Modello NN 2+1D

IL DATASET

Il dataset (preso da Keggle) su cui è stato trainato un modello è composto di circa 600 video strutturati come 30 immagini di dimensioni quadrate e 100 video per il test set. Il vantaggio sta nella quantià di dati su cui si puo trainare il modello. Dall’altra parte non c’è troppa varietà per quanto riguarda persone o luoghi in cui sono state scattate le immagini.

GLI INPUT AL MODELLO

Poiché nella nostra applicazione di base abbiamo un video come dato, una funzione fondamentale per il riconoscimento della gesture è stata “load\_most\_moving\_frames\_center\_crop”. Il suo scopo è selezionare automaticamente dai video registrati un set di 30 frame che catturano la parte del video con più movimento. Questo consente di isolare le parti del video più rilevanti per l'analisi del gesto, eliminando porzioni piu statiche. Inoltre viene fatto un crop su ogni frame tenendo solamente il quadrato centrale e scartando le fasce sopra e sotto.

Tutto cio è stato fatto sfruttando la libreria OpenCV con lo scopo di estrarre i frame dai video nel modo più simile possibile al dataset su cui il modello è stato trainato.

<Immagine: grafico movimento->video->30 frame->crop centrale>

Immagine che contiene testo, linea, Diagramma, diagramma

Il contenuto generato dall'IA potrebbe non essere corretto.

IL MODELLO

Il modello è una rete neurale convoluzionale 3D pensata per riconoscere gesti a partire da video.

Ogni input è una sequenza di 30 frame RGB, e l'obiettivo è classificare il gesto presente nel video, sfruttando sia le informazioni spaziali (nei singoli frame) che temporali (nel movimento tra i frame).

Nel modello Vengono usate le ResNet ovvero una particolare architettura per reti neurali utile nel caso di CNN profonde.

3D CNN e (2+1)D CNN

Per creare il un modello che potesse analizzare video, quindi una sequenza di frame, è stata usata una rete convoluzionale. Di solito una CNN viene usata per classificare immagini singole e quindi viene fatta una interpolazione 2D. Per i video è diverso poiche presentano un concetto in piu: sequenza temporale di frame. Per questo serve un kernel 3D che è capace non solo di interpretare frame singoli ma anche una sequenza di essi. Infatti, a differenza di un filtro 2D, il filtro 3D è capace di muoversi anche nella dimensione temporale.

2D altezza larghezza -> classifica immagini

3D altezza larghezza tempo -> classifica video

Percio se immaginiamo di avere un set di 30 frame uno affianco al altro creerebbero un volume (array) di dimensioni: (30, altezza, larghezza). Perciò un filtro 3D funzionerebbe esattamente come uno 2D.

Le reti convoluzionali tridimensionali (3D CNN) estendono il concetto delle CNN 2D alle sequenze video, aggiungendo la dimensione temporale al processo di convoluzione. Mentre nelle CNN 2D i filtri operano su altezza e larghezza di un’immagine per estrarne caratteristiche spaziali, le CNN 3D applicano filtri volumetrici di forma (T × H × W), dove T è la profondità temporale, per analizzare blocchi di più fotogrammi consecutivi. Questo permette al modello non solo di riconoscere strutture spaziali all’interno di ogni singolo frame, ma anche di catturare dinamiche e movimenti nel tempo, fondamentali per comprendere il contenuto di un video. L’input viene trattato come un volume tridimensionale, ad esempio con forma (30, altezza, larghezza), dove 30 rappresenta il numero di frame. Un kernel 3D scorre su questo volume e, in ogni posizione, aggrega le informazioni di un piccolo cubo, generando una nuova rappresentazione volumetrica. Sebbene le CNN 3D siano molto efficaci nell’analisi spaziotemporale, esse richiedono un numero elevato di parametri: ad esempio, un filtro 3D di dimensione (3 × 3 × 3) con C canali in ingresso e uscita necessita di 27 · C² pesi. Questo le rende computazionalmente costose.

Per ovviare a questo problema, sono state introdotte le reti convoluzionali (2+1)D, che scompongono la convoluzione 3D in due operazioni separate: una convoluzione spaziale 2D applicata ai singoli frame (1 × 3 × 3), seguita da una convoluzione temporale 1D (3 × 1 × 1) che elabora la sequenza dei frame nel tempo. Questa fattorizzazione riduce significativamente il numero di parametri (ad esempio 12 · C² invece di 27 · C² per un filtro equivalente) e migliora l’efficienza dell’apprendimento, pur mantenendo la capacità di estrarre informazioni sia spaziali che temporali.

RESNET

Le ResNet sono sono architetture di NN progettate per risolvere un grosso problema che emerge quando le reti diventano molto profonde: il degrado delle prestazioni.

Questo è dovuto a:

* Vanishing/exploding gradients
* Difficoltà nel propagare informazioni utili all’indietro (backpropagation)

Una ResNet permette di trasportare l’input lungo tutta la rete, in questo modo si evita che durante la rete venga “scordato” l’input. Infatti, funzioni come la ReLU rischiano di cancellare informazioni (ad esempio valori del neurone minore di 0) utili mentre i residual block del modello risolvono anche questo problema.

Le residual connections risolvono questi problemi creando una scorciatoia che trasmette direttamente l’input di un blocco alla sua uscita, facendo sì che il blocco debba imparare solo la differenza (il residuo) rispetto a ciò che riceve in ingresso.

Immagine che contiene Carattere, schermata, calligrafia, simbolo

Il contenuto generato dall'IA potrebbe non essere corretto. Immagine che contiene Carattere, testo, schermata, Elementi grafici

Il contenuto generato dall'IA potrebbe non essere corretto.

La somma x + f(x) serve a non perdere l’informazione iniziale di x. In pratica ad ogni passo modifichiamo leggermente x con le nuove informazioni apprese dalla convoluzione.

Nel nostro caso la f(x) è proprio la convoluzione 3D spiegata di sopra.

Di solito una rete prende una immagine con 3 canali RGB e ne estrae n (ad esempio 16) feature: bordi verticali, orizzontazi, curve… quindi si fa del pooling e l’immagine (le 16 feature) viene ridotta di dimensioni (questa riduzione di dimensioni simula il catturare feature sempre piu estese e grandi dell’immagine, noi riduciamo l’immagine ma in realtà è come se stessimo rendendo i filtri piu grandi: Aumenta il campo visivo e ogni unità nel layer successivo guarda un’area più ampia dell’immagine originale) e così via. Si passa dai pixel → a bordi → a texture → a parti del viso → a una testa. All’ultimo passo si prendono tutte le feature estratte (teste, zampe…) e con “Flatten + Dense + Softmax”(capire) si decide se è un cane o un gatto. Con l’aumentare dei layer Il gradiente si disperde → l’input originale non viene più "sentito" e La rete diventa difficile da ottimizzare. Invece le reti con residual block funzionano portandosi la rappresentazione x dell’immagine originale lungo tutta la rete e i vari layer potenziano le aree dove trovano certe feature come occhi o orecchie ma mantenendo comunque x che viene ritoccata ad ogni layer. Evitando di alterare completamente l’immagine iniziale come farebbero le calssiche CNN ma modificando l’input pian piano.

Un fattore importante è appunto rimanere costanti con le dimensioni dell’input e output di ogni residual block, insomma x e f(x) devono avere le stesse dimensioni di altezza e larghezza. Altrettanto importante è ridurre la dimensione dei canali e quindi dopo ogni residual block i canali vengono riscalati (tipicamente di un fattore 2) e ridotti di dimensioni, dopodichè si può riapplicare un residual block (che comprende la convoluzione 3D). Questo è il principio che sta alla base del nostro modello.

ARCHITETTURA DEL MODELLO

È un modello di deep learning che usa Keras ed è composto da diversi livelli grazie a Functional API: classi Python che ereditano da keras.layers.Layer che poi vanno a specializzarsi; quindi, vengono assemblate e riutilizzate per comporre il modello finale. Qui un punto fondamentale, che poi verrà approfondito, è il fatto di poter costruire un grafo anzi che il classico modello lineare di keras.sequential. Infatti, c’è la possibilità di mettere i layer in sequenza ma anche in parallelo.

<immagine modello>

PIPELINE MODELLO

In generale la rete neurale sfrutta la combinazione di convoluzioni 2+1D e residual block per apprendere in modo efficiente movimenti complessi, applicando più Resize il che consente di catturare informazioni a diverse scale spaziali.

Immagine che contiene testo, Carattere, ricevuta, documento

Il contenuto generato dall'IA potrebbe non essere corretto.Una pipeline di input al modello serve innanzitutto per inserire l’immagine e passarla attarverso i nodi. Un punto fondamentale è un layer di pre-processing che usa data augmentation per migliorare la generalizzazione del modello. Questo layer applica ad ogni video una variazione di luminosità costante su tutti i frame simulando condizioni di luce diverse. Questa strategia rende il modello meno prono all’overfitting rispetto alle condizioni di luminosità viste durante l’addestramento.

<immagini io che divento chiaro>

Di seguito un singolo layer che applica una convoluzione 2+1D.

Quindi i dati vengono normalizzati e passati ad una ReLu.

Dopo questa fase iniziale ci sono una serie di blocchi con lo scopo di ridurre man mano la risoluzione dei video e nel frattempo inserire blocchi di ResNet come descritto di sopra.

Dopodiché viene fatto un pooling sul tensore finale che aggrega i dati lungo tutte e 3 le dimensioni e ne fa una media. Ogni feature è rappresentato da un unico valore. Infine, c’è un Dense layer semplicemente per la classificazione del gesto.

RISULTATI

Il modello si comporta bene sia durante l’addestramento che nella fase di test. È abbastanza generalizzato e non ci sono segni di overfitting, riesce a predirre le classi in modo eguale senza prediligenre una e raggiunge una una accuracy del 90%.Immagine che contiene schermata, testo, quadrato, Rettangolo

Il contenuto generato dall'IA potrebbe non essere corretto.Immagine che contiene testo, diagramma, schermata, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

Detto ciò, è importante fare una considerazione sul dataset utilizzato: i dati con cui il modello è stato addestrato contengono un gruppo omogeneo di persone e questo limita la capacità del modello di generalizzare quando viene testato su soggetti con caratteristiche differenti. Si può dire che è un overfitting indotto dal dataset. Dai risultati la CNN non sembra essere il problema ed è plausibile aspettarsi risultati migliori con una ampliazione del dataset introducendo appunto diversità nel set di training.