Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

TITOLO ITALIANO

TITOLO INGLESE

GIULIANO GAMBACORTA

Relatore: *Paolo Frasconi*
Correlatore: *Correlatore*

Anno Accademico 2017-2018

INDICE

ELENCO DELLE FIGURE

"citazione"
— Autore

1

INTRODUZIONE

1.1 PREFAZIONE

Questa tesi consiste nella riproduzione e nello studio di *sketch-rnn* [1], una rete neurale in grado di generare disegni di semplici oggetti, composti da sequenze di tratti, appresi da un dataset di disegni creati da esseri umani. Il dataset è costantemente ampliato tramite *Quick Draw!* [2], un gioco online in cui agli utenti viene chiesto di riprodurre alcuni oggetti entro 20 secondi, che al momento della stesura di questa tesi costituisce la più vasta collezione di disegni al mondo. In questo lavoro viene proposta un'implementazione in *Keras* [3], un framework che a sua volta poggia su *tensorflow* [4], che è la libreria utilizzata per il lavoro originale.

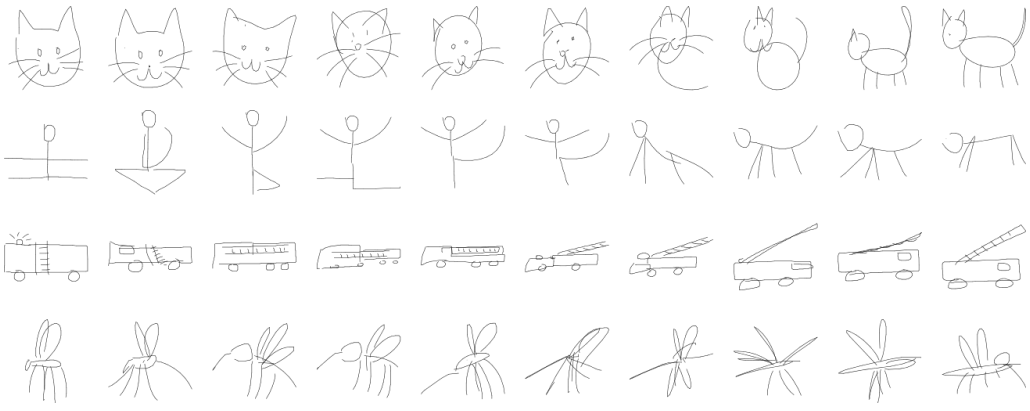


Figura 1: Interpolazioni nello spazio di latenza di immagini vettoriali generate dal modello.

1.2 STATO DELL'ARTE

Negli ultimi anni la generazione di immagini attraverso l'uso di reti neurali ha avuto ampia diffusione, fra i modelli più importanti possiamo

citare *Generative Adversarial Networks (GANs)* [5], *Variational Inference (VI)* [6] e *Autoregressive Density Estimation (AR)*. [7] Il limite della maggior parte di questi algoritmi è che lavorano con immagini in pixel a bassa risoluzione, a differenza degli esseri umani che, piuttosto che vedere il mondo come una griglia di pixel, astraggono concetti per rappresentare ciò che osservano. Allo stesso modo degli esseri umani, che fin da piccoli imparano a riportare i concetti appresi attraverso una sequenza di tratti su un foglio, questo modello generativo apprende da, e produce, immagini vettoriali.

L'obbiettivo è di addestrare una macchina a riprodurre ed astrarre concetti, in maniera analoga a come farebbe una persona. Ciò può avere numerose applicazioni in campo didattico come artistico, ad esempio assistendo il processo creativo, così come l'analisi della rappresentazione prodotta può offrire spunti di ricerca.

1.3 LAVORI CORRELATI

1.4 RETI NEURALI

Per spiegare le reti neurali e il Deep Learning si possono usare diversi approcci: si può ad esempio seguire il corso storico, introducendo il concetto di *Percettrone*, passando ai Percettroni Multi-Strato ed ai primi metodi di ottimizzazione che furono applicati a questi modelli. Un'altra possibilità consiste nel partire da un punto di vista stocastico, definendo una regressione logistica che implica in modo naturale la minimizzazione di una *loss function*. Ciò permette la definizione del concetto stesso di loss function e di come modificare dei parametri per ottenere una soluzione migliore, ciò si ricollega perfettamente al concetto di Percettrone Multi-Strato come classificatore di una regressione logistica.

Da qui in poi verrà proposta la spiegazione di alcuni modelli fondamentali del Deep Learning, che saranno considerati come "mattoni" costitutivi del modello studiato in questo progetto. Ciò aiuterà a comprendere agevolmente l'implementazione realizzata nel codice, seguendo un punto di vista coerente con le piattaforme più diffuse.

1.5 RETI DENSAMENTE CONNESSE

Un neurone artificiale consiste in una funzione matematica che costituisce l'unità computazionale di base di una rete neurale artificiale. Come si nota

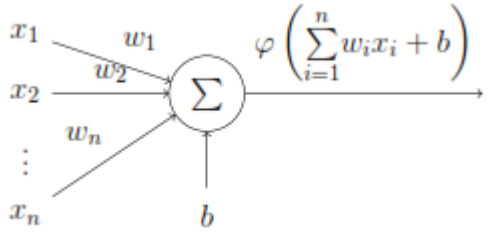


Figura 2: Rappresentazione di un neurone artificiale.

dalla figura 2 il neurone artificiale riceve n input pesati $(w_1x_1, ..., w_nx_n)$ e un bias b. Successivamente li somma e applica una funzione non lineare nota come *activation function* φ per generare l'output. In sostanza calcolare l'output di un singolo neurone corrisponde a formare una combinazione lineare dei suoi input pesati e poi passarla ad una funzione di attivazione.

Per ragioni storiche, nel caso particolare in cui la funzione di attivazione consista di una funzione con una soglia lineare (e output binario), il neurone è detto *Percettrone*.

Da qui in poi, quando ci riferiremo ad un *layer* di una rete neurale staremo parlando di un raggruppamento di neuroni che formano, nello specifico, una colonna nel grafo della figura 2, ovvero neuroni che si trovano allo stesso livello di profondità nella rete. Inoltre, ogni layer che si trova fra quello di input e quello di output verrà chiamato *hidden layer*.

Come detto in precedenza, un Percettrone Multi-Strato (MLP da Multi-Layer Perceptron) può essere visto come un classificatore a regressione logistica dove l'input è trasformato utilizzando una trasformazione non lineare appresa $h(\mathbf{x})$ che costituisce l'hidden layer della nostra rete neurale (come in figura 2). Questa trasformazione proietta l'input in uno spazio dove diventa linearmente separabile.

Questa computazione eseguita da una rete neurale multi-Strato, con un singolo hidden layer con una funzione di attivazione non lineare per elementi φ_i sul vettore di input $\mathbf{x}$ e l'hidden output può essere scritta, in forma matriciale, come $\mathbf{y} = \varphi_2(W_2\varphi_1(W_1\mathbf{x} + \mathbf{b_1}) + \mathbf{b_2})$ dove $W_i, \mathbf{b_i}$ sono la matrice dei pesi e il vettore del bias dell'i-esimo layer.

Seguendo il teorema di approssimazione universale [11] possiamo affermare che una rete neurale con un singolo hidden layer contenente un numero finito di neuroni (ovvero un MLP) è abbastanza per approssimare qualunque funzione continua su un sottoinsieme compatto di $\mathbb{R}^n$.

Alcuni sinonimi rilevanti utilizzati al posto di reti neurali Multi-Strato sono *dense layers*, *fully-connected layers* o, meno diffuso, *inner product layers*.

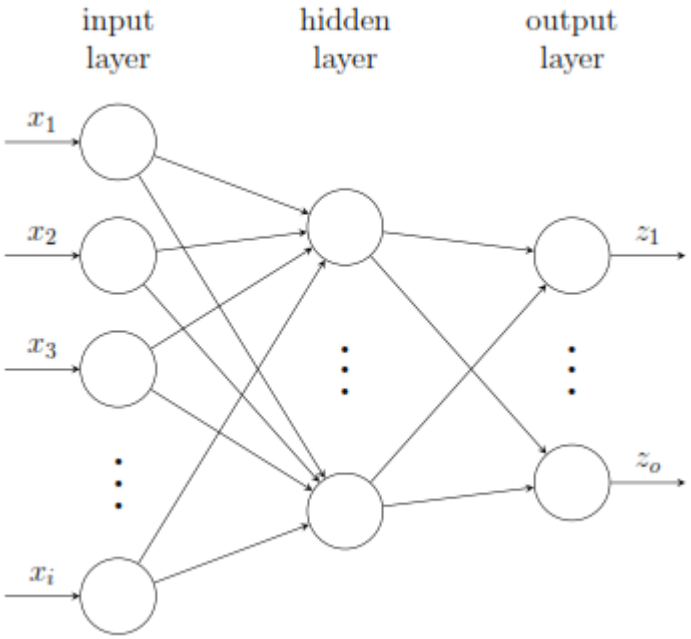


Figura 3: Rappresentazione di un modello Multi-Strato.

1.6 RETI RICORRENTI

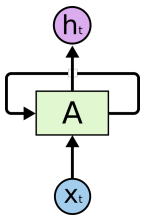


Figura 4: Una semplice RNN (Recurrent Neural Network).

Il cervello umano, nello specifico il lobo frontale, è in grado di elaborare conseguenze future risultanti da azioni nel presente, è in grado di selezionare fra buone e cattive azioni (o fra migliori e ideali) e può determinare somiglianze e differenze fra oggetti ed eventi.

Molti problemi richiedono, per essere risolti, di un certo grado di conoscenza pregressa. Un esempio può riguardare le variazioni della luce di un semaforo: se, per esempio, osserviamo che nel momento attuale la luce accesa è quella gialla, il nostro scopo sarebbe quello di sapere quale sarà la prossima ad accendersi. Le diverse posizioni delle luci in un semaforo sono irrilevanti: ciò che ci interessa è sapere quale colore

apparirà, sapendo che quello appena apparso è il giallo. Nella maggior parte delle città sappiamo che la risposta sarebbe che il prossimo sarà il rosso. Per rispondere a questa domanda è venuta in nostro aiuto la nostra esperienza ma se provassimo a risolvere il problema utilizzando una rete neurale come quella proposta in precedenza non otterremmo una risposta soddisfacente. Ciò accade perché la soluzione presenta una dipendenza temporale, che corrisponde al metodo attraverso cui un essere umano apprende a risolvere problemi: analizzando sequenze di eventi.

Per trovare una soluzione a questo problema, com'è uso nel campo dell'Intelligenza Artificiale, viene tratta nuovamente ispirazione dai modelli basati sui principi biologici per elaborare una classe di reti neurali artificiali, in cui le connessioni fra le unità formano un ciclo orientato, dette Reti Neurali Ricorrenti (RNN da Recurrent Neural Networks, Fig. 4). Queste connessioni creano uno stato interno della rete che le permette di esibire un comportamento dinamico nel tempo.

Per le reti neurali non ricorrenti possediamo già molte conoscenze per ottenere buoni parametri. Di conseguenza si rende necessario trovare un modo di trasferire le potenzialità già discusse anche su questo nuovo tipo di reti neurali. Un'idea primitiva potrebbe essere quella di elaborare la rete ricorrente attraverso copie molteplici di una singola rete, come si vede in fig. 5, trasferendo le informazioni dall'hidden layer A alla copia successiva per realizzare una forma di memorizzazione.

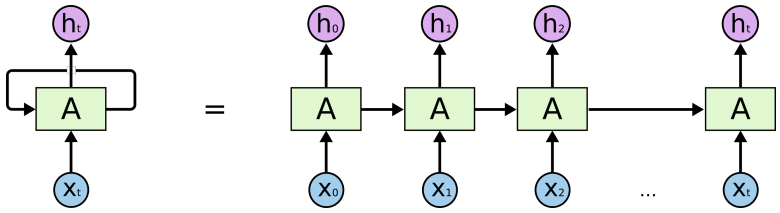
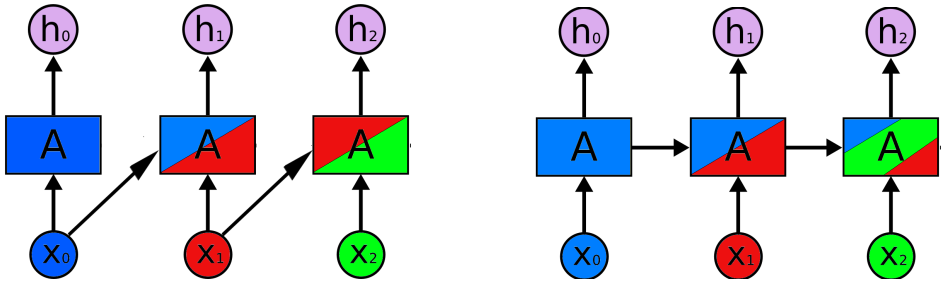


Figura 5: Una RNN dispiegata lungo la linea temporale.

Tuttavia ci potrebbero essere diversi modi di intendere la memoria e di combinare l'input del *time-step* attuale con le informazioni ottenute dal precedente. Ad esempio potremmo scegliere di ottenere le informazioni precedenti conservando l'input oppure l'hidden layer del time-step passato, ottenendo risultati completamente diversi. in fig. 6 sono riportati i due esempi sopra descritti, dove ogni colore rappresenta gli effetti sulla memoria dell'hidden layer A.

Come si può vedere in fig. 6(a), usando la ricorrenza dell'input viene ricordato solo l'attuale input e il precedente, invece nel caso della ricorrenza dell'hidden layer (fig. 6(b)) viene ricordata una mistura di



(a) Visualizzazione della memoria usando la ricorrenza dell'input

(b) Visualizzazione della memoria usando la ricorrenza dell'hidden layer

Figura 6: Due diversi metodi di implementazione della memoria in una RNN

tutti gli input precedenti. Per comprendere perché la seconda ipotesi è migliore si può usare questo esempio: si immagini di voler dedurre la parola seguente a *"I love"* ("io amo") e che il testo contega le affermazioni *"I love you"* ("ti amo") e *"I love carrots"* ("amo le carote"). Se lo scopo è predire la decisione che verrà presa fra queste due opzioni e non sono note altre informazioni al di là dell'input (nel caso della ricorrenza dell'input), la rete neurale non avrà abbastanza informazioni per decidere. Viceversa, se la rete neurale possiede informazioni riguardanti il contesto, ad esempio se precedentemente si è parlato di cibo o di pasti, la sceltà diventa più chiara. A prima vista la ricorrenza dell'hidden layer può essere interpretata come un tentativo di ricordare ogni informazione con cui la rete neurale è entrata in contatto ma in pratica vige un limite di al più qualche passo.

Un'altra caratteristica delle RNN, che le avvantaggia ulteriormente rispetto alle MLN, è la versatilità. Le RNN, infatti, sono in grado di operare su sequenze di vettori, a differenza delle MLN o delle CNN (che non saranno trattate in questo elaborato) che operano su vettori di dimensioni fissate, così come fissato è il numero di passi computazionali (limitato ad esempio dal numero di hidden layers). Si possono elaborare sequenze sia in input che in output (o entrambi), in fig. 7 distinguiamo cinque casi.

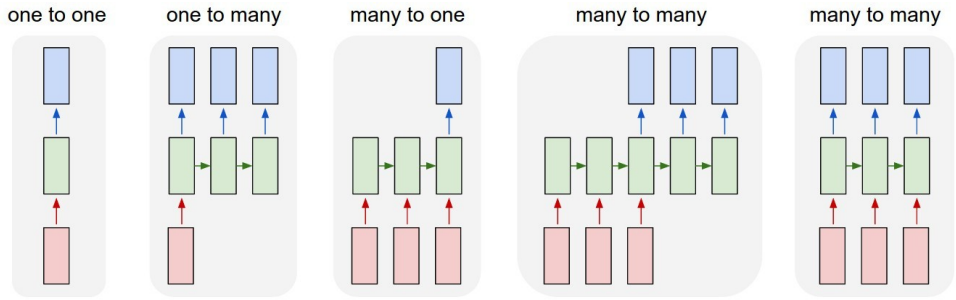


Figura 7: Combinazioni di sequenze vettoriali. [12]

Uno a uno: il caso più semplice, in cui non vi è ricorrenza, ad esempio nella classificazione di immagini.

Uno a molti: il caso in cui è presente una sequenza in output. Tipicamente usato nella creazione di sottotitoli, dove ad una sola immagine corrisponde una frase.

Molti ad uno: in questo caso la sequenza è presente solo in input, è la struttura della *sentiment analysis*, dove da una frase viene estratta la sensazione positiva/negativa contenuta in essa.

Molti a molti: qui abbiamo una sequenza sia in input che in output, si potrebbe trattare di un modello di traduzione che assegna ad una frase in una lingua la frase corrispondente in un'altra.

Molti a molti(sincronizzato): come nell'esempio precedente ma l'output è in sincronia con l'input. Utilizzato ad esempio nella classificazione video, in cui un'etichetta va assegnata ad ogni frame in tempo reale.

1.6.1 *Dipendenze a lungo termine*

Nel progettare una RNN va presa in considerazione la possibilità di incontrare la necessità di fornire un vasto numero di informazioni dal contesto per risolvere un problema. Tornando al compito della previsione di parole in un testo, si supponga di dover completare la frase "sto per andare a nuotare in", le informazioni a breve termine suggeriscono che la parola successiva sia un luogo dove sia possibile nuotare, sapendo che la frase precedente è "la spiaggia è molto assolata" si potrebbe dedurre che la parola da predire sia "mare". Il problema in questione è che si incontrano spesso difficoltà nell'identificare informazioni rilevanti. La formulazione di RNN data finora, in teoria, è perfettamente in grado di

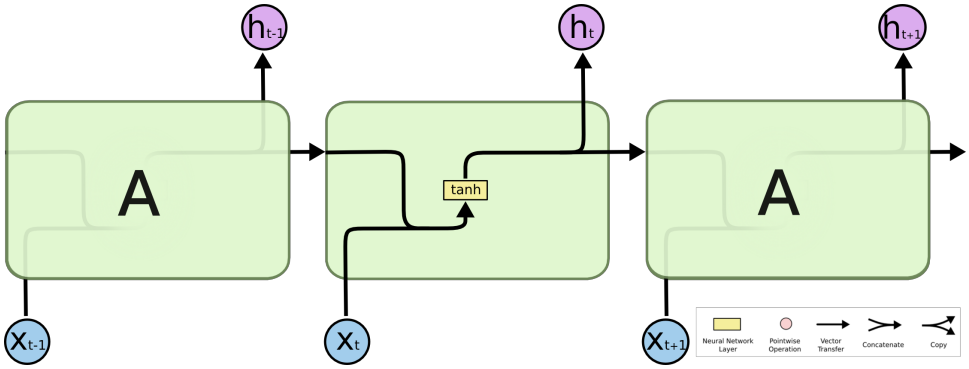


Figura 8: Struttura interna di una RNN standard con un singolo hidden layer.

gestire dipendenze a lungo termine. In pratica, come spesso accade, si tratta di una prova tipicamente banale per una mente umana ma che una semplice implementazione (come in fig. 8) non sarebbe in grado di risolvere efficacemente. Uno degli ostacoli più frequenti da risolvere è il cosiddetto problema del gradiente evanescente (*vanishing gradient* [13]), per superarlo si rendono necessarie varianti più elaborate della RNN semplice, come le LSTM.

1.6.2 *Long Short Term Memory*

1.6.3 *Reti bidirezionali*

1.7 RETI AUTOREGRESSIVE

1.8 AUTOENCODER

1.8.1 *Variational Autoencoder*

1.9 MDN

2

STRUMENTI

2.1 LIBRERIE SOFTWARE

2.1.1 *Keras*

2.1.2 *Edward*

2.1.3 *Recurrentshop*

2.2 DATASET

3

ESPERIMENTI

3.1 KERAS SKETCH-RNN

4

CONCLUSIONI

BIBLIOGRAFIA

[1] David Ha, Douglas Eck - *A Neural Representation of Sketch Drawings* - arXiv:1704.03477, 2017 (Cited on page 7.)

[2] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. - *The Quick, Draw! - A.I. Experiment.* - `https://quickdraw.withgoogle.com/`, 2016. (Cited on page 7.)

[3] Chollet, François and others - *Keras* - GitHub, `https://github.com/keras-team/keras` (Cited on page 7.)

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. - *TensorFlow: Large-scale machine learning on heterogeneous systems* - `tensorflow.org`, 2015 (Cited on page 7.)

[5] I. Goodfellow. - *NIPS 2016 Tutorial: Generative Adversarial Networks.* - ArXiv e-prints, Dec. 2017. (Cited on page 8.)

[6] D. P. Kingma and M. Welling. - *Auto-Encoding Variational Bayes.* - ArXiv e-prints, Dec. 2013. (Cited on page 8.)

[7] S. Reed, A. van den Oord, N. Kalchbrenner, S. Gómez Colmenarejo, Z. Wang, D. Belov, and N. de Freitas. - *Parallel Multiscale Autoregressive Density Estimation.* - ArXiv e-prints, Mar. 2017. (Cited on page 8.)

[8] Dustin Tran and Alp Kucukelbir and Adji B. Dieng and Maja Rudolph and Dawen Liang and David M. Blei - *Edward: A library for probabilistic modeling, inference, and criticism* - arXiv preprint arXiv:1610.09787, 2016

[9] Axel Brando - *Mixture Density Networks (MDN) for distribution and uncertainty estimation* - https://github.com/axelbrando/Mixture-Density-Networks-for-distribution-and-uncertainty-estimation/blob/master/ABrando-MDN-MasterThesis.pdf, 2017

[10] Colah - *Understanding lstm networks, colah's blog.* - http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[11] G. Cybenko - *Approximation by Superpositions of a Sigmoidal Function* - 1989 (Cited on page 9.)

[12] A. Karpathy - *The Unreasonable Effectiveness of Recurrent Neural Networks* - http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015 (Cited on pages 3 and 13.)

[13] S. Hochreiter and J. Schmidhuber *Long short-term memory* - Neural computation, 9 (1997), pp. 1735 (Cited on page 14.)