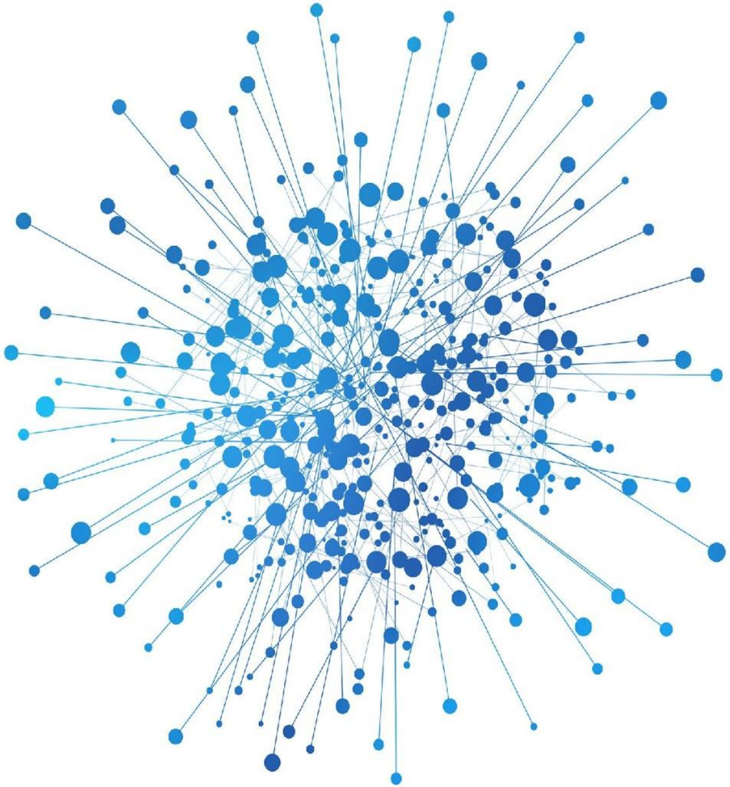
Predictive and Generative Deep Learning for Graphs

Amir Saffari

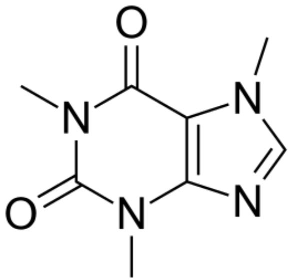
@amirsaffari - amir.saffariazar@benevolent.ai

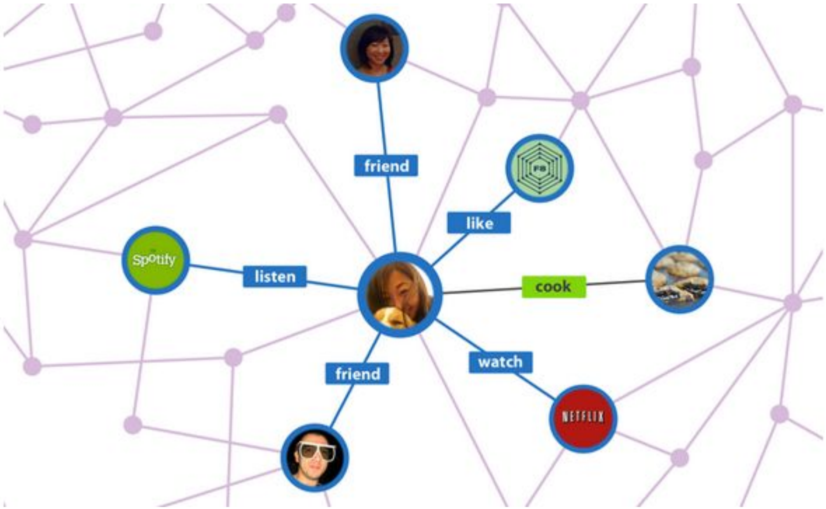
BenevolentAI

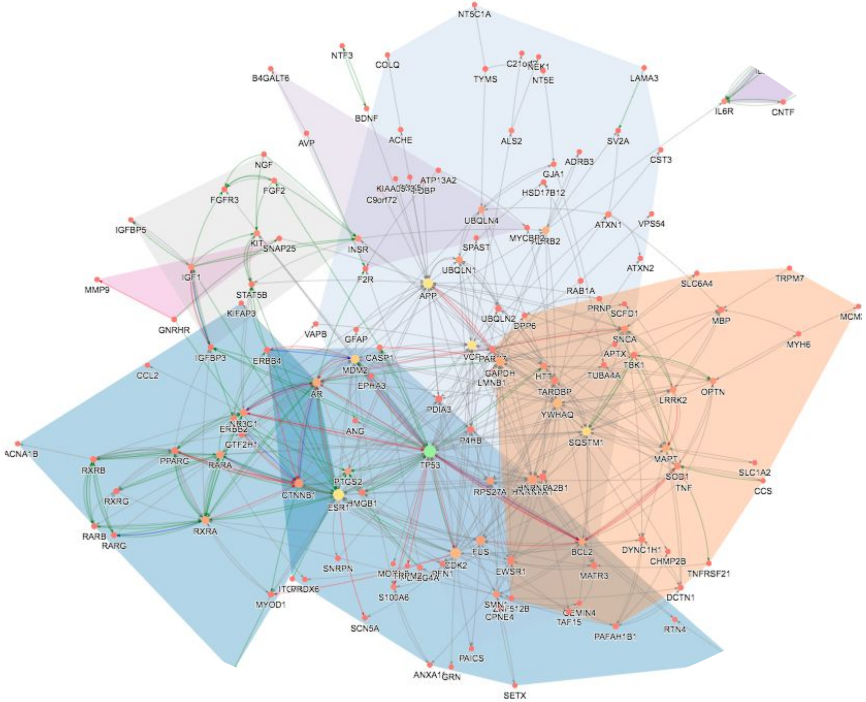


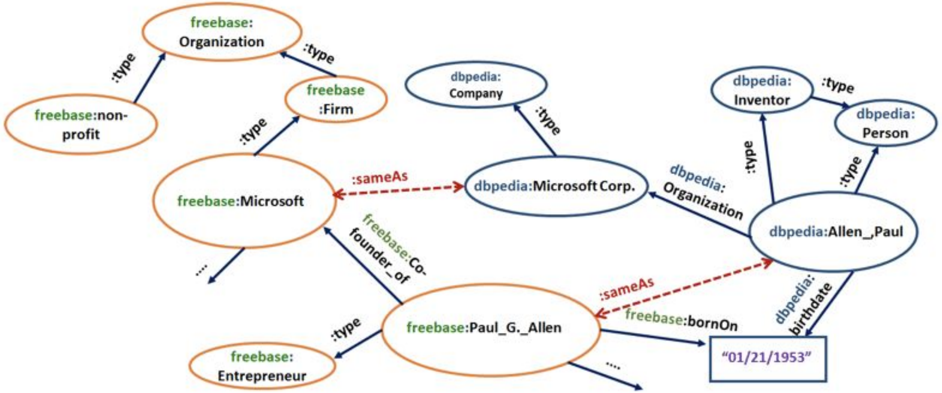
Introduction

Graphs as Primary Data Structures



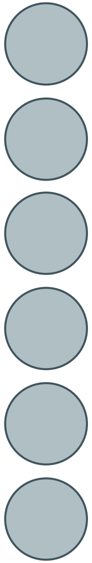




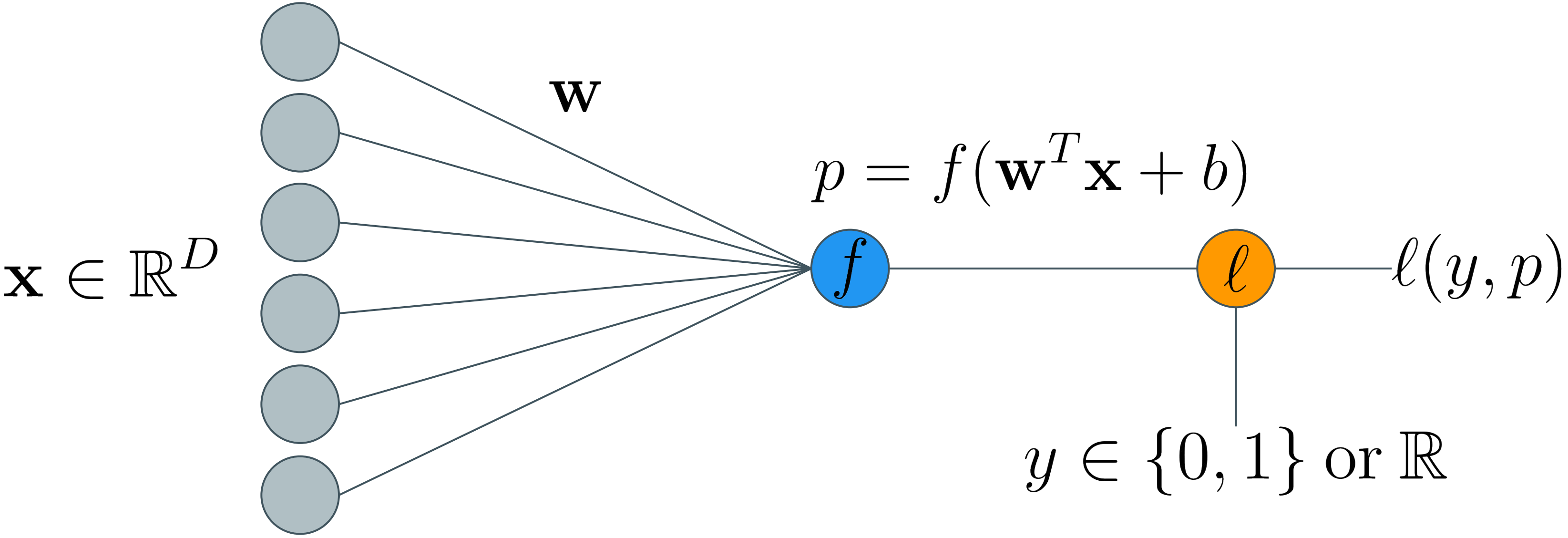


Vector Data

$$\mathbf{x} \in \mathbb{R}^D$$

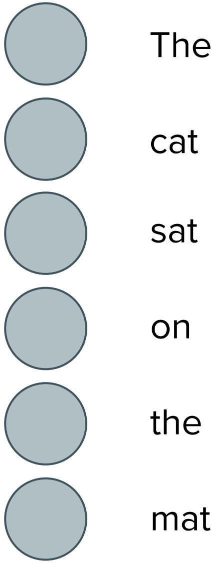


Dense Models



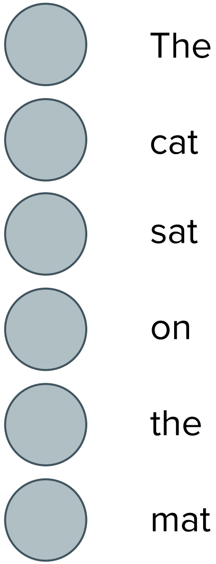
Categorical Data

$$\mathbf{x} \in \mathcal{V}^T$$



Categorical Data: Bag of Words

$$\mathbf{x} \in \mathcal{V}^T$$



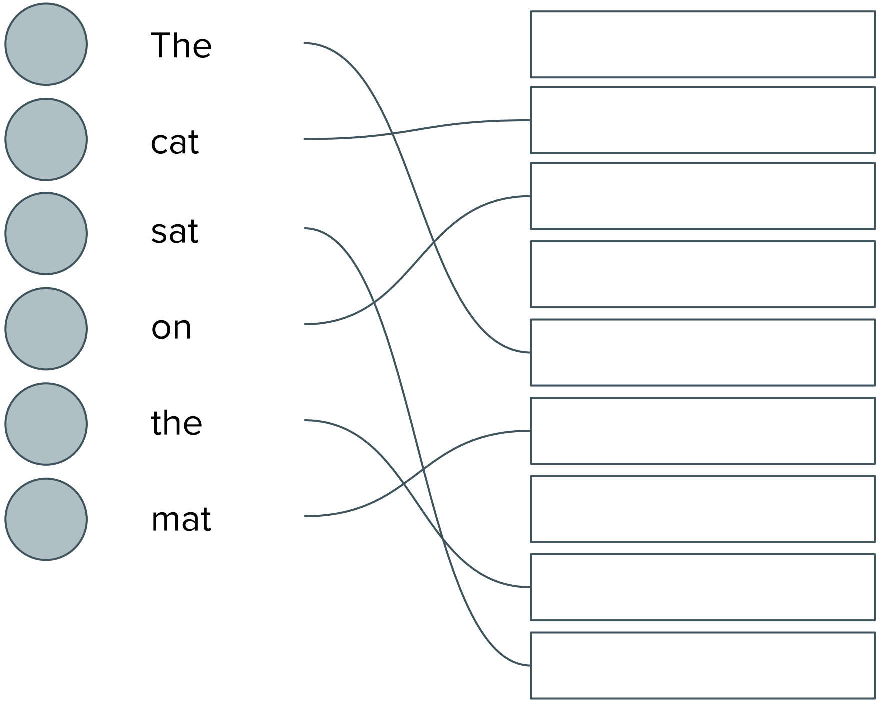
$$\mathrm{BOW}(\mathrm{The}) = [0, \cdots, 0, 1, 0, 0]$$

$$\mathrm{BOW}(\mathrm{cat}) = [0, 1, \cdots, 0, 0]$$

Embedding Models

$$E \in \mathbb{R}^{|\mathcal{V}| \times D}$$

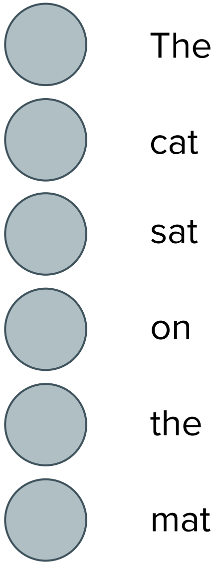
$$\mathbf{x} \in \mathcal{V}^T$$

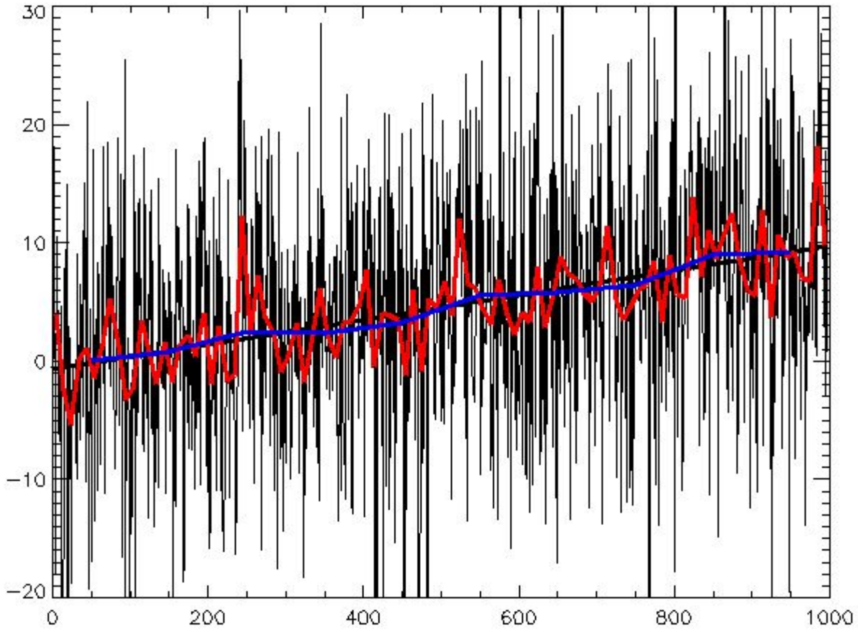


$$E(\mathbf{x}) \in \mathbb{R}^{T \times D}$$

Sequential Data: 1D

$\mathbf{x} \in \mathcal{V}^T$



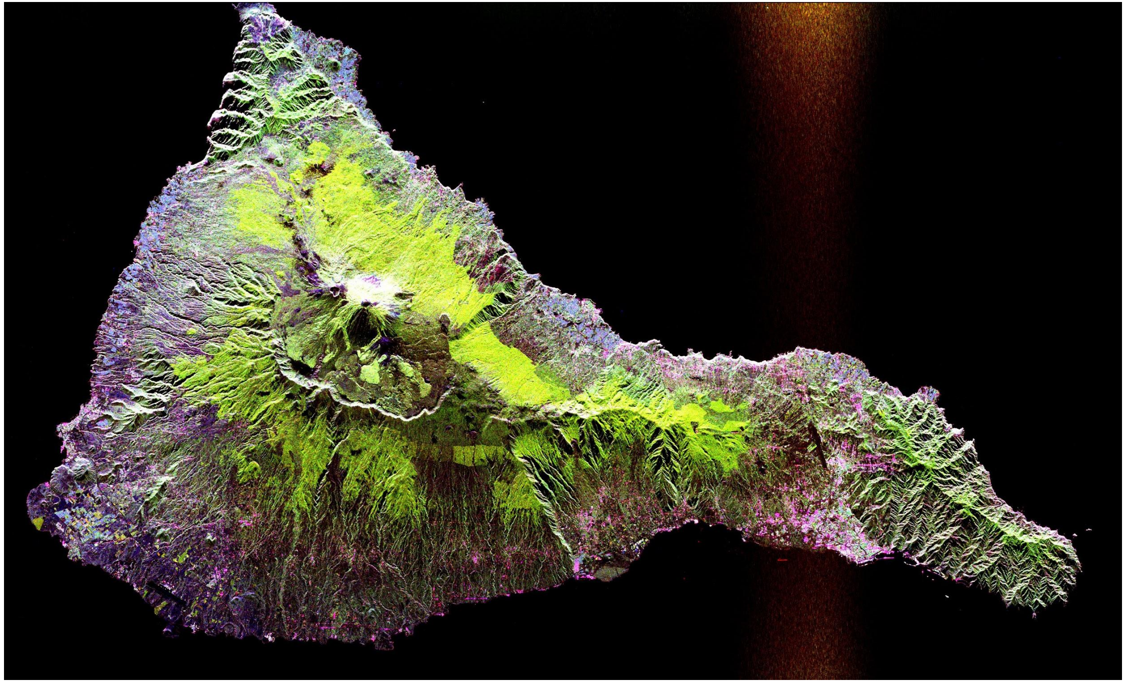


$\mathbf{x} \in \mathbb{R}^T$

https://en.wikipedia.org/wiki/Time_series

Sequential Data: 2D

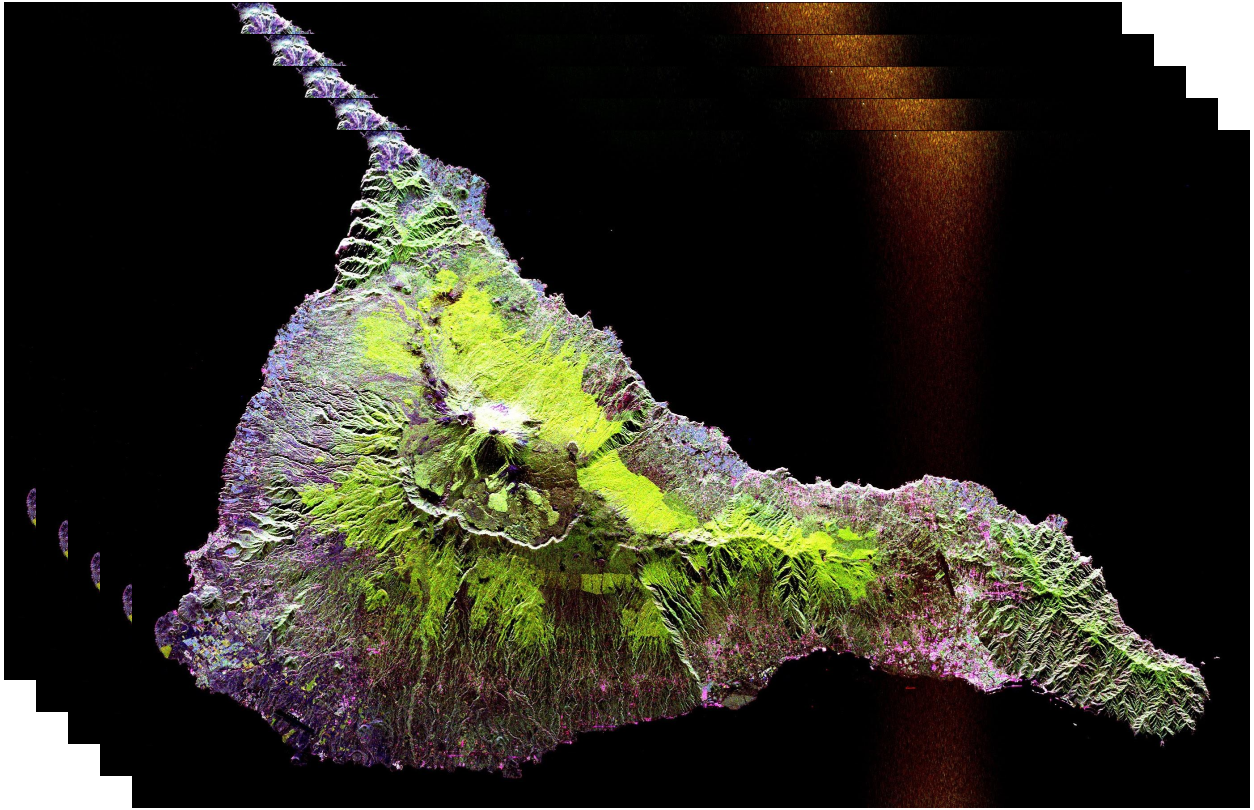
$$\mathbf{x} \in \mathbb{R}^{W \times H}$$



https://en.wikipedia.org/wiki/Image

Sequential Data: 3D

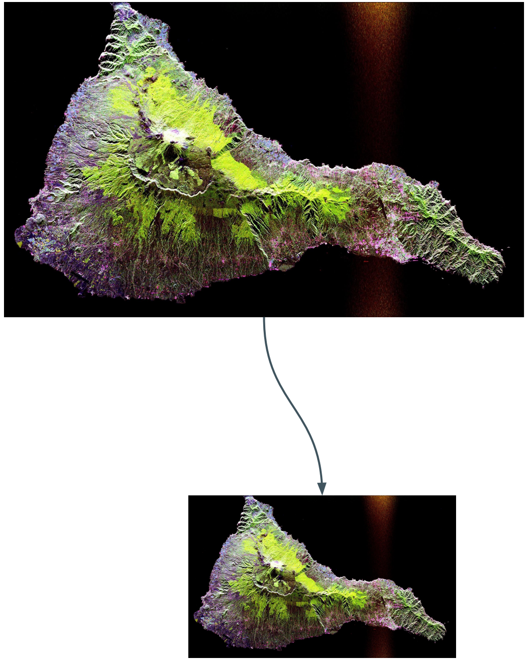
$$\mathbf{x} \in \mathbb{R}^{W \times H \times T}$$



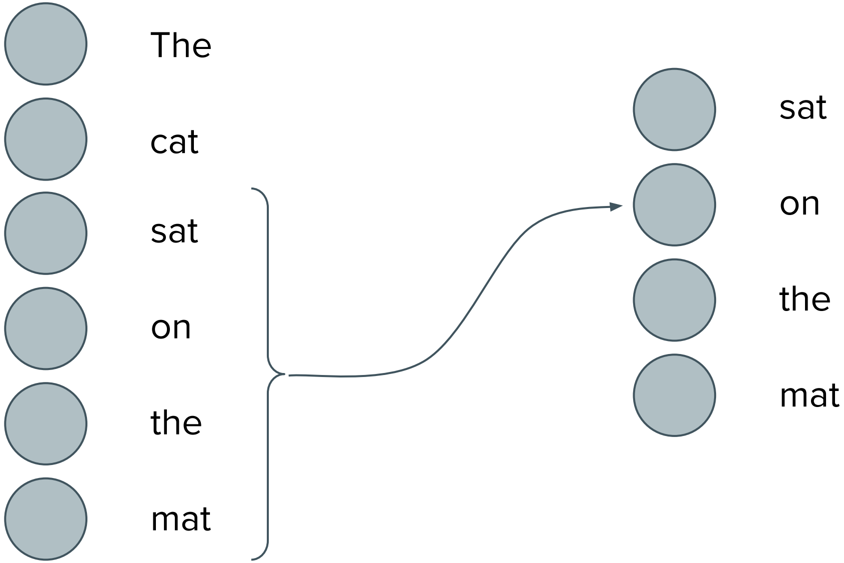
https://en.wikipedia.org/wiki/Image

How to Deal with Variable Length Sequences?

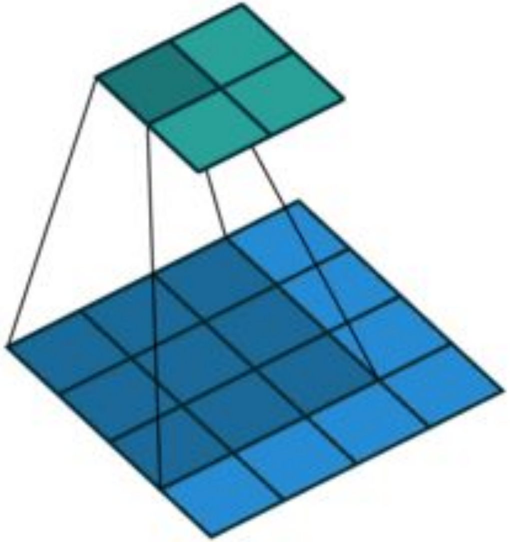
Resize to standard size

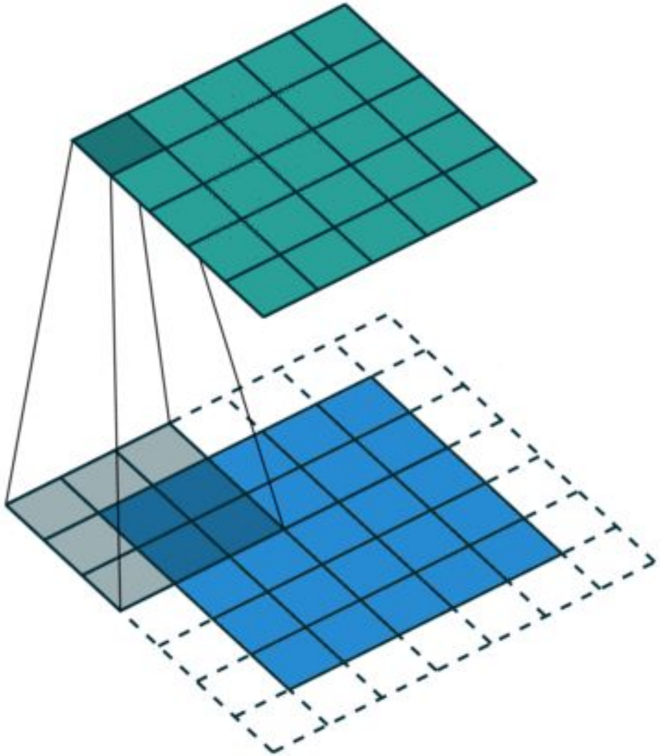


Fix context size



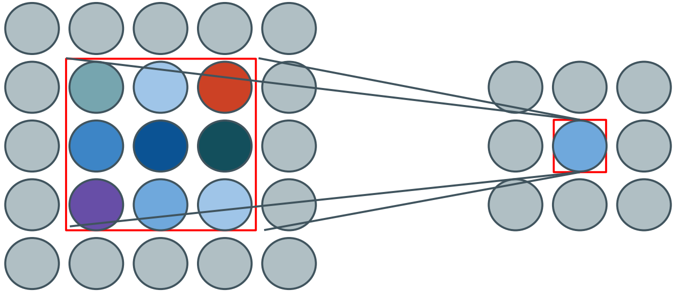
Convolutional Models



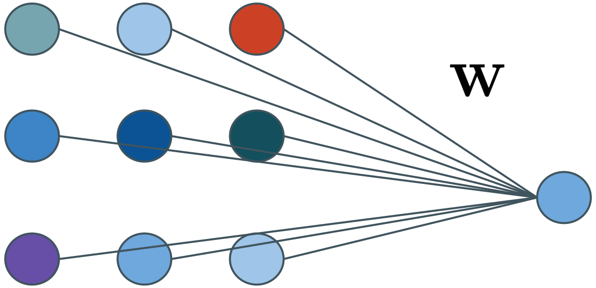


https://github.com/vdumoulin/conv_arithmetic

Convolutional Models

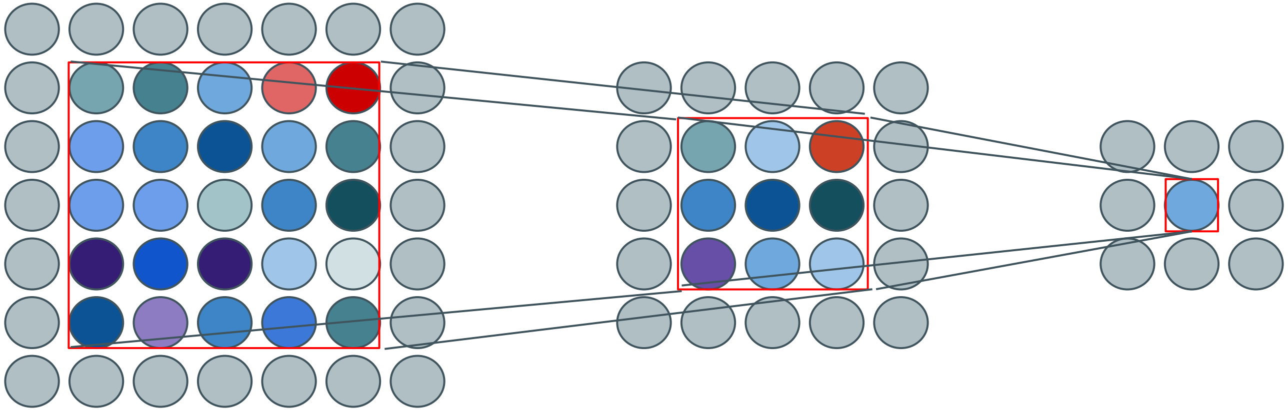


$$\mathbf{x} \in \mathbb{R}^{W \times H}$$



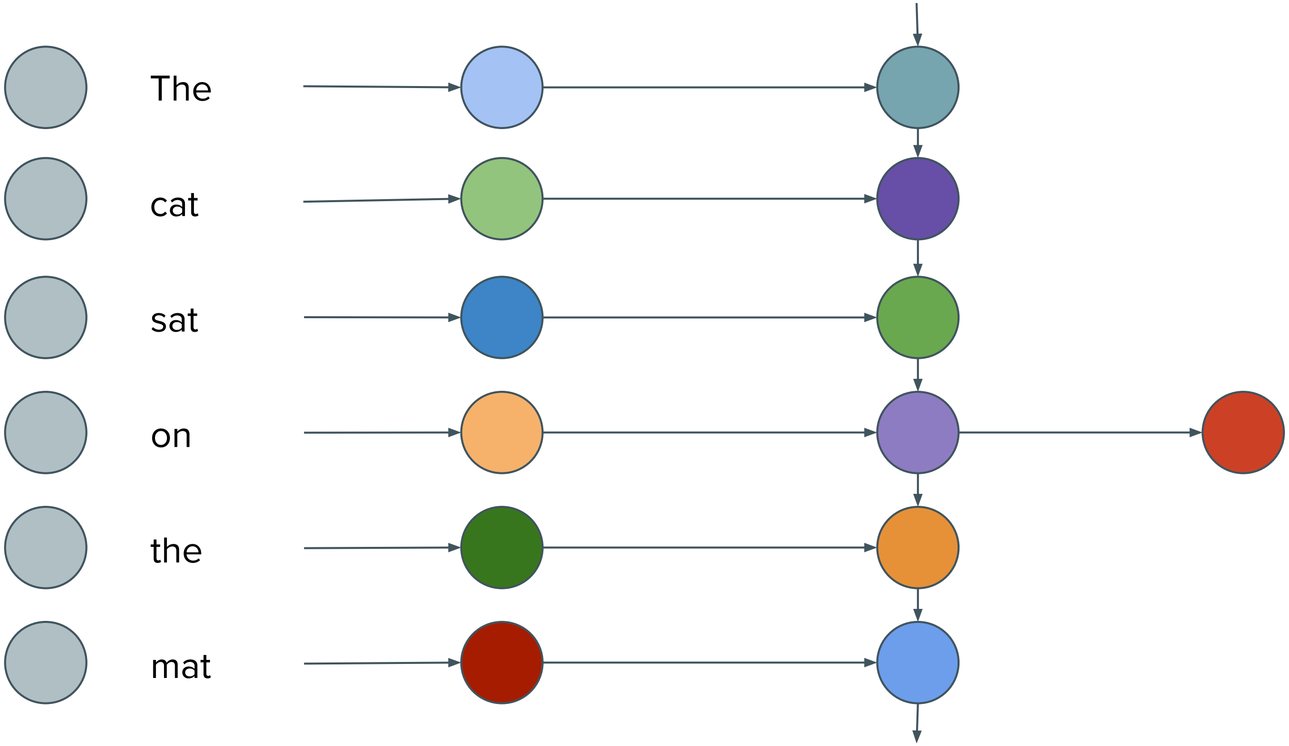
$$p = f(\mathbf{w}^T \mathbf{x} + b)$$

Convolutional Models: Multiple Layers and Receptive Field



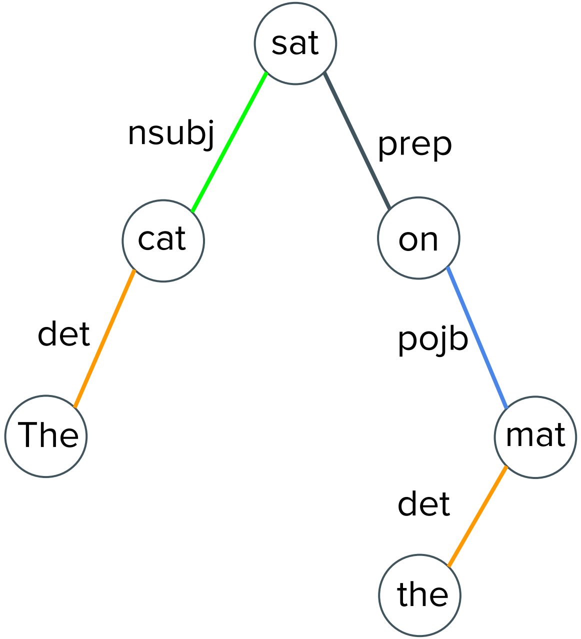
Recurrent Models

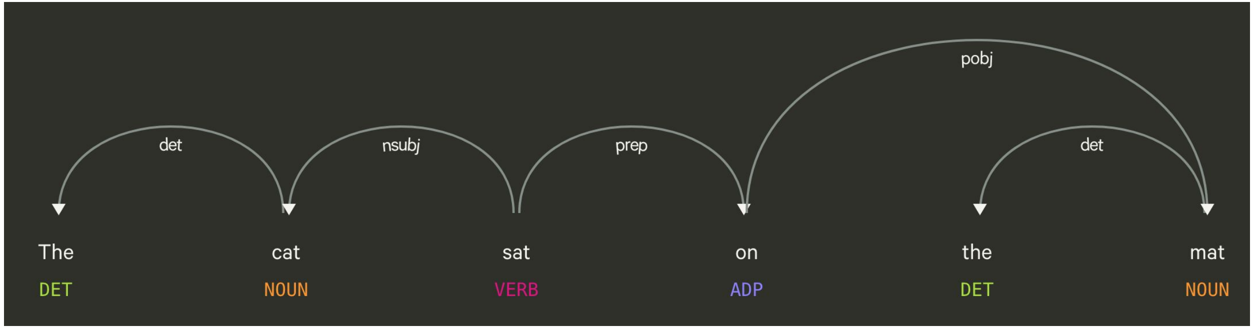
$$\mathbf{e}_t = e(x_t; \mathbf{w}_e) \qquad \mathbf{h_t} = g(\mathbf{e}_t, \mathbf{h}_{t-1}; \mathbf{w}_h)$$



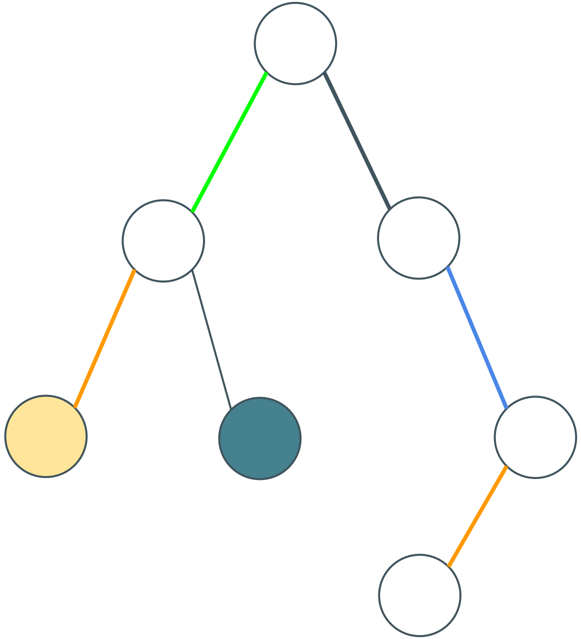
$$p_t = f(\mathbf{h_t}; \mathbf{w}_o)$$

Complex Structures: Trees



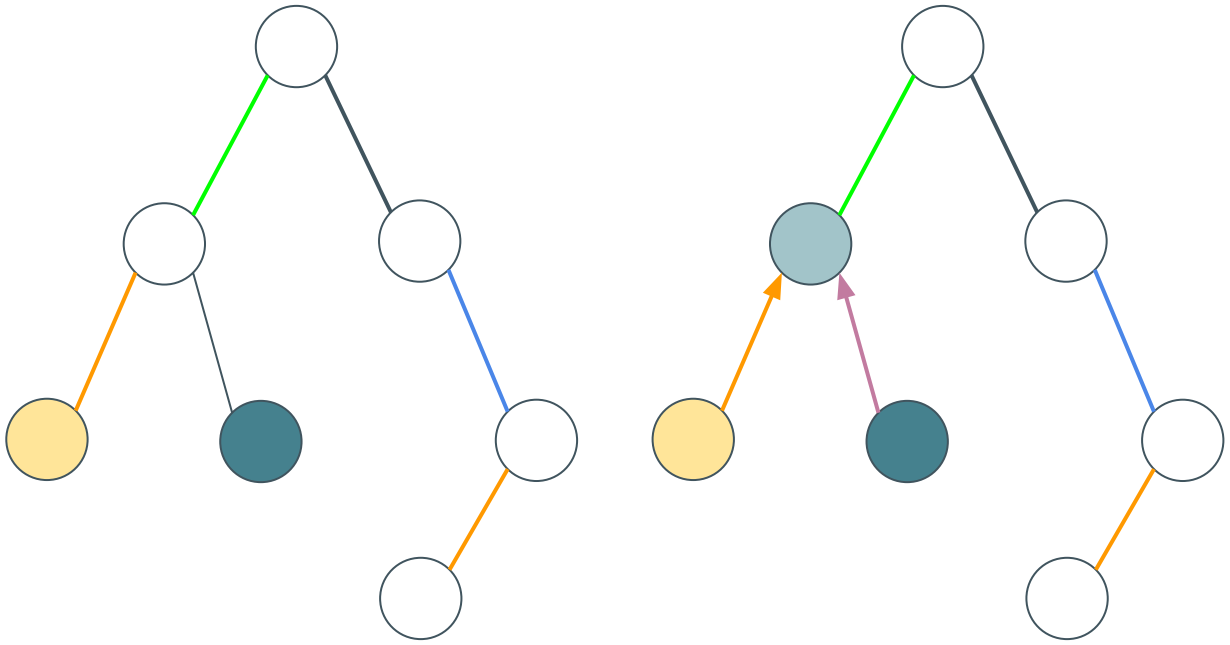


Recursive Models



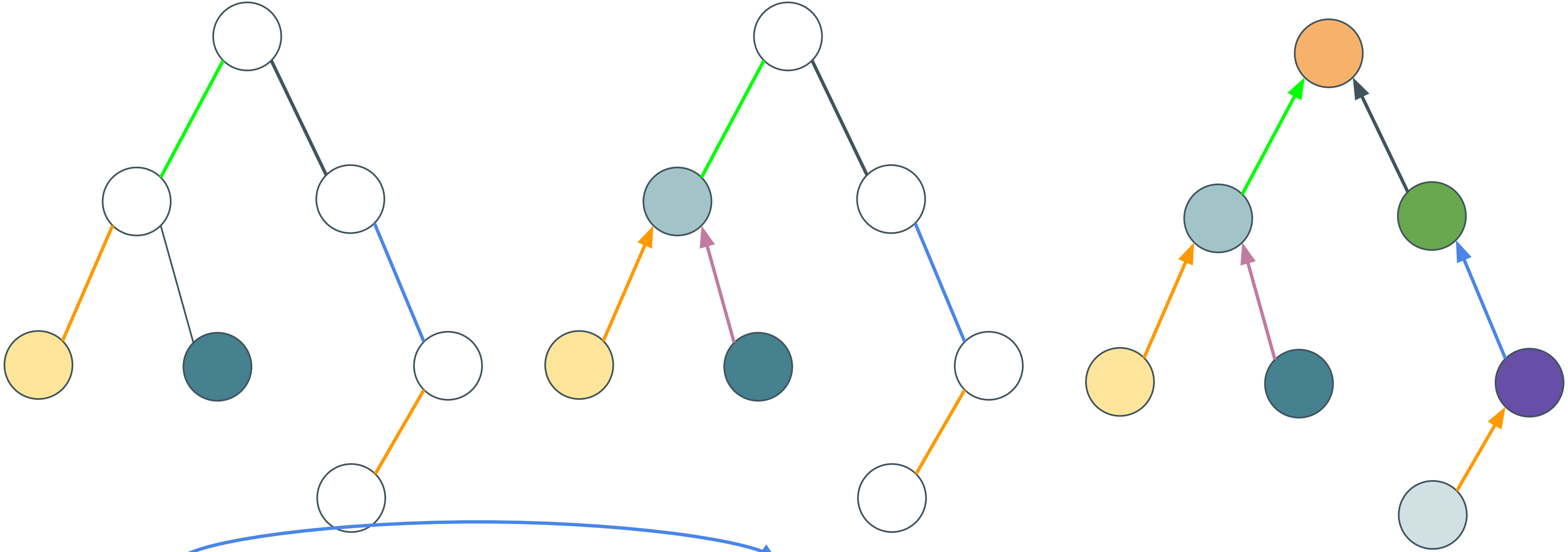
$$\mathbf{e}_t = e(x_t; \mathbf{w}_e)$$

Recursive Models



$$\mathbf{e}_t = e(x_t; \mathbf{w}_e) \qquad \mathbf{h_t} = g(\mathbf{e}_t, \mathbf{h}_{t-1}; \mathbf{w}_h)$$

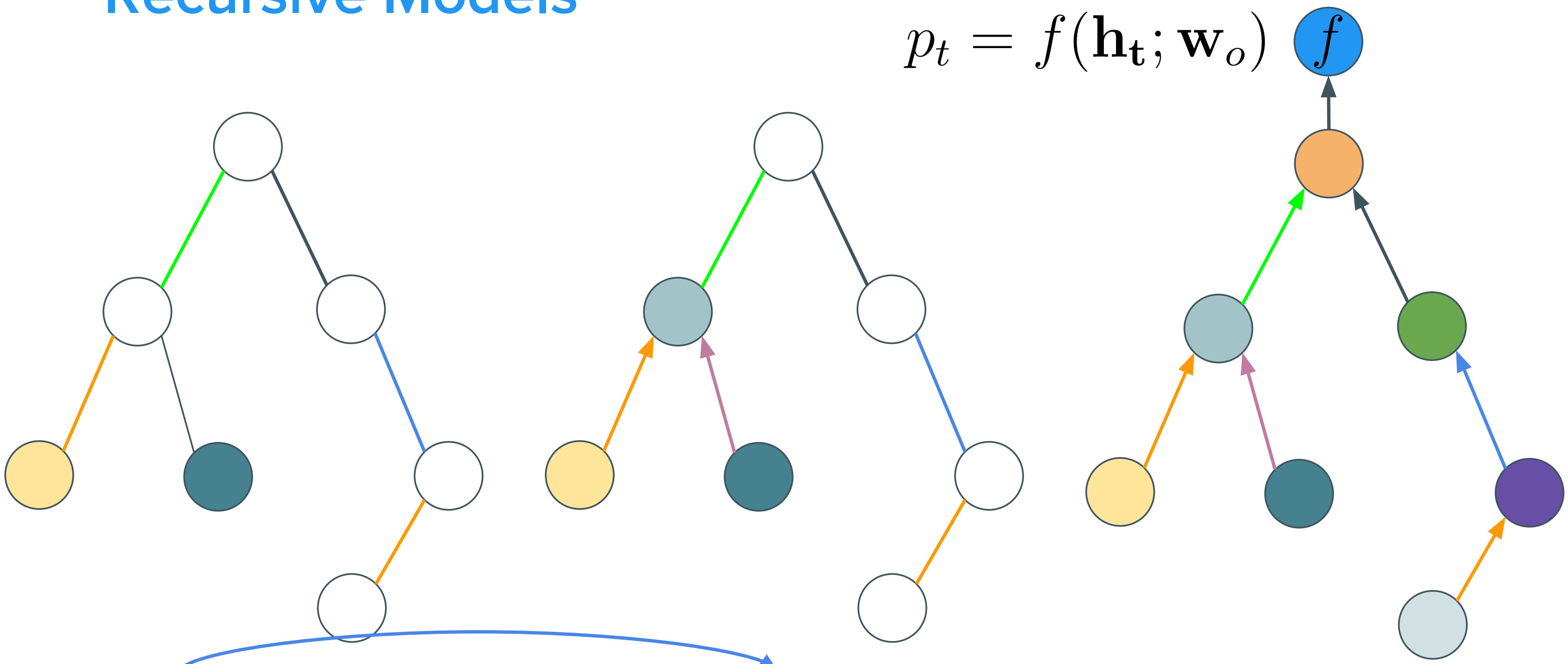
Recursive Models



$$\mathbf{e}_t = e(x_t; \mathbf{w}_e) \qquad \mathbf{h_t} = g(\mathbf{e}_t, \mathbf{h}_{t-1}; \mathbf{w}_h)$$

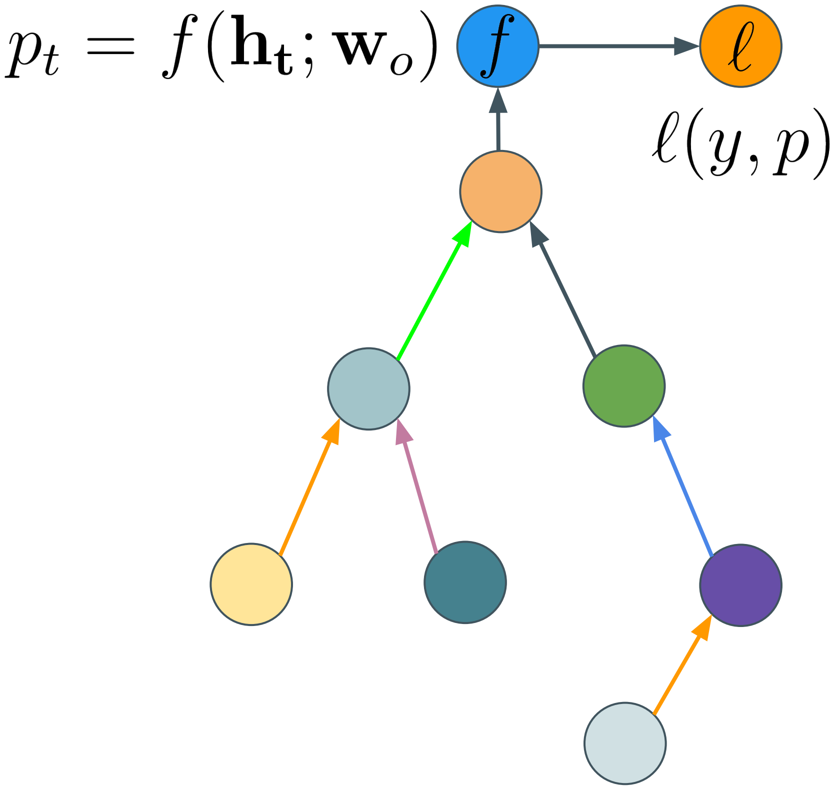
Recursive Models

$$p_t = f(\mathbf{h_t}; \mathbf{w}_o)$$

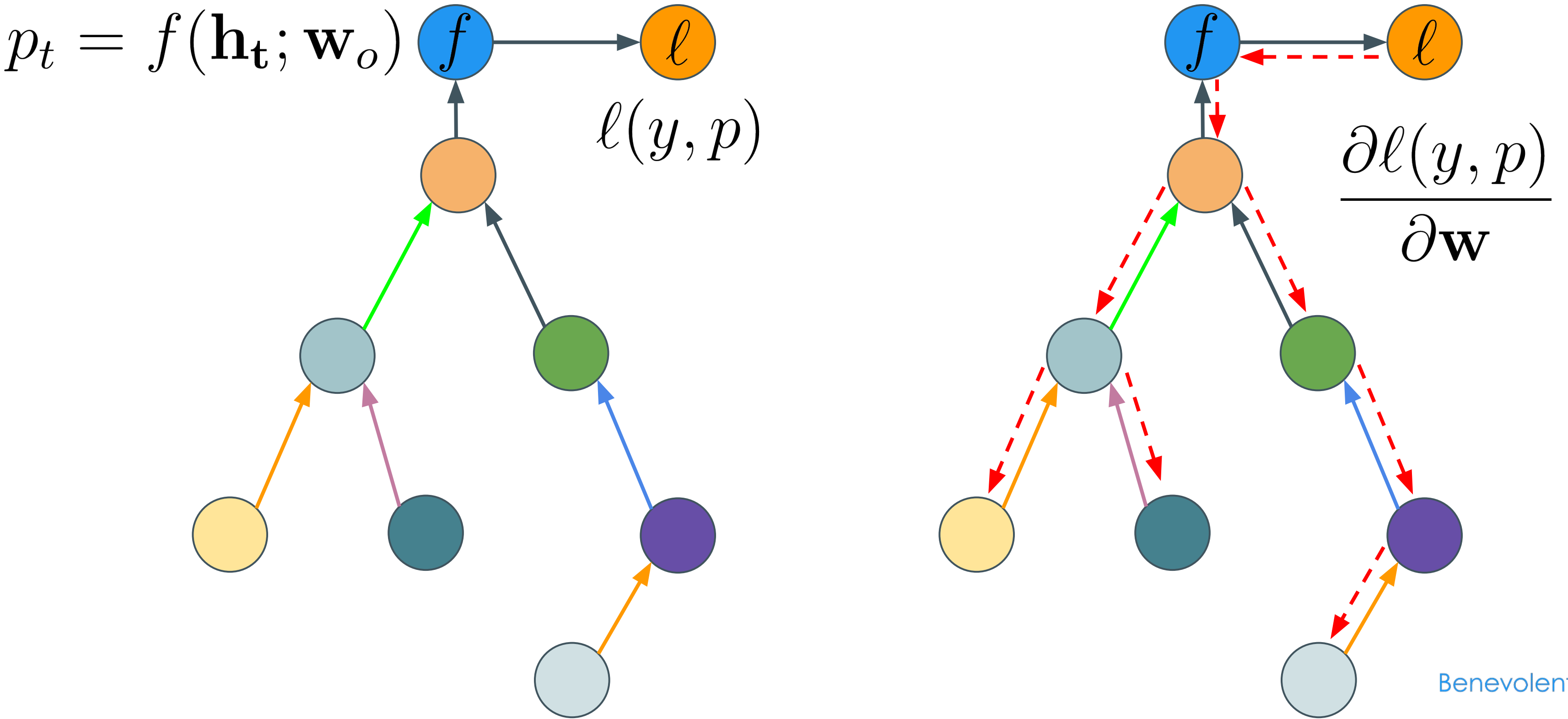


$$\mathbf{e}_t = e(x_t; \mathbf{w}_e) \qquad \mathbf{h_t} = g(\mathbf{e}_t, \mathbf{h}_{t-1}; \mathbf{w}_h)$$

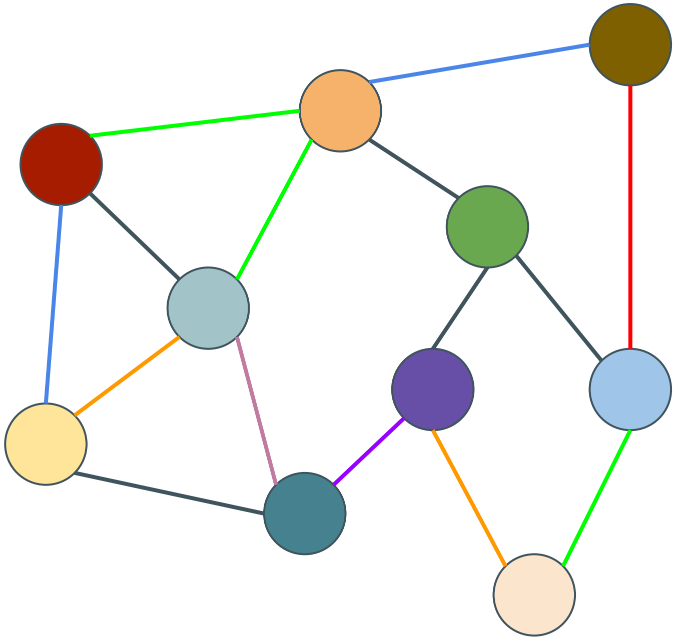
Complex Structures: Structure as Computational Graph



Complex Structures: Backpropagation through Structure



Complex Structures: Graph

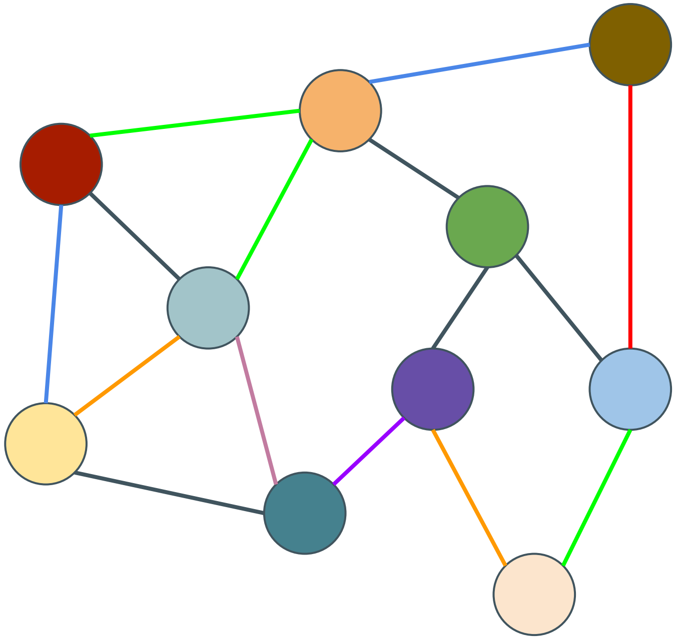


Graph $G = (\mathcal{N}, \mathcal{R})$

Nodes $\mathcal{N} = \{n_1, \cdots, n_{|\mathcal{N}|}\}$

Relations $\mathcal{R} = \{r_1, \cdots, r_{|\mathcal{R}|}\}$

Complex Structures: Graph



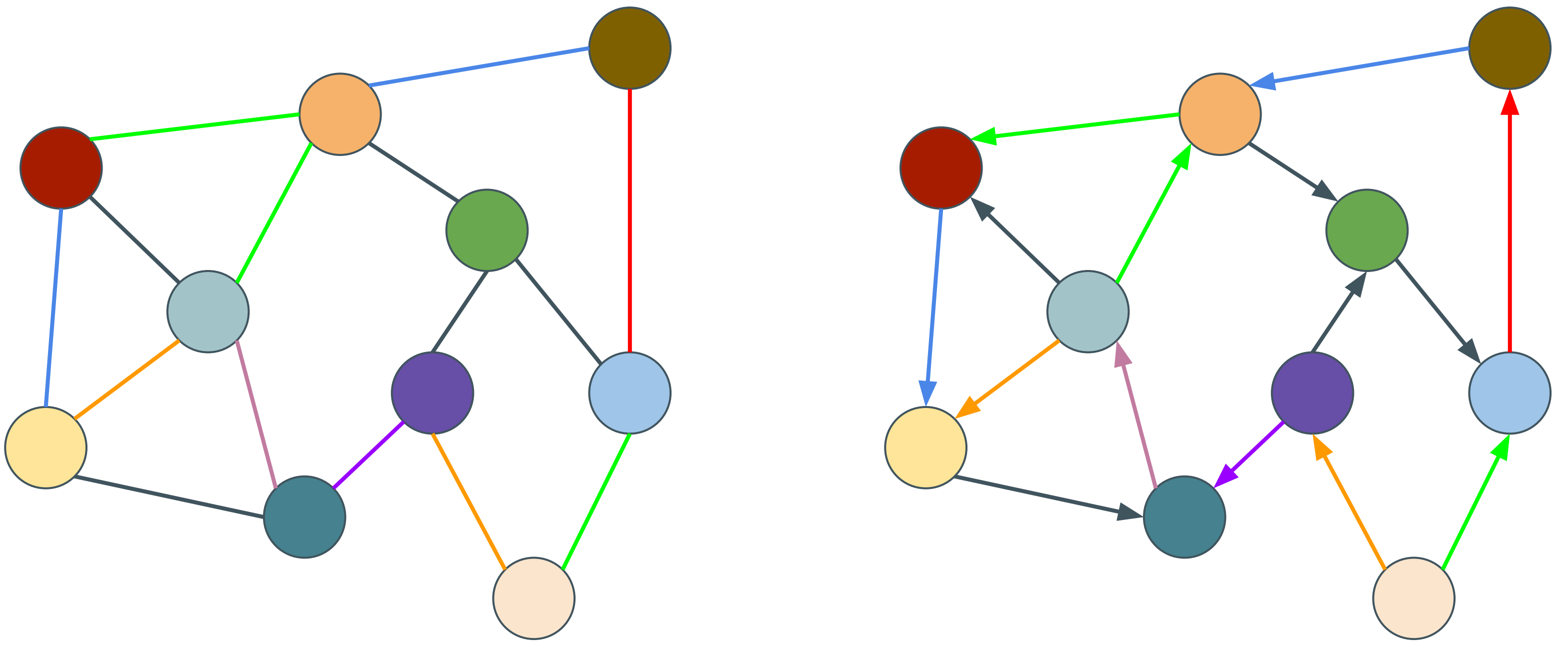
Graph $G = (\mathcal{N}, \mathcal{R})$

Nodes $\mathcal{N} = \{n_1, \cdots, n_{|\mathcal{N}|}\}$

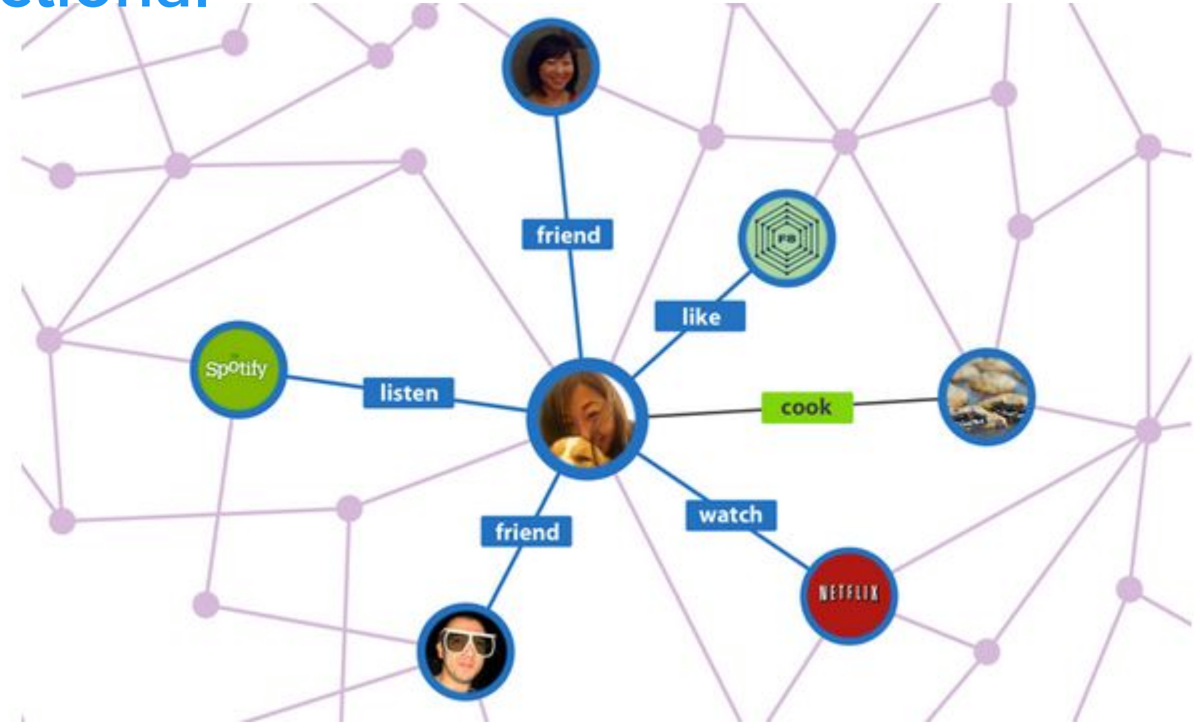
Relations $\mathcal{R} = \{r_1, \cdots, r_{|\mathcal{R}|}\}$

Features $\mathbf{x}_n$ $\mathbf{x}_r$

Complex Structures: Directed Graph

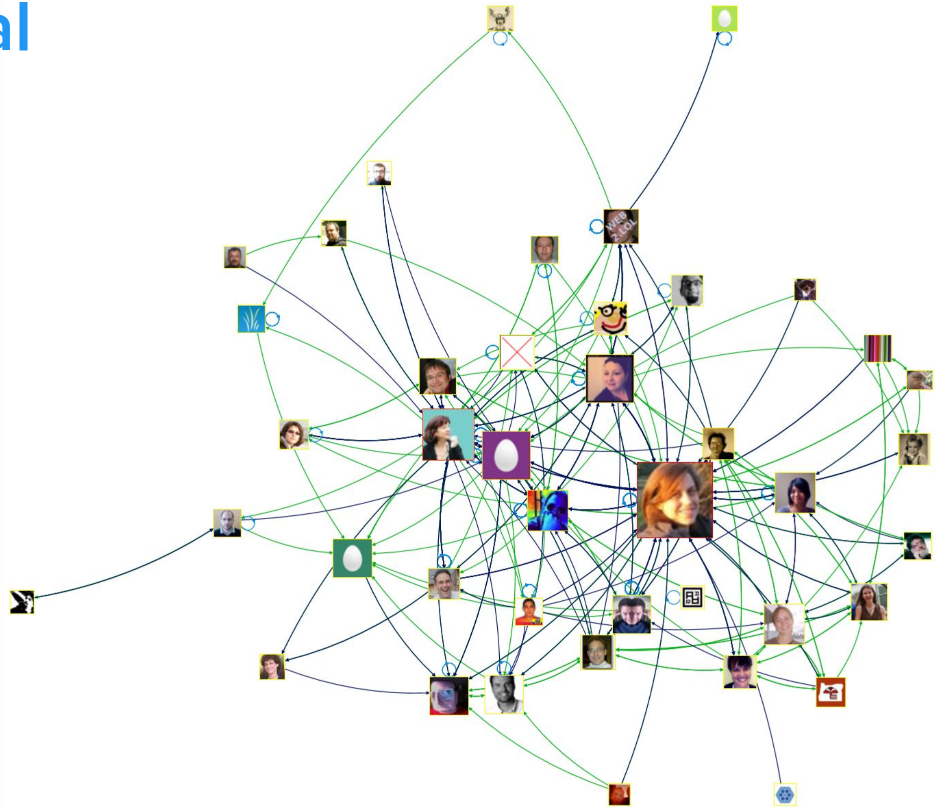


Examples of Graphs: Facebook Friends - Undirectional



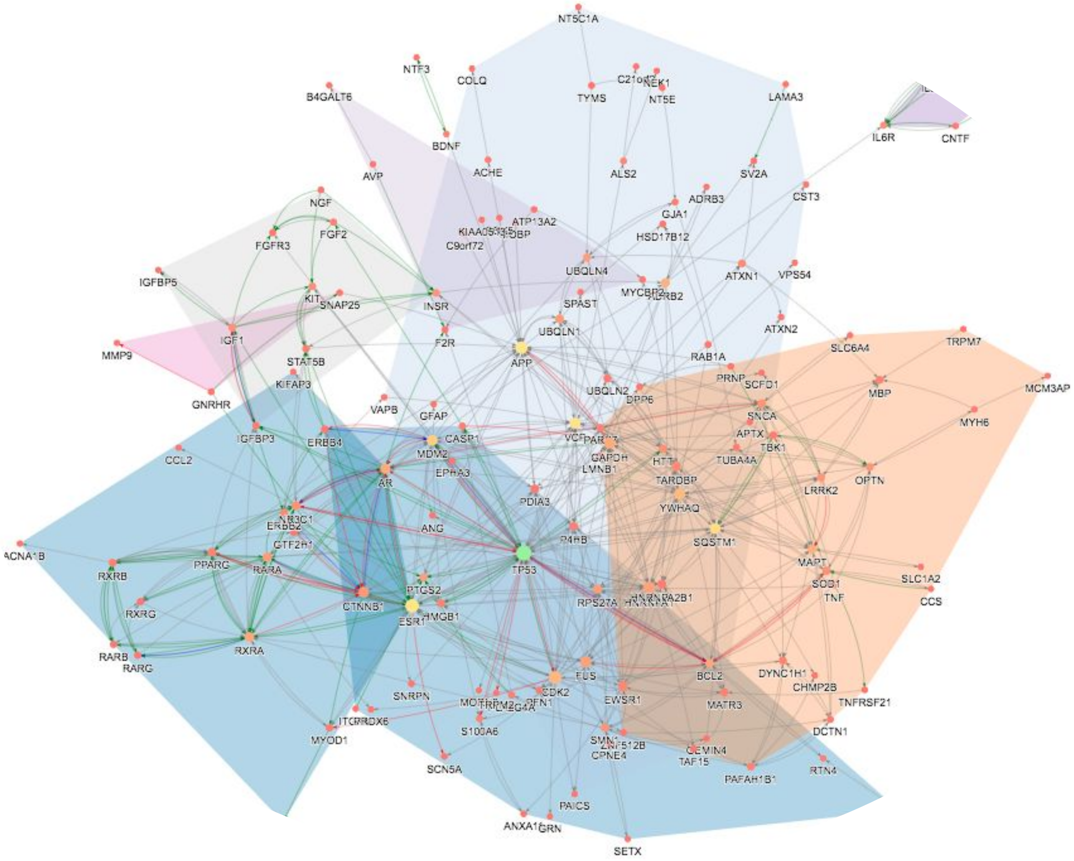
https://techcrunch.com/2015/04/28/facebook-api-shut-down/

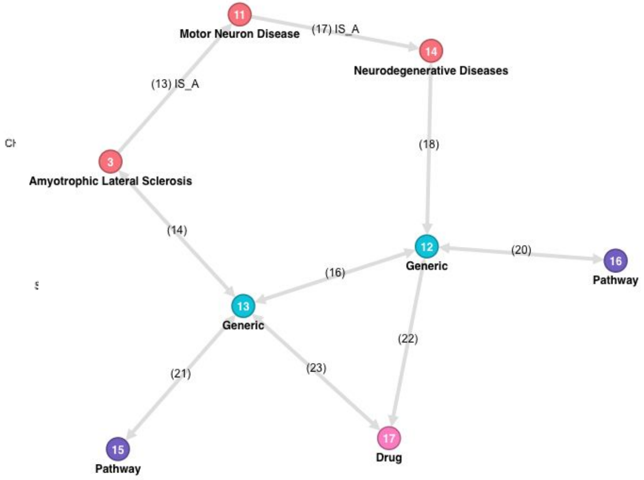
Examples of Graphs: Twitter Followers - Directional



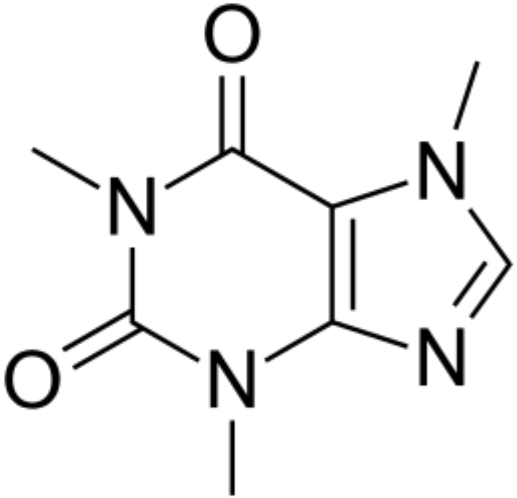
http://tasos-spiliotopoulos.com/ecscw11.html

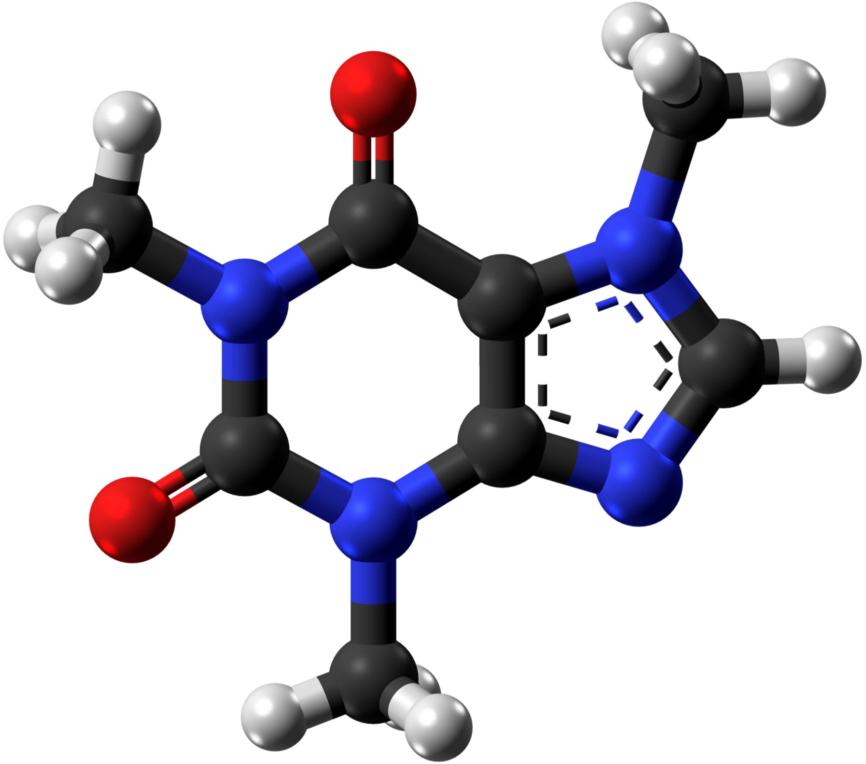
Examples of Graphs: Biological Knowledge Graph





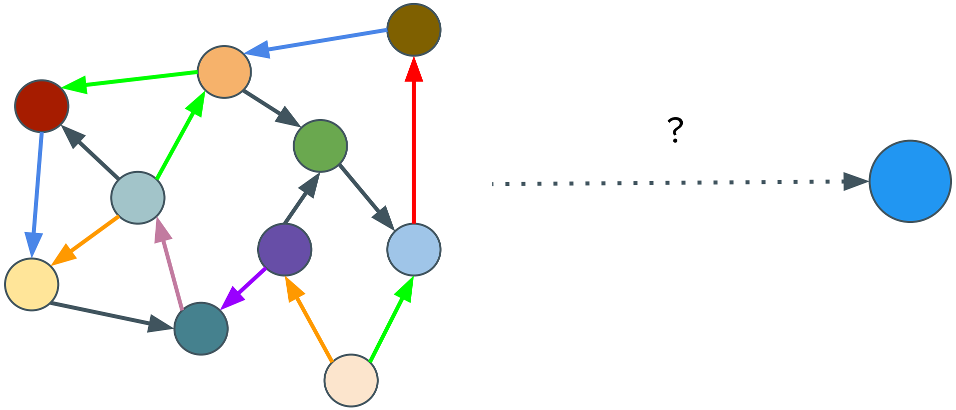
Examples of Graphs: Molecular Graph





https://en.wikipedia.org/wiki/Caffeine

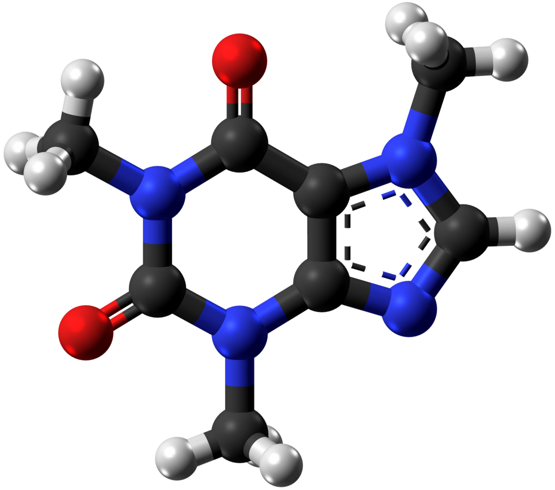
ML Tasks for Graphs: Graph Classification or Regression

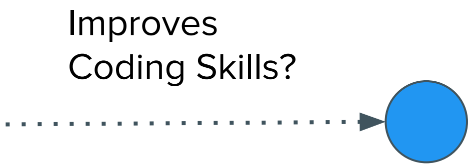


$$y \in \{0, 1\} \text{ or } \mathbb{R}$$

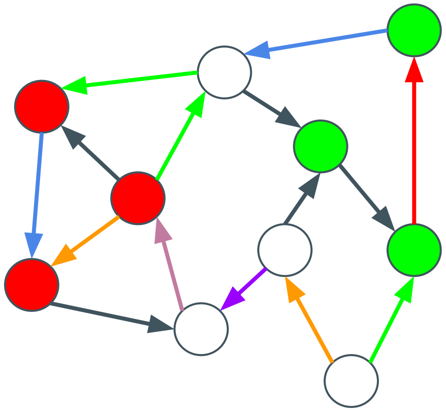
ML Tasks for Graphs: Graph Classification or Regression

Caffeine

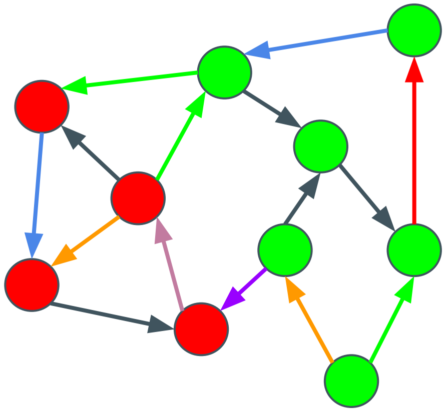




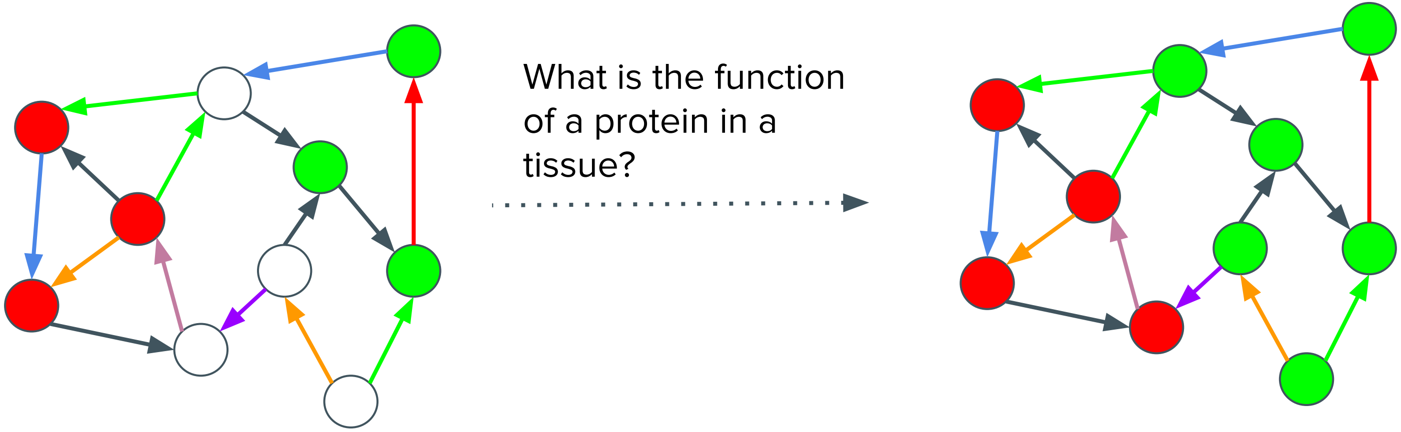
ML Tasks for Graphs: Node Classification



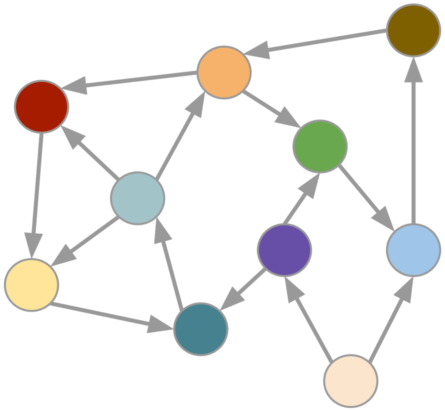
Who should I buy coffee next?



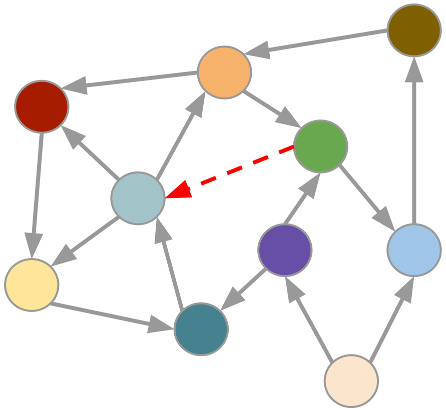
ML Tasks for Graphs: Node Classification



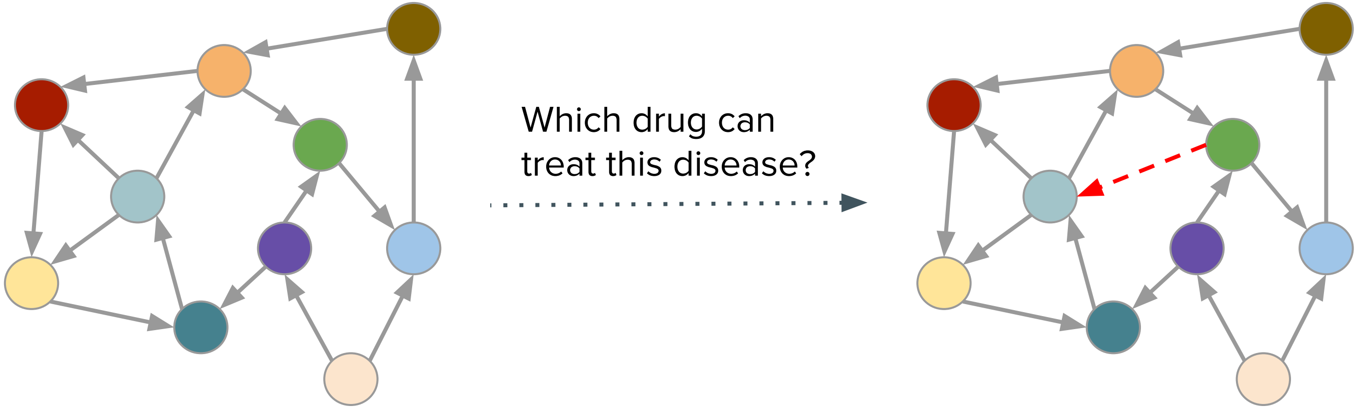
ML Tasks for Graphs: Relationship Inference

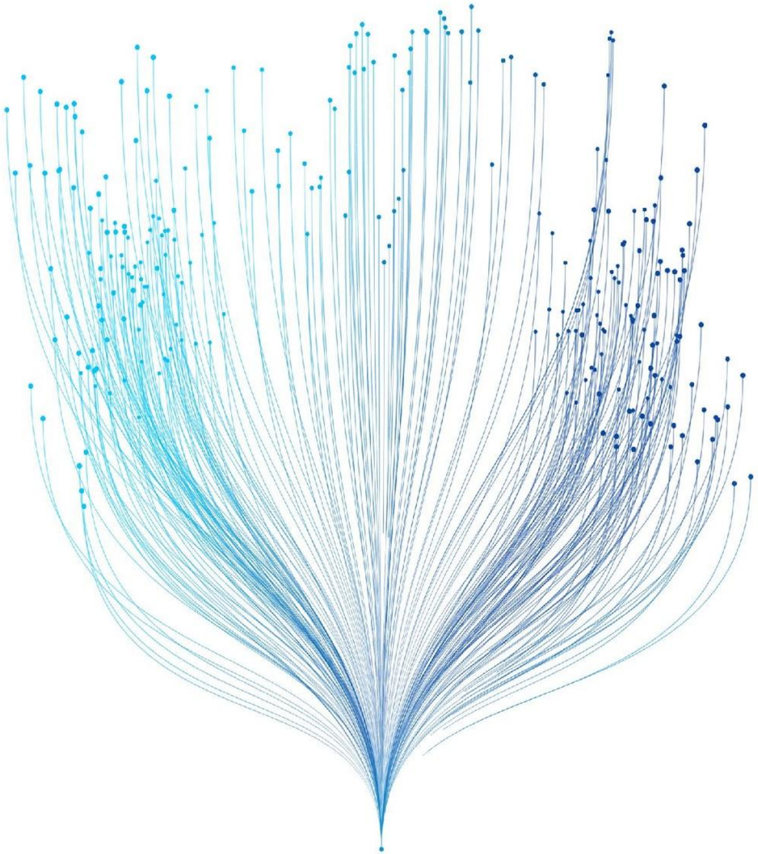


Which coffee shop should I try next?



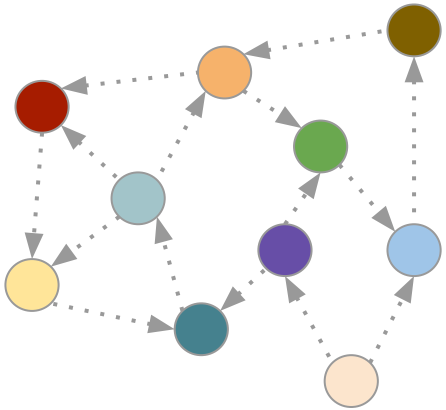
ML Tasks for Graphs: Relationship Inference





Graph Convolutional Neural Networks

Node Embedding

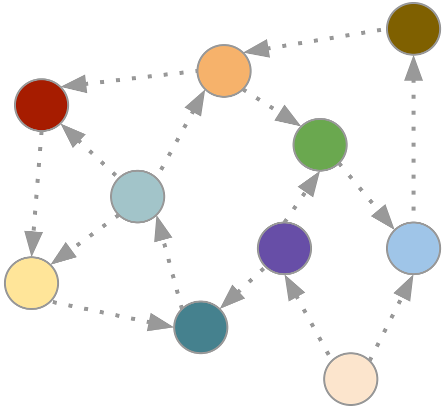


$$\mathbf{e}_n = e(n; \mathbf{w}_e^n)$$

$$\mathbf{e}_n = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

$$\mathbf{e}_n = e(\mathbf{x}_n; \mathbf{w}_e^n)$$

Node Embedding

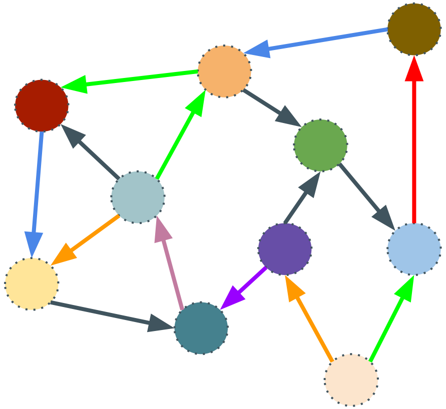


Transductive $$\mathbf{e}_n = e(n; \mathbf{w}_e^n)$$

Transductive $$\mathbf{e}_n = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

Inductive $$\mathbf{e}_n = e(\mathbf{x}_n; \mathbf{w}_e^n)$$

Relation Embedding

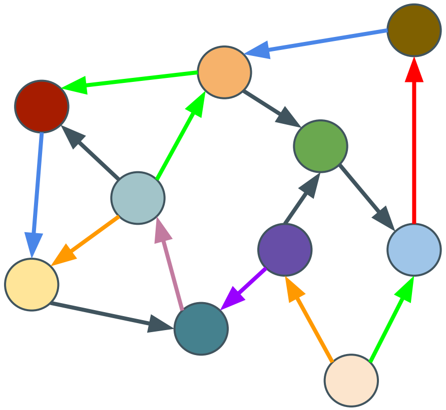


Transductive $\mathbf{e}_r = e(r; \mathbf{w}_e^r)$

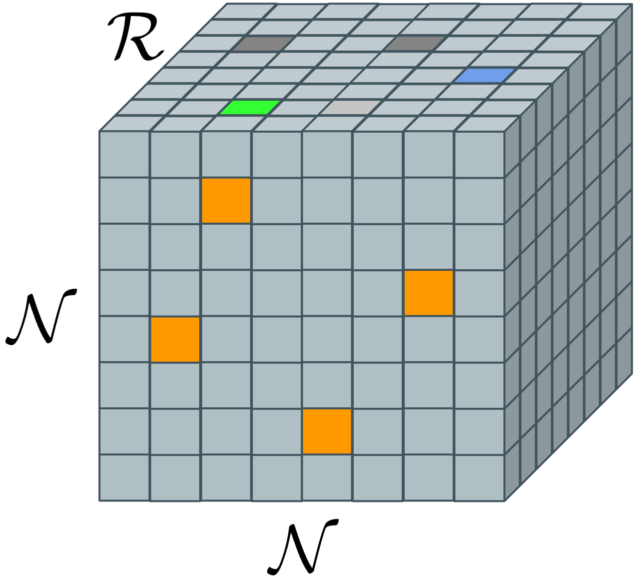
Transductive $\mathbf{e}_r = e(r, \mathbf{x}_r; \mathbf{w}_e^r)$

Inductive $\mathbf{e}_r = e(\mathbf{x}_r; \mathbf{w}_e^r)$

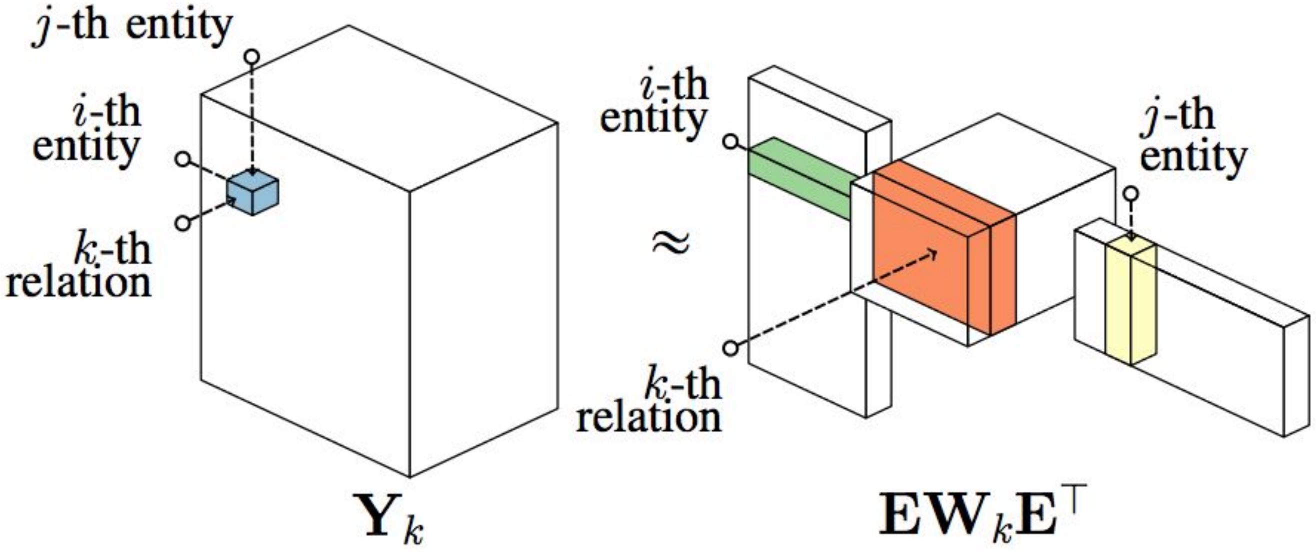
Graphs as Tensors



$$T(G) \in \{0,1\}^{|\mathcal{N}| \times |\mathcal{N}| \times |\mathcal{R}|}$$



Tensor Factorisation Models for Relationship Inference: RESCAL



https://arxiv.org/pdf/1503.00759.pdf

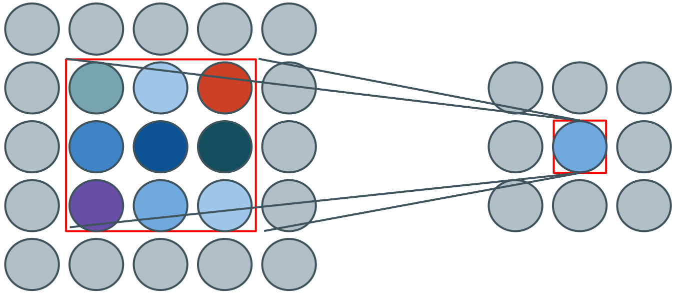
BenevolentAI

Graph Embedding Models for Relationship Inference

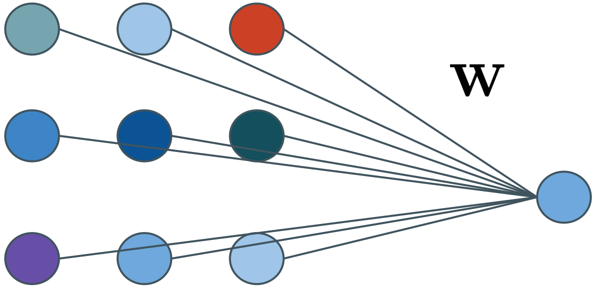
Model	Scoring Function $\psi_r(\mathbf{e}_s, \mathbf{e}_o)$	Relation Parameters	Space Complexity
SE (Bordes et al. 2014)	$\left\lVert \mathbf{W}_r^L \mathbf{e}_s - \mathbf{W}_r^R \mathbf{e}_o \right\rVert_p$	$\mathbf{W}_r^L, \mathbf{W}_r^R \in \mathbb{R}^{k \times k}$	$\mathcal{O}(n_e k + n_r k^2)$
TransE (Bordes et al. 2013a)	$\lVert \mathbf{e}_s + \mathbf{r}_r - \mathbf{e}_o \rVert_p$	$\mathbf{r}_r \in \mathbb{R}^k$	$\mathcal{O}(n_e k + n_r k)$
DistMult (Yang et al. 2015)	$\langle \mathbf{e}_s, \mathbf{r}_r, \mathbf{e}_o \rangle$	$\mathbf{r}_r \in \mathbb{R}^k$	$\mathcal{O}(n_e k + n_r k)$
ComplEx (Trouillon et al. 2016)	$\langle \mathbf{e}_s, \mathbf{r}_r, \mathbf{e}_o \rangle$	$\mathbf{r}_r \in \mathbb{C}^k$	$\mathcal{O}(n_e k + n_r k)$
ConvE	$f(\mathrm{vec}(f([\overline{\mathbf{e}_s}; \overline{\mathbf{r}_r}] * \omega))\mathbf{W})\mathbf{e}_o$	$\mathbf{r}_r \in \mathbb{R}^{k'}$	$\mathcal{O}(n_e k + n_r k')$

Dettmers et al, Convolutional 2D Knowledge Graph Embeddings, AAAI 2018

Convolution

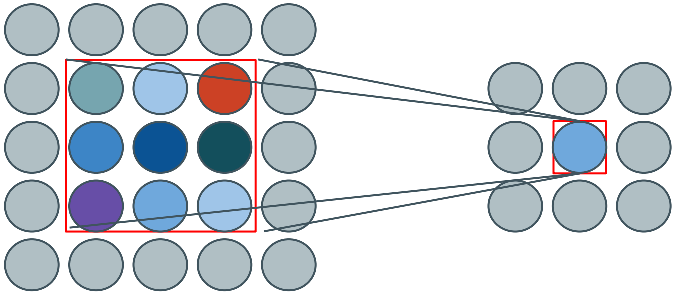


$$\mathbf{x} \in \mathbb{R}^{W \times H}$$

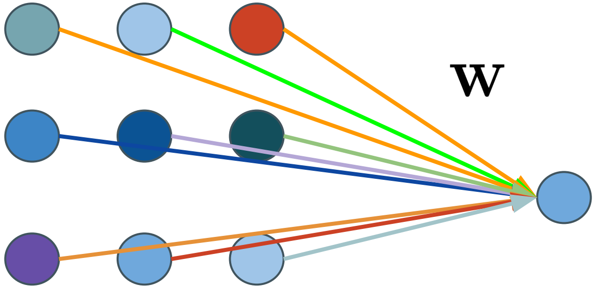


$$p = f(\mathbf{w}^T \mathbf{x} + b)$$

From Convolution on a Grid to Graph

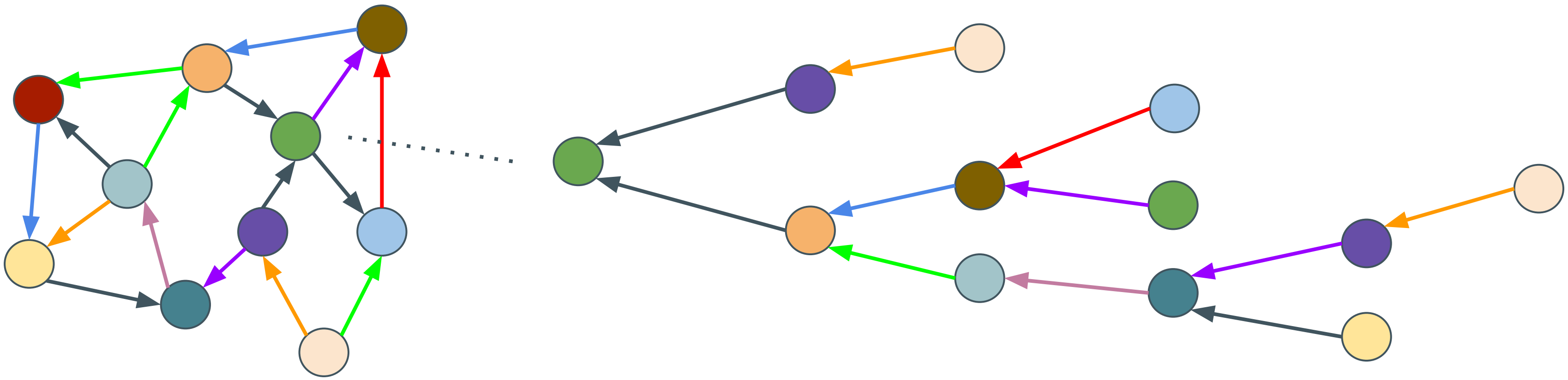


$$\mathbf{x} \in \mathbb{R}^{W \times H}$$



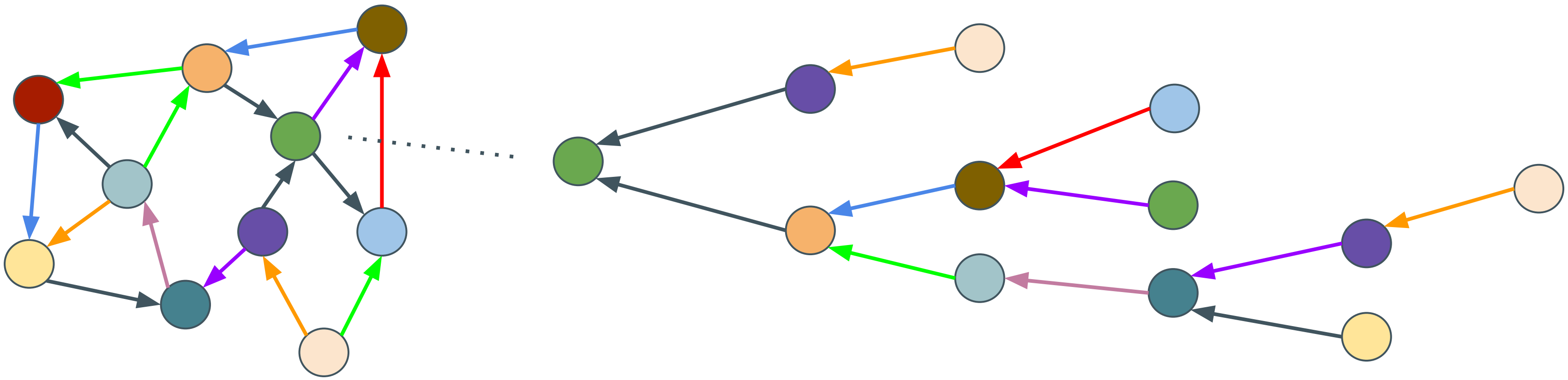
$$p = f(\mathbf{w}^T \mathbf{x} + b)$$

Graph Convolution: Recursive Computation with Shared Parameters



Represent each node based on its neighbourhood

Graph Convolution: Recursive Computation with Shared Parameters

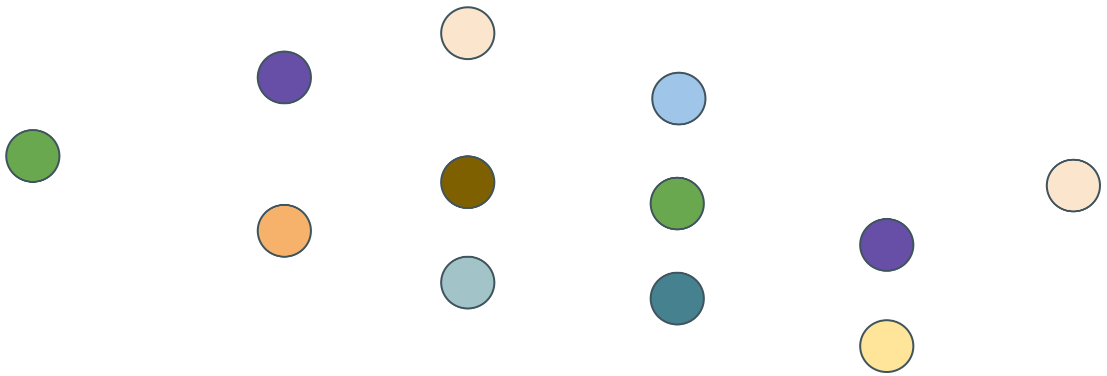


Represent each node based on its neighbourhood

Recursively compute the state of each node by propagating previous states using relation specific transformations

Graph Convolution: Step 0 - Node Embedding

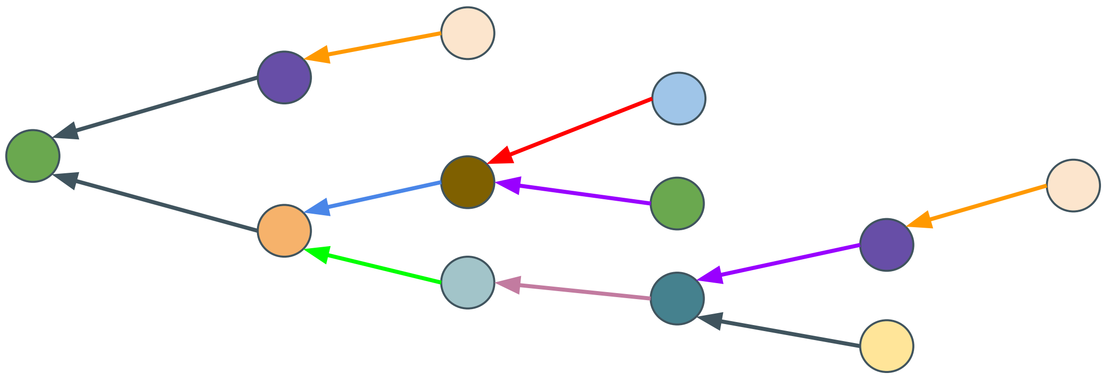
$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$



Graph Convolution: Step k - Messages

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

$$\sum_{v \in N(n)} \frac{\mathbf{h}_v^{k-1}}{|N(n)|}$$

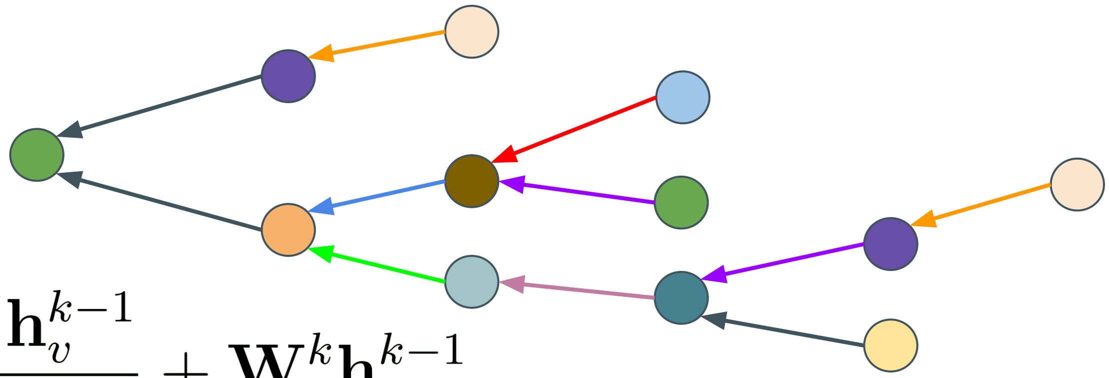


Graph Convolution: Step k - Aggregation

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

$$\sum_{v \in N(n)} \frac{\mathbf{h}_v^{k-1}}{|N(n)|}$$

$$\mathbf{m}_n^k = \sum_{v \in N(n)} \sum_{(n,r,v) \in G} \frac{\mathbf{W}_i^{k,r} \mathbf{h}_v^{k-1}}{|N(n)|} + \mathbf{W}_s^k \mathbf{h}_n^{k-1}$$



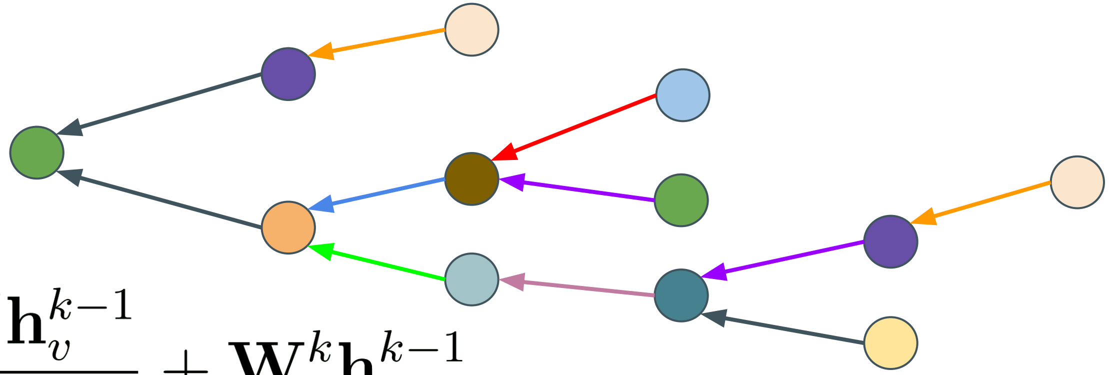
Graph Convolution: Step k - State Update

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

$$\sum_{v \in N(n)} \frac{\mathbf{h}_v^{k-1}}{|N(n)|}$$

$$\mathbf{m}_n^k = \sum_{v \in N(n)} \sum_{(n,r,v) \in G} \frac{\mathbf{W}_i^{k,r} \mathbf{h}_v^{k-1}}{|N(n)|} + \mathbf{W}_s^k \mathbf{h}_n^{k-1}$$

$$\mathbf{h}_n^k = g(\mathbf{m}_n^k)$$



Graph Convolution: Generalisations

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

Embedding model

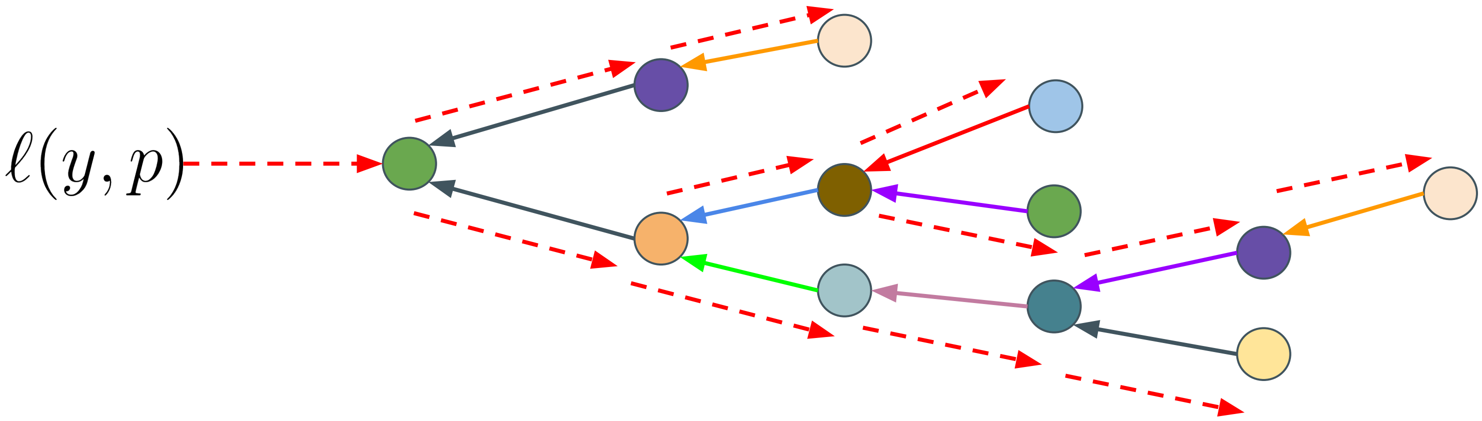
$$\mathbf{a}_n^k = a(\{\mathbf{h}_v^{k-1} | v \in N(n)\}; \mathbf{w}_a)$$

Aggregation model

$$\mathbf{h}_n^k = g(\mathbf{a}_n^k, \mathbf{h}_n^{k-1}; \mathbf{w}_h)$$

State update model

Graph Convolution: Backpropagation through Structure



Graph Convolution: Generalisations

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

Embedding model

$$\mathbf{a}_n^k = a(\{\mathbf{h}_v^{k-1} | v \in N(n)\}; \mathbf{w}_a)$$

Aggregation model

$$\mathbf{h}_n^k = g(\mathbf{a}_n^k, \mathbf{h}_n^{k-1}; \mathbf{w}_h)$$

State update model

Average
Max pooling
LSTM
Attention
...

Graph Convolution: Generalisations

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$ Embedding model

$$\mathbf{a}_n^k = a(\{\mathbf{h}_v^{k-1} | v \in N(n)\}; \mathbf{w}_a)$$ Aggregation model

$$\mathbf{h}_n^k = g(\mathbf{a}_n^k, \mathbf{h}_n^{k-1}; \mathbf{w}_h)$$ State update model

Average
Max pooling
LSTM
Attention
...

Nonlinear map (e.g. Dense + ReLU, ...)
MLP
...

Graph Convolution: Generalisations

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$

Embedding model

$$\mathbf{a}_n^k = a(\{\mathbf{h}_v^{k-1} | v \in N(n)\}; \mathbf{w}_a)$$

Aggregation model

$$\mathbf{h}_n^k = g(\mathbf{a}_n^k, \mathbf{h}_n^{k-1}; \mathbf{w}_h)$$

State update model

Average
Max pooling
LSTM
Attention
…

Nonlinear map (e.g. Dense + ReLU, …)
MLP
…

Dropout
Batch normalisation
…

Graph Convolution: Generalisations

$$\mathbf{h}_n^0 = e(n, \mathbf{x}_n; \mathbf{w}_e^n)$$
Embedding model

$$\mathbf{a}_n^k = a(\{\mathbf{h}_v^{k-1} | v \in N(n)\}; \mathbf{w}_a)$$
Aggregation model

$$\mathbf{h}_n^k = g(\mathbf{a}_n^k, \mathbf{h}_n^{k-1}; \mathbf{w}_h)$$
State update model

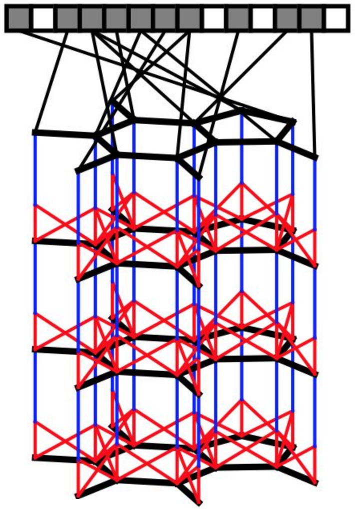
$$p_n = f(\mathbf{h}_n^K; \mathbf{w}_o)$$
Node classification model

$$p_G = f(s(\{\mathbf{h}_n^K | v \in G\}; \mathbf{w}_s); \mathbf{w}_o)$$
Graph classification model

$$p_{n,r,v} = f(\mathbf{h}_n^K, \mathbf{w}_i^{K,r}, \mathbf{h}_v^K; \mathbf{w}_o)$$
Relation inference model

Convolutional Networks on Graphs for Learning Molecular Fingerprints

David Duvenaud†, Dougal Maclaurin†, Jorge Aguilera-Iparraguirre
Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, Ryan P. Adams
Harvard University



Dataset Units	Solubility [4] log Mol/L	Drug efficacy [5] EC_{50} in nM	Photovoltaic efficiency [8] percent
Predict mean	4.29 ± 0.40	1.47 ± 0.07	6.40 ± 0.09
Circular FPs + linear layer	1.71 ± 0.13	**1.13 ± 0.03**	2.63 ± 0.09
Circular FPs + neural net	1.40 ± 0.13	1.36 ± 0.10	2.00 ± 0.09
Neural FPs + linear layer	0.77 ± 0.11	**1.15 ± 0.02**	2.58 ± 0.18
Neural FPs + neural net	**0.52 ± 0.07**	**1.16 ± 0.03**	**1.43 ± 0.09**

Table 1: Mean predictive accuracy of neural fingerprints compared to standard circular fingerprints.

Duvenaud et al, Convolutional Networks on Graphs for Learning Molecular Fingerprints, NIPS 2015

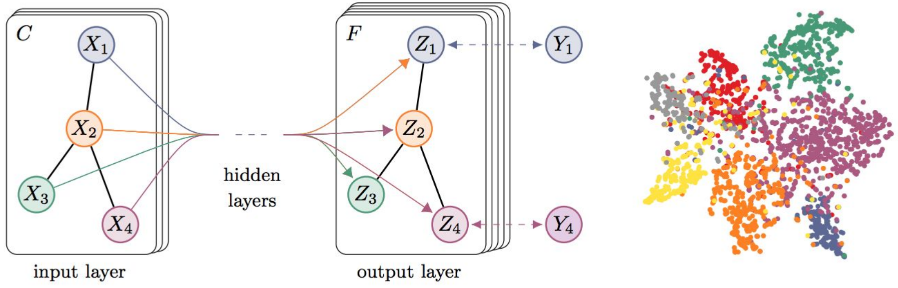
Semi-Supervised Classification with Graph Convolutional Networks

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Table 2: Summary of results in terms of classification accuracy (in percent).

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7



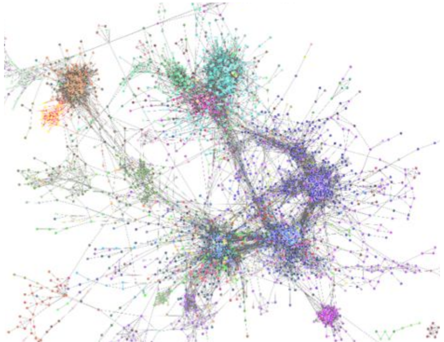
(a) Graph Convolutional Network

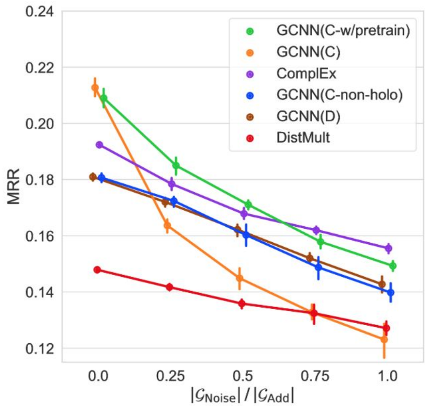
(b) Hidden layer activations

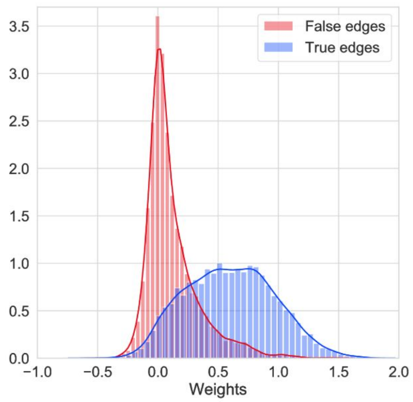
Figure 1: *Left*: Schematic depiction of multi-layer Graph Convolutional Network (GCN) for semi-supervised learning with C input channels and F feature maps in the output layer. The graph structure (edges shown as black lines) is shared over layers, labels are denoted by Y_i. *Right*: t-SNE (Maaten & Hinton, 2008) visualization of hidden layer activations of a two-layer GCN trained on the Cora dataset (Sen et al., 2008) using 5% of labels. Colors denote document class.

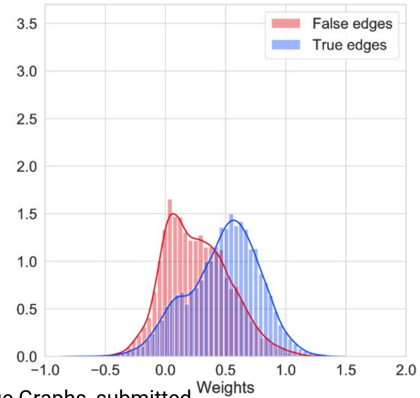
Interpretable Graph Convolutional Neural Networks for Inference on Noisy Knowledge Graphs

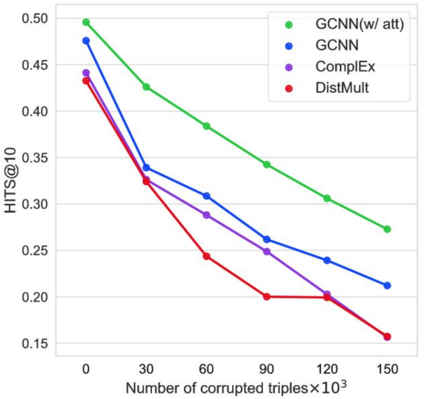
- Real-world large-scale knowledge graphs are automatically extracted
  - Using NER, entity linking, relationship extraction, …
  - Making them noisy
- GCNNs are not interpretable
  - For many applications, interpretability is key
- Added novel edge-specific attention mechanism to GCNNs









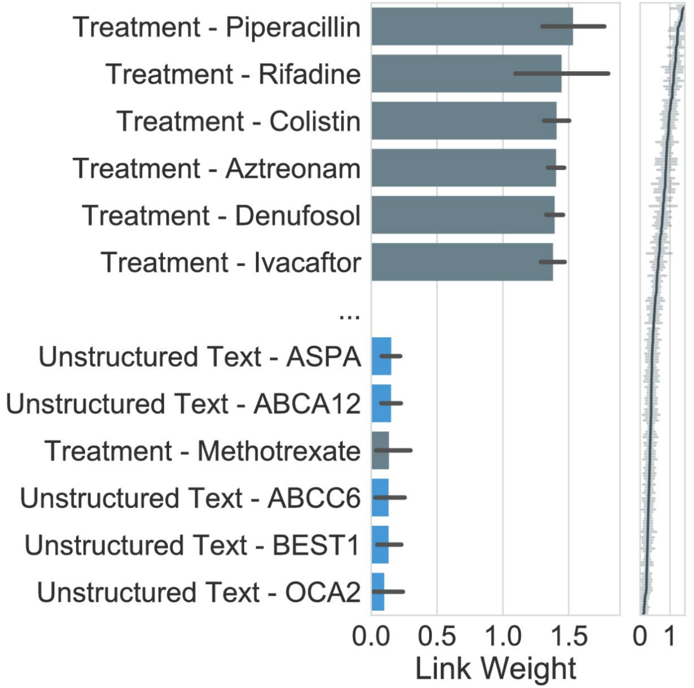


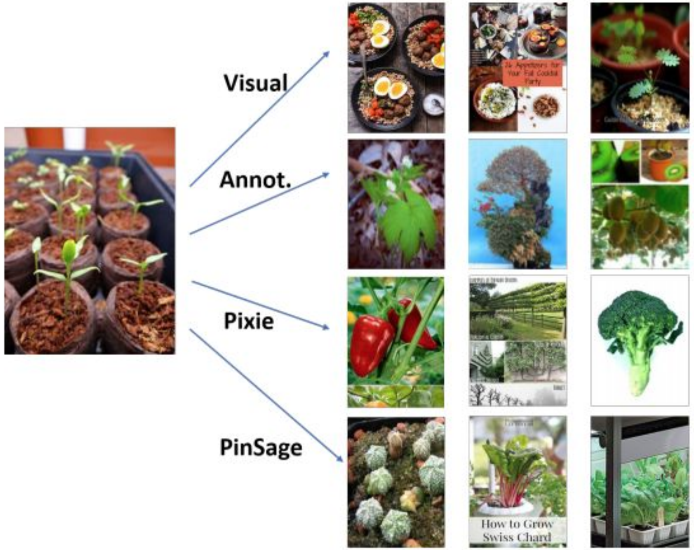
Neil et al, Interpretable Graph Convolutional Neural Networks for Inference on Noisy Knowledge Graphs, submitted

Interpretable Graph Convolutional Neural Networks for Inference on Noisy Knowledge Graphs

	Hits@10				MRR			
Algorithm	100%	50%	Skip	Noised	100%	50%	Skip	Noised
DistMult	43.2	20.2	N/A	20.6	23.9	8.69	N/A	8.93
ComplEx	44.1	24.1	N/A	24.3	25.9	10.9	N/A	11.0
GCNN	47.5	33.2	25.8	21.4	27.2	16.8	13.3	11.1
GCNN w/att	**48.2**	**34.7**	**34.0**	**35.6**	**28.3**	**18.5**	**18.8**	**19.1**

Table 1: Performance on the FB15k-237 Dataset. For "100%" we train using the full train set. In "50%" only half this data is used. For "Skip" and "Noised," half of the remaining data is corrupted. "Noised" is trained in a normal way also using this noisy data. In "Skip" it is used exclusively to populate the adjacency matrix.





Method	Hit-rate	MRR
Visual	17%	0.23
Annotation	14%	0.19
Combined	27%	0.37
max-pooling	39%	0.37
mean-pooling	41%	0.51
mean-pooling-xent	29%	0.35
mean-pooling-hard	46%	0.56
PinSage	67%	**0.59**

Table 1: Hit-rate and MRR for PinSage and content-based deep learning baselines. Overall, PinSage gives 150% improvement in hit rate and 60% improvement in MRR over the best baseline.[5]

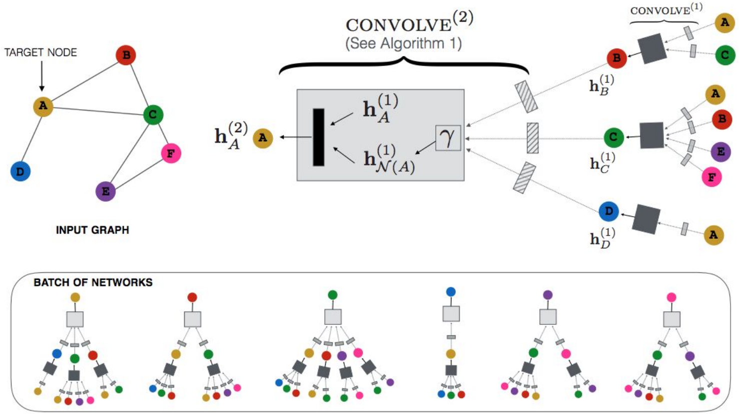
Graph Convolutional Neural Networks for Web-Scale Recommender Systems

Rex Ying*†, Ruining He*, Kaifeng Chen*†, Pong Eksombatchai*,

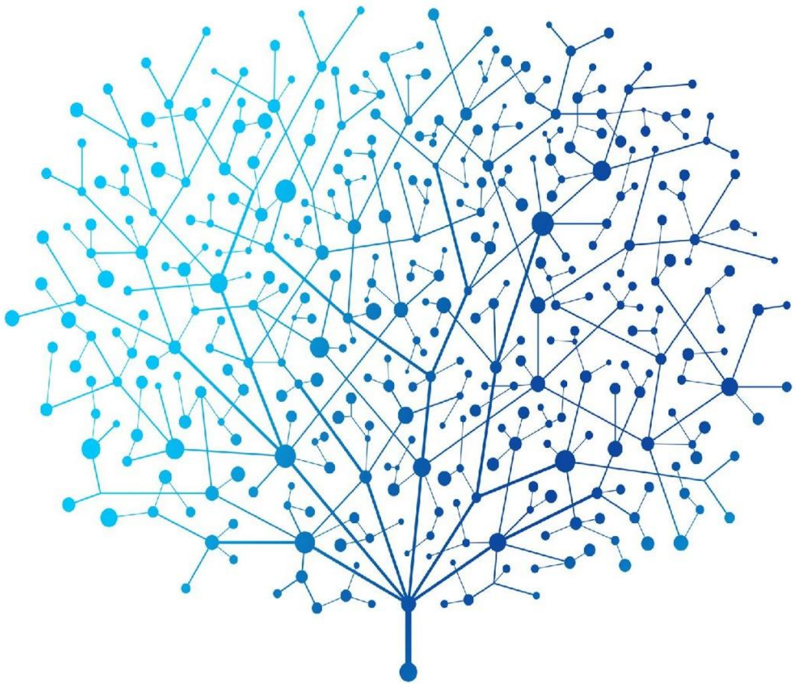
William L. Hamilton†, Jure Leskovec*†

*Pinterest, †Stanford University

{rhe,kaifengchen,pong}@pinterest.com,{rexying,wleif,jure}@stanford.edu



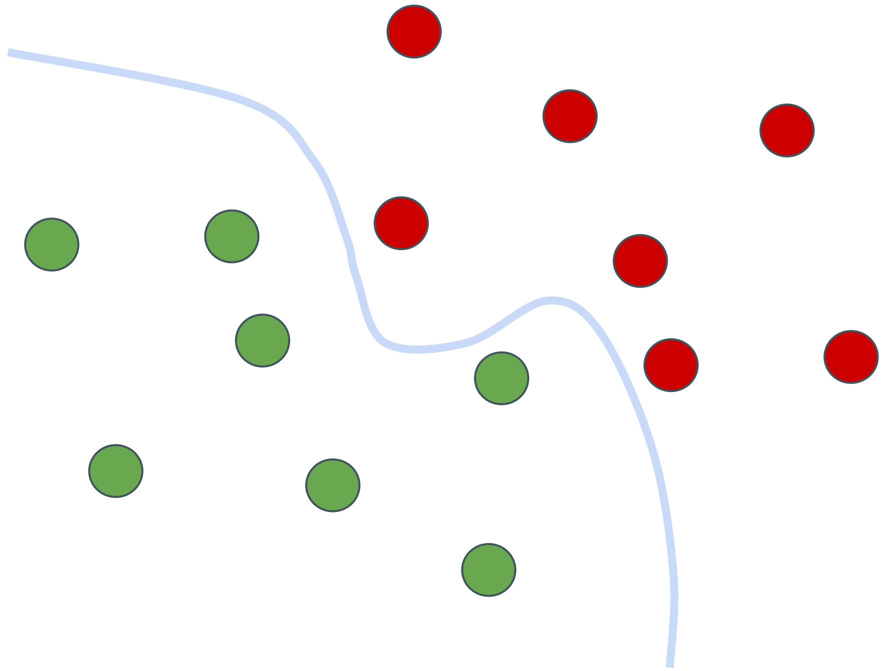
We deploy PinSage at Pinterest and train it on 7.5 billion examples on a graph with 3 billion nodes representing pins and boards, and 18 billion edges. According to offline metrics, user studies and



Generative Models for Graphs

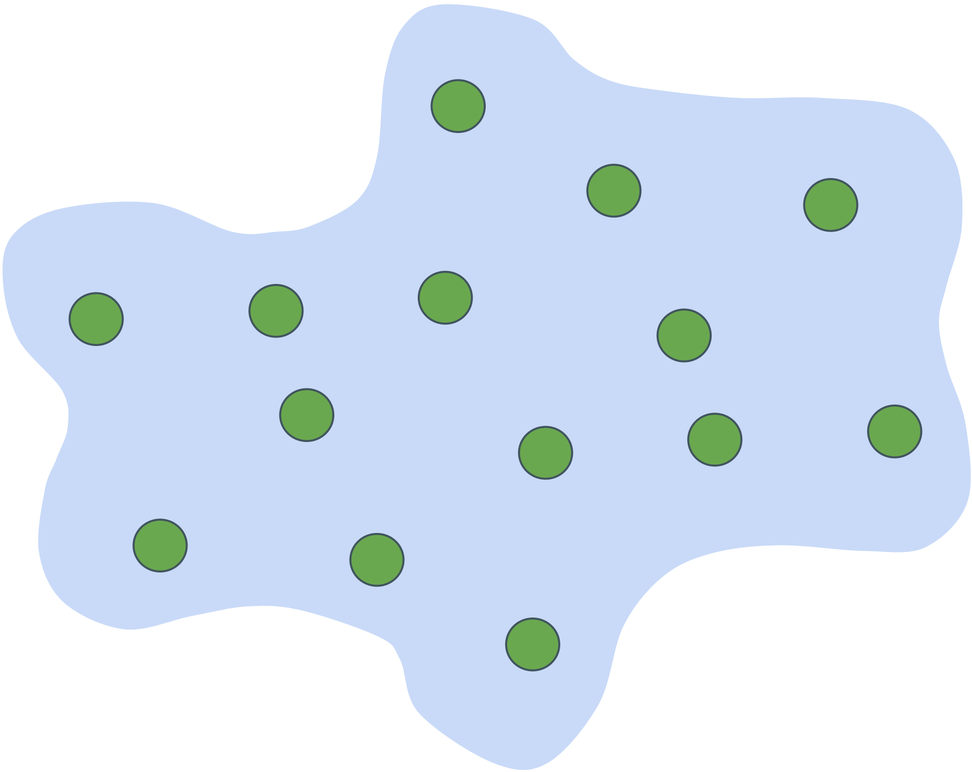
Discriminative Models

$$p(y|\mathbf{x};\mathbf{w})$$



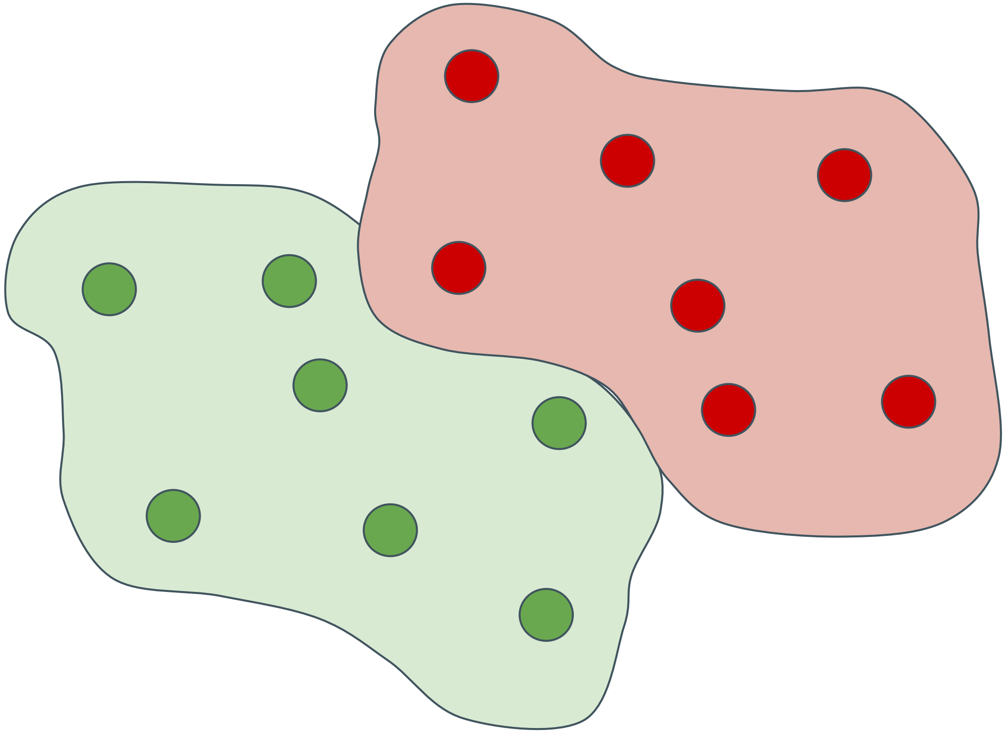
Generative Models

$$p(\mathbf{x}; \mathbf{w})$$



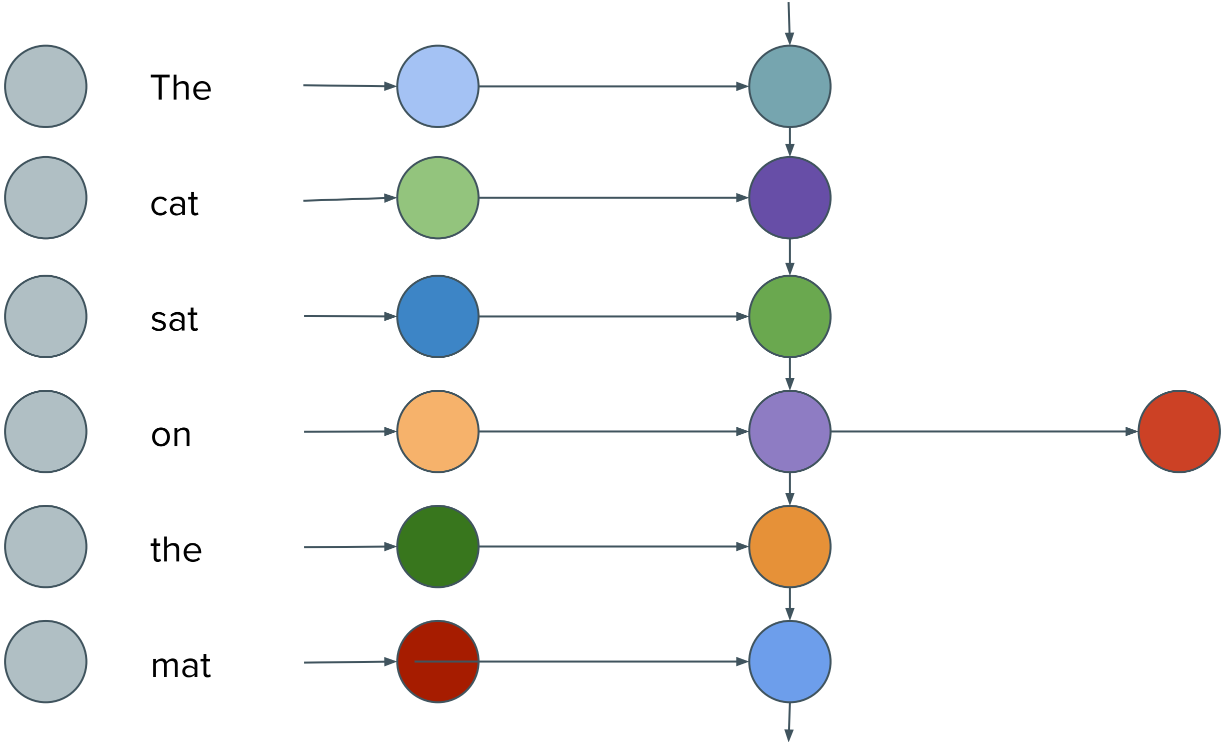
Generative Models

$$p(y, \mathbf{x}; \mathbf{w})$$



Autoregressive Models for Sequential Data

$$\mathbf{e}_t = e(x_t; \mathbf{w}_e) \qquad \mathbf{h_t} = g(\mathbf{e}_t, \mathbf{h}_{t-1}; \mathbf{w}_h)$$

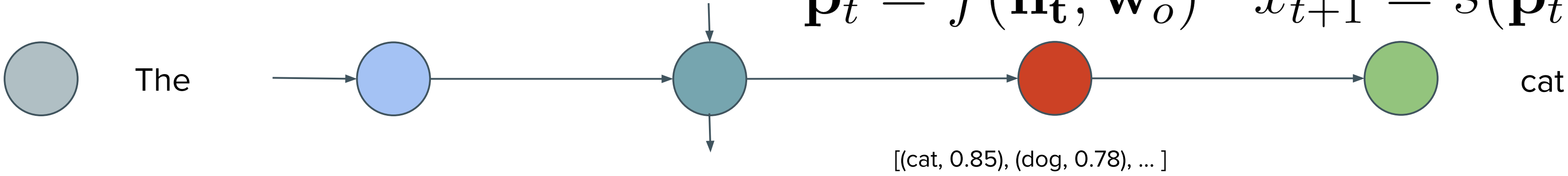


$$p_t = f(\mathbf{h_t}; \mathbf{w}_o)$$

Autoregressive Models for Sequential Data

$$\mathbf{e}_t = e(x_t; \mathbf{w}_e) \qquad \mathbf{h_t} = g(\mathbf{e}_t, \mathbf{h}_{t-1}; \mathbf{w}_h)$$

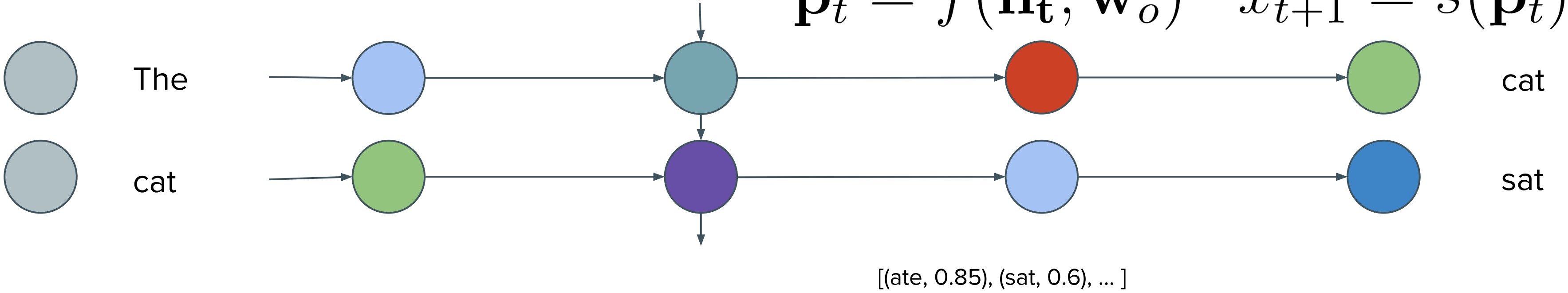
$$\mathbf{p}_t = f(\mathbf{h_t}; \mathbf{w}_o) \qquad x_{t+1} = s(\mathbf{p}_t)$$



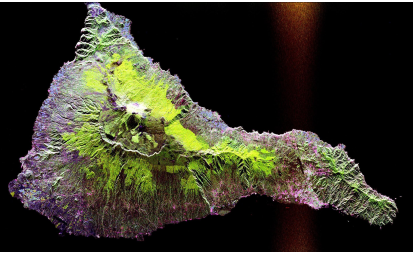
Autoregressive Models for Sequential Data

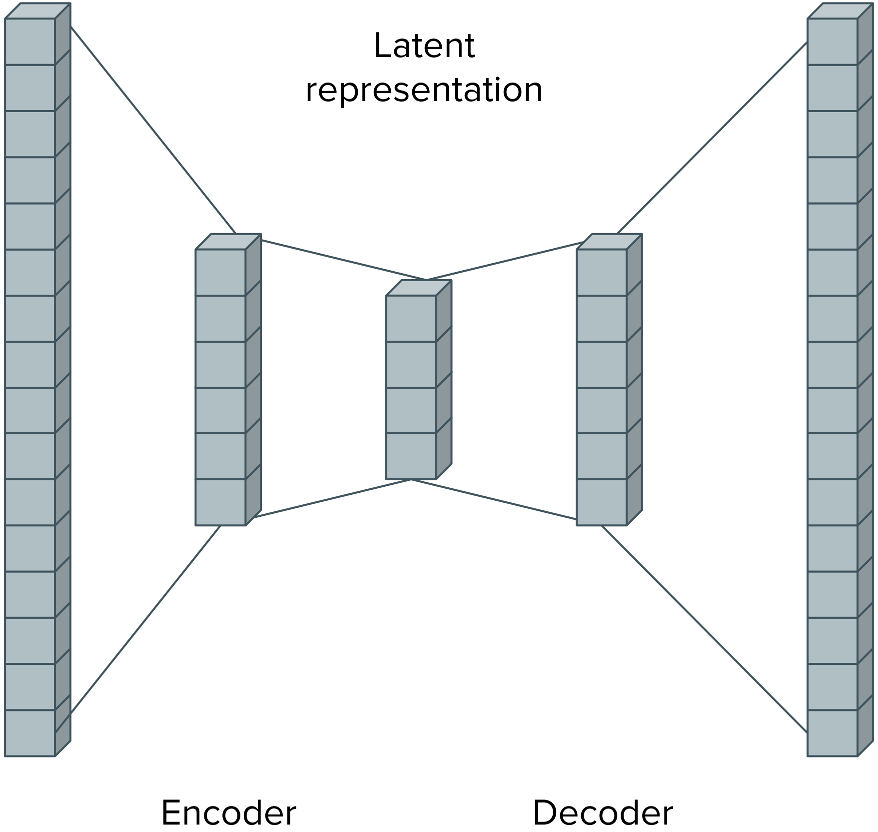
$$\mathbf{e}_t = e(x_t; \mathbf{w}_e) \quad \mathbf{h_t} = g(\mathbf{e}_t, \mathbf{h}_{t-1}; \mathbf{w}_h)$$

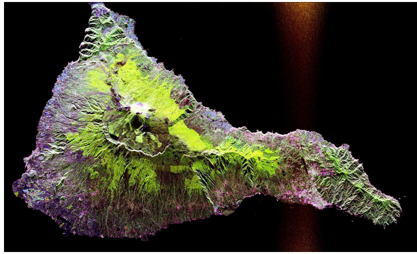
$$\mathbf{p}_t = f(\mathbf{h_t}; \mathbf{w}_o) \quad x_{t+1} = s(\mathbf{p}_t)$$



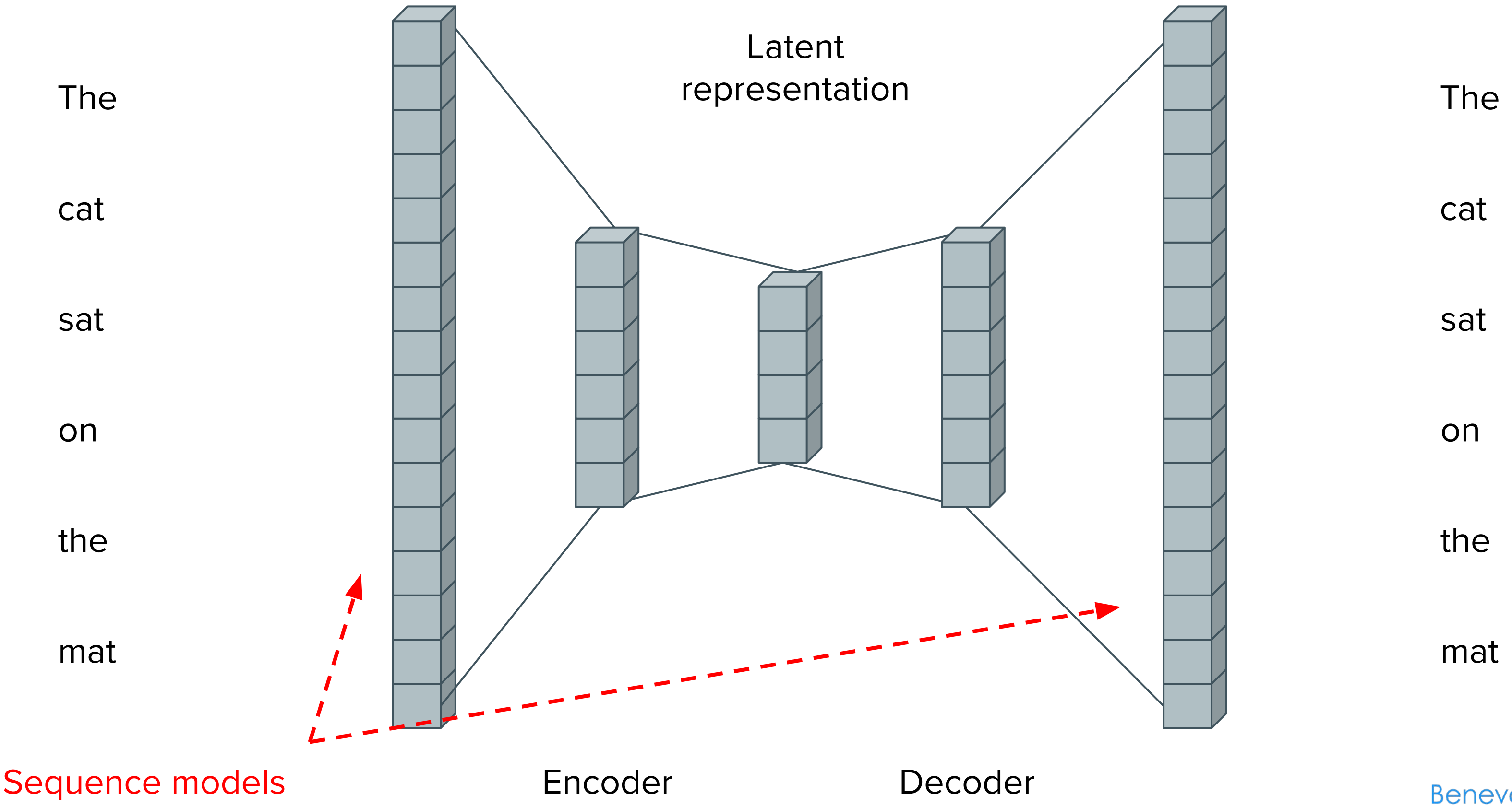
AutoEncoder Models



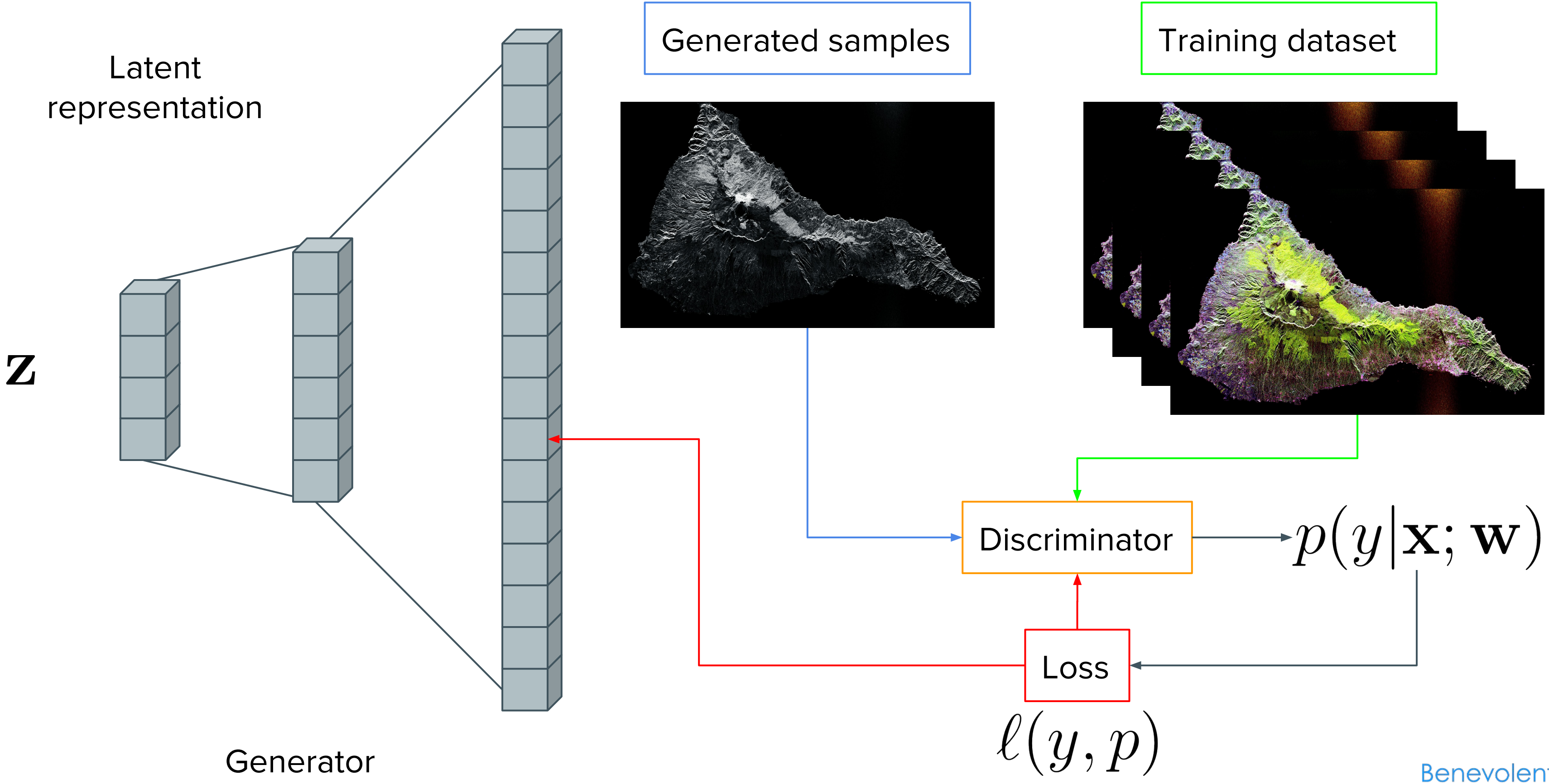




AutoEncoder Models

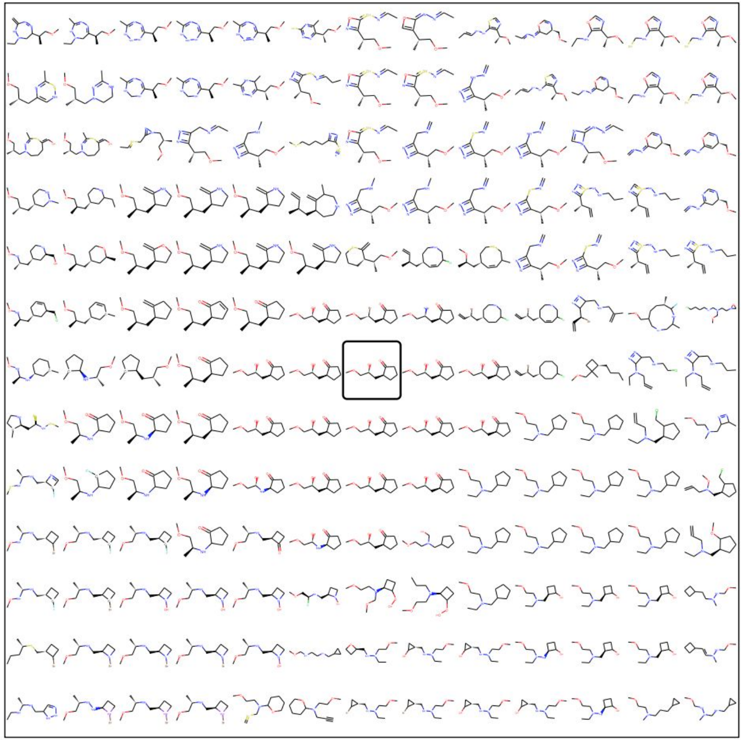


Generative Adversarial Network Models



Grammar Variational Autoencoder

Matt J. Kusner [1 2] Brooks Paige [1 3] José Miguel Hernández-Lobato [3]



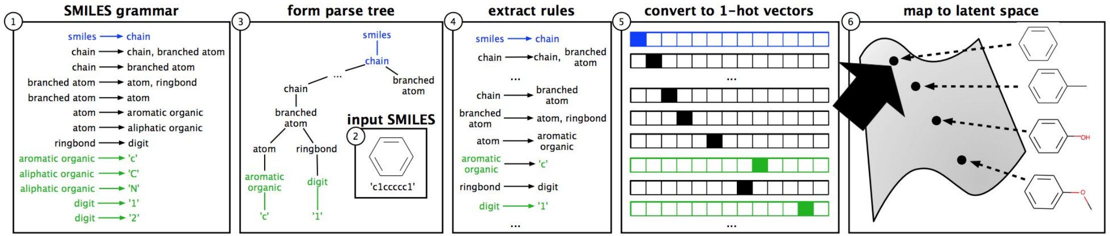


Figure 1. The encoder of the GVAE. We denote the start rule in blue and all rules that decode to terminal in green. See text for details.

Table 5. Test Log-likelihood (LL) and RMSE for the sparse GP predictions of penalized LogP score from the latent space

Objective	Method	Expressions	Molecules
LL	GVAE	**-1.320±0.001**	**-1.739 ±0.004**
	CVAE	-1.397±0.003	-1.812±0.004
RMSE	GVAE	**0.884 ±0.002**	**1.404 ±0.006**
	CVAE	0.975±0.004	1.504±0.006

Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design

Daniel Neil, Marwin Segler, Laura Guasch, Mohamed Ahmed, Dean Plumbley, Matthew Sellwood, Nathan Brown

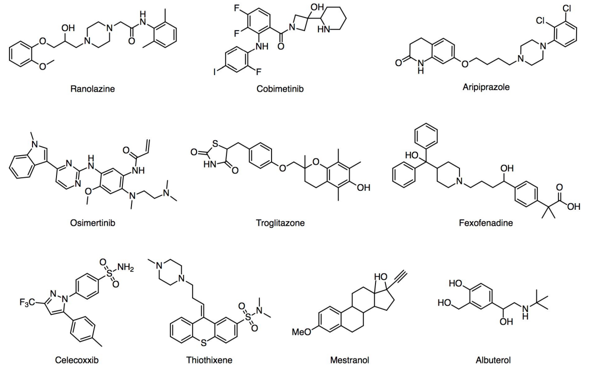


Figure 6: Target molecules for the Tanimoto benchmark.

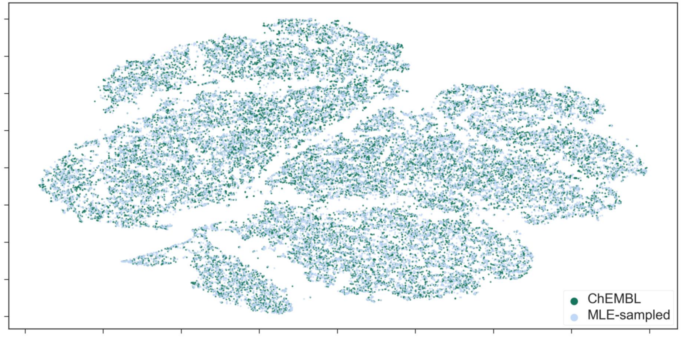


Figure 5: t-SNE visualization (Maaten & Hinton, 2008) of MLE sampling of generated space. The MLE model effectively covers the space of ChEMBL and even reproduces the subspaces around the ChEMBL molecules.

Table 1: Model performance, given by mean fitness in the final timestep over three random initializations, while single-best SMILES result from the plotted runs is given in parentheses.

		Baseline	Reg. PG	A2C	PPO	HC-MLE
Property	LogP=-1	1.00	0.66 (1.00)	0.98 (1.00)	1.00 (1.00)	0.97 (1.00)
	LogP=0	1.00	0.78 (1.00)	0.98 (1.00)	1.00 (1.00)	0.98 (1.00)
	LogP=1	1.00	0.83 (1.00)	0.98 (1.00)	1.00 (1.00)	0.97 (1.00)
	LogP=2	1.00	0.86 (1.00)	0.97 (1.00)	1.00 (1.00)	0.97 (1.00)
	LogP=3	1.00	0.86 (1.00)	0.97 (1.00)	0.91 (1.00)	0.97 (1.00)
Mult. Obj.	MPO	1.00	0.82 (1.00)	0.95 (1.00)	1.00 (1.00)	0.98 (1.00)
	Ro5	1.00	0.77 (1.00)	0.96 (1.00)	1.00 (1.00)	0.59 (1.00)
Tanimoto	Albuterol	0.02	-0.55 (0.41)	0.14 (-0.08)	0.04 (-0.10)	0.32 (0.83)
	Aripiprazole	-0.15	-0.34 (0.63)	0.38 (-0.12)	0.40 (0.29)	0.51 (1.00)
	Celecoxxib	-0.22	-0.35 (0.69)	0.20 (-0.06)	0.25 (0.14)	0.43 (1.00)
	Cobimetinib	-0.18	-0.47 (0.17)	-0.01 (-0.01)	0.11 (0.06)	0.32 (0.57)
	Fexofenadine	-0.26	-0.33 (0.50)	-0.24 (-0.13)	0.18 (0.19)	0.47 (0.82)
	Mestranol	-0.17	-0.46 (0.62)	0.14 (-0.22)	0.06 (0.30)	0.34 (0.85)
	Osimertinib	-0.44	-0.43 (0.15)	-0.36 (-0.26)	-0.11 (0.11)	0.13 (0.48)
	Ranolazine	-0.20	-0.32 (0.49)	0.32 (-0.19)	0.14 (0.47)	0.50 (1.00)
	Thiothixene	-0.26	-0.35 (0.28)	-0.09 (-0.19)	0.07 (0.29)	0.33 (0.57)
	Troglitazone	-0.28	-0.39 (0.27)	-0.19 (-0.27)	0.06 (0.18)	0.24 (0.56)
Summary	Mean	0.30	0.09 (0.66)	0.42 (0.32)	0.48 (0.53)	0.59 (0.81)
	Runtime	0.025s	0.68s	2.5s	8.54s	0.31s

Neil et al, Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design, ICLR 2018

GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models

Jiaxuan You [*1] **Rex Ying** [*1] **Xiang Ren** [2] **William L. Hamilton** [1] **Jure Leskovec** [1]

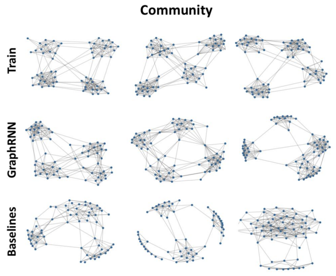


Figure 7. Visualization of graph dataset with four communities. Graphs from training set (First row), graphs generated by GraphRNN(Second row) and graphs generated by Kronecker, MMSB and B-A baselines respectively (Third row) are shown.

Table 2. GraphRNN compared to state-of-the-art deep graph generative models on small graph datasets using MMD and negative log-likelihood (NLL). $(\max(|V|), \max(|E|))$ of each dataset is shown. (DeepVAE and GraphVAE cannot scale to the graphs in Table 1.)

	Community-small (20,83)					Ego-small (18,69)				
	Degree	Clustering	Orbit	Train NLL	Test NLL	Degree	Clustering	Orbit	Train NLL	Test NLL
GraphVAE	0.35	0.98	0.54	13.55	25.48	0.13	0.17	0.05	12.45	14.28
DeepGMG	0.22	0.95	0.40	106.09	112.19	0.04	0.10	0.02	21.17	22.40
GraphRNN-S	**0.02**	0.15	**0.01**	31.24	35.94	0.002	**0.05**	**0.0009**	8.51	9.88
GraphRNN	0.03	**0.03**	**0.01**	28.95	35.10	**0.0003**	**0.05**	**0.0009**	9.05	10.61

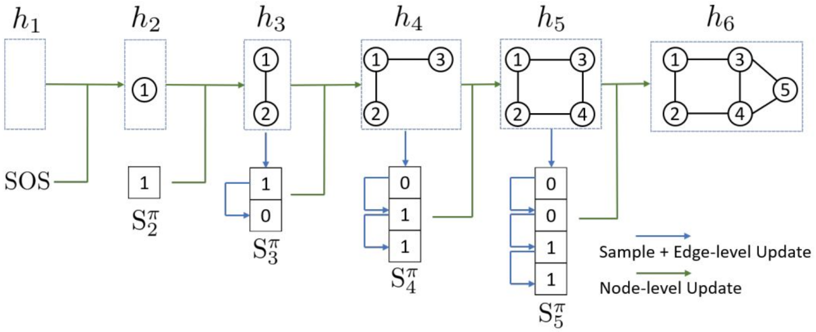


Figure 1. GraphRNN at inference time. Green arrows denote the graph-level RNN that encodes the "graph state" vector h_i in its hidden state, updated by the predicted adjacency vector S_i^π for node $\pi(v_i)$. Blue arrows represent the edge-level RNN, whose hidden state is initialized by the graph-level RNN, that is used to predict the adjacency vector S_i^π for node $\pi(v_i)$.

You et al, GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models, ICML 2018

Learning Deep Generative Models of Graphs

Yujia Li [1] Oriol Vinyals [1] Chris Dyer [1] Razvan Pascanu [1] Peter Battaglia [1]

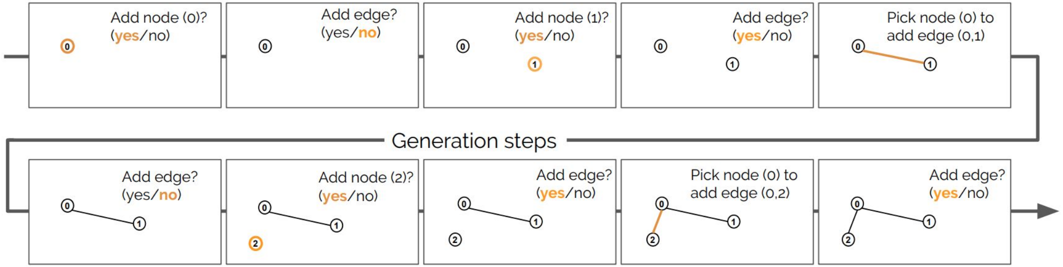


Figure 1. Depiction of the steps taken during the generation process.

Table 2. Molecule generation results. N is the number of permutations for each molecule the model is trained on. Typically the number of different SMILES strings for each molecule < 100.

Arch	Grammar	Ordering	N	NLL	%valid	%novel
LSTM	SMILES	Fixed	1	21.48	93.59	81.27
LSTM	SMILES	Random	< 100	**19.99**	93.48	83.95
LSTM	Graph	Fixed	1	22.06	85.16	80.14
LSTM	Graph	Random	$O(n!)$	63.25	91.44	91.26
Graph	Graph	Fixed	1	20.55	**97.52**	90.01
Graph	Graph	Random	$O(n!)$	58.36	95.98	**95.54**

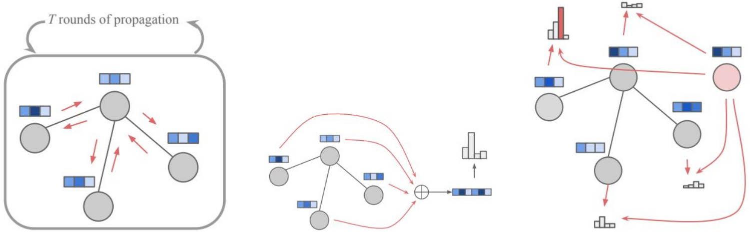


Figure 2. Illustration of the graph propagation process (left), graph level predictions using $f_{addnode}$ and $f_{addedge}$ (center), and node selection f_{nodes} modules (right).

Li et al, Learning Deep Generative Models of Graphs, ICML 2018

Constrained Graph Variational Autoencoders for Molecule Design

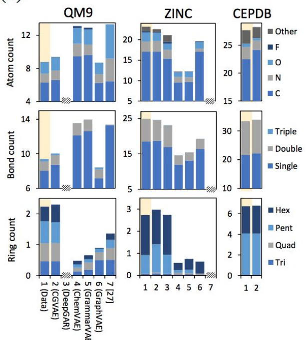
Qi Liu[*1], **Miltiadis Allamanis**[2], **Marc Brockschmidt**[2], and **Alexander L. Gaunt**[2]

[1]Singapore University of Technology and Design
[2]Microsoft Research, Cambridge, UK
qiliu@u.nus.edu, {miallama, mabrocks, algaunt}@microsoft.com

(a)

	Measure	2: CGVAE	3: [21]	4: [8]	5: [17]	6: [29]	7: [27]
QM9	% valid	100	-	10.00	30.00	61.00	98.00
	% novel	94.35	-	90.00	95.44	85.00	100
	% unique	98.57	-	67.50	9.30	40.90	99.86
ZINC	% valid	100	89.20	17.00	31.00	14.00	-
	% novel	100	89.10	98.00	100	100	-
	% unique	99.82	99.41	30.98	10.76	31.60	-
CEPDB	% valid	100	-	8.30	0.00	-	-
	% novel	100	-	90.05	-	-	-
	% unique	99.62	-	80.99	-	-	-

(b)



(c)

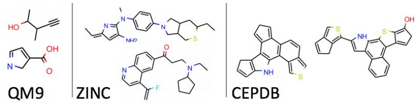


Figure 3: Overview of statistics of sampled molecules from a range of generative models trained on different datasets. In (b) We highlight the target statistics of the datasets in yellow and use the numbers 2, ..., 7 to denote different models as shown in the axis key. A hatched box indicates where other works do not supply benchmark results. Two samples from our model on each dataset are shown in (c), with more random samples given in supplementary material A.

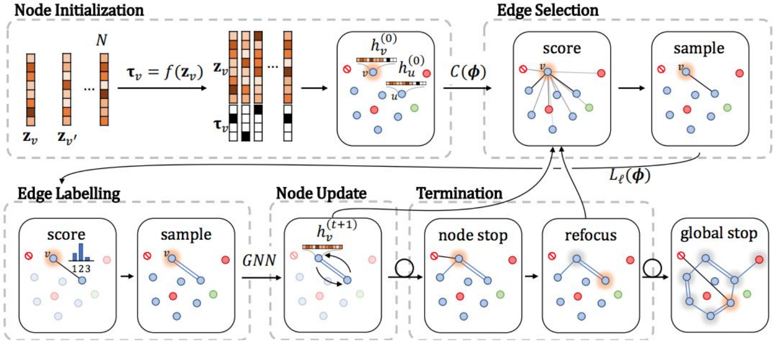


Figure 1: Illustration of the phases of the generative procedure. Nodes are initialized with latent variables and then we enter a loop between edge selection, edge labelling and node update steps until the special stop node $\oslash$ is selected. We then refocus to a new node or terminate if there are no candidate focus nodes in the connected component. A looped arrow indicates that several loop iterations may happen between the illustrated steps.

Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation

Jiaxuan You[1]* jiaxuan@stanford.edu

Bowen Liu[2]* liubowen@stanford.edu

Rex Ying[1] rexying@stanford.edu

Vijay Pande[2] pande@stanford.edu

Jure Leskovec[1] jure@cs.stanford.edu

[1]Department of Computer Science, [2]Department of Chemistry
Stanford University
Stanford, CA, 94305

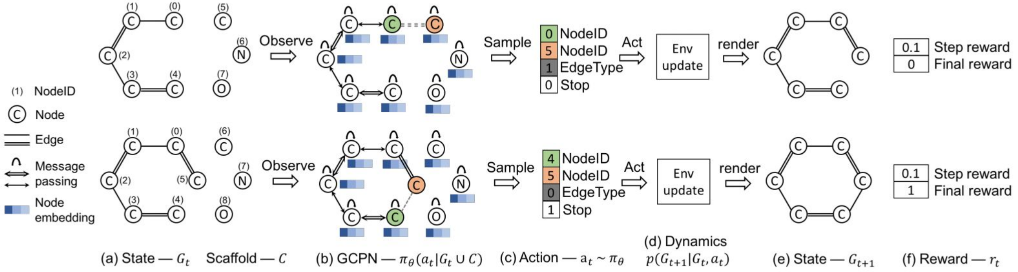


Figure 1: An overview of the proposed iterative graph generation method. Each row corresponds to one step in the generation process. **(a)** The state is defined as the intermediate graph G_t, and the set of scaffold subgraphs defined as C is appended for GCPN calculation. **(b)** GCPN conducts message passing to encode the state as node embeddings then produce a policy π_θ. **(c)** An action a_t with 4 components is sampled from the policy. **(d)** The environment performs a chemical valency check on the intermediate state, and then returns **(e)** the next state G_{t+1} and **(f)** the associated reward r_t.

Table 2: Comparison of the effectiveness of property targeting task.

Method	$-2.5 \leq \text{logP} \leq -2$		$5 \leq \text{logP} \leq 5.5$		$150 \leq \text{MW} \leq 200$		$500 \leq \text{MW} \leq 550$	
	Success	Diversity	Success	Diversity	Success	Diversity	Success	Diversity
ZINC	0.3%	0.919	1.3%	0.909	1.7%	0.938	0	–
JT-VAE	11.3%	**0.846**	7.6%	0.907	0.7%	0.824	16.0%	0.898
ORGAN	0	–	0.2%	**0.909**	15.1%	0.759	0.1%	0.907
GCPN	**85.5%**	0.392	**54.7%**	0.855	**76.1%**	**0.921**	**74.1%**	**0.920**

Table 3: Comparison of the performance in the constrained optimization task.

δ	JT-VAE			GCPN		
	Improvement	Similarity	Success	Improvement	Similarity	Success
0.0	1.91 ± 2.04	0.28 ± 0.15	97.5%	$\mathbf{4.20 \pm 1.28}$	$\mathbf{0.32 \pm 0.12}$	**100.0%**
0.2	1.68 ± 1.85	0.33 ± 0.13	97.1%	$\mathbf{4.12 \pm 1.19}$	$\mathbf{0.34 \pm 0.11}$	**100.0%**
0.4	0.84 ± 1.45	$\mathbf{0.51 \pm 0.10}$	83.6%	$\mathbf{2.49 \pm 1.30}$	0.47 ± 0.08	**100.0%**
0.6	0.21 ± 0.71	0.69 ± 0.06	46.4%	$\mathbf{0.79 \pm 0.63}$	$\mathbf{0.68 \pm 0.08}$	**100.0%**

References

- Duvenaud et al, Convolutional Networks on Graphs for Learning Molecular Fingerprints, NIPS 2015
- Niepert et al, Learning Convolutional Neural Networks for Graphs, ICML 2016
- Kipf et al, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017
- Ying et al, Graph Convolutional Neural Networks for Web-Scale Recommender Systems, KDD 2018
- Dettmers et al, Convolutional 2D Knowledge Graph Embeddings, AAAI 2018
- Neil et al, Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design, ICLR 2018
- You et al, GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models, ICML 2018
- Li et al, Learning Deep Generative Models of Graphs, ICML 2018
- Liu et al, Constrained Graph Variational Autoencoders for Molecule Design, arXiv:1805.09076, 2018
- You et al, Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, arXiv:1806.02473, 2018

Thanks!

Amir Saffari

@amirsaffari, amir.saffariazar@benevolent.ai

BenevolentAI