3 Menu

Questo modulo contiene la classe menù che gestisce il menu del programma, comunica con il modulo object detector attraverso gestureQueue aspettando che un gesto diventi disponibile per poi catturarlo e usarlo.

2.4 Capture

Capture gestisce sia la cattura dei frame dalla webcam che la creazione dell'interfaccia. A ogni frame chiama la funzione detectionW in object detector e riceve il frame che, in caso di gesto riconosciuto, ha sopra disegnato la bounding box con il nome del gesto e la probabilità associata. Usa poi il frame per aggiornare la finestra del programma.

2.5 Object_detector

Questo modulo contiene la classe object detector. Il file si basa sul modulo presente in questo github di esempi tensorflow. Con le modifiche da noi effettuate l'abbiamo reso compatibile con il programma multithreaded (per poter sfruttare la coda di gesti) e la rete neurale che abbiamo usato. La funzione detectionW è stata aggiunta come wrapper per aggiungere tre funzionalità principali:

- stabilizzazione dei gesti riconosciuti
- inserimento nella coda dei gesti dei gesti riconosciuti
- ritorno del frame modificato al modulo capture.py

3 Rete Neurale

Uno dei punti chiave per il nostro progetto è stata quella di addestrare una rete neurale artificiale in grado di riconoscere gesti predefiniti per usarli come comandi.
Si tratta di un task di **Object Detection**. Nel nostro caso le immagini su cui si effettuerà l'object detection saranno i singoli frame video ripresi dalla webcam.

3.1 Scelta del modello

Per il nostro progetto siamo partiti da un modello pre-allenato, ovvero da un modello che è stato precedentemente addestrato per un task di Object Detection su un dataset molto grande. L'utilizzo di un modello pre-allenato per task di Object Detection è spesso l'approccio più utilizzato in quanto non sempre si dispone di un dataset abbastanza grande da permettere al modello di generalizzare opportunamente, inoltre i parametri sono già ottimizzati e il training risulterà più veloce.

Si effettuerà quindi **transfer learning**: sfrutteremo ciò che ha imparato il modello e utilizzeremo tale "conoscenza" per risolvere il nostro specifico obiettivo. Servirà però applicare le opportune modifiche al modello (che avrà un dataset diverso, un numero diverso di classi etc..). Possiamo trovare diversi modelli pre-addestrati sul github di tensorflow.

Tra i vari modelli abbiamo optato per "SSD MobileNet v2 320x320", che presenta un'ottima velocità e che, per il nostro task, ci ha consentito di arrivare ad un riconoscimento dei gesti soddisfacente.

3.2 Dataset

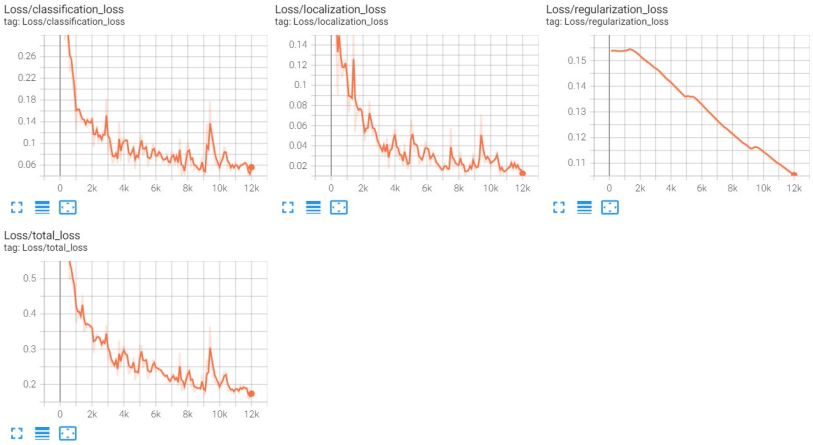
Per la creazione del dataset, abbiamo raccolto personalmente diverse immagini tramite webcam, per ogni relativo gesto e abbiamo etichettato tali immagini utilizzando il programma **LabelImg**, che permette di evidenziare un box (label detection) contenente, nel nostro caso, soltanto il gesto. Questo processo creerà delle annotations (.XML) che contengono informazioni (e.g. dimensioni binding box, label) necessarie per il training. Per aiutare il modello a generalizzare abbiamo scattato fotografie con angoli ed illuminazione diversa, oltre che utilizzare le fotografie da tutti i membri del gruppo.
Inizialmente abbiamo utilizzato una trentina di immagini per gesto, per un totale di 300 immagini. Non avendo raggiunto la accuratezza necessaria in alcuni gesti che apparivano simili tra di loro abbiamo aumentato il dataset ad un totale di 1000 immagini, suddivise in circa 90% per il **training set**, 10% per il **test set**.
Prima di iniziare la raccolta delle immagini abbiamo esplorato la possibilità di effettuare dell'image processing per rendere il compito della rete più semplice. In particolare abbiamo provato ad applicare tecniche di skin thresholding presentate nei paper [2]e [3]. Se applicate con successo queste tecniche ci avrebbero permesso di inviare alla rete delle immagini bianco/nero allegerendo il compito di riconoscimento grazie alla ridotta complessità delle immagini eliminando informazioni ridondanti come lo sfondo o la presenza di falsi positivi. Purtroppo non siamo riusciti a applicare una tecnica che generalizzasse abbastanza bene il compito di escludere solo la pelle dal frame della webcam sotto ogni forma di illuminazione.

3.3 Training

Dopo aver creato i label abbiamo cominciato ad addestrare la rete. Sono stati effettuati diversi training, agendo di volta in volta sugli iper-parametri (parametri che determinano in che modo la rete viene allenata) al fine di migliorare il risultato.

La **loss** è l'errore (o la differenza) tra ciò che la rete predice rispetto alla true label dell'immagine. L'algoritmo di ottimizzazione di una rete neurale cerca di minimizzare tale valore in modo tale che le predizioni siano più accurate possibili. Alla fine di ogni epoca di training, la loss viene calcolata e poi accumulata tra tutti gli output (definendo quindi la cost function).

Il grafico mostrato successivamente mostra il decadimento della loss nel corso dell'addestramento:



4 Interfaccia Grafica

L'interfaccia grafica con cui si presenta l'applicazione è composta da due sezioni principali: superiore e inferiore.

La **sezione inferiore** è usata per visualizzare l'immagine che si vuole modificare , applicare le modifiche e visualizzarle , senza che l'immagine originale venga modificata nella sua sede in memoria. Solo dopo aver effettuato il gesto di Conferma, l'immagine originale verrà sovrascritta con quella nuova.

La **sezione superiore** invece è composta da 3 aree, una in cui si visualizza il flusso video raccolto della webcam, dopo aver applicato la rete neurale, in modo da far capire all'utente se ha eseguito il gesto corretto e con quale percentuale di correttezza, al fine di fargli apprendere in maniera autonoma come eseguirlo nel migliore dei modi.

Le altre due aree hanno lo scopo di segnalare se è stato eseguito un gesto corretto/errato o se si è in attesa di un gesto, mentre l'ultima funge da menù indicativo , indicando I possibili gesti che possono essere eseguiti , ed in quale area del menù ci si trova.

4.1 Uso dei gesti

Nel menù principale è possibile effettuare il gesto **Visualizza** per visualizzare l'elenco dei gesti disponibili. E' possibile inoltre eseguire il gesto **Esci** per uscire dall'applicazione. Tramite I gesti **Più** e **Meno** è possibile cambiare l'immagine corrente. Una volta selezionato un gesto di modifica dell'immagine si entrerà nel rispettivo menù dove si potranno utilizzare I gesti **Più** e **Meno** per aumentare o diminuire l'effetto di modifica o, nel caso dei filtri, decidere il filtro da utilizzare e il gesto **Conferma** per salvare l'immagine e tornare al menù principale.

5 Funzioni di modifica immagine

5.1 Saturazione

Nel caso della **Saturazione**, si converte l'immagine da BGR a HSV, si dividono i 3 canali, e si lavora solo sul canale s per gestire accuratamente la saturazione. Se si vuole aumentare la saturazione , si moltiplica s per un valore standard, definito a priori, se si vuole diminuire la saturazione si divide s per lo stesso valore. Dopodichè, per visualizzare l'mmagine correttamente la si riconverte in BGR.

5.2 Luminosità

Procedimento simile si applica nel caso della **Luminosità**, dove si somma o si sottrae (aumentare/diminuire luminosità) un valore standard alla luminosità attuale del pixel, tenendo però presente che la somma con il nuovo valore non superi 255 , perchè altrimenti il valore del pixel ripartirebbe da zero, e si avrebbero quindi pixel più scuri rispetto all'immagine originale. Stesso discorso nel caso in cui si volesse diminuire la luminsotià, ma facendo attenzione che la sottrazione tra luminosità del pixel e valore standard non sia inferiore a zero, altrimenti in questo caso , il restante valore da sottrare verrà sottratto da 255 e si avrà un pixel più luminoso del precedente invece che più scuro.

5.3 Contrasto

Il **Contrasto** lavora invece con una funzione di OpenCV, ConvertScaleAbs, alla quale passiamo due valore diversi di alpha nel caso in cui si voglia aumentare o diminuire il contrasto.

5.4 Rotazione

La funzione di **Rotazione** permette di ruotare di 90 gradi , da destra o da sinistra l'immagine, usando la funziona di OpenCV, warpAffine, che permette di applicare una trasformazione affine all'immagine, previa passaggio di una matrice di trasformazione , che ci procuriamo mediante la funzione openCV , getRotationMatrix2D , e l'uso di funzioni e calcoli matematici basati su seno e coseno.

5.5 Filtri

Per quanto riguarda la gestione dei **Filtri** , una volta entrati nella suddetta area si appica il filtro base che ci restituisce l'immagine originale , e con il gesto di più e meno si può scegliere che filtro applicare. IL filtro **Normal** serve per tornare all'immagine originale nel caso non si vogliano più applicare i filtri.
Gli altri filtri invece sono realizzati in vari modi, alcuni come **Cartoon** e **HDR** , sfruttano direttamente funzioni implementate in openCV, rispetivamente stylization e detailEnhance.

Blur ed **Emboss** , applicano una matrice di convoluzione all'immagine originale, il kernel è fisso per ogni funzione e definito nella stessa, per poi essere applicato all'immagine mediante la funzione openCV filter2D.

Anche il filtro **Seppia** usa un kernel di convoluzione, applicato però tramite la funzione transform , preceduta da una conversione dei pixel da interi 8bit a float 64bit, per evitare perdite ed overflow, a cui poi infine è applicata una normalizzazione per eliminare i valori superiori a 255 , assegnandogli suddetto valore , ed infine un riconversione da float 64bit a interi 8bit, per poter visualizzare correttamente l'immagine.

Infine il filtro **Invert** separa i canali BGR in tre variabili diverse , che vengono poi usate per cambiare l'ordine dei canali dell'immagine principale , quindi se l'immagine originale ha ordine BGR , ci sono 5 possibili diverse combinazioni per il filtro invert (RGB , RBG, BRG, GRB,GBR). Una volta selezionato il filtro invert , dunque, I comandi + e – non serviranno per aumentare/diminuire intensità, ma per cambiare le varie combinazioni dei 3 canali.

Riferimenti bibliografici

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] T. J. McBride, N. Vandayar, and K. J. Nixon. A comparison of skin detection algorithms for hand gesture recognition. In *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, pages 211–216, 2019.

[3] P. Thwe and M. Yu. Analysis on skin colour model using adaptive threshold values for hand segmentation. *International Journal of Image, Graphics and Signal Processing*, 11:25–33, 08 2019.