Tracciamento di condivisioni multiple di immagini sui social network

Elaborato di Tesi

Francesco Fantechi

A.A. 2020-2021

Relatori: Alessandro Piva, Andrew D. Bagdanov

UNIVERSITA' DEGLI STUDI DI FIRENZE
Facolta di Ingegneria
Corso di Laurea in Ingegneria Informatica

1. Abstract

I social networks (SN) e le app di messaggistica istantanea hanno assunto sempre più un ruolo fondamentale e dominante nella vita delle persone e tramite essi grandissime quantità di foto e contenuti multimediali vengono scambiati e condivisi ogni giorno in modo così veloce da non riuscire a tenerne traccia. Quest'ultimo aspetto ha come conseguenza negativa quella di consentire la trasmissione in modo anonimo di informazioni false e mendaci fino a sfociare in veri e propri crimini come il cyberbullismo che per loro natura ignota rimangono per essere impuniti. Per arginare questo fenomeno le indagini forensi lavorano al fine di trovare metodi sempre più efficaci atti al tracciamento dei cammini seguiti da questi contenuti durante la loro vita. Questo lavoro si pone infatti come obiettivo quello di tracciare e ricostuire in ordine cronologico il percorso di condivisioni multiple sui social network al quale delle immagini sono andate in contro estraendo da esse le informazioni lasciate da ogni piattaforma. Per farlo sono state utilizzate strategie risolutive basate sul deep learning riportando degli ottimi risultati nel tracciamento fino a tre condivisioni.

2. Introduzione

L'introduzione di fotocamere digitali sempre di maggior qualità negli smartphone ha favorito una produzione crescente di dati multimediali e i social network sono diventati i canali di maggior diffusione di tali contenuti. L'incremento di popolarità di editor di facile utilizzo ha reso inoltre la manipolazione e la distorsione di tali contenuti sempre più accessibile a chiunque. L'abuso di queste pratiche se non controllato può avere un impatto molto negativo sulla società con conseguenze non facilmente misurabili. Negli ultimi anni molti metodi forensi sono stati proposti per validare contenuti multimediali, come ad esempio l'identificazione della fotocamera usata per l'acquisizione, e tali metodi sono stati resi sempre più sofisticati con l'avvento di techiniche basate sul deep learning. La loro efficacia viene messa però in discussione nella trattazione di immagini provenienti dai social network. Questi ultimi infatti sottopongono tali contenuti a forti compressioni JPEG che vanno a rimuovere le tracce forensi presenti sui file immagine rendendo tali tecniche poco efficaci. I social network infatti attuano delle compressioni che alterano e cancellano i dati associati ad un'immagine e spesso tali compressioni seguono delle proprie politiche ne fissate o stabilite pubblicamente. Questo aspetto introduce la necessità di ricercare metodi per tracciare le condivisioni sui social network in modo da recuperare parzialmente la conoscenza sulla provenienza di una immagine sotto indagine. I migliori risultati fino a questo momento sono stati riportati in [1] dove una rete convoluzionale è stata addestrata sulle tracce lasciate nei coefficienti DCT e nei metadati JPEG di alcune immagini al fine di imparare a predirne la storia classificandole di conseguenza. Il dataset di immagini da loro utilizzato, R-SMUD, appartiene a quello generato dal dataset RAISE [2] e contiene immagini condivise da una a tre volte sui social network Facebook (FB), Flickr (FL) e Twitter (TW). Riportiamo in figura ??? la tabella con i loro risultati ottenuti classificando su 3 classi per la sola ultima condi-

visione $C1$, su 12 classi per la predizione delle due ultime possibili condivisioni $C2$ e su 39 classi per la predizione di tutte e tre le possibili condivisioni $C3$. In questo lavoro sperimentiamo altri metodi per tracciare e ricostruire la storia di immagini che sono state condivise da una fino a tre volte sui social network sopra elencati sfruttando le tracce lasciate nei coefficienti DCT e le informazioni presenti nei metadati e header JPEG.

3. Metodologia proposta

In questa sezione introduciamo la metodologia proposta al fine di tracciare le condivisioni multiple di immagini sui social network (Sezione 3.1) e il dataset e le features utilizzate per addestrare i nostri algoritmi di apprendimento supervisionato (Sezione 3.2).

3.1.

Il metodo da noi proposto e implementato prevede alla base un Multi Layer Perceptron (MLP), ossia una rete neurale che presenta più strati di "neuroni" per processare dei valori ed eseguire una classificazione su un numero fissato di etichette restituendo una distribuzione di probabilità su di esse. Un neurone - una cellula celebrale la cui funzione principale è raccogliere, elaborare e propagare segnali elettrici - a grandi linee "scatta" quando una combinazione lineare dei suoi input supera una certa soglia. Le reti neurali sono composte da un certo numero di "neuroni", detti nodi o unità, uniti da un collegamento diretto. Ad ogni collegamento, per esempio dal nodo j al nodo i, è inoltre associato un peso $W_{j,i}$ che determina la sua importanza. Ogni unità i calcola per prima cosa una somma pesata dei propri input x_j e poi vi applica una funzione di attivazioe g (solitamente una funzione soglia o sigmoide) per derivarne l'output in base al valore assunto dalla somma:
$output = g(\sum_{j=0}^{n} W_{j,i} x_j)$.
Una rete neurale a singolo strato, ossia dove gli input sono collegati direttamente agli output, è detta "Percettrone a soglia" e presenta il limite di poter rappresentare solamente funzioni linearmente separabili. Il problema dei semplici percettroni è superato calibrando i pesi nella rete in modo da minimizzare una qualche misura di errore sull'insieme di training. Ad esempio con il metodo della discesa stocastica di gradiente (SGD) i pesi sono aggiornati ad ogni iterazione di addestrmaneto secondo la regola $W_j \leftarrow W_j + \alpha \times (-\frac{\partial E}{\partial W_j})$, con α tasso di apprendimento e $-\frac{\partial E}{\partial W_j}$ l'antigradiente dell'errore. Come si può apprezzare in figura ??? un MLP è una rete neurale multistrato. Esso presenta infatti un livello di input la cui dimensione è pari al numero di features a disposizione per l'addestramento, un livello di output con dimensione pari al numero di classi su cui predire e un numero non fisso di livelli nascosti di dimensione variabile. Le formule per l'aggiornamento dei pesi sono simili a quelle precedentemente riportatate con la differenza che l'errore viene retropropagato all'indietro dal

livello di output ai livelli nascosti al fine di aggiornare i pesi con l'idea che ogni nodo è responsabile in parte dell'errore in ognuno dei nodi a cui è collegato. Il numero e la dimensione degli strati del MLP da una parte aumenta la capacità di predizione della rete riducendo la linearità e dall'altra può introdurre un maggior rischio di incorrere nel fenomeno dell'overfitting. Ad ogni iterazione di addestramento (117 immagini alla volta come Batch Size) i pesi delle nostre reti vengono aggiornati attraverso una retropropagazione dell'errore calcolato confrontando la distribuzione di probabilità predetta con la vera classe al quale l'oggetto sotto indagine appartiene secondo il metodo dell'entropia incrociata. L'aggiornamento dei parametri é stato effettuato con l'ottimizzatore Adam, un ottimizzatore che selezionando in modo adattivo un tasso di apprendimento separato per ciascun parametro riesce ad trovare con maggiore probabilità il minimo globale rispetto ad altri ottimizzatori come SGD.

Questa rete alimentata in avanti è stata impiegata al fine di ottenere una rete neuarle ricorrente (RNN) ossia una rete che riprende di nuovo in input i propri stessi output. Supportando una memoria a breve termine e facendo dipendere da essa la risposta ad un dato input questa rete é stata implementata con l'idea di riuscire a predire in ordine cronologico tutte le condivisioni al quale un immagine è andata incontro.

In reti troppo profonde una RNN può presentare il problema dei gradienti che svaniscono andando a vanificare l'effetto della retropropagazione durante l'addestramento. Per questo è stata implementata anche una rete Long Short Term Memory (LSTM) che, a differenza di una RNN, prende tre input: i dati di input corrente, la memoria a breve termine che coincide con l'ultimo livello nascosto del MLP dell'iterazione precedente e la memoria a lungo termine (stato della cella) che racchiude informazioni su tutte iterazioni precedenti. Come è possibile osservare in figura ???, la rete percorre vari step per regolare le informazioni da conservare o scartare ad ogni passaggio prima di passare le informazioni a lungo termine e a breve termine alla cella successiva. Il primo step noto come "Input Gate" si compone in due strati. Il primo passa la memoria a breve termine e l'input corrente in una funzione sigmoide. La funzione sigmoide trasforma i valori in un range compreso tra 0 e 1, con 0 che indica che parte dell'informazione non è importante e 1 che indica quale parte dell'informazione verrà utilizzata. Questo aiuta a decidere i valori da conservare e utilizzare, e anche i valori da scartare. Il secondo livello prende sempre la memoria a breve termine e l'input corrente e li fa passare attraverso una funzione di attivazione per regolare la rete. Gli output di questi 2 strati vengono quindi moltiplicati e il

risultato finale rappresenta le informazioni da conservare nella memoria a lungo termine. Nel secondo step noto come "Forget Gate" la memoria a breve termine e l'input corrente vengono passati attraverso una funzione sigmoide e il vettore risultante viene moltiplicato con la memoria a lungo termine per scegliere quali parti di essa conservare e quali no. Il terzo step noto come "Output Gate" riprende l'input corrente, la precedente memoria a breve termine e la memoria a lungo termine appena calcolata per produrre il nuovo stato di memoria a breve termine che verrà passato alla cella nel passaggio temporale successivo. Da questo stato nascosto può essere ricavato anche l'output della fase temporale corrente.

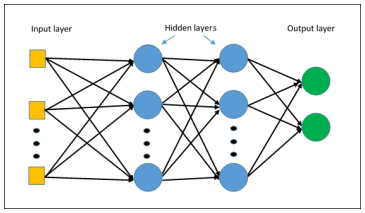

Figure 1: Architettura di un MLP

3.2. Il dataset utilizzato per addestrare e testare le performance della nostra rete é il dataset usato in [1] ossia quello generato dal dataset RAISE. Questo dataset, detto R-SMUD (RAISE Social Multiple Up-Download) contiene delle immagini condivise su tre SN: Facebook (FB), Flickr (FL) e Twitter (TW). 50 immagini (formato raw) sono state selezionate da RAISE e ritagliate nelle dimensioni: 377 × 600, 1012 × 1800 e 1687 × 3000 con un rapporto di 9: 16. Tutte le immagini ritagliate sono state successivamente compresse in formato JPEG a fattori di qualità QF = 50, 60, 70, 80, 90, 100. Questo ha prodotto 900 immagini in totale che sono state poi condivise al massimo 3 volte attraverso le tre piattaforme. Il dataset presenta 21060 immagini per il trainset, 7200 immagini per il validset e 7200 immagini per il testset. Per ogni immagine sono state estratte tramite la libreria software ExifTool [1] le tre features sulle quali è stato eseguito l'addestramento della rete. Per ogni file immagine risultano 369 coefficenti DCT che mantengono le tracce delle compressioni lossy JPEG, 152 valori contenuti nei metadati che forniscono le informazioni riguardanti le impostazioni della compressione JPEG, come ad esempio la tabella di quantizzazione e la tabella per la decodifica di Huffman, e 10 valori contenuti nell'header JPEG che codificano proprietà strutturali della compressione JPEG. Per addestrare e testare le nostre reti queste features sono state concatenate per ogni immagine ottenendo degli array caratteristici $[1, 531]$. Per ognuno dei tre set questi array sono stati poi impilati a formare delle matrici i cui valori sono stati poi normalizzati per colonna rispetto alla matrice del trainset, $x = \frac{x - x_{min}}{x_{max} - x_{min}}$. Ogni immagine è etichettata con una delle 39 classi ottenute considerando tutte le possibili combinazioni con ripetizione delle condivisioni sui tre SN, $3^1 + 3^2 + 3^3 = 39$.

4. Esperimenti condotti e risultati

L'intero codice è stato scritto in linguaggio Python servendosi in particolar modo per l'implementazione delle varie reti neurali della libreria Pytorch. Il codice è versionato sulla piattaforma github, clicca qui per accedervi.

I primi esperimenti condotti si concentrano sulla predizione della sola ultima condivisione. La rete implementata è un MLP che esegue una classificazione su 3 classi e per questo le etichette sono state portate da 39 a 3 a seconda dell'ultimo SN nella sequenza di condivisione. La rete, i cui risultati sono riportati in tabella ??? alla voce "Direct3", presenta un livello di input di dimensione 531, tre livelli nascosti di dimensione rispettivamente $256, 128$ e 32, e un livello di output di dimensione 3. L'addestramento è stato eseguito per 20 epoche con un batch size di 117 immagini.
Gli esperimenti poi sono stati condotti al fine di classificare sia l'ultima che la penultima condivisione. È stata testata sia una rete che classifica diret-

tamente su 12 classi (tutte le combinazioni doppie dei 3 SN e le condivisioni singole) sia dei metodi che eseguono due classificazioni in sequenza per provare ad identificare le ultimi due condivisioni. La rete che classifica su 12 classi è un MLP che presenta un livello di input di dimensione 531, tre livelli nascosti di dimensione rispettivamente $256, 128$ e 64, e un livello di output di dimensione 12. L'addestramento di questa rete e delle seguenti è stato eseguito aiutandosi con la libreria di Pytorch "Tensorboard" al fine di analizzare ad ogni epoca di addestramento l'accuaratezza e la loss riportata sul trainset e sul validset. Questo ci ha permesso di osservare la possibile presenza di overfitting e di decidere quando interrompere l'addestramento al fine di massimizzare i risultati. I risultati di accuratezza sul testset riportati per questa rete in tabella ??? alla voce "Direct12", sono ottenuti con 77 epoche di addestramento e un batch size di 117 immagini.

Per quanto riguarda la classificazione dell'ultimo e del penultimo social in sequenza sono state implementate due reti differenti. Una utilizza un MLP che restituisce due classificazioni: una su 3 classi per il primo social e una su 4 classi per il secondo social ("NONE" in caso l'immagine fosse stata condivisa una sola volta). Questo tipo di architettura assume che due predizioni in sequenza sugli stessi valori siano sufficienti a classificare le ultime due condivisioni. I risultati di questa rete sono riportati in tabella ??? alla voce "DoublePrediction" e sono ottenuti tramite un addestramento di 71 epoche e un batch size di 117 immagini. Il La seconda assomiglia ad una rete ricorrente "srotolata" in quanto presenta un MLP identico per ogni predizione da effettuare. Qui i MLP prendono un extra input con l'idea che questo riesca ad incapsulare tutte le informazioni sull'ultimo social predetto e influenzare la predizione sul penultimo. Questo extra input corrisponde all'ultimo layer del MLP e viene concatenato all'input formato dalle features dell'immagine. La rete esegue quindi in sequenza i due MLP: il primo sull'input concatenato con un vettore di zeri e il secondo sull'input concatenato con l'ultimo layer dell'esecuzione precedente. Nella prima esecuzione restituisce una classificazione su 3 classi mentre sulla seconda una classificazione su 4 classi. risultati di questa rete sono riportati in tabella ??? alla voce "Unrolled2" e sono ottenuti tramite un addestramento di 132 epoche e un batch size di 117 immagini. Sia in questa rete che nella rete "DoublePrediction" i MLP presentano un livello di input di dimensione 531, tre livelli nascosti di dimensione rispettivamente $256, 128$ e 32, e un livello di output di dimensione 3 o 4.

Successivamente sono state costruite delle reti per il riconoscimento di tutte e tre le ultime condivisioni. È stata testata sia una rete che classifica direttamente su 39 classi (tutte le combinazioni triple e doppie dei 3 social e le condivisioni singole), il metodo simile alla rete ricorrente "srotolata", la rete ricorrente vera e propria e una rete LSTM.

Il risultato del MLP che classifica direttamente su 39 classi è riportato in tabella ??? alla voce "Direct39". Questa rete presenta un livello di input di 531 features, due livelli nascosti di dimensione 256 e 128 e un livello di output di dimensione 39. La rete

è stata addestrata per 149 epoche con un batch size di 117 immagini.
I risultati del metodo simile ad una rete ricorrente srotolata sono riportati in tabela ??? alla voce "Unrolled3". Questa rete è stata addestrata per 80 epoche con un batch size di 117 immagini e presenta 3 livelli nascosti (531 $[256, 128, 32]$ $3 - 4$).
La rete ricorrente (RNN), la cui architettura può essere apprezzata in figura ???, presenta un solo MLP che prende un extra input con l'idea che questo riesca ad incapsulare tutte le informazioni sull'ultimo social predetto e influenzare la predizione sul penultimo. Questo extra input corrisponde all'ultimo layer del MLP e viene concatenato all'input formato dalle features dell'immagine. La rete esegue quindi tre volte il MLP: la prima iterazione sull'input concatenato con un vettore di zeri e la seconda e la terza sull'input concatenato con l'ultimo layer dell'esecuzione precedente. Nella prima esecuzione restituisce una classificazione su 3 classi mentre sulla seconda e sulla terza una classificazione su 4 classi. I risultati di questa rete sono riportati in tabella ??? alla voce "RNN" e sono ottenuti tramite un addestramento di 67 epoche e un batch size di 117 immagini. Il MLP presenta un livello di input di dimensione 531, un solo livello nascosto di dimensione rispettivamente 267, e un livello di output di dimensione 3 o 4.
Infine è stata testata la rete LSTM. I risultati sono riportati in tabella ??? alla voce "LSTM" e sono ottenuti tramite un addestramento di 80 epoche e un batch size di 117 immagini. Il MLP presenta un livello di input di dimensione 531, un solo livello nascosto di dimensione rispettivamente 267, e un livello di output di dimensione 3 o 4.

5. Conclusioni

Come