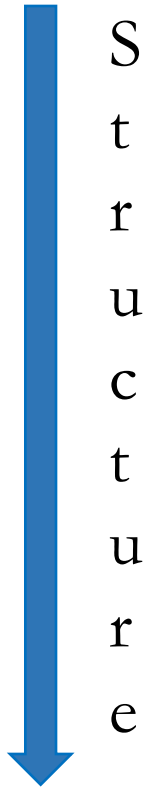
A Gentle Introduction to Graph Neural Networks: From Graph Attention Networks to Social Influence Modeling

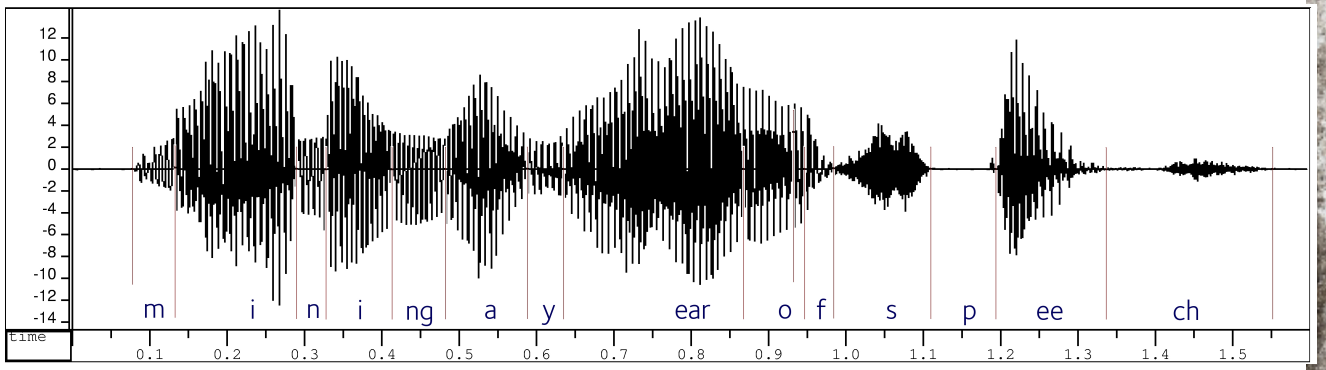
Simone Azeglio

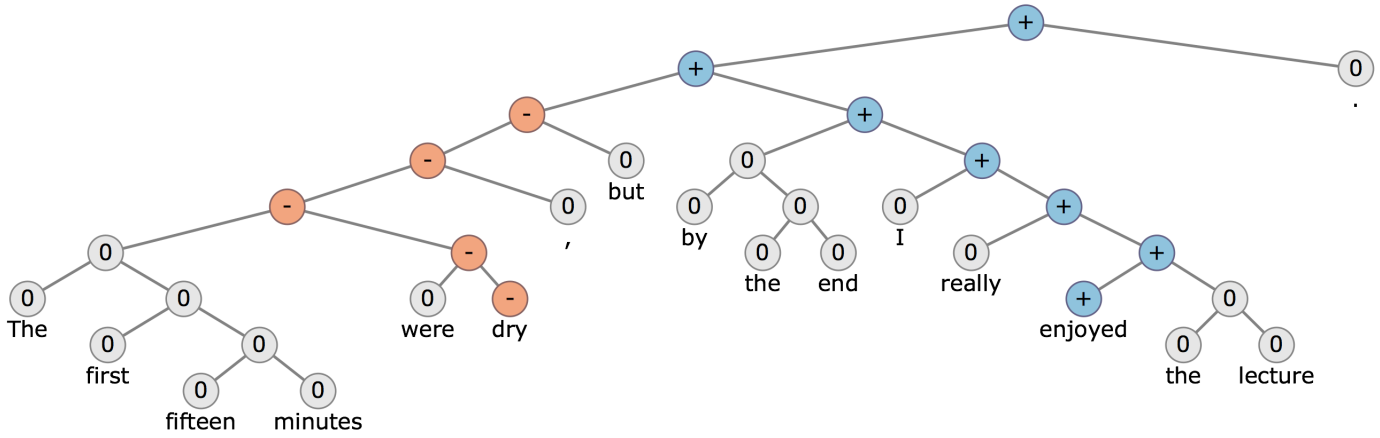
Overview

- The Success of Deep Learning: Euclidean Data
- A New Challenge: Graph Structured Data
- CNN on Graphs with Spatial Filters
- Attention Mechanisms: GATs
- DeepInf: Social Influence Prediction



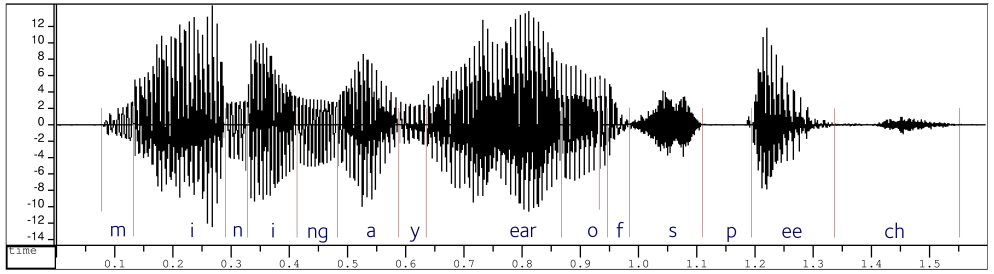
The Success of Deep Learning: Euclidean Data



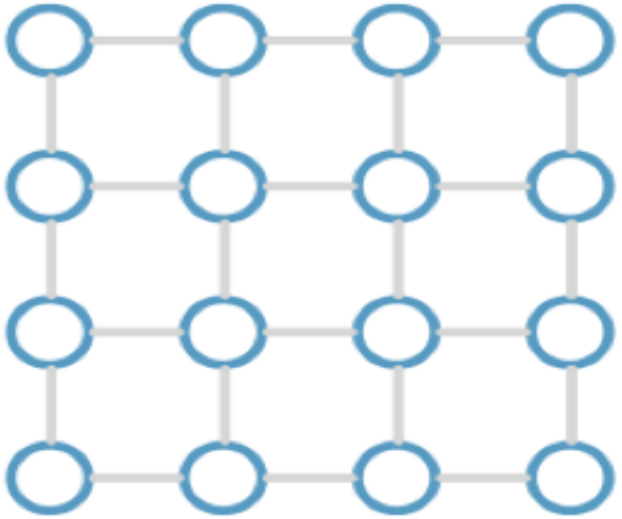


IMAGENET

The Structure is Stable: Euclidean Data



1D grid



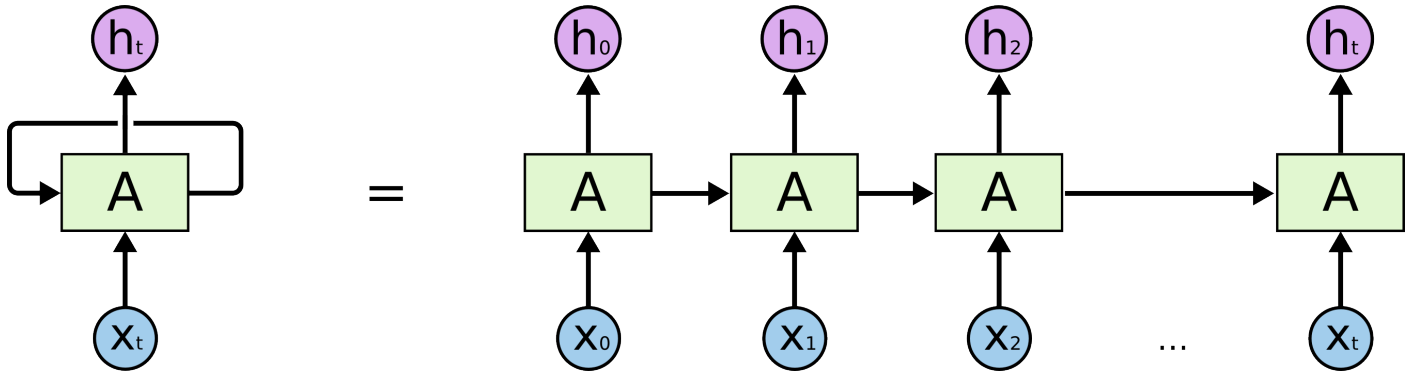
2D grid

RNNs and CNNs

How to deal with that?

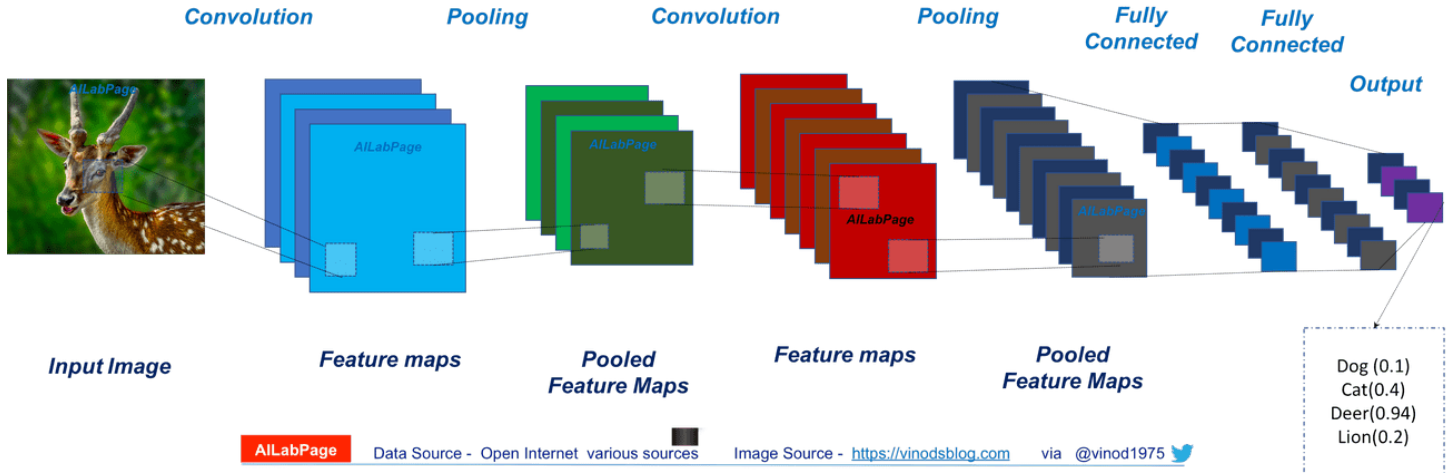
Introduce *time* in your network and you get a **RNN**

1-D **Recurrent Neural Networks**



[Image Source : colah]

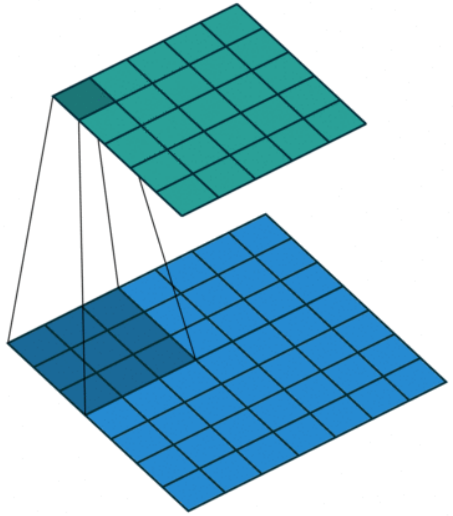
2-D **Convolutional Neural Networks**



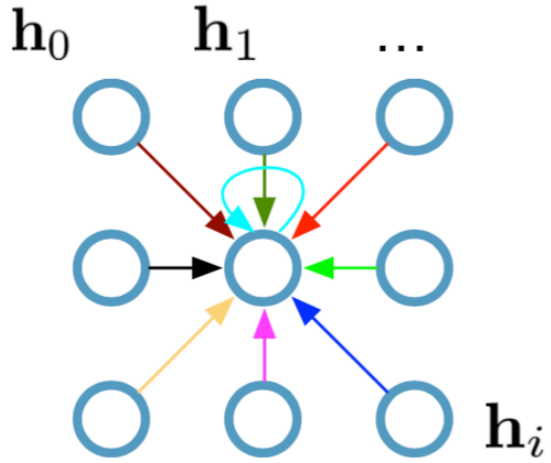
CNNs are Multi-Layer-Perceptrons on Steroids (i.e. *weights sharing*)

CNN Layers

Single CNN layer with **3x3** filter



[Animation Source : vdumoulin]



Update for a single pixel:

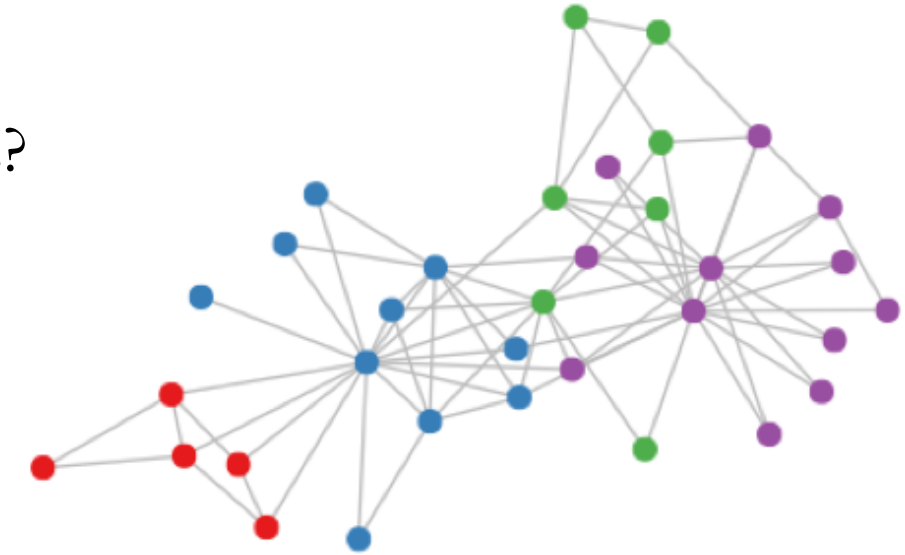
- Transform neighbors individually $\mathbf{W}_i h_i$
- Add everything up $\sum_i \mathbf{W}_i h_i$

Full update: $$\mathbf{h}_4^{l+1} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

A New Challenge: Graph Structured Data

What if our data look like this?

Or like this?

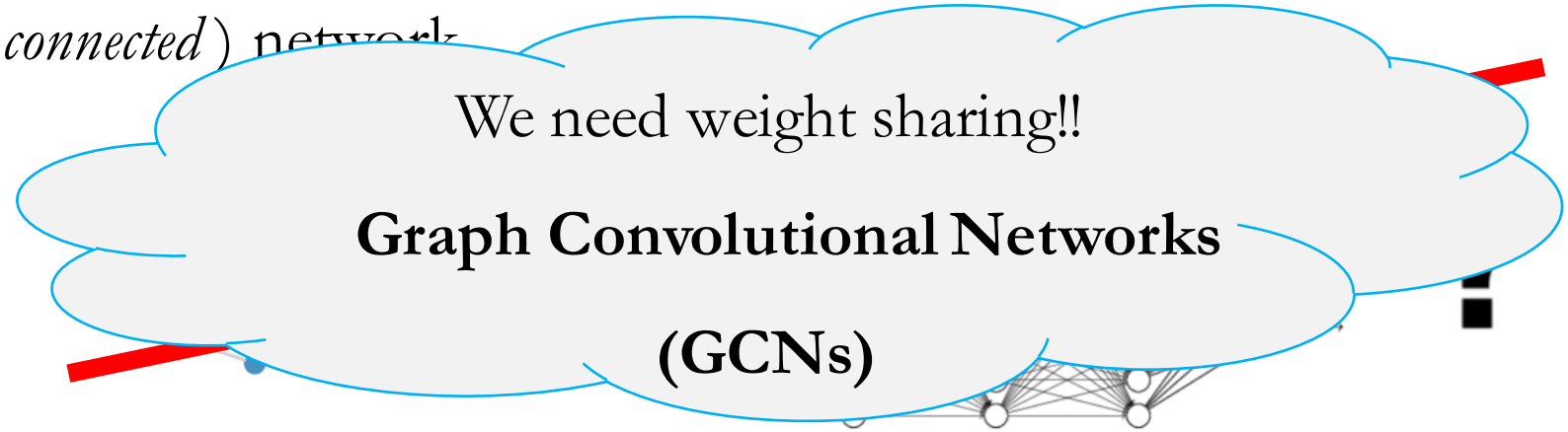


Real-world examples:

- Social Networks
- World-Wide-Web
- Telecommunication Networks
- …

A Naive Approach

- Take adjacency matrix $\mathbf{A}$ and feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$
- Concatenate them in $\mathbf{X}_{in} = [\mathbf{X}, \mathbf{A}] \in \mathbb{R}^{N \times (N+F)}$
- Feed them into a deep (*fully connected*) network
- Done?



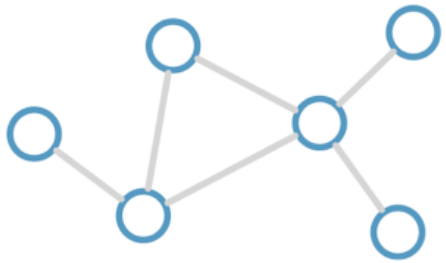
Problems:

- Huge number of parameters $\mathcal{O}(N)$
- Does not generalize across graphs
- Needs to be re-trained if number of parameters changes

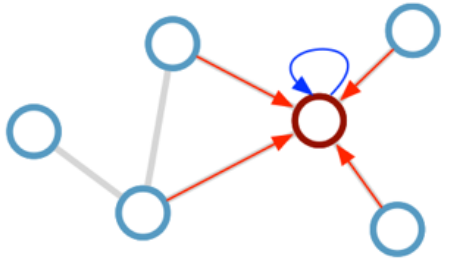
CNNs on Graphs with Spatial Filters

F. Scarselli et al, IEEE Transaction on Neural Networks (2009)

Consider this undirected graph



Calculate update for node in red



Update rule: $$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

$\mathcal{N}_i$ neighbor. indices

c_{ij} norm. constant

How is this related to spectral graphs?

- Localized 1st-order approximation of spectral filters

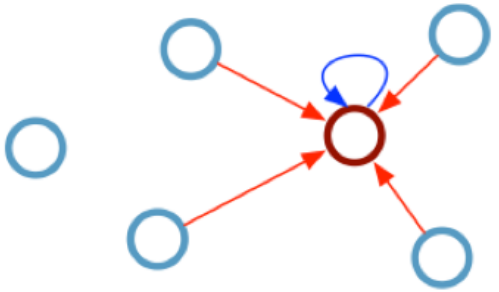
Kipf & Welling, ICLR (2017)

Vectorized Form

Kipf & Welling, ICLR (2017)

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{H}^{(l)}\mathbf{W}_0^{(l)} + \tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}_1^{(l)}\right)$$

With $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$

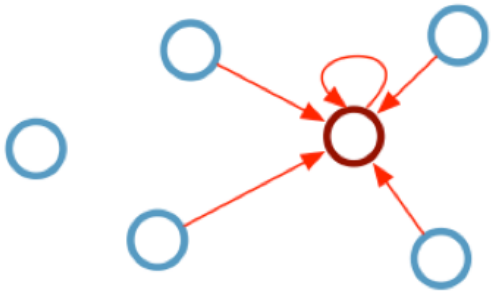


$$\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)^T}, \ldots, \mathbf{h}_N^{(l)^T}]^T$$

Or treat self-connections the same way:

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}_1^{(l)}\right)$$

With $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_N)\tilde{\mathbf{D}}^{-\frac{1}{2}}$



$$\tilde{D}_{ii} = \sum_j (A_{ij} + \delta_{ij})$$

A is tipically **sparse**:

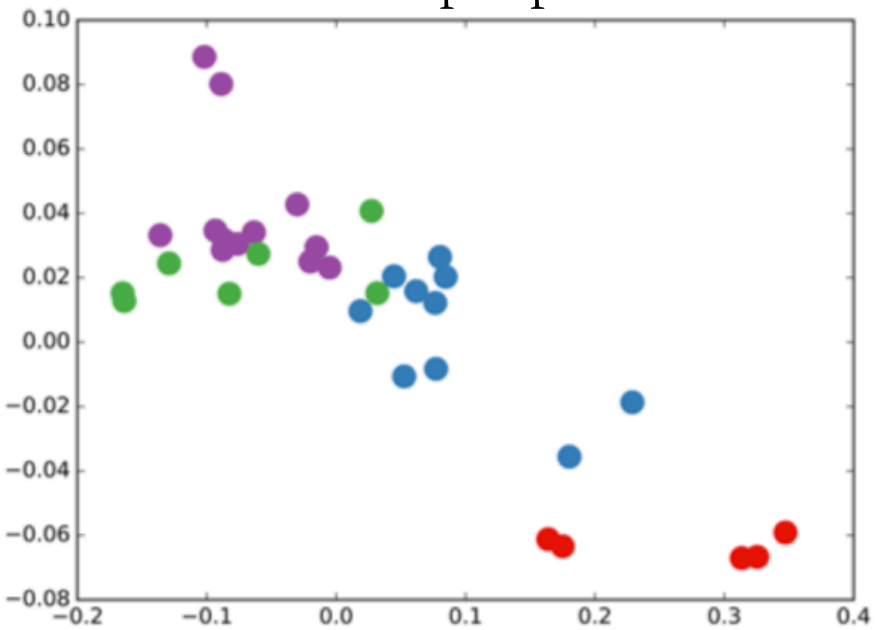
- We can use sparse matrix multiplications! $\mathcal{O}(|L|)$

What does it do?

Forward pass trough untrained 3-layers **GCN** model

Randomly initialized parameters

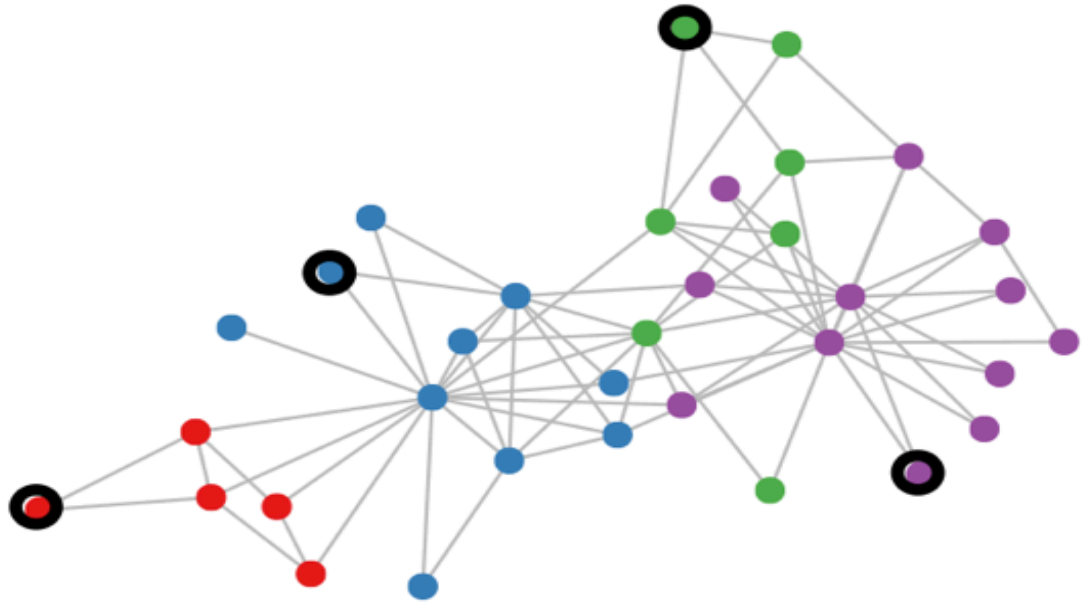
2-dim output per node



Produces (useful?) embeddings

Semi-Supervised Classification on Graphs

Setting:
Some nodes are labeled (black circles)
All other nodes are unlabeled



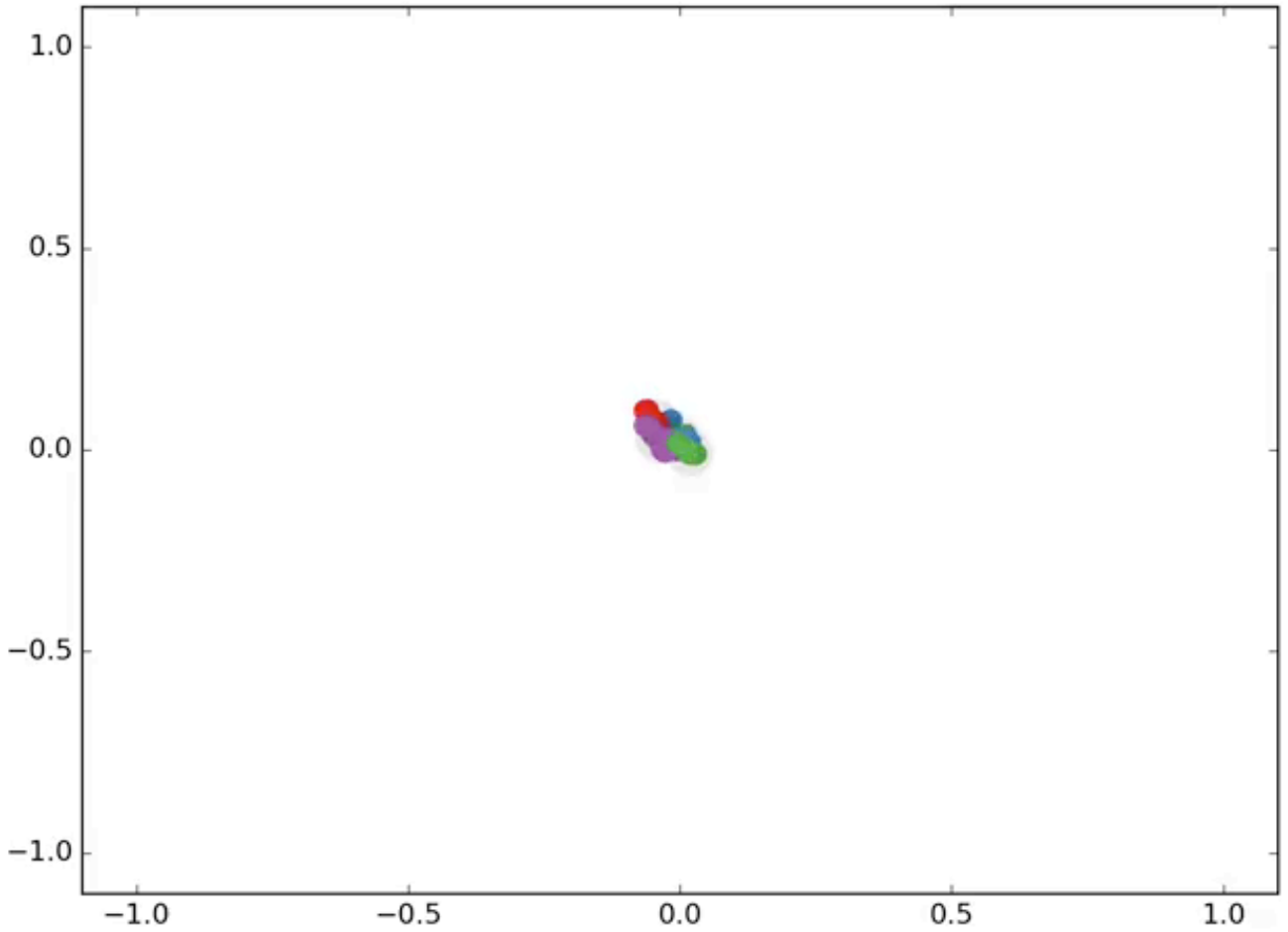
Task:
Predict labels of unlabeled nodes

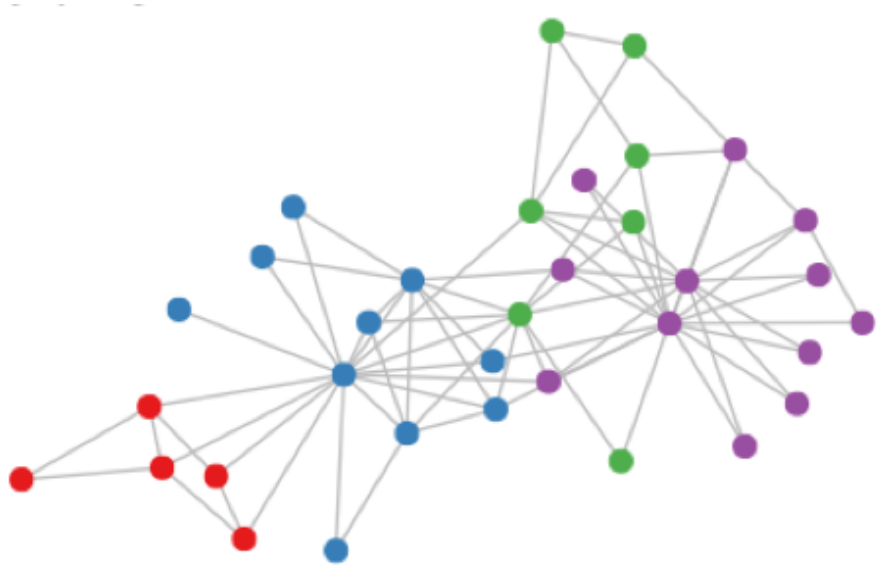
Standard approach:
Graph-based regularization ("smoothness constraints")

X.Zhu et al, ICML (2003)

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{reg} \quad \text{with} \quad \mathcal{L}_{reg} = \sum_{i,j} A_{ij} ||f(X_i) - f(X_j)||^2$$

Toy Example (Semi-Supervised Learning)

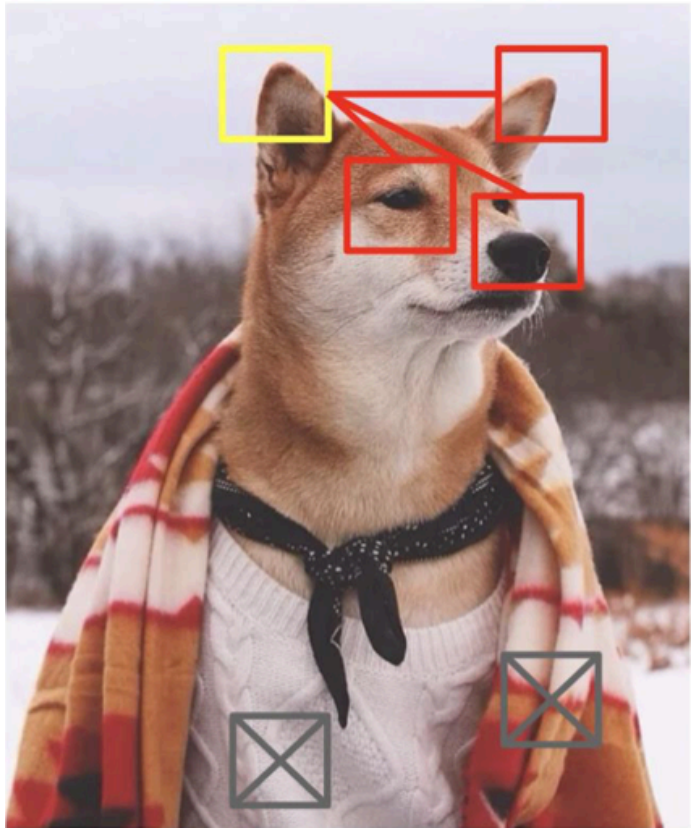




[Animation Source: tkpif]

Attention Mechanisms

We deduce something by paying attention to something **which is relatively more important**



Graph Attention Networks

P.Velickovic, arXiv:1710.10903 [stat.ML](2017)

Do you recall GCNs?

They learn Convolution Weights

$$\mathbf{h}_i^{(l+1)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)} \Big)$$

What's new with GATs?

They learn Convolution Weights

and Attention Coefficients

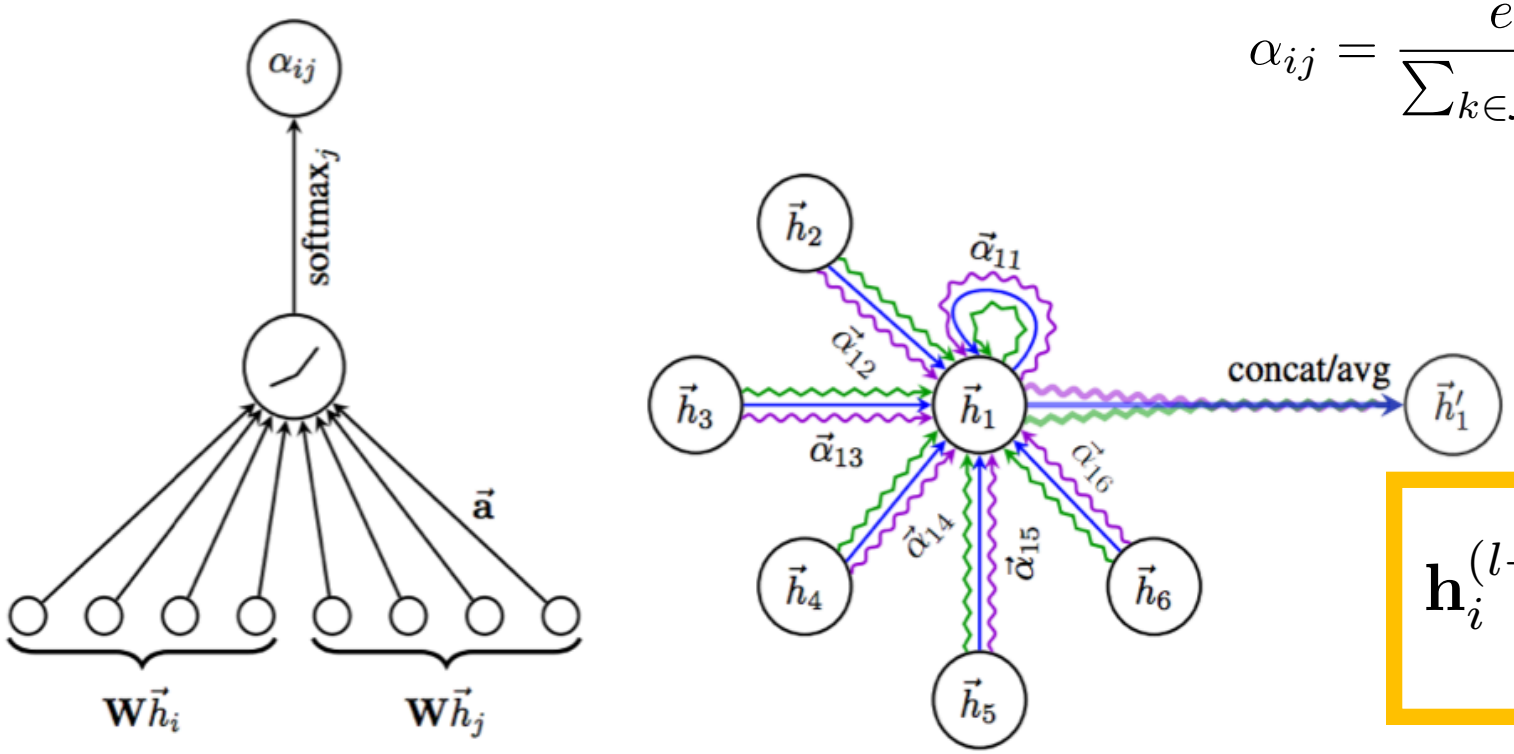
$$\mathbf{h}_i^{(l+1)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)} \Big)$$

With $$\alpha_{ij} = \frac{exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}h_i || \mathbf{W}h_j]))}{\sum_{k \in \mathcal{N}_i} exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}h_i || \mathbf{W}h_k]))}$$

A.Vaswani et al, arXiv:1706.03762 [cs.CL](2017)

Graph Attention Networks: Multi-head Attention

$$\alpha_{ij} = \frac{exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}h_i||\mathbf{W}h_j]))}{\sum_{k\in\mathcal{N}_i} exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}h_i||\mathbf{W}h_k]))}$$



Average:

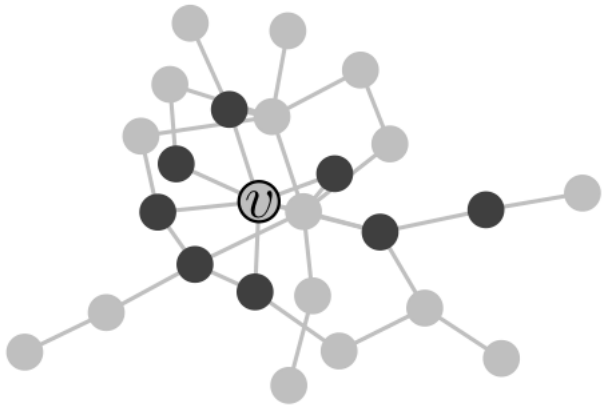
$$\mathbf{h}_i^{(l+1)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)}\mathbf{W}^{(l)}\mathbf{h}_j^{(l)}\right)$$

Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

DeepInf: Social Influence Prediction with GATs

J.Qiu et al, arXiv:1807.05560 [cs.SI](2018)

1. Let $G = (V, E)$ be a static network where it's *r-neighbors* are defined as $\Gamma_v^r = \{u : d(u, v) \leq r\}$



2. The *r-ego network* of v is the subnetwork G_v^r induced by Γ_v^r

3. Users in social networks perform *social actions*: at each timestamp t, we observe a binary action status of user u, $s_u^t \in \{0, 1\}$

Problem: Social Influence Locality $P(s_v^{t+\Delta t} | G_v^r, S_v^t)$ $S_v^t = \{s_u^t : u \in \Gamma_v^r \setminus \{v\}\}$

It can be reformulated as a binary graph classification problem

$$\mathcal{L}(\Theta) = -\sum_{i=1}^{N} \log\left(P_\Theta\left(s_{v_i}^{t_i+\Delta t} \middle| G_{v_i}^r, S_{v_i}^{t_i}\right)\right)$$

DeepInf: Social Influence Prediction with GATs

	OAG	Digg	Twitter	Weibo
$\|V\|$	953,675	279,630	456,626	1,776,950
$\|E\|$	4,151,463	1,548,126	12,508,413	308,489,739
N	499,848	24,428	499,160	779,164

Data	Model	AUC	Prec.	Rec.	F1
OAG	LR	65.55	32.26	**69.97**	44.16
	SVM	65.48	32.17	69.82	44.04
	PSCN	69.16	36.45	64.64	46.61
	DeepInf-GAT	**71.79**	**40.77**	60.97	**48.86**
Digg	LR	84.72	56.78	73.12	63.92
	SVM	86.01	63.42	67.34	65.32
	PSCN	87.37	64.75	68.15	66.40
	DeepInf-GAT	**90.65**	**66.82**	**78.49**	**72.19**
Twitter	LR	78.07	45.86	**69.81**	55.36
	SVM	79.42	**49.12**	67.31	56.79
	PSCN	78.74	47.36	67.29	55.59
	DeepInf-GAT	**80.22**	48.41	69.08	**56.93**
Weibo	LR	77.10	42.34	72.88	53.56
	SVM	77.11	43.27	70.79	53.71
	PSCN	81.31	47.72	71.53	57.24
	DeepInf-GAT	**82.72**	**48.53**	**76.09**	**59.27**

Features

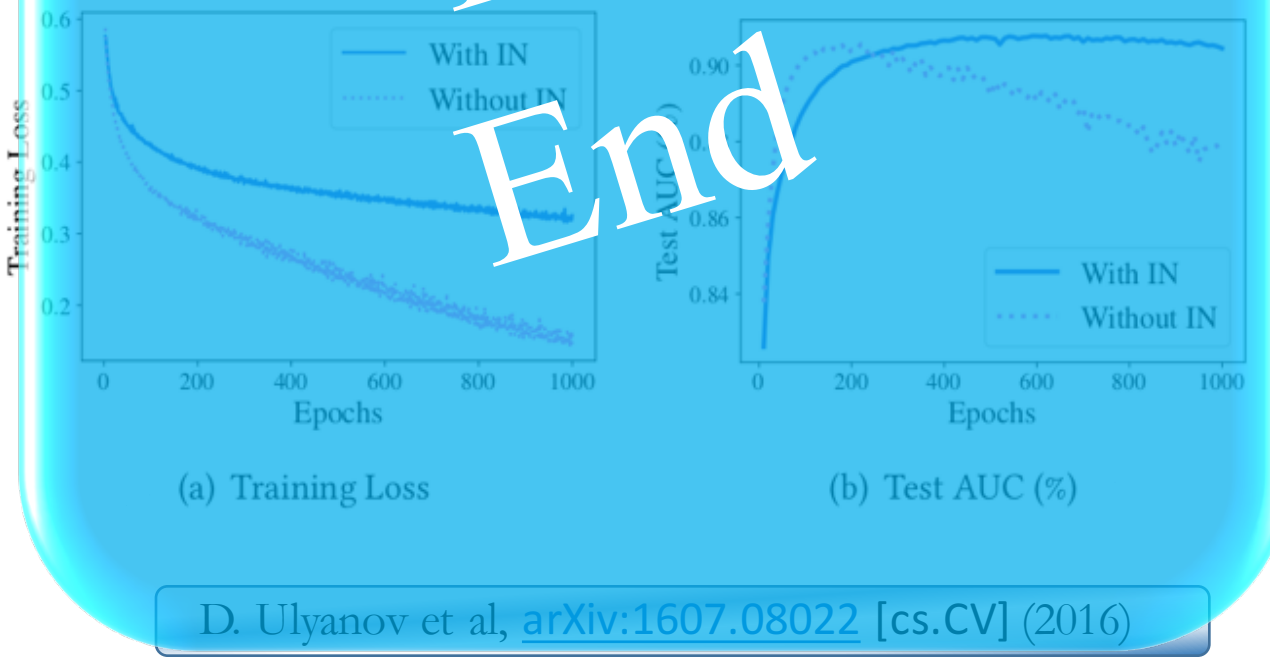
Name	Description
	Coren… [·]
	…], 64-dim).
Ego	Density of … duced by active neighbors [46].
	#Connected components formed by active neighbors [46].

End-to-end = No Features

DeepInf: Social Influence Prediction with GATs

Avoiding Overfitting: Instance Normalization

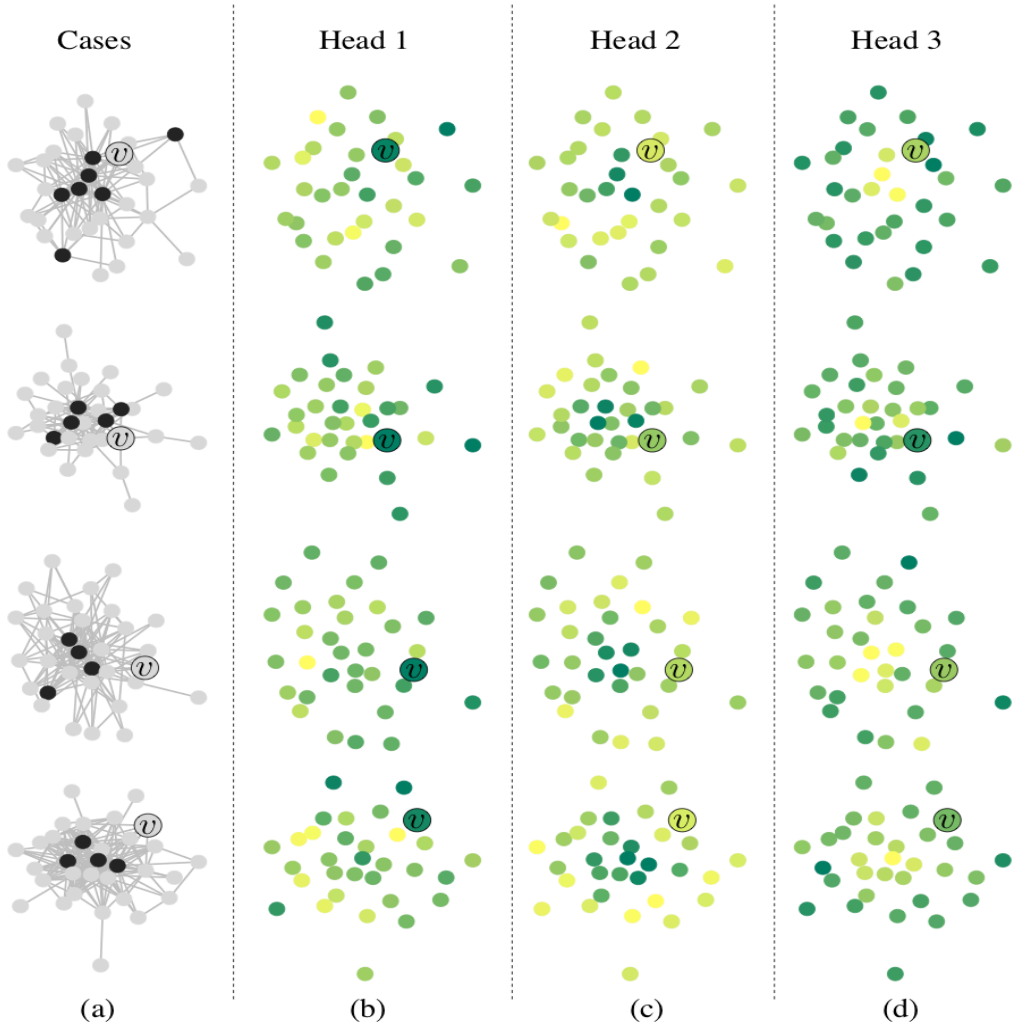
$$y_{ud} = \frac{x_{ud} - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}}$$



The End

D. Ulyanov et al, arXiv:1607.08022 [cs.CV] (2016)

Case Study



Appendix - Model Framework of DeepInf

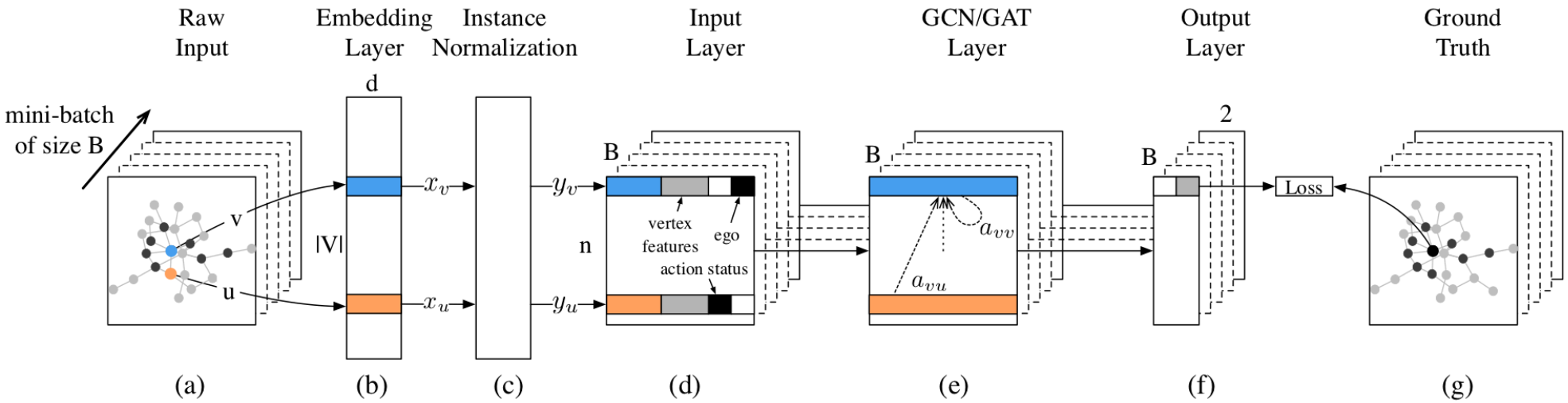


Figure 2: Model Framework of DeepInf. (a) Raw input which consists of a mini-batch of B instances; Each instance is a sub-network comprised of n users who are sampled using random walk with restart as described in Section 3.1. In this example, we keep our eyes on ego user v (marked as blue) and one of her active neighbor u (marked as orange). (b) An embedding layer which maps each user to her D-dimensional representation; (c) An Instance Normalization layer [47]. For each instance, this layer normalizes users' embedding x_u's according to Eq. 3. The output embedding y_u's have zero mean and unit variance within each instance. (d) The formal input layer which concatenates together network embedding, two dummy features (one indicates whether the user is active, the other indicates whether the user is the ego), and other customized vertex features (see Table 2 for example). (e) A GCN or GAT layer. a_{vv} and a_{vu} indicate the attention coefficients along self-loop (v, v) and edge (v, u), respectively; The value of these attention coefficients can be chosen between Eq. 5 and Eq. 7 according to the choice between GCN and GAT. (f) and (g) Compare model output and ground truth, we get the negative log likelihood loss. In this example, ego user v was finally activated (marked as black).

Appendix – Data Preparation

M. De Domenico et al, Scientific Reports 3, 2980 (2013)

Overview of the Dataset

Our dataset consists of messages posted on the Twitter social network, crawled by means of the Application Programming Interface (API) made available by the service itself. We collected tweets sent between 00:00 AM, 1st July 2012 and 11:59 PM, 7th July 2012 containing at least one of the following keywords or hashtags: `lhc`, `cern`, `boson`, `higgs` (see *Methods*). The final amount of tweets we analyzed was 985,590. Hence, we built the corresponding social network of the authors of the tweets: the resulting graph is composed of 456,631 nodes and 14,855,875 directed edges. Nodes correspond to the authors of the tweets and edges represent the followee/follower relationships between them. We discarded 70,838 users from the original dataset containing 527,469 users because of the non accessibility of the list of their followees and followers due to privacy settings. Twitter users can specify their location by filling the *Location* field of their profile, on optional basis and at different levels of granularity (e.g., United States, New York, Chelsea, etc.). We use this information, when available, to assign a geographic position to each tweet: the resulting number of geo-located tweets is 632,027 (see *Methods*).

J. Zhang et al, TKDD Vol 9,3 (2015)

A few issues : Imbalanced Data

As observed by Zhang et al. [54], structural features become significantly correlated with social influence locality when the ego user has a relatively large number of active neighbors. However, the

Therefore, when we train our model on such imbalanced datasets, the model will be dominated by observations with few active neighbors. To deal with the imbalance issue and show the superiority of our model in capturing local structural information, we filter out observations with few active neighbors.

Imbalanced Labels

The second problem comes from label imbalance. For example, in the Weibo dataset, the ratio between negative instances and positive instances is about 300:1. To address this issue, we sample a more balanced dataset with the ratio between negative and positive to be 3:1.