

Tesi di Laurea Magistrale in Informatica

Reti Convulsive e a Capsule per riconoscimento di post incitanti all'odio

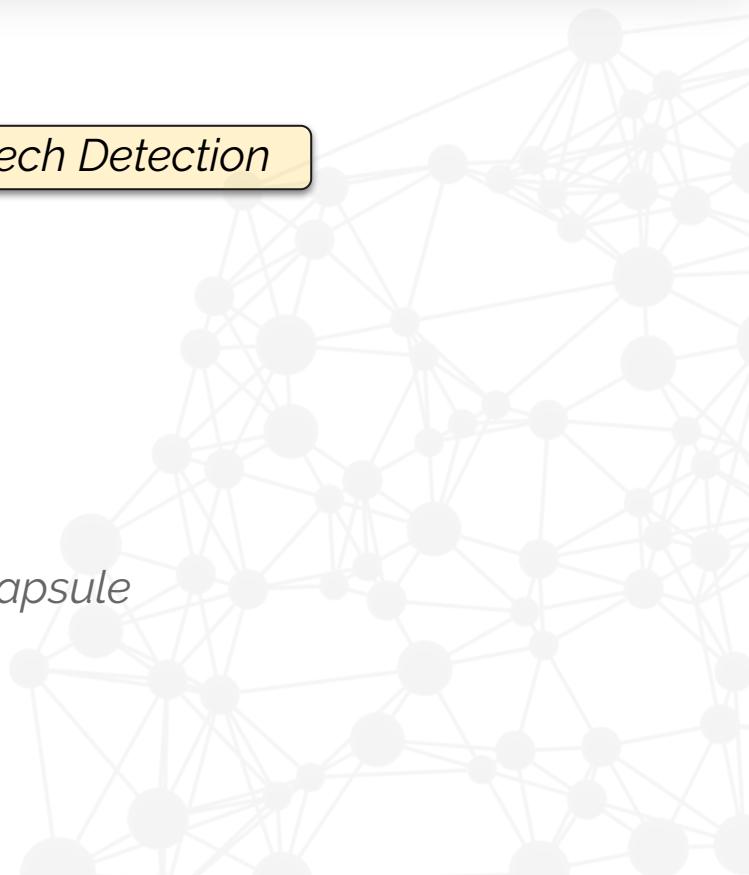
Laureando
Lorenzo Ferri

Relatori
Valentina Poggioni
Alina-Elena Baia

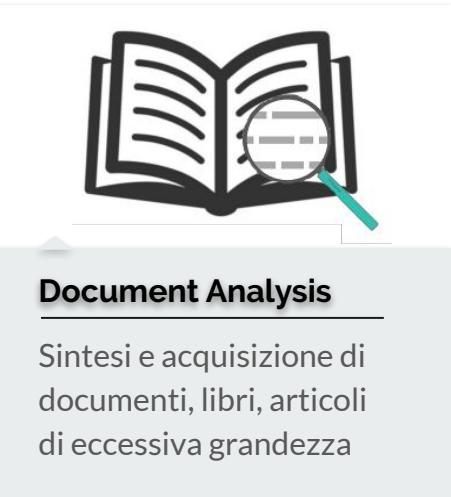


Outline

- Cos' è il *Natural Language Processing* e l'*Hate Speech Detection*
- Baseline model: *Multilayer Perceptron*
- Reti Convolutive
- Reti a Capsule
- Hate Speech Detection con *Reti Convolutive* e *a Capsule*
- Risultati



Natural Language Processing (NLP)



Document Analysis

Sintesi e acquisizione di documenti, libri, articoli di eccessiva grandezza

Speech Recognition

Riconoscimento e sintesi vocale per assistenti personali intelligenti



Alexa Siri Google Now



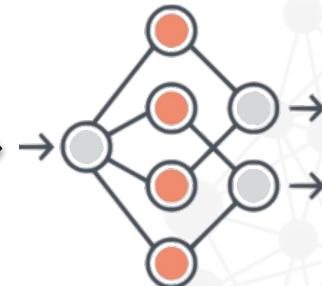
Sentimental Analysis

Riconoscimento di stati d'animo da testo scritto

Hate Speech Detection

Applicazione del *Natural Language Processing* nel riconoscimento di **testo offensivo**

E' ritenuta offensiva qualsiasi comunicazione (vocale o testuale) con il fine di danneggiare una persona o un gruppo di persone in base alla **razza, cultura, colore, etnia, orientamento sessuale, religione e pensiero**.



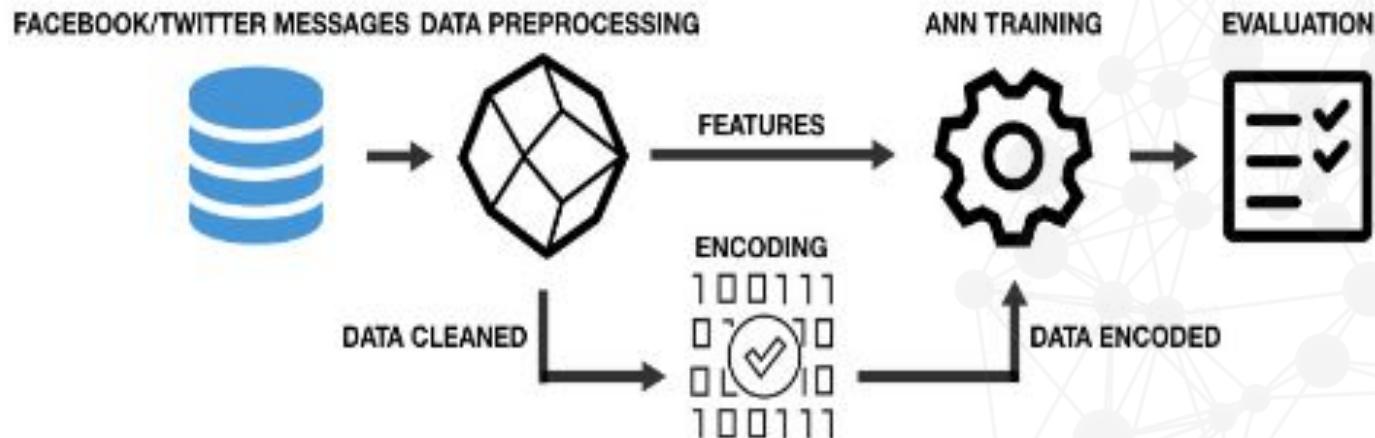
Outline

- Cos' è il *Natural Language Processing* e l'*Hate Speech Detection*
- Baseline model: *Multilayer Perceptron*
- Reti Convolutive
- Reti a Capsule
- Hate Speech Detection con *Reti Convolutive* e a *Capsule*
- Risultati

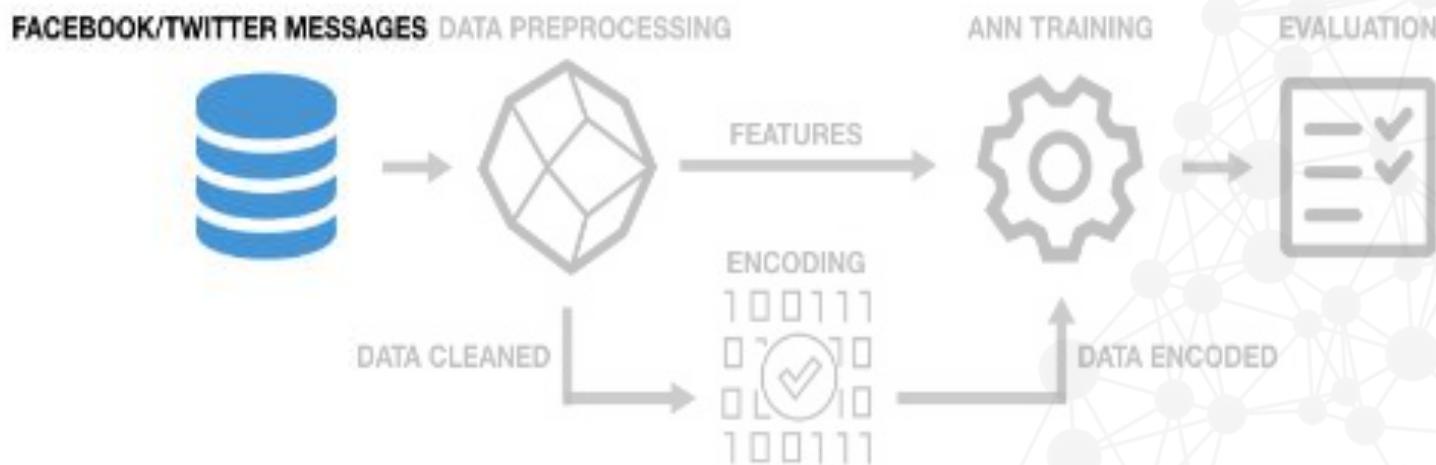


EVALITA

Giulio Bianchini, Lorenzo Ferri e Tommaso Giorni. "**Text Analysis for Hate Speech Detection in Italian Messages on Twitter and Facebook**." In: EVALITA@ CLiC-it. 2018



EVALITA



Dataset

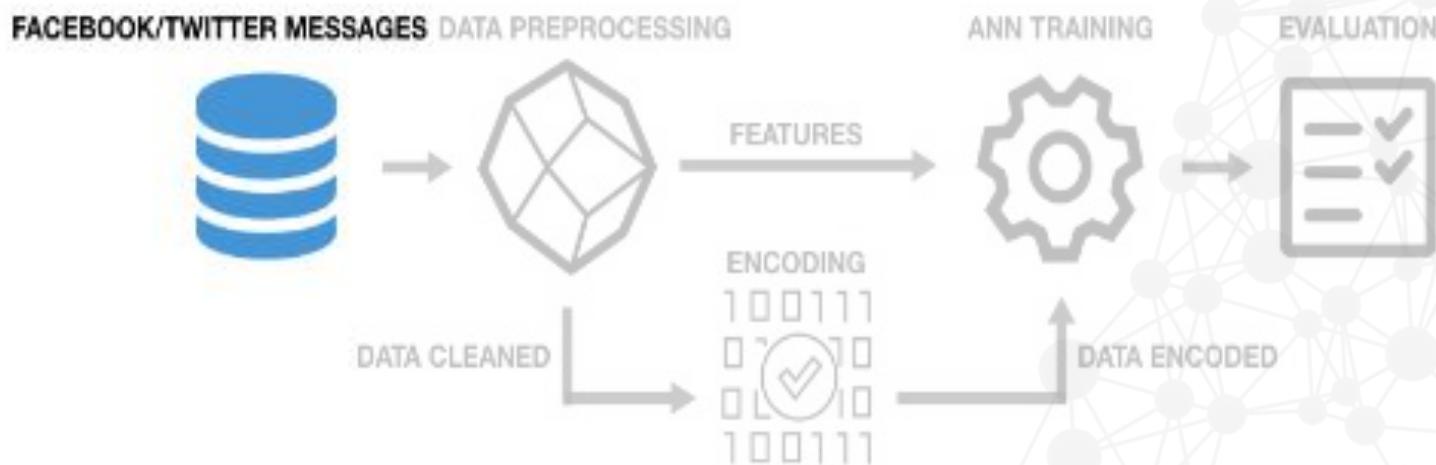
Sezione Facebook del dataset fornito da **EVALITA** per la competizione *HaSpeeDe*
4000 commenti Facebook (3000 per il training-set e 1000 per il test-set) a tema politico
Ogni commento è fornito di una label (**1** se il commento è offensivo / **0** se non è offensivo)

Io voterò no😊😊😊Renzi deve andare a casa😢😢😢	0
.....AZZI.....LORO !!!!!!!!!!	1
Malpezzi ma vaaaaaaaaaaa finiscila x piacere sei ridicolaaaaaaaaaa	1

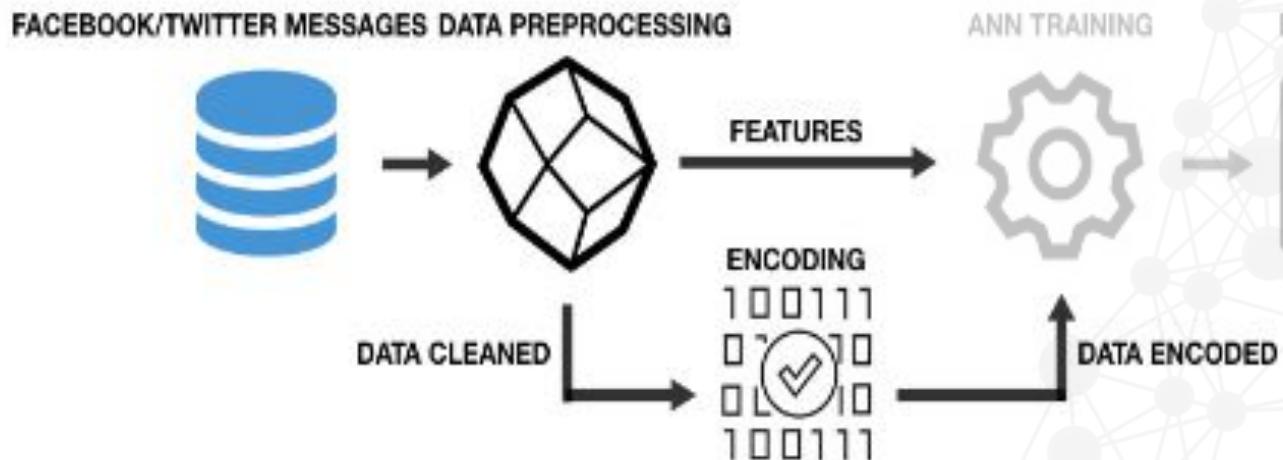
Il linguaggio di un'utenza di cultura medio-bassa e tipico dei social network, è solitamente povero, errato, sporco e ricco di imperfezioni

Necessaria fase di **Preprocessing**

EVALITA



EVALITA



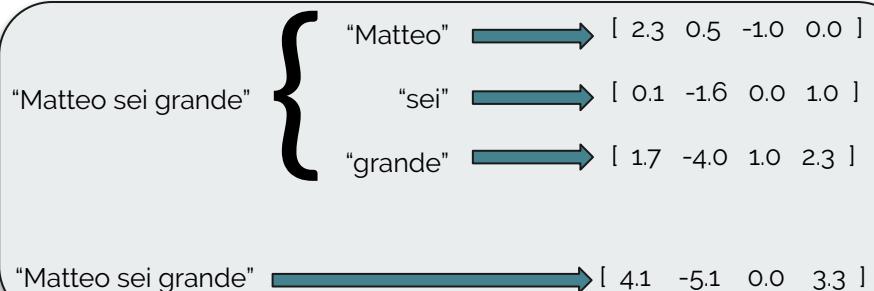
Word2Vec & Features

Word2Vec

Modello standard per l'**embedding** di parole.

Usa una rete neurale a 2 livelli allenata con un vasto corpus testuale. I valori associati ad una parola sono i pesi della rete

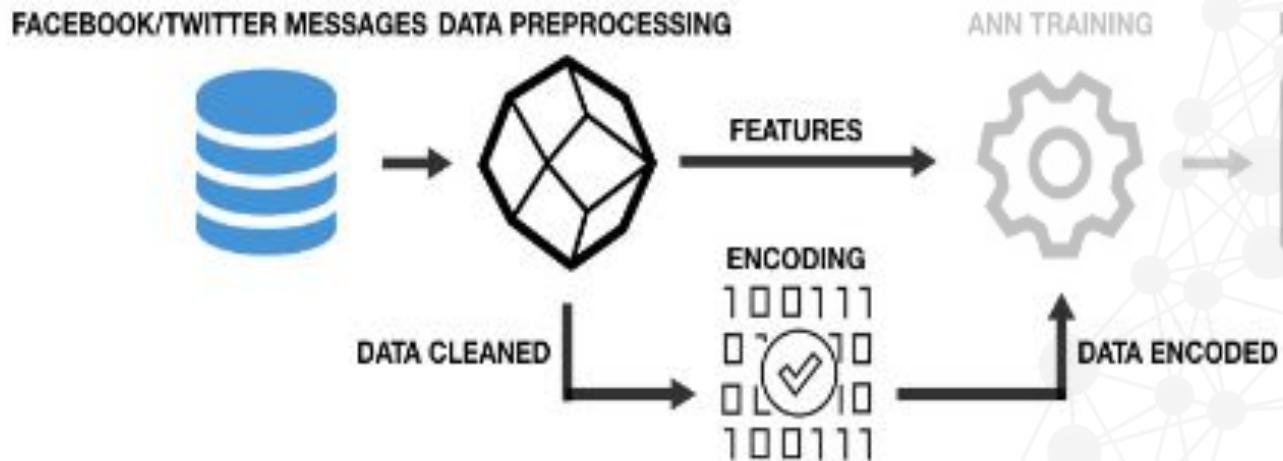
Crea uno spazio vettoriale che memorizza le informazioni sintattiche e semantiche delle parole



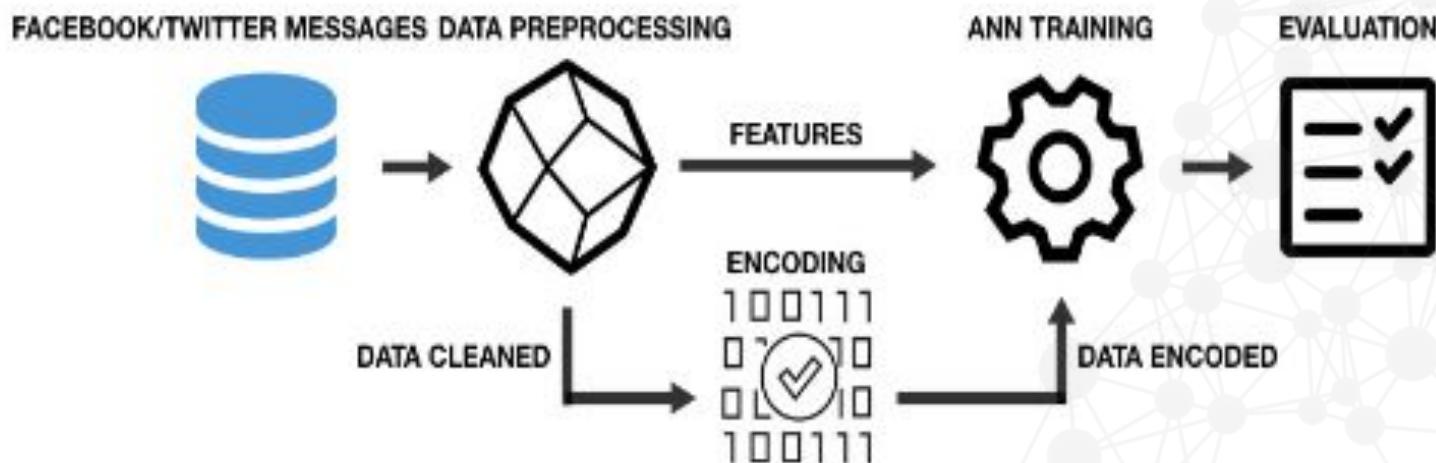
Features

- 1) Percentuale parolacce nella frase
- 2) Numero di parolacce
- 3) Polarità del commento tramite ***SentiWordNet***
- 4) Numero di proposizioni che compongono un commento
- 5) Numero di **?** e **!**
- 6) Percentuale di parole scritte in **CAPS-LOCK**
- 7) Numero di parole del commento
- 8) Somma Word2Vec delle parole
- 9) Media Word2Vec delle parole

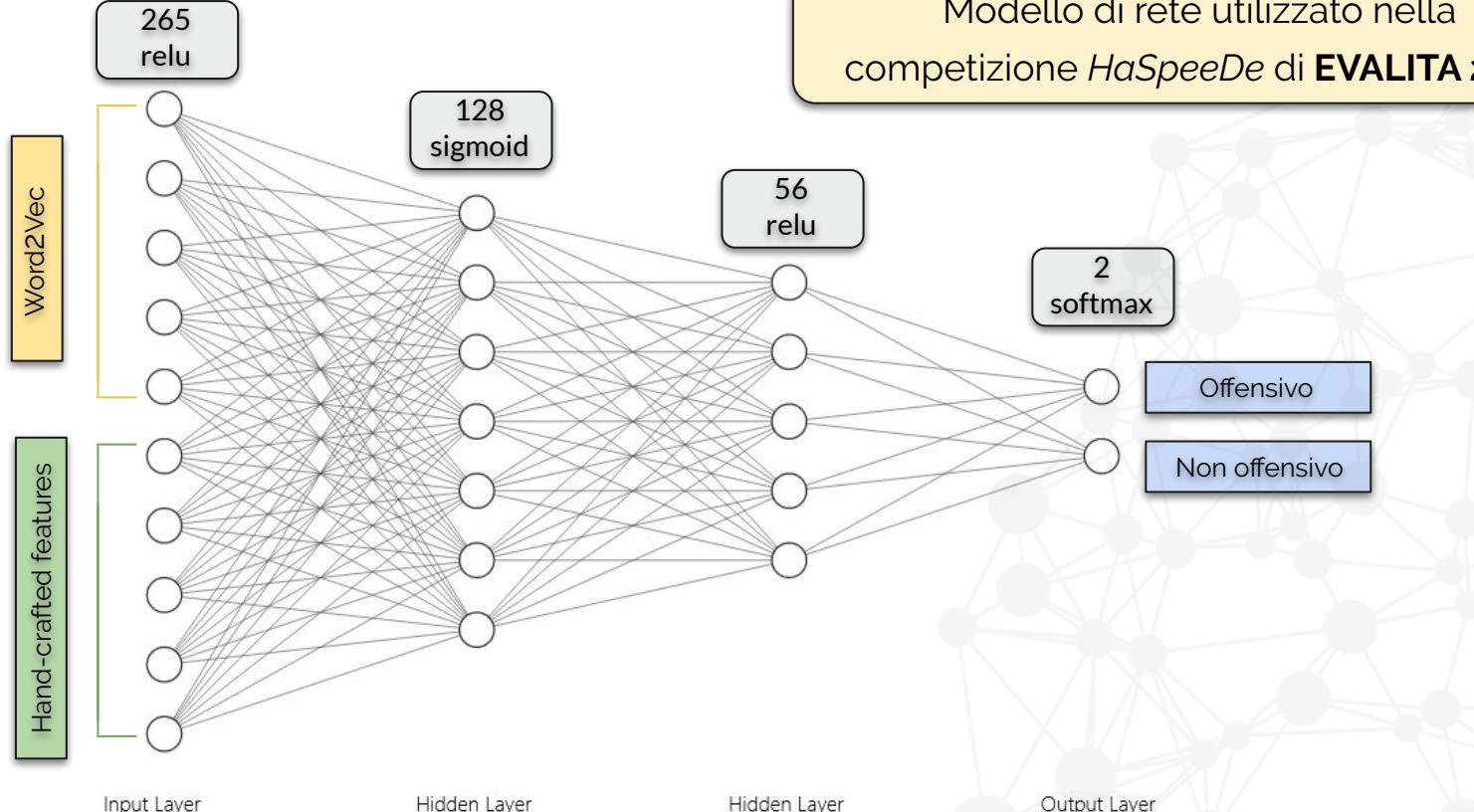
EVALITA



EVALITA



Multilayer Perceptron (MLP)

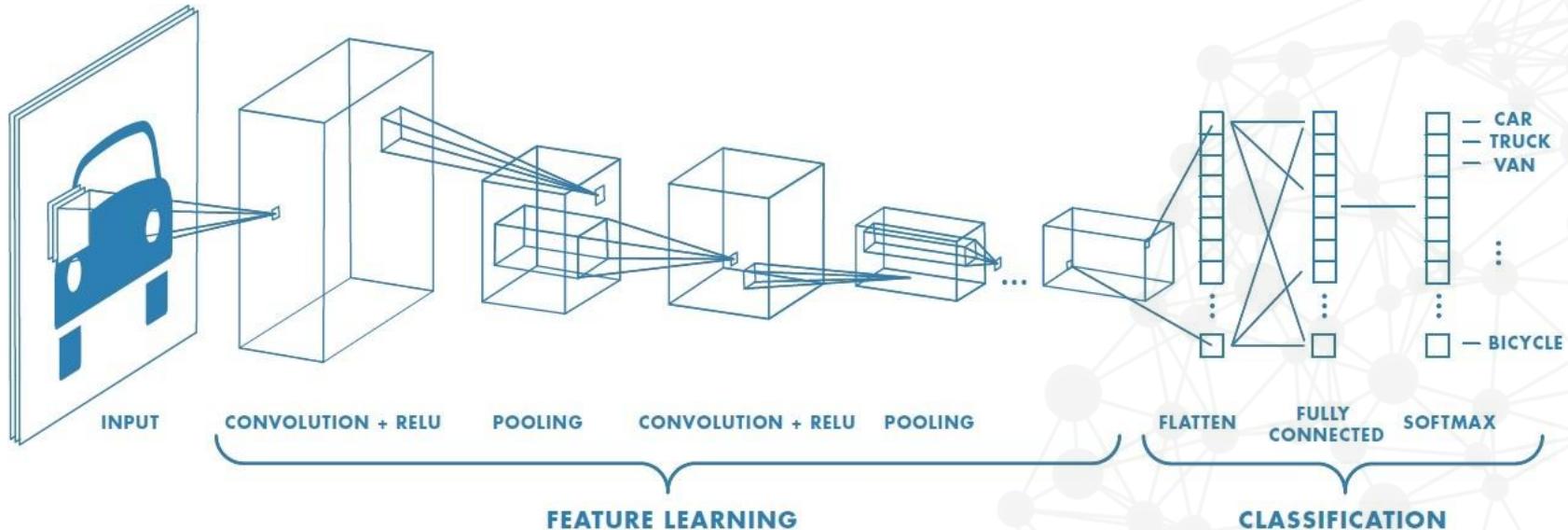


Outline

- Cos' è il *Natural Language Processing* e l'*Hate Speech Detection*
- Baseline model: *Multilayer Perceptron*
- Reti Convolutive
- Reti a Capsule
- Hate Speech Detection con *Reti Convolutive* e *a Capsule*
- Risultati



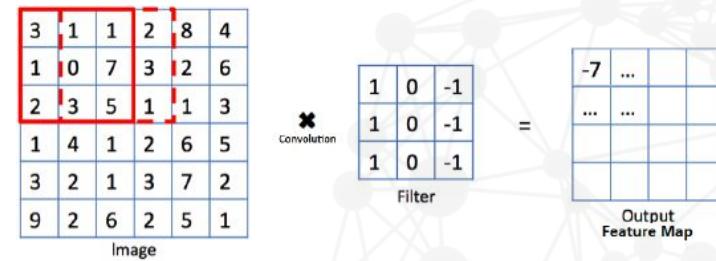
Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)

Il termine “*convolutional*” deriva dall'utilizzo dell'operazione di convoluzione utilizzata dalla rete per l'estrazione delle features dai dati in input

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x - y)dy$$



Vengono calcolate le **informazioni spaziali** tra features di uno stesso input

Il filtro, muovendosi nell'immagine, calcola le informazioni spaziali tra pixel vicini. Ad ogni passo non viene analizzato un solo dato singolarmente ma l'insieme di più dati ricoperti dallo stesso filtro

I dati ora hanno un'**importanza spaziale**, comportando un maggior contenuto informativo

Convolutional Neural Network (CNN)

Convolutional Layer: è il livello dove viene applicata l'operazione di convoluzione tra immagine e filtro. L'output è una **feature map** che evidenzia una certa caratteristica dell'immagine (angoli, colori, forme, ...)

Activation Layer: livello usato per garantire la *non-linearietà* delle feature maps

Pooling Layer: livello di compressione delle feature maps per ridurre la grandezza delle immagini mantenendo comunque le informazioni più importanti.
Garantisce la **Translation Invariance**

0	0	0	0	0	0
0	1	1	1	0	
0	1	1	1	0	
0	1	1	1	0	
0	0	0	0	0	0

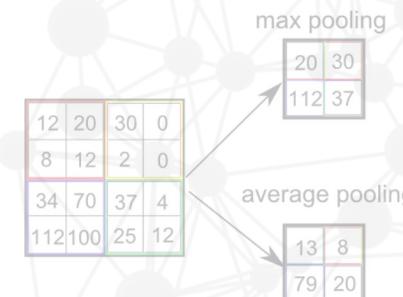
MaxPool

1	1	0
1	1	0
0	0	0

0	1	1	1	0	
0	1	1	1	0	
0	1	1	1	0	
0	0	0	0	0	0
0	0	0	0	0	0

MaxPool

1	1	0
1	1	0
0	0	0



Convolutional Neural Network (CNN)

Convolutional Layer: è il livello dove viene applicata l'operazione di convoluzione tra immagine e filtro. L'output è una **feature map** che evidenzia una certa caratteristica dell'immagine (angoli, colori, forme, ...)

Activation Layer: livello usato per garantire la *non-linearietà* delle feature maps

Pooling Layer: livello di compressione delle feature maps per ridurre la grandezza delle immagini mantenendo comunque le informazioni più importanti.
Garantisce la **Translation Invariance**

0	0	0	0	0	0
0	1	1	1	0	
0	1	1	1	0	
0	1	1	1	0	
0	0	0	0	0	0

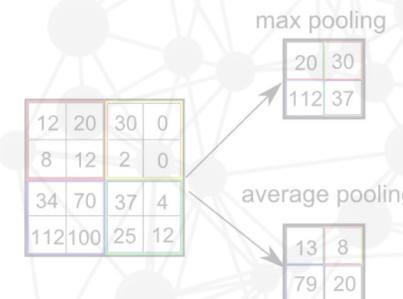
MaxPool

1	1	0
1	1	0
0	0	0

0	1	1	1	0	
0	1	1	1	0	
0	1	1	1	0	
0	0	0	0	0	0
0	0	0	0	0	0

MaxPool

1	1	0
1	1	0
0	0	0



Convolutional Neural Network (CNN)

Convolutional Layer: è il livello dove viene applicata l'operazione di convoluzione tra immagine e filtro. L'output è una **feature map** che evidenzia una certa caratteristica dell'immagine (angoli, colori, forme, ...)

Activation Layer: livello usato per garantire la *non-linearietà* delle feature maps

Pooling Layer: livello di compressione delle feature maps per ridurre la grandezza delle immagini mantenendo comunque le informazioni più importanti.
Garantisce la **Translation Invariance**

0	0	0	0	0	0
0	1	1	1	0	
0	1	1	1	0	
0	1	1	1	0	
0	0	0	0	0	0

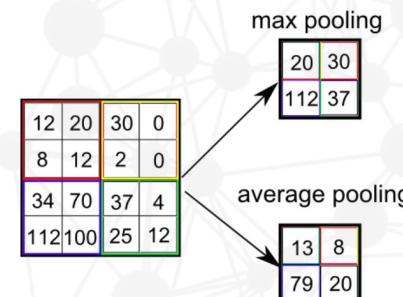
MaxPool

1	1	0
1	1	0
0	0	0

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0
0	0	0	0	0

MaxPool

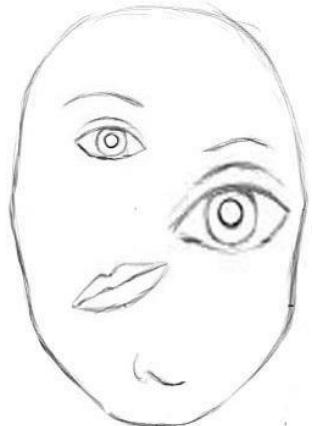
1	1	0
1	1	0
0	0	0



Outline

- Cos' è il *Natural Language Processing* e l'*Hate Speech Detection*
- Baseline model: *Multilayer Perceptron*
- Reti Convolutive
- Reti a Capsule
- Hate Speech Detection con *Reti Convolutive* e *a Capsule*
- Risultati





È una faccia?

Le CNN non mantengono le relazioni spaziali tra le varie componenti di un'immagine. La causa maggiore è il **Maxpooling**

Una CNN riconosce le singole componenti (occhio, naso) ma non riesce a relazionarle tra loro (gli occhi devono trovarsi sopra il naso)

Le CNN garantiscono la *Translation Invariance* ma non l' **Equivarianza**

Per questi motivi, una CNN classificherà l'immagine come una **FACCIA** sebbene non lo sia

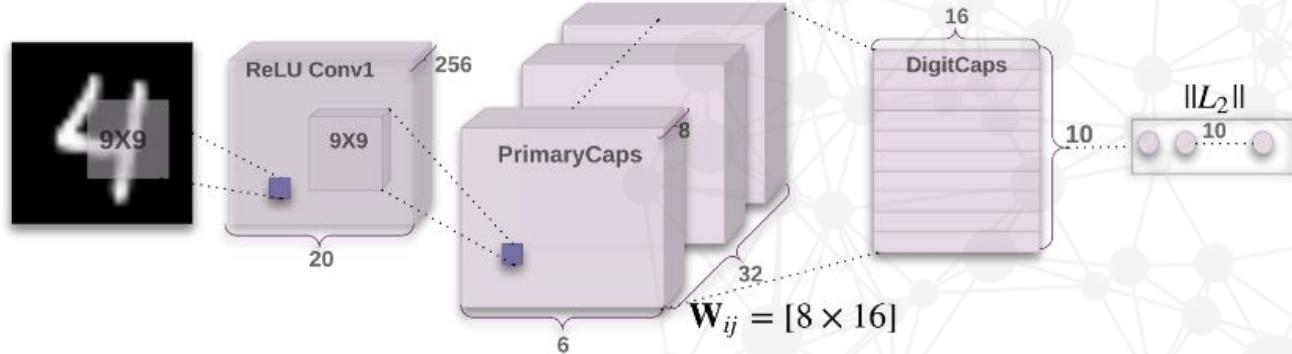
Capsule Network (CapsNet)

Nuovo modello di rete introdotto da **Geoffrey Hinton** nel 2017

Livelli di rete formati da **capsule**, un insieme di neuroni il cui output è chiamato **activity vector** caratterizzato da due grandezze: *lunghezza* e *direzione*

Un livello i è formato da n capsule e ognuna di queste è connessa a tutte le capsule del livello $i+1$

Ogni connessione è definita da un *coefficiente di accoppiamento* \mathbf{C}_{ij} e una *matrice peso* \mathbf{W}_{ij}



Routing-by-agreement

Capsule Network (CapsNet)

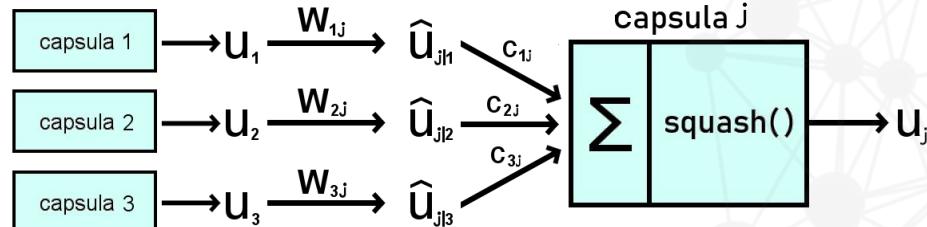
Il livello di pooling nelle CNN viene sostituito dal **routing-by-agreement**

Routing basato sulle **affinità** (*agreement*) tra le varie capsule di livelli consecutivi

$$\text{prediction-vector } i = W_{ij} \times \text{activity-vector } i$$

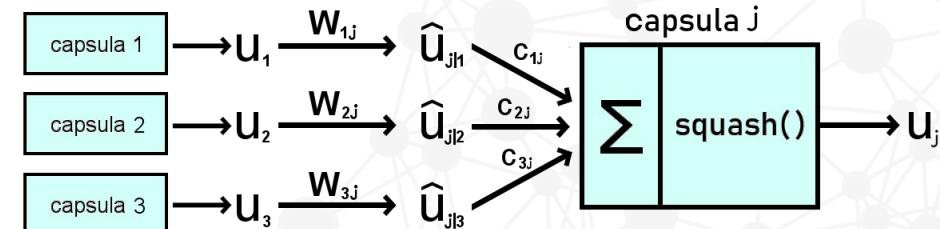
La capsula i cerca di prevedere qual è la capsula $i+1$ che calcola le features più affini alla propria

Dai vari C_{ij} dipende l'importanza o meno delle informazioni passate dalla capsula i alla capsula j



Capsule Network (CapsNet)

u_i	activity vector della capsula i
v_j	activity vector della capsula j
$\hat{u}_{j i} = W_{ij} \cdot u_i$	prediction vector della capsula i riguardo l'output della capsula j
$s_j = \sum_i c_{ij} \cdot \hat{u}_{j i}$	input totale alla capsula j
$a_{ij} = v_j \cdot \hat{u}_{j i}$	affinità tra capsula i e capsula j
$b_{ij} = b_{ij} + a_{ij}$	aggiornamento coefficiente di accoppiamento tra capsula i e capsula j
$c_{ij} = \frac{\exp(b_{ij})}{\sum_n \exp(b_{in})}$	softmax del coefficiente di accoppiamento tra capsula i e capsula j



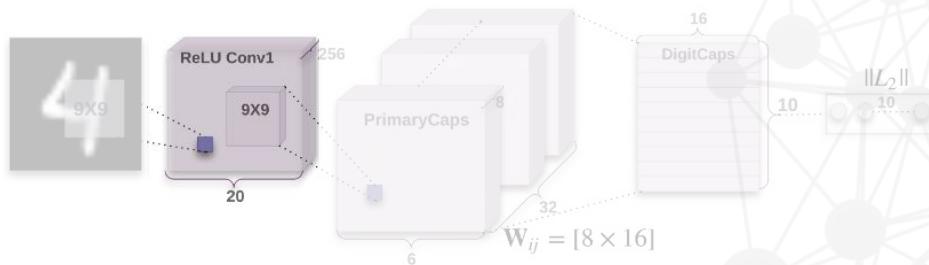
Capsule Network (CapsNet)

Convolutional Layer: è il livello dove viene applicata l'operazione di convoluzione tra immagine e filtro

Primary Capsule Layer: livello di capsule che utilizza le informazioni passate dal *Convolutional Layer*

Output Capsule Layer: livello di output (1 capsula per ogni output)

Length Layer: livello che calcola la *magnitudine* degli *activity vector* delle capsule-output per la classificazione



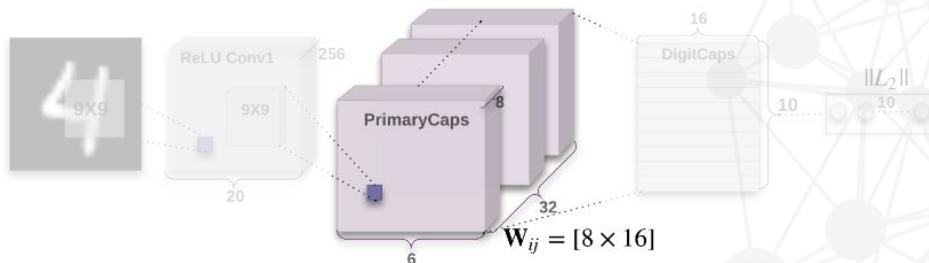
Capsule Network (CapsNet)

Convolutional Layer: è il livello dove viene applicata l'operazione di convoluzione tra immagine e filtro

Primary Capsule Layer: livello di capsule che utilizza le informazioni passate dal *Convolutional Layer*

Output Capsule Layer: livello di output (1 capsula per ogni output)

Length Layer: livello che calcola la *magnitudine* degli *activity vector* delle capsule-output per la classificazione



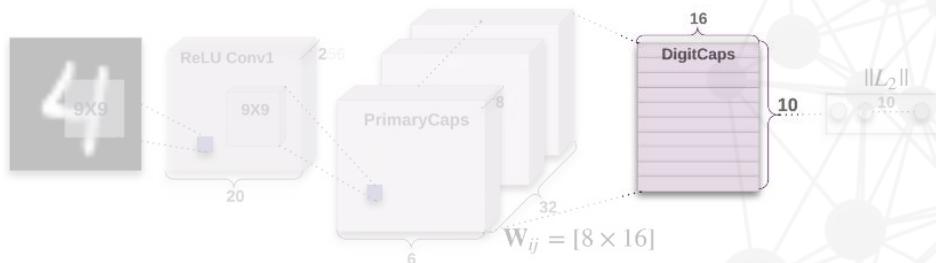
Capsule Network (CapsNet)

Convolutional Layer: è il livello dove viene applicata l'operazione di convoluzione tra immagine e filtro

Primary Capsule Layer: livello di capsule che utilizza le informazioni passate dal *Convolutional Layer*

Output Capsule Layer: livello di output (1 capsula per ogni output)

Length Layer: livello che calcola la *magnitudine* degli *activity vector* delle capsule-output per la classificazione



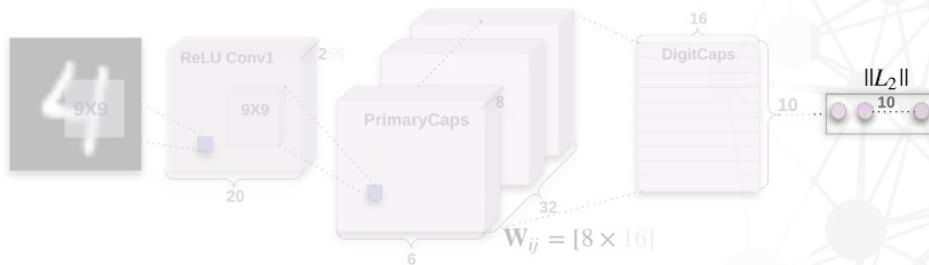
Capsule Network (CapsNet)

Convolutional Layer: è il livello dove viene applicata l'operazione di convoluzione tra immagine e filtro

Primary Capsule Layer: livello di capsule che utilizza le informazioni passate dal *Convolutional Layer*

Output Capsule Layer: livello di output (1 capsula per ogni output)

Length Layer: livello che calcola la *magnitudine* degli *activity vector* delle capsule-output per la classificazione

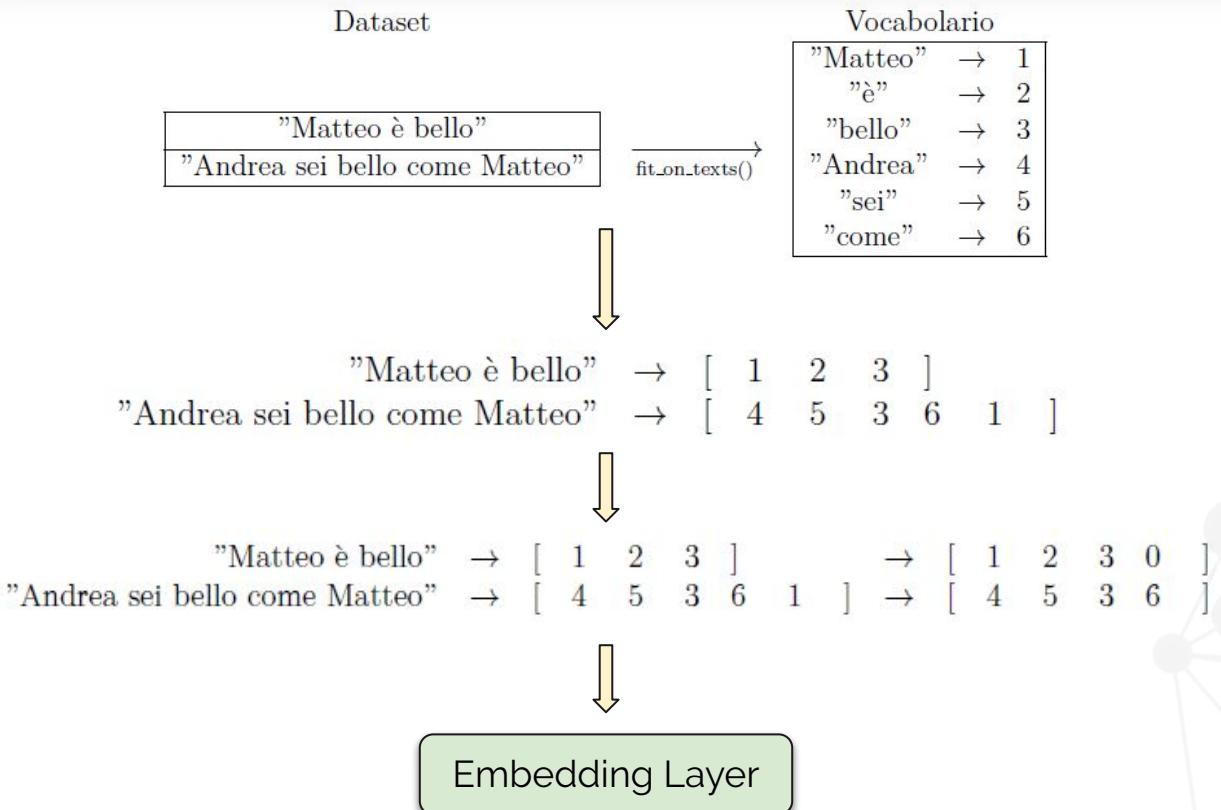


Outline

- Cos' è il *Natural Language Processing* e l'*Hate Speech Detection*
- Baseline model: *Multilayer Perceptron*
- Reti Convolutive
- Reti a Capsule
- Hate Speech Detection con *Reti Convolutive* e *a Capsule*
- Risultati



Da frase (stringa) a immagine (matrice)



Individuazione

Sostituzione

Padding

Embedding

Embedding

Da frase (stringa) a immagine (matrice)

Pesi Word2Vec

Matteo →	[-0.5 2.1 0.0]
è →	[1.2 0.1 -2.0]
bello →	[0.7 1.0 -0.1]

“Matteo è bello”

Tokenizer

[1 2 3 0]

Embedding

$$\begin{bmatrix} -0.5 & 2.1 & 0.0 \\ 1.2 & 0.1 & -2.0 \\ 0.7 & 1.0 & -0.1 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

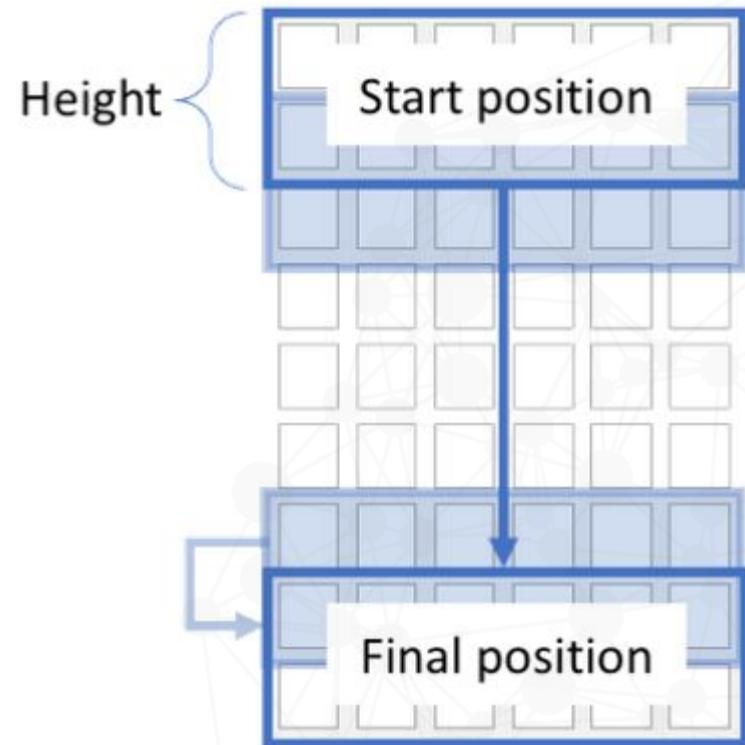
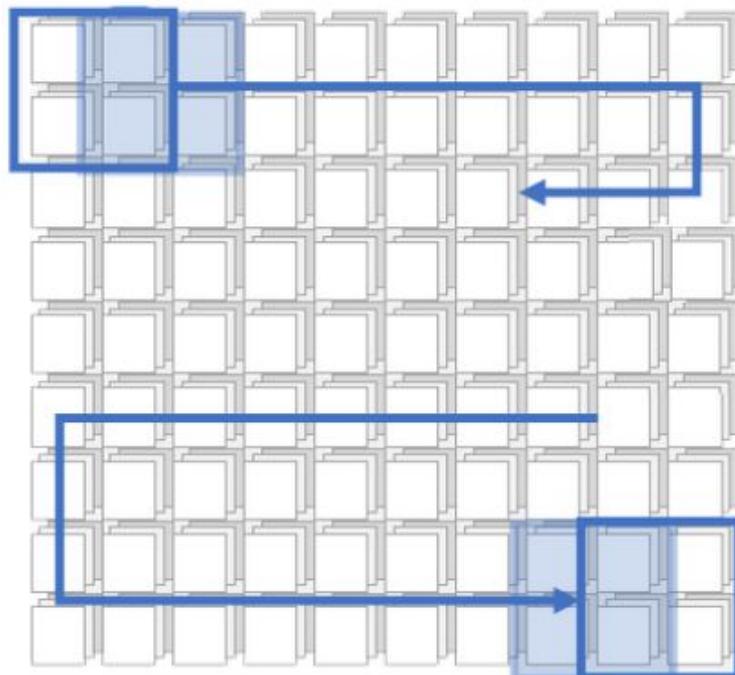
Stringa

Vettore

Matrice

2D Convolution VS 1D Convolution

Da frase (stringa) a immagine (matrice)



Modelli Sviluppati

1. Multilayer Perceptron (**MLP**)
2. Convolutional Network single-channel con 3 livelli convolutivi (**CNN**)
3. Come CNN + regolarizzatori kernel e bias (**CNNreg**)
4. Come CNN + regolatizzatori kernel e bias + adaptive learning rate (**CNNtot**)
5. Convolutional Newtork multichannel come CNNtot con 2 input + features hand-normalizzate (**CnnHN**)
6. Come CnnHN ma features NON normalizzate (**CnnNN**)
7. Come CnnHN ma features con normalizzazione min-max (**CnnSN**)
8. Convolutional Network multichannel con 3 input (**CNN3inp**)
9. Capsule Network con 1 nodo output e function-loss di Hinton (**CapNet_1_H**)
10. Capsule Network con 2 nodi output e function-loss di Hinton (**CapNet_2_H**)
11. Capsule Network con 1 nodo output e function-loss CrossEntropy (**CapNet_1_CE**)
12. Capsule Network con 2 nodi output e function-loss di CrossEntropy (**CapNet_2_CE**)

Outline

- Cos' è il *Natural Language Processing* e l'*Hate Speech Detection*
- Baseline model: *Multilayer Perceptron*
- Reti Convolutive
- Reti a Capsule
- Hate Speech Detection con *Reti Convolutive* e a *Capsule*
- **Risultati**



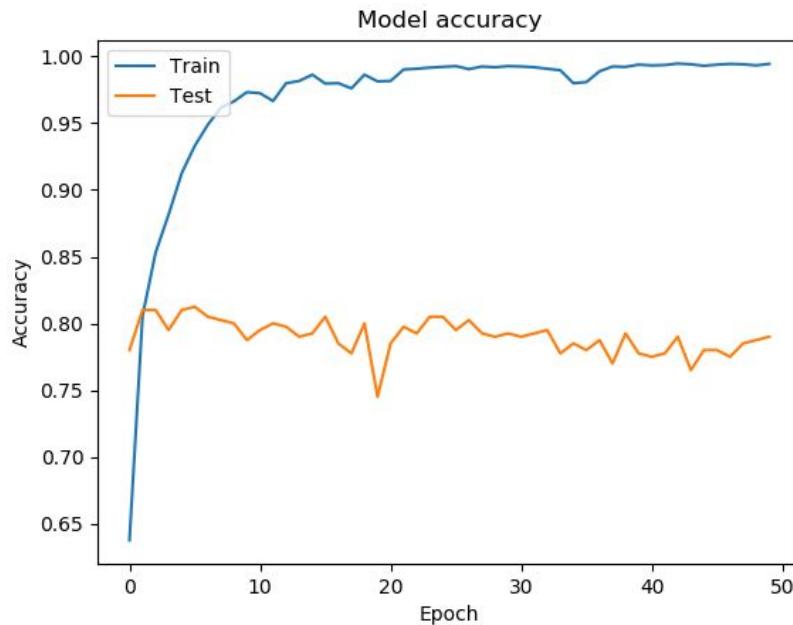
Risultati

Accuratezza media di 3 run con 50 epochhe divise per batch-size 64 e 128

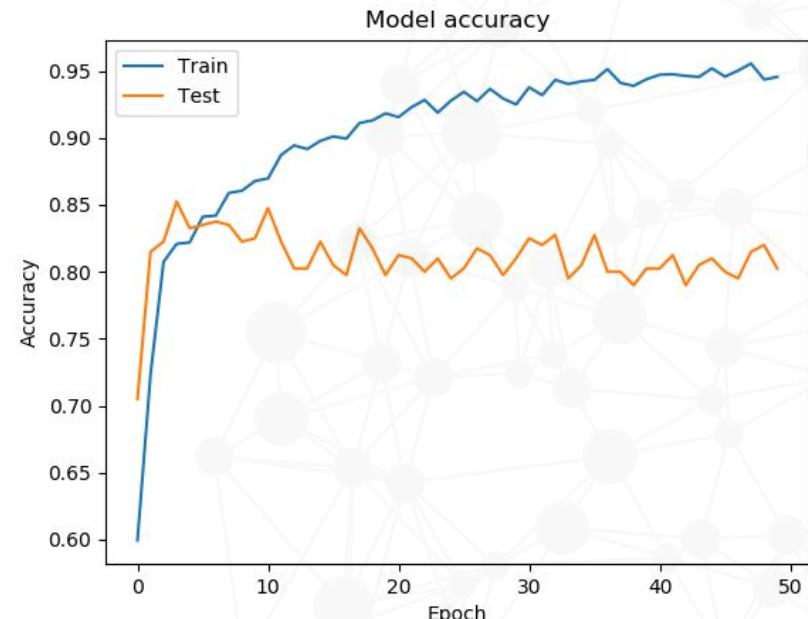
	64						128					
	5	10	15	20	25	30	5	10	15	20	25	30
MLP	0.815	0.819	0.813	0.816	0.81	0.816	0.811	0.809	0.819	0.819	0.818	0.811
CNN	0.78	0.788	0.784	0.789	0.781	0.775	0.805	0.793	0.811	0.799	0.8	0.813
CNNreg	0.836	0.831	0.828	0.82	0.812	0.823	0.831	0.816	0.838	0.81	0.822	0.82
CNNtot	0.837	0.839	0.842	0.842	0.839	0.836	0.845	0.845	0.835	0.838	0.841	0.844
CnnHN	0.7	0.696	0.689	0.688	0.688	0.688	0.701	0.753	0.763	0.769	0.769	0.766
CnnNN	0.637	0.666	0.659	0.659	0.657	0.663	0.647	0.671	0.677	0.676	0.678	0.678
CnnSN	0.832	0.833	0.827	0.831	0.82	0.828	0.806	0.803	0.82	0.821	0.828	0.826
CNN3inp	0.825	0.84	0.842	0.84	0.842	0.846	0.818	0.827	0.831	0.831	0.834	0.834
CapNet_1_H	0.81	0.798	0.798	0.801	0.798	0.798	0.812	0.805	0.801	0.8	0.8	0.798
CapNet_1_CE	0.778	0.772	0.772	0.769	0.77	0.768	0.785	0.775	0.773	0.768	0.768	0.768
CapNet_2_H	0.269	0.327	0.325	0.332	0.343	0.358	0.435	0.614	0.569	0.558	0.582	0.579
CapNet_2_CE	0.829	0.823	0.821	0.825	0.825	0.822	0.829	0.827	0.821	0.821	0.823	0.821

CNN VS CNNreg

Convolutional Network single-channel
con 3 livelli convolutivi

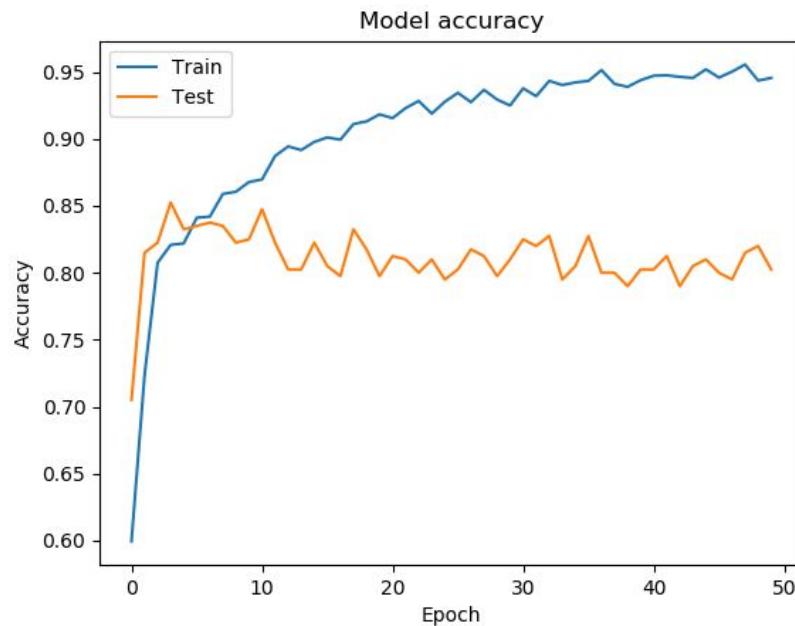


Convolutional Network single-channel con 3
livelli convolutivi + regolarizzatori kernel/bias

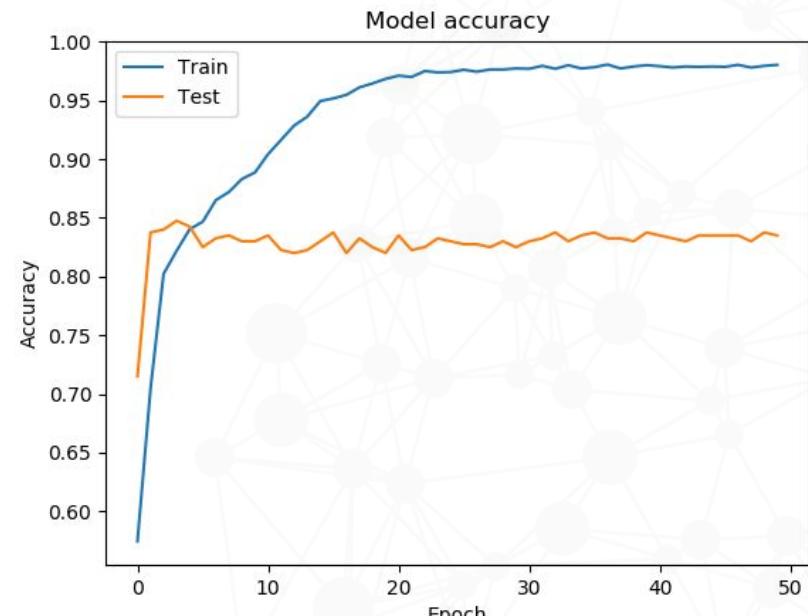


CNNreg VS CNNtot

Convolutional Network single-channel con 3 livelli convolutivi + regolarizzatori kernel/bias

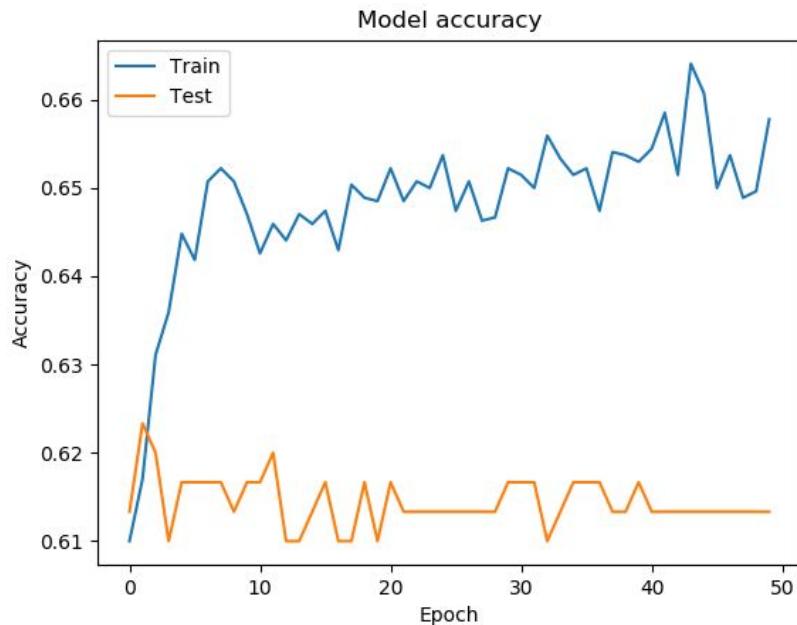


Convolutional Network single-channel con 3 livelli convolutivi + regolarizzatori kernel/bias + adaptive learning rate

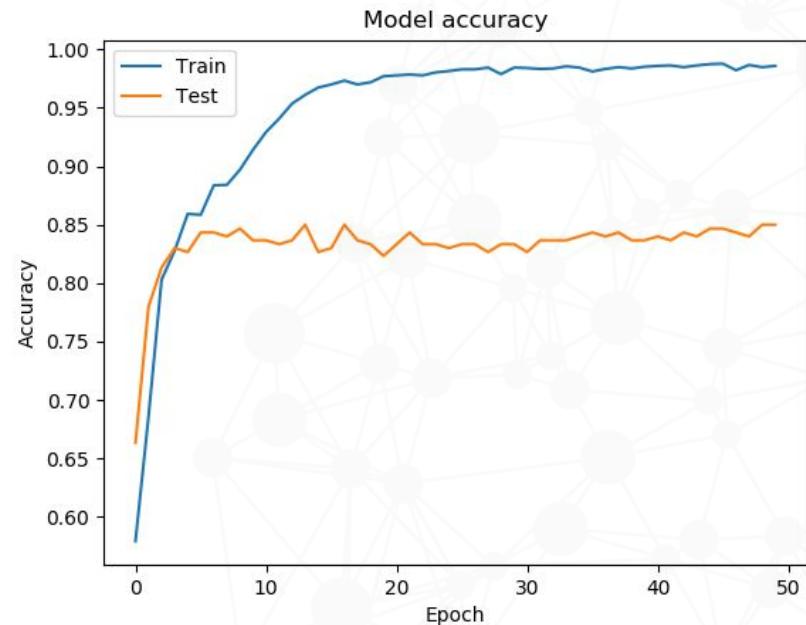


CnnNN VS CnnSN

Convolutional Network multi-channel
con features NON normalizzate

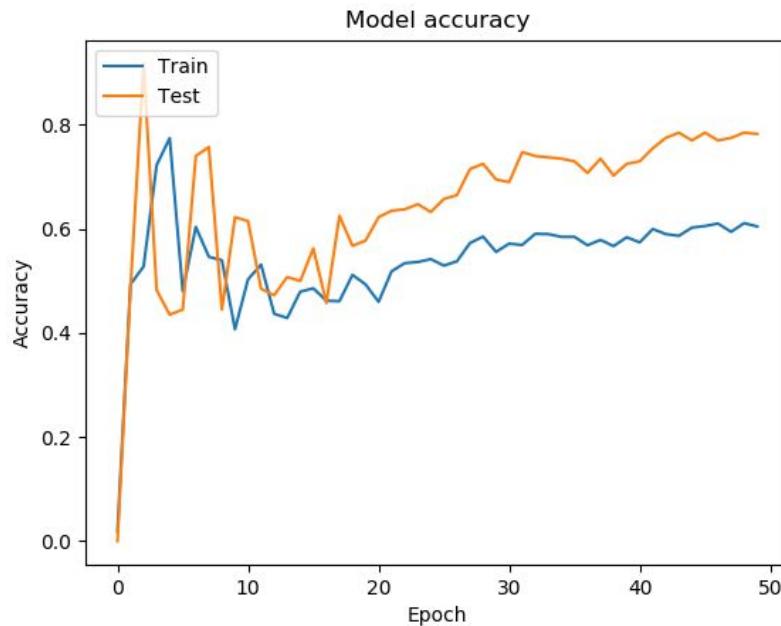


Convolutional Network multi-channel
con normalizzazione min-max

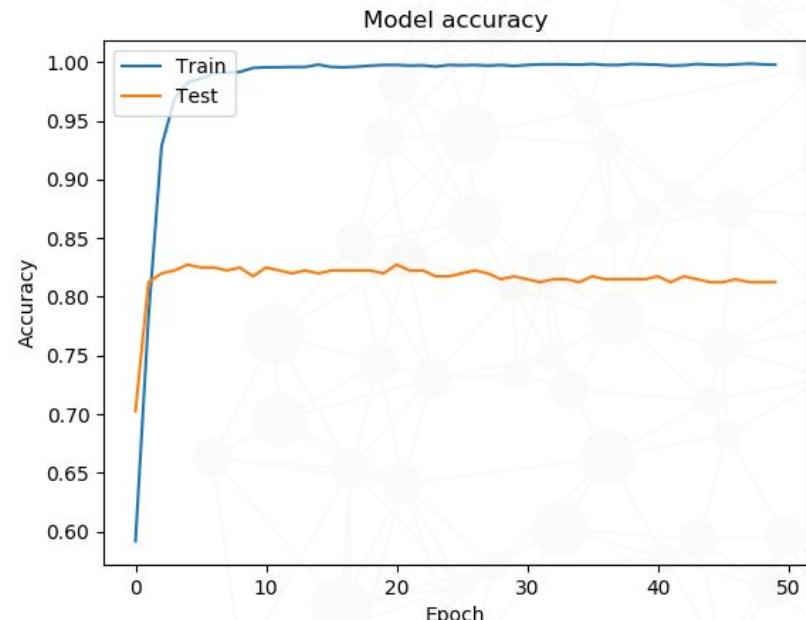


CapNet_2_H VS CapNet_2_CE

Capsule Network di Hinton con 2 nodi output e loss-function di Hinton



Capsule Network di Hinton con 2 nodi output e loss-function crossentropy



Considerazioni Finali

	64						128					
	5	10	15	20	25	30	5	10	15	20	25	30
MLP	0.815	0.819	0.813	0.816	0.81	0.816	0.811	0.809	0.819	0.819	0.818	0.811
CNNtot	0.837	0.839	0.842	0.842	0.839	0.836	0.845	0.845	0.835	0.838	0.841	0.844
CnnSN	0.832	0.833	0.827	0.831	0.82	0.828	0.806	0.803	0.82	0.821	0.828	0.826
CNN3inp	0.825	0.84	0.842	0.84	0.842	0.846	0.818	0.827	0.831	0.831	0.834	0.834
CapNet_2_CE	0.829	0.823	0.821	0.825	0.825	0.822	0.829	0.827	0.821	0.821	0.823	0.821

	MLP	CNNtot	CnnSN	CNN3inp	CapNet_2_CE
accuracy	0.819	0.845	0.833	0.846	0.829
parametri	108 074	310 257	312 305	712 305	1 107 008
secondi	6	143	103	331	802

Lavori Futuri

- Testare modelli più complessi di Capsule Network
- Capire perché non funziona il Word2Vec nelle CapsNet e testare altre soluzioni di embedding
- Metodo **out-of-the-box** per l'utilizzo di features aggiuntive nelle Reti Convolute

Grazie a tutti per l'attenzione!



Lorenzo Ferri
ferrilorenzo62@gmail.com