Presentazione

\*saluti\*, il progetto che vorrei esporre oggi riguarda il rilevamento degli audio deepfake e dunque riuscire a distinguerli dagli audio genuini.

Per deepfake si intendono tutti quei contenuti multimediali, audio, foto, video, generati tramite apprendimento profondo, il deep learning

Chiaramente il progresso che si ha avuto nel campo dell’intelligenza artificiale negli ultimi 2-3 decenni ha fatto sì che la creazione di contenuti deepfake sia più facile che mai ed è oggi disponibile al grande pubblico, senza richiedere competenze che vadano oltre il saper navigare in rete e seguire semplici istruzioni, grazie a piattaforme online come Dall-E o StableDiffusion. Anche per gli addetti ai lavori il tutto è molto più semplice grazie a librerie quali Tensorflow che permettono di sviluppare pipeline di machine learning usando API quali Keras, oltre chiaramente a modelli esistenti per la generazione di deepfake

Il primo deepfake definibile tale fu il programma Video Rewrite, 1997, lavoro seminale il cui obiettivo, raggiunto con ottimi risultati, era di riadattare il labiale di una persona in un video per farlo combaciare con una differente traccia audio. Nel 2014 Goodfellow et.al pubblicarono un paper introducendo l’idea delle reti generative a meccanismo avversariale. E come possiamo vedere dal grafico, la quantità di letteratura su questo campo è in aumento.

Il grafico a sinistra invece rappresenta una tipica pipeline per generare deepfake, quindi a partire da una traccia audio e un video, se ne crea una rappresentazione intermedia dopo una fase di preprocessing e la si usa come input ai modelli

Tipicamente la generazione di deepfake passa attraverso l’uso di modelli ad architettura di tipo autoencoder. Un autoencoder è definito da due insiemi finiti, quello dei messaggi codificati e dei messaggi decodificati e da due funzioni che hanno per dominio e codominio uno di questi due insiemi.

Vi è una distribuzione di probabilità sull’insieme dei decodificati e una funzione di misurazione della qualità della ricostruzione del messaggio. Un autoencoder ideale fornisce codice arbitrario al decoder e questo ottiene un messaggio che esiste realmente nella distribuzione di probabilità, generando così messaggi “finti” indistinguibili dai veri.

Ad un autoencoder può poi essere agganciata una rete GAN, avversariale. Le reti di questo tipo presentano due moduli, un generatore e un discriminatore. Il generatore crea contenuti basandosi sulla codifica data dall’encoder e il discriminatore tenta di capire se si tratta di contenuto genuino o generato. Il meccanismo di apprendimento è un gioco a somma zero dove il generatore cerca di far combaciare la distribuzione dei dati reali con quella dei dati generati e il discriminatore di mandare in output un valore prossimo ad 1 per dati rilevati come reali e 0 se rilevati come fake

**\*discorsino sui vantaggi e svantaggi, andare a braccio principalmente\***

I quattro tipi di manipolazione img/video che sono face-swap (scambio di faccia input a faccia target, risultato rigido, perdità di espressività), lip-syncing (si prende un video input e una traccia audio differente e si manipola il video in modo che la regione della bocca combaci con la traccia audio), puppet master (è come il face swap ma si lavora sul target anziché sull’input, trasferendo le caratteristiche facciali dall’input al target) e face synthesis(facce completamente inesistenti nella realtà) e facial attribute manipulation(modifica di una sola regione del volto mantenendo il resto intoccato)

I due tipi di manipolazione audio sono Text To Speech e Voice Conversion

TTS è la tecnica più “antica” con decenni di lavoro alle spalle. Agli inizi era un tipo di tecnica concatenativa, si spezzava il parlato in frammenti da concatenare in un nuovo parlato. Questa tecnica è andata in disuso con l’avvento delle reti neurali ed oggi si fa uso di encoder-decoder e/o GAN. Progetti pionieri nel TTS sono stati, tra gli altri, WaveNet e TacoTron. WaveNet usa il segnale audio in input così com’è per ricavarci feature acustiche quali gli spettrogrammi, passandole in una rete neurale generativa addestrata su parlato reale

TacoTron è un progetto Google che a differenza di WaveNet, il quale addestra ogni suo sottomodulo indipendentemente, è un sistema end-to-end (si parte direttamente dall’input grezzo e si arriva direttamente all’output, in un unico modello, senza la necessità di architetture complesse fatte di sottomoduli diversi per preprocessare, estrarre feature, ottimizzare e poi predire). Sintetizza parlato a partire da coppie testo-audio e ha buoni doti di generalizzazione. E infine arrivò DeepVoice3, anch’esso encoder-decoder con una rete postprocessing convolutiva per predire i parametri finali. Ognuno di questi sistemi ha i suoi limiti: i primi due sono computazionalmente costosi e affamati di dati di training, il terzo generalizza peggio degli altri

La Voice Conversion invece manipola una voce di input per adattarla alla voce desiderata mantenendo il contenuto linguistico del parlato. Può venire usata per il doppiaggio, per assistenza alle persone con difficoltà uditive, assistenti digitali personalizzati. Per fare ciò si ha bisogno di quelle caratteristiche del parlato che sono di alto livello, come il timbro vocale e la prosodia (cioè il ritmo, l’intonazione, l’accento ecc..)

Si iniziò la ricerca cercando modelli che richiedessero dati di training paralleli, ossia si richiedevano campioni audio dove sia la voce input che quella target proferiscono lo stesso parlato. Chiaramente risulta più facile costruire modelli accurati ma la richiesta dei dati è improponibile. Oggi si studiano modelli non paralleli, con approcci basati su GAN che generano parlato di qualità buona che però degrada su target nuovi e questo è il loro vero scoglio.

**\*slide Processing audio file\***

**\*reti neurali usate\***: iniziare a parlare di cos’è una rete convoluzionale e come funziona (parte difficile)

Cos’è una rete convoluzionale: è innanzitutto una rete neurale feed-forward, ossia di quelle che si “muovono” in una sola direzione:avanti, a differenza ad esempio delle reti ricorrenti dove si hanno dei cicli tra i layer. E quindi è innanzitutto una rete neurale e in quanto tale implementa una funzione f(x) che dipende da n funzioni concatenate, ognuna che mappa da un vettore di valori ad un altro, una funzione per ogni strato della rete. Ogni componente del vettore è un’unità. Si chiama rete neurale perché tenta di simulare il meccanismo dei neuroni cerebrali.

Ogni strato della rete è una funzione lineare e i parametri di ogni output sono da apprendere. Questa funzione in uscita è seguita da una funzione nonlineare detta di attivazione, tipicamente la Rectified Linear Unit (ReLU) definita come phi(x) = max(x,0). E questo per ogni strato tranne che per l’ultimo, dove nel nostro caso di classificazione binaria con l’obiettivo di minimizzare la log loss binaria, la funzione di attivazione sarà la sigmoide, che schiaccia la linea dei reali in un intervallo (0,1) da usare per predire classe uno oppure classe due

Facendo noi apprendimento supervisionato, forniamo un insieme di dati di training che sono coppie input-output in modo che la rete apprenda la relazione che li lega. Poi tramite la backpropagation si calcola il gradiente della funzione d’errore considerata, rispetto ai pesi della rete, per ogni coppia di dati, uno strato alla volta, iterando “all’indietro” per ottimizzare questi calcoli. A questo punto l’aggiornamento dei pesi della rete può essere effettuato seguendo un qualche metodo di ricerca del minimo di una funzione, quale la discesa del gradiente. Il concetto è sempre uno, la rete performa al massimo quando la funzione d’errore è minimizzata e questo lo si raggiunge con la discesa del gradiente. I pesi vengono aggiornati in proporzione alla derivata parziale dell’errore

La peculiarità di una rete convoluzionale è che riesce a tenere in considerazione le dipendenze spazio (pixel vicini sono correlati)-temporali(frame consecutivi sono quasi uguali dunque pixel in un frame dipende dal frame precedente) in un immagine/frame di un video. Vengono applicati dei filtri su piccole parti dell’immagine e i parametri poi ottenuti lungo l’addestramento vengono condivisi di modo che la rete impara ovunque essi compaiano, cioè i parametri vengono condivisi: per un singolo kernel si usano gli stessi parametri in ogni punto dell’immagine

Questi filtri sono i kernel, array bidimensionali che vengono fatti passare sull’immagine.

Un’immagine è tridimensionale, ha altezza, larghezza e profondità(rgb). Il kernel viene applicato su una porzione dell’immagine e viene calcolato il prodotto scalare tra il valore nel kernel e il corrispondente nell’immagine. Il kernel viene poi fatto strisciare, spostare, di un certo numero di pixel detto “stride”. Una volta terminato il processo, fatto strisciare il kernel su tutta l’immagine, si ha per output una “feature map”, che tecnicamente è di minore dimensione perché gli elementi ai margini vengono rimossi, il kernel “esce fuori” dall’immagine. Andando in profondità si avrebbe poi una feature map nulla, quindi si applica lo zero-padding, aggiungendo zeri ai bordi dell’immagine.

Altra caratteristica delle reti convolutive sono i layer di pooling. Questi si occupano di fare down-sampling, ridurre la dimensione delle feature map. Qui si applicano funzioni non lineari, tipicamente una detta di max pooling: viene fatto girare sull’immagine piccoli simil-kernel e al suo interno viene scelto il valore maggiore. Per ridurre la dimensionalità chiaramente lo stride dev’essere maggiore di 1 altrimenti scansionerebbe tutta l’immagine senza fare niente. Il risultato del pooling è ridurre la dimensione, quindi ridurre i parametri, quindi ridurre le computazioni necessarie e ridurre l’overfitting. Tipicamente si inserisce un layer di pooling tra layer convolutivi successivi, i quali sono a loro volta seguiti da funzione di attivazione. Notare come nella resnet non si applichi il pooling, non ve n’è necessità, si usano layer convoluzionali con kernel 1x1 che riducono la dimensionalità (la possono anche aumentare, dipende dal numero di neuroni)

Alla fine dei blocchi conv/pooling si ha la predizione finale attuata dai layer “fully connected”, si usano dei layer “Dense” la cui caratteristica è che ogni neurone è collegato a tutti i neuroni del layer precedente (per questo è denso). Prima di entrare nel fully connected, il suo input dev’essere appiattito da mappa a vettore.

Inoltre nelle nostre reti, in particolare nei blocchi della resnet, vi è un layer di Batch Normalization, questo consiste nel normalizzare gli input, lavorare su media e varianza, quindi sottraendo ad ogni input la media e dividendo per deviazione standard (radice varianza più un piccolo termine epsilon per stabilità numerica). Non si è arrivati ad un verdetto unanime sul perché la BN porti benefici ma sappiamo empiricamente che aiuta con l’overfitting e l’efficienza computazionale

La peculiarità delle reti residuali sono le connessioni residuali. Il vero motivo per cui è stata introdotta resnet non è nemmeno il problema dei gradienti svaniti/esplosi, come detto dagli autori stessi nel paper “Deep Residual Learning for Image Recognition”: questo problema viene in gran parte superato con layer intermedi di normalizzazione che permettono di raggiungere convergenza con SGD anche a modelli con decine di layer. Il problema allo scendere di profondità è un problema di degradazione: l’accuracy raggiunge un picco e scende rapidamente e questo non è a causa di overfitting, più layer portano ad errori ancora più grandi. Gli autori dimostrarono che presa una rete di n strati, se se ne aggiungono altri che applicano solo funzione identità e nient’altro la performance peggiora lo stesso. La soluzione è avere delle shortcut connections: se un certo numero di layer producono in uscita un certo h(x), facciamo in modo che invece producano f(x) = h(x) – x e che questa x sottratta venga poi direttamente sommata ad h(x), producendo f(x) + x. Questa x proviene da una shortcut connection, cioè l’input salta i layer e viene direttamente aggiunta a f(x), non aumentando né il numero dei parametri del modello né complessità computazionale. Il paper dimostra come una rete residuale outperforma alla grande una rete ugualmente profonda che non fa uso di shortcuts.