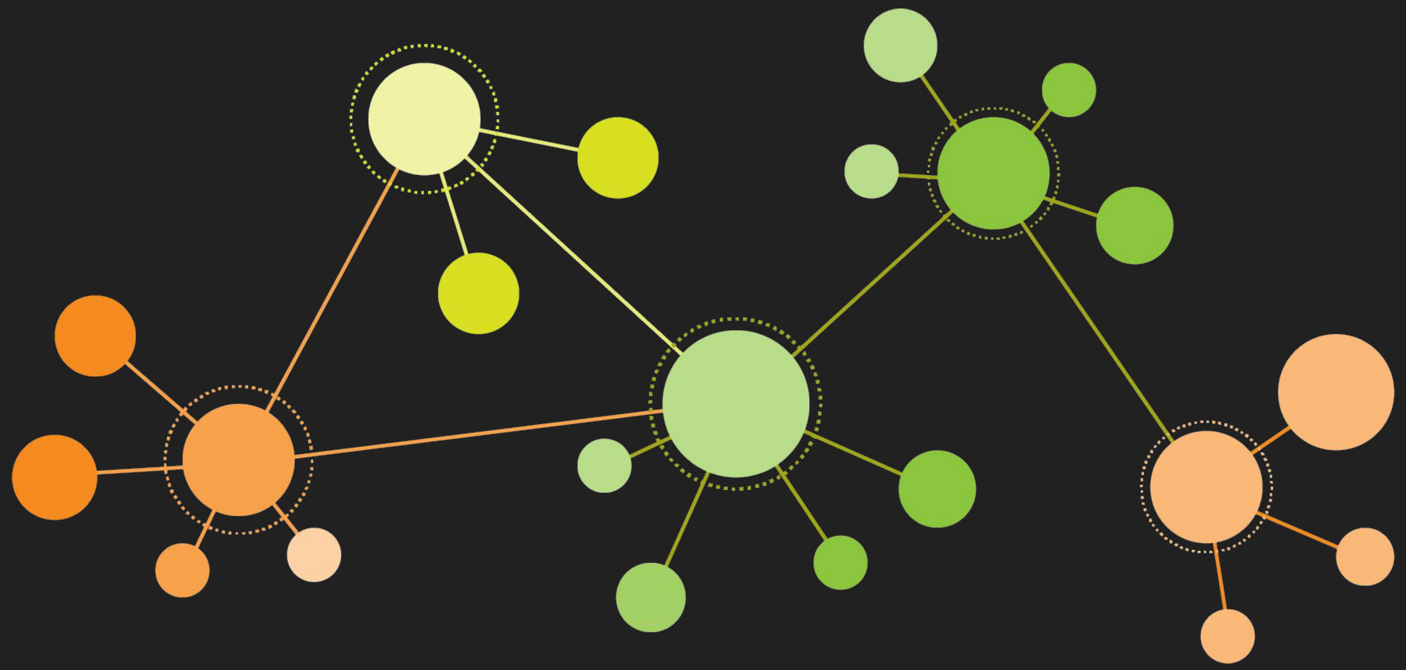
Graph Neural Networks

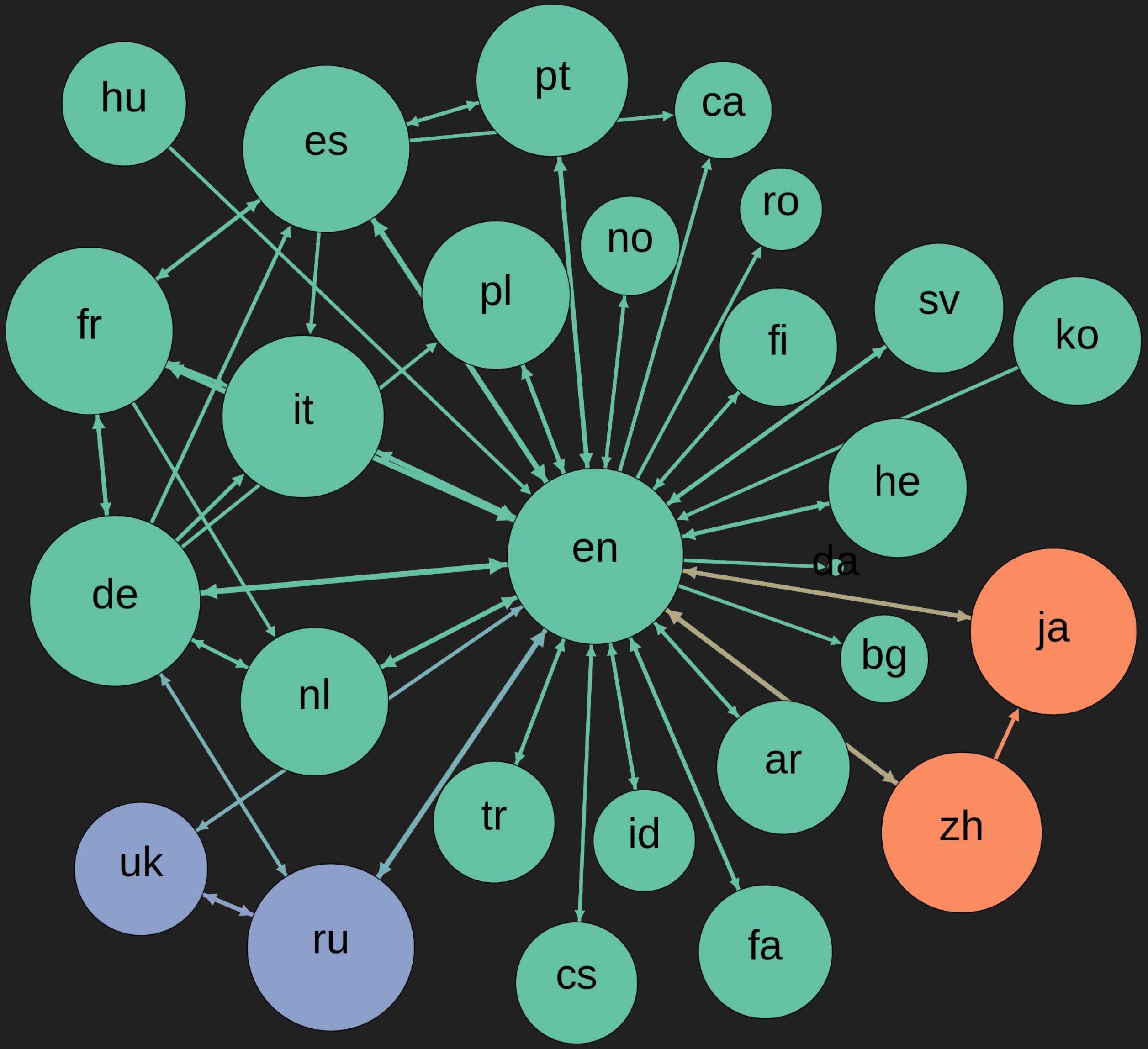
And how they changed the landscape of graph processing

Peter Jung

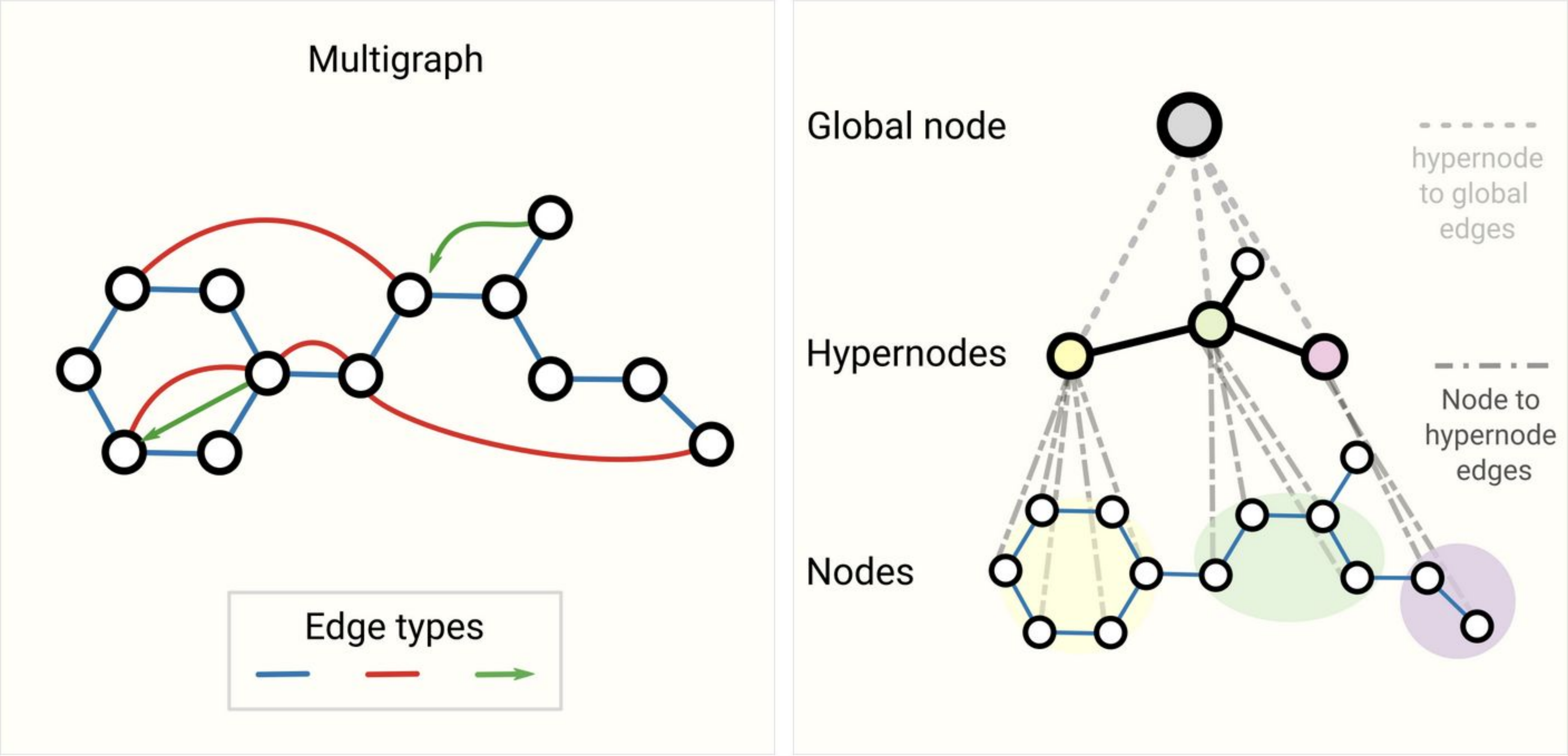


Introduction

The Graph

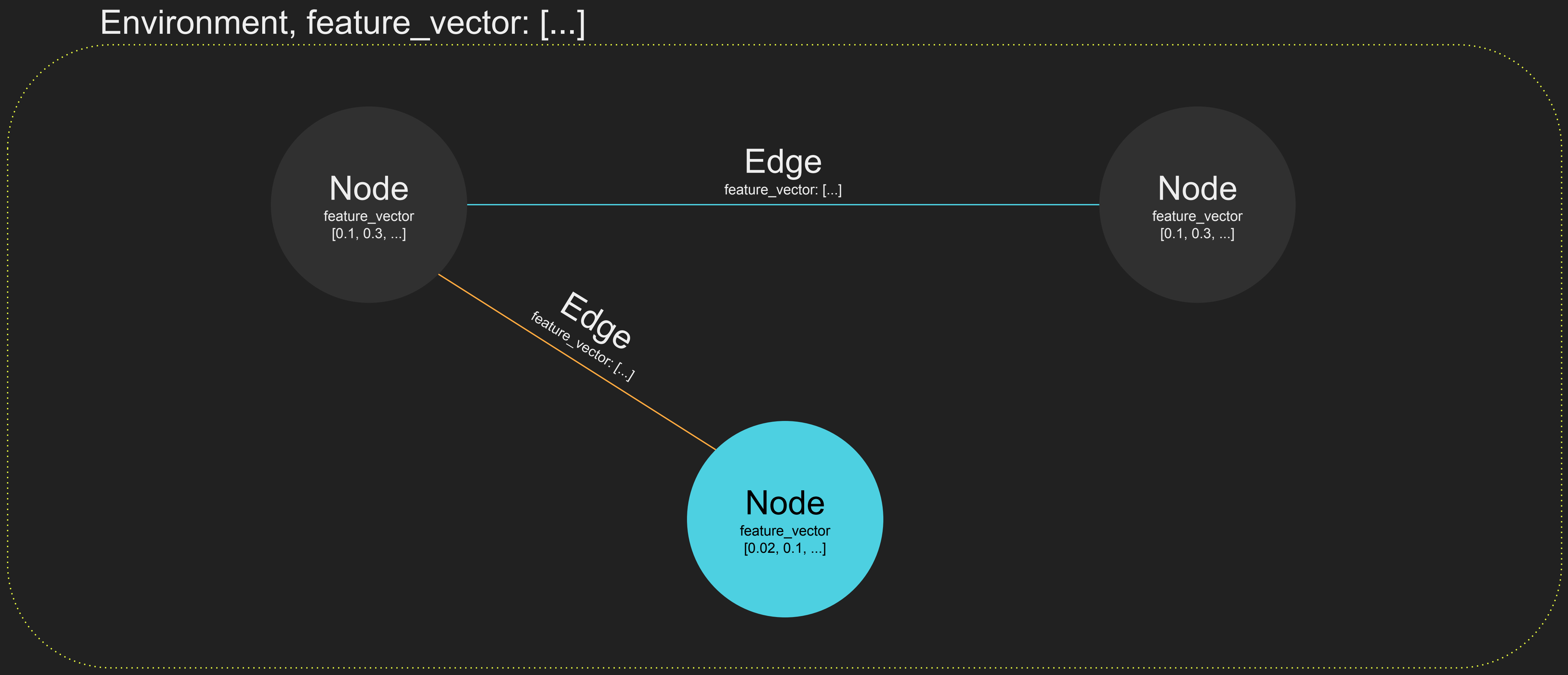


hu
pt
es
ca
ro
no
pl
sv
fr
fi
ko
it
he
en
da
de
ja
bg
nl
ar
tr
zh
id
uk
ru
fa
cs



Multigraph
Edge types
Global node
hypernode
to global
edges
Hypernodes
Node to
hypernode
edges
Nodes

Generalized Graph Environment



Strict Types of Graphs

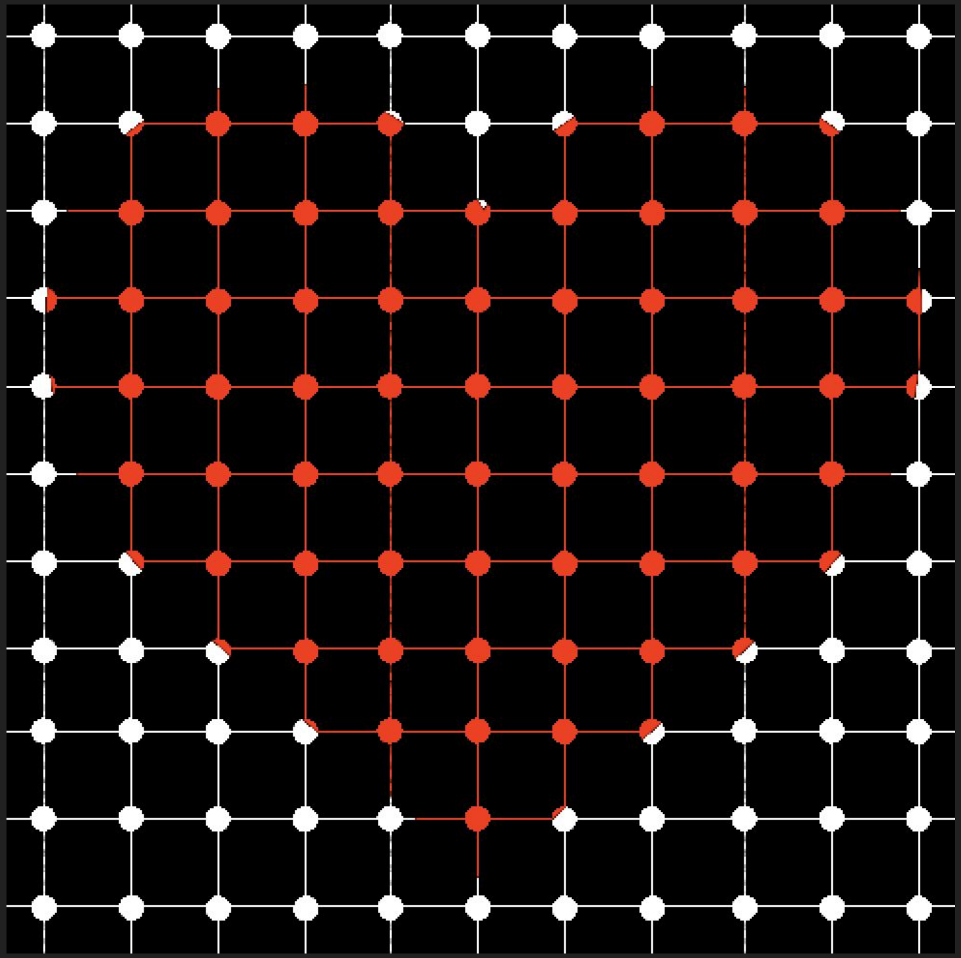
- A lot of things can be generalized to a graph

- Usually we work with strict types

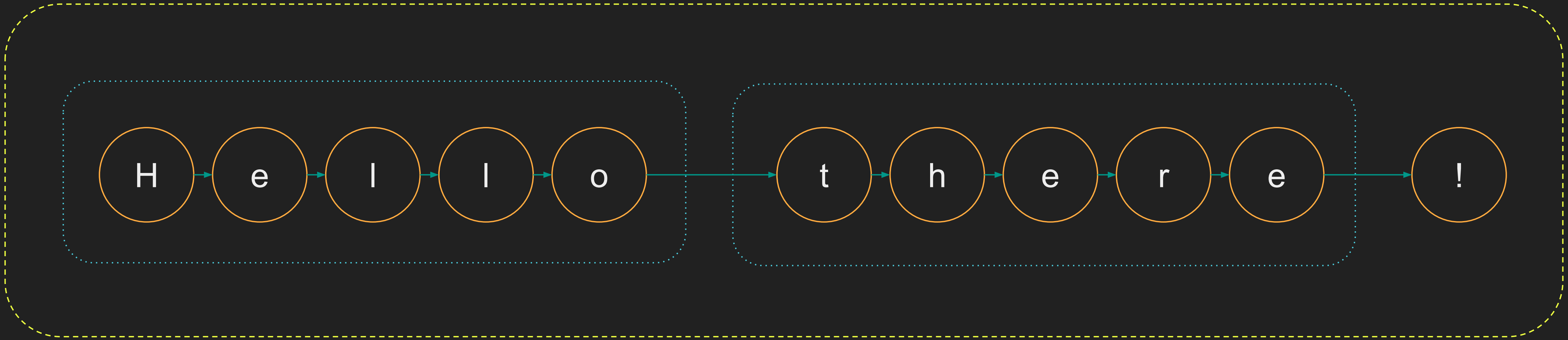
    - Grid
    - Sequential and directed

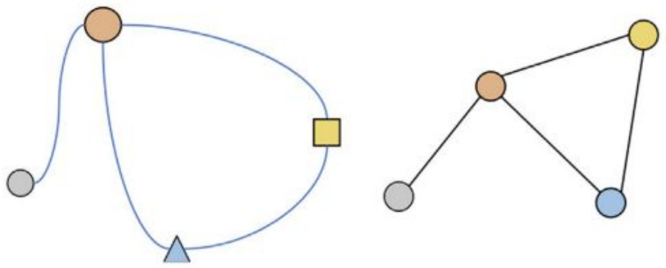
    --> Structure information is built into the models itself

Images are just special kind of graphs

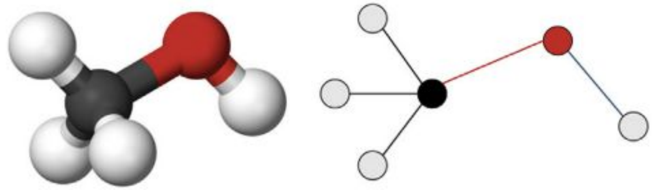


Texts are special kind of graphs

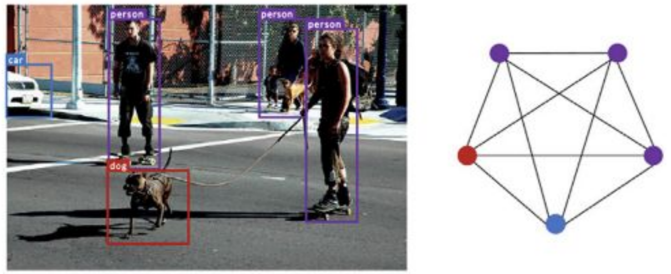




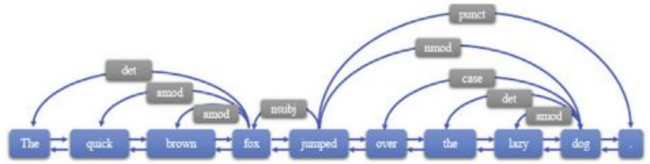
(a) Physics



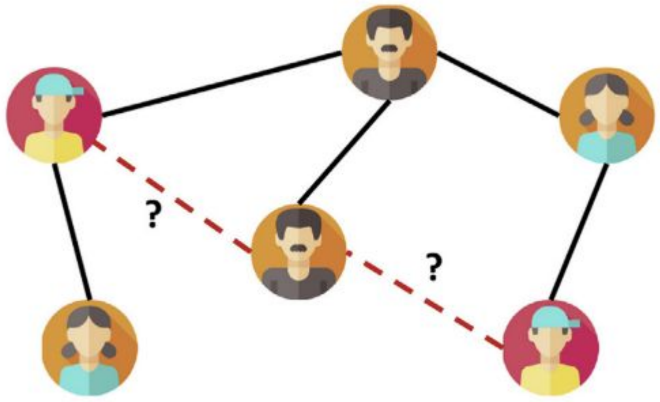
(b) Molecule



(c) Image



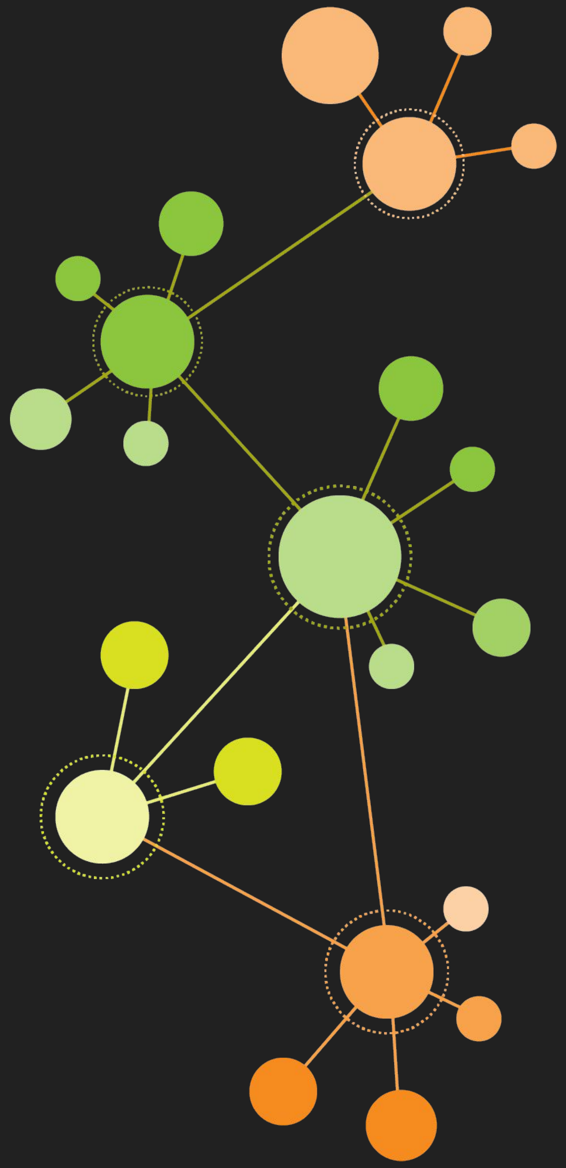
(d) Text



(e) Social Network

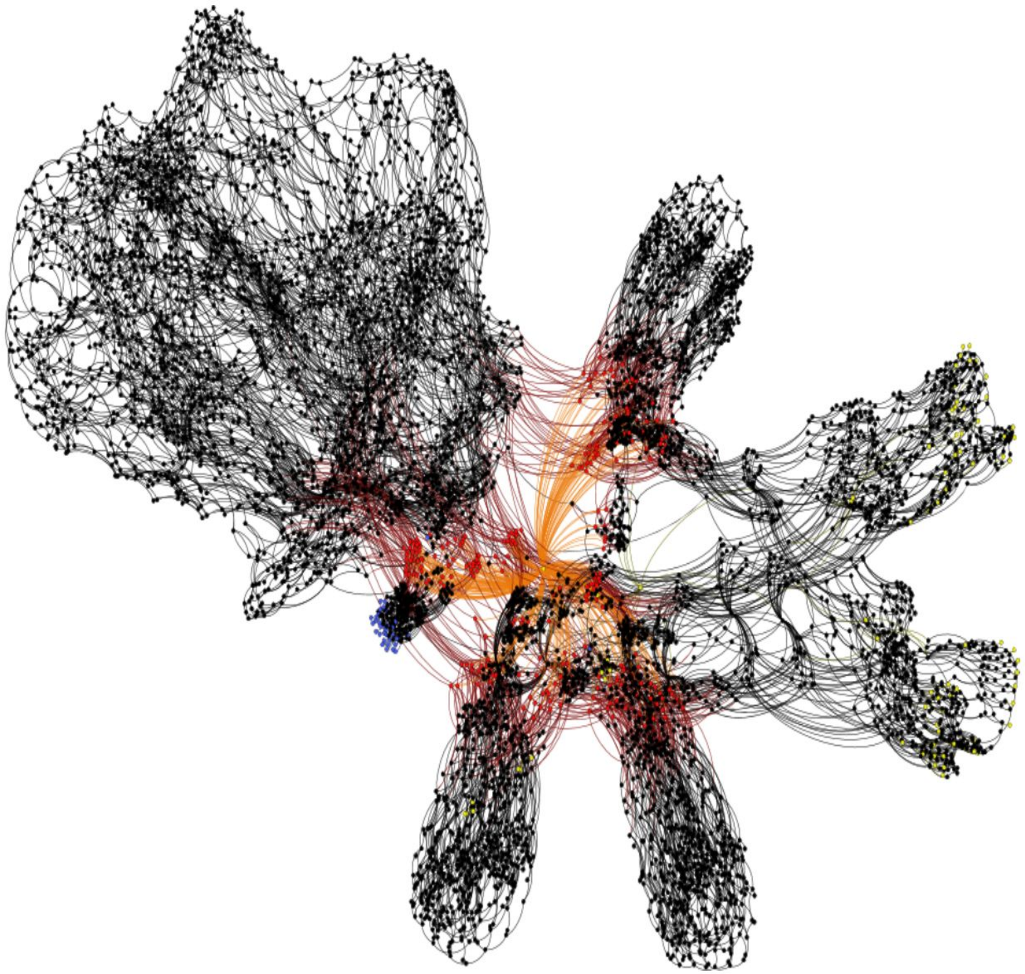
About Graph Neural Networks

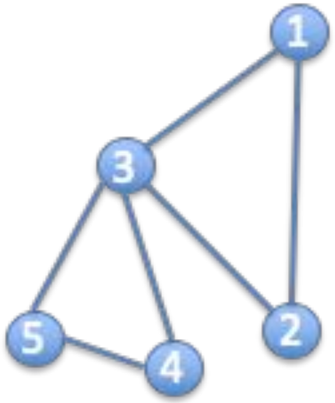
- **NLP / CV takes these structures as granted**
- Encodes **arbitrary** relationships and interactions into the data
- Types:
    - Graph *(Spatial, Spectral)* Convolutional Network
    - Graph Attention Network
    - Gated Graph Neural Network
    - Recurrent Graph Neural Network
    - Generative Graph Neural Networks
    - …

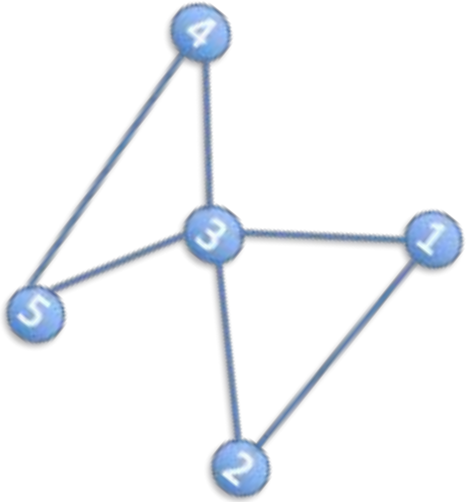


Challenges

Hard to visualize in human-labable form

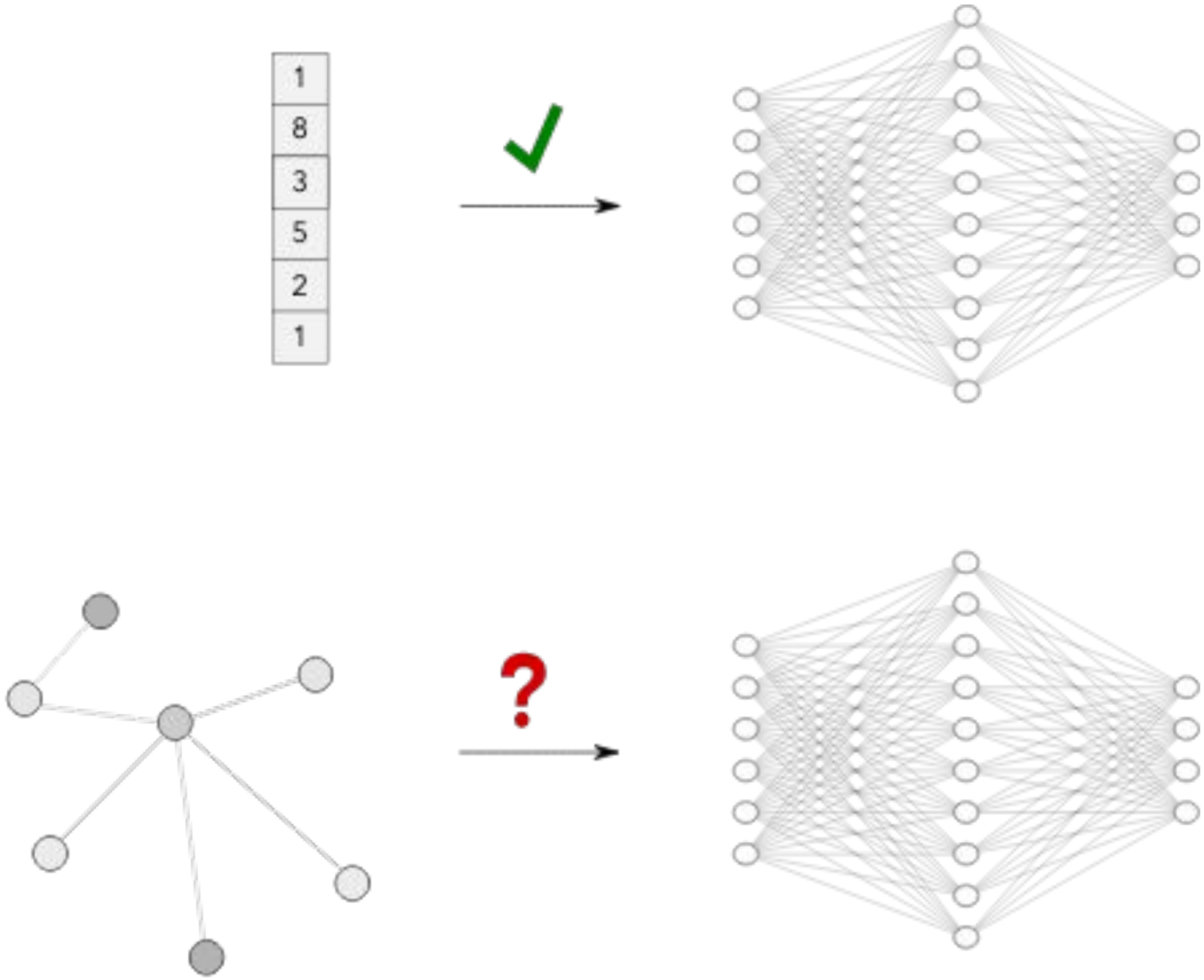




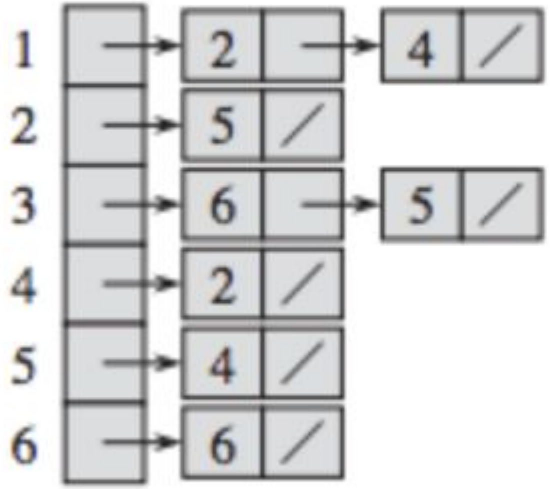


data > IMDB-BINARY

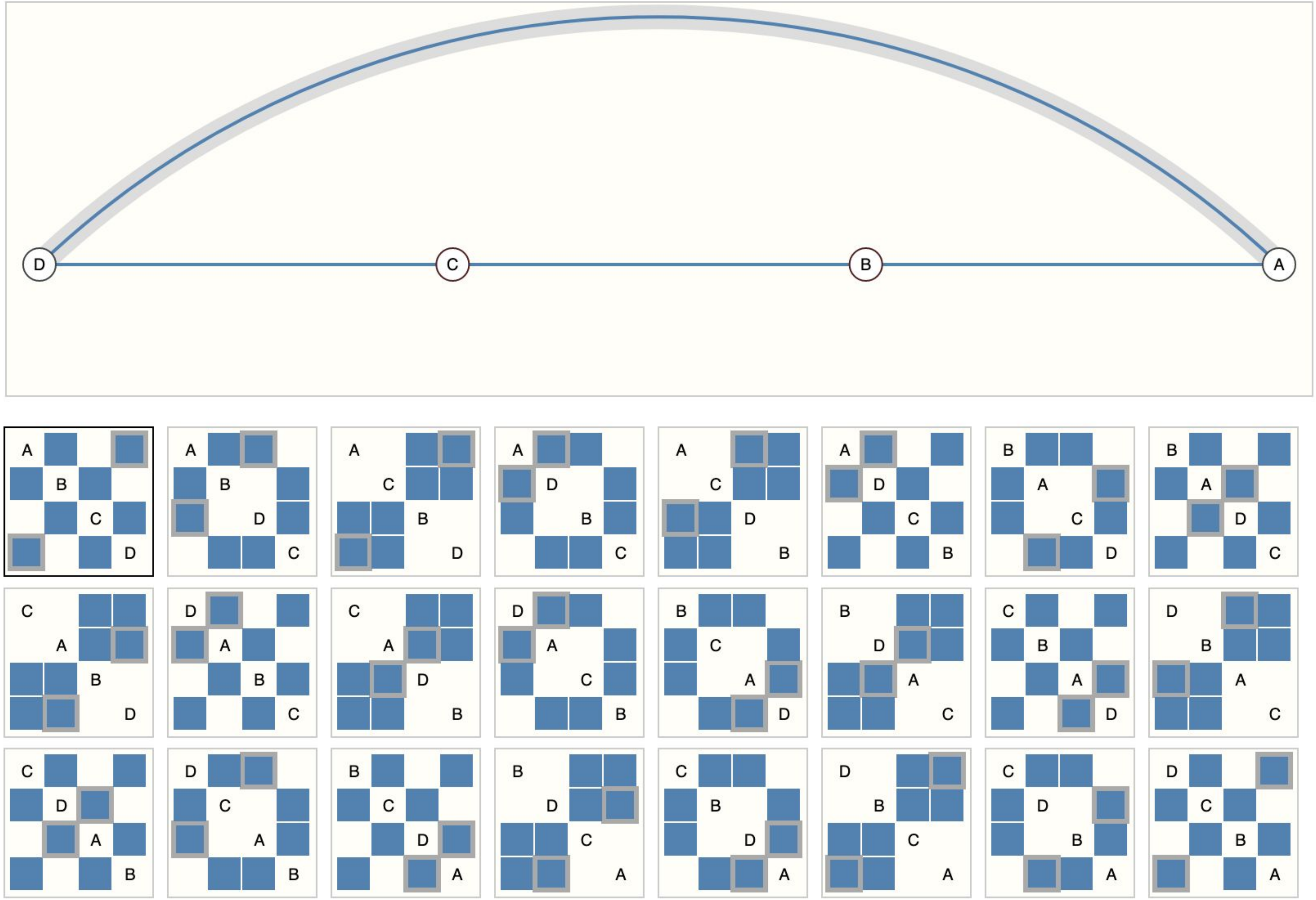
```
20 20 73
1 2
1 3
1 4
1 5
1 6
7 2
7 8
7 9
7 10
7 11
7 12
7 13
7 14
2 15
2 3
2 4
2 8
2 16
2 9
2 5
2 6
2 17
```


1
8
3
5
2
1
?

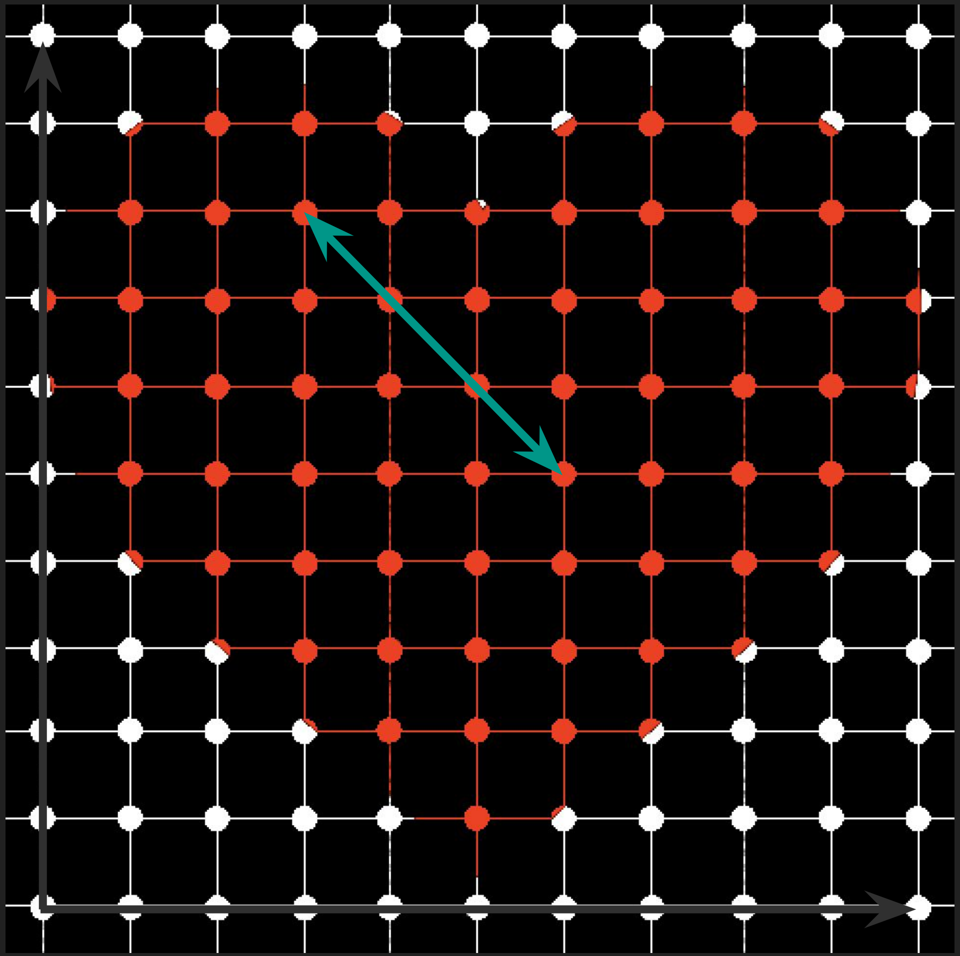
Various possible representations

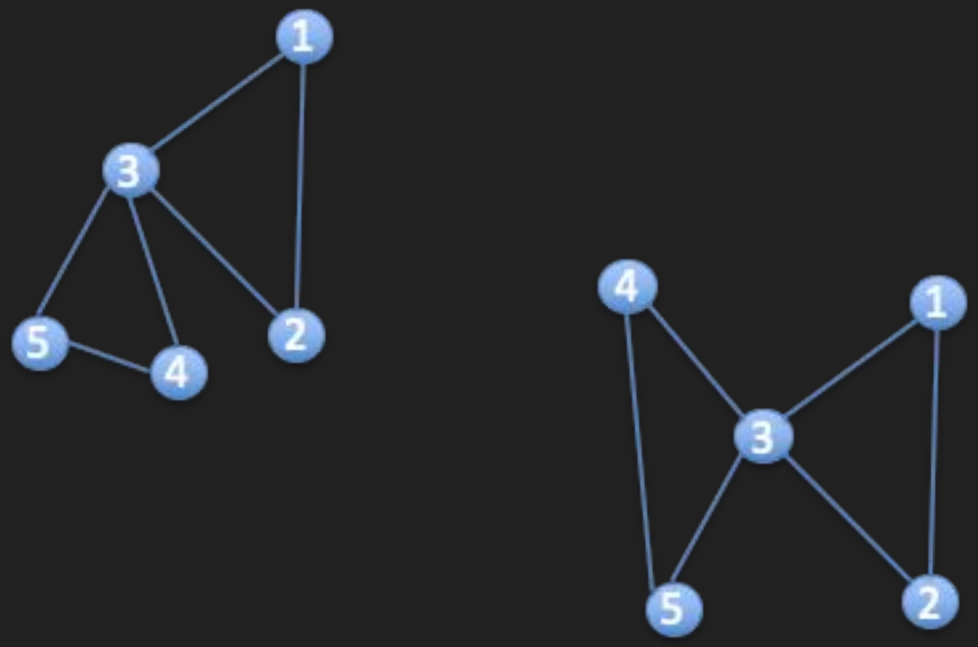


	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1


D C B A
A B C D
A B D C
A C B D
A D B C
A C D B
A D C B
B A C D
B A D C
C A B D
D A B C
C A D B
D A C B
B C A D
B D A C
C B A D
D B A C
C D A B
D C A B
B C D A
B D C A
C B D A
D B C A
C D B A
D C B A

Does not exist in euclidean space and / or with fixed form





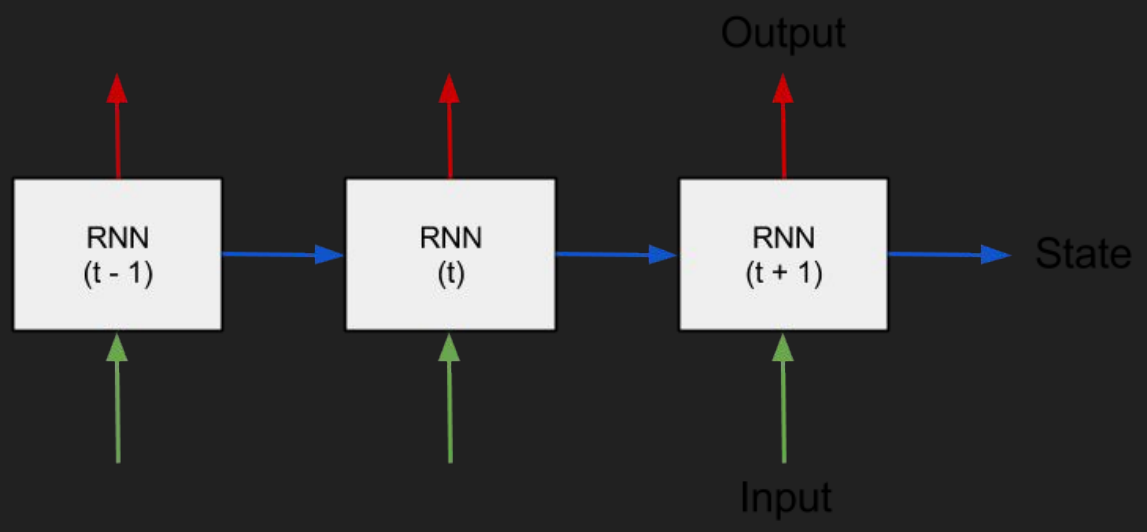
- Basic operations like convolution that are taken for granted in the Euclidean case are not well defined

Why not just standard NNs?

- Variable inputs
- Isomorphic graphs

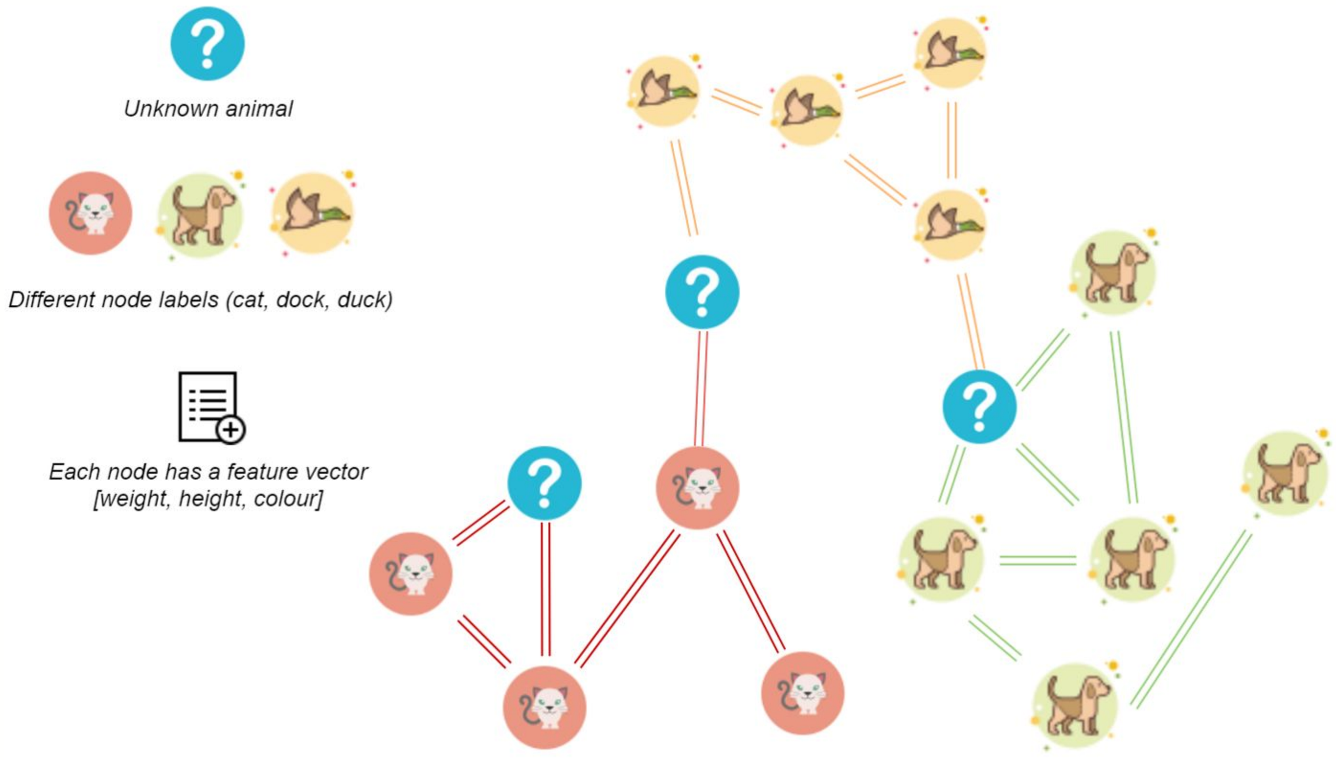
- CNN or RNN stack features in some specific order
    - We should traverse in all possible orderings
    - Computationally expensive
    - Convolution on adj. matrix does not capture neighbors

- Output of the GNN should be invariant for the input order of nodes



Tasks

- **Node classification**
  - **Citation networks**
  - **R / TW / FB / IG posts**
  - **Friends relationships**
- Link prediction
  - Recommendation system
- Graph classification
  - Classify whole graph
- Community detection
- Graph Completion/Generation



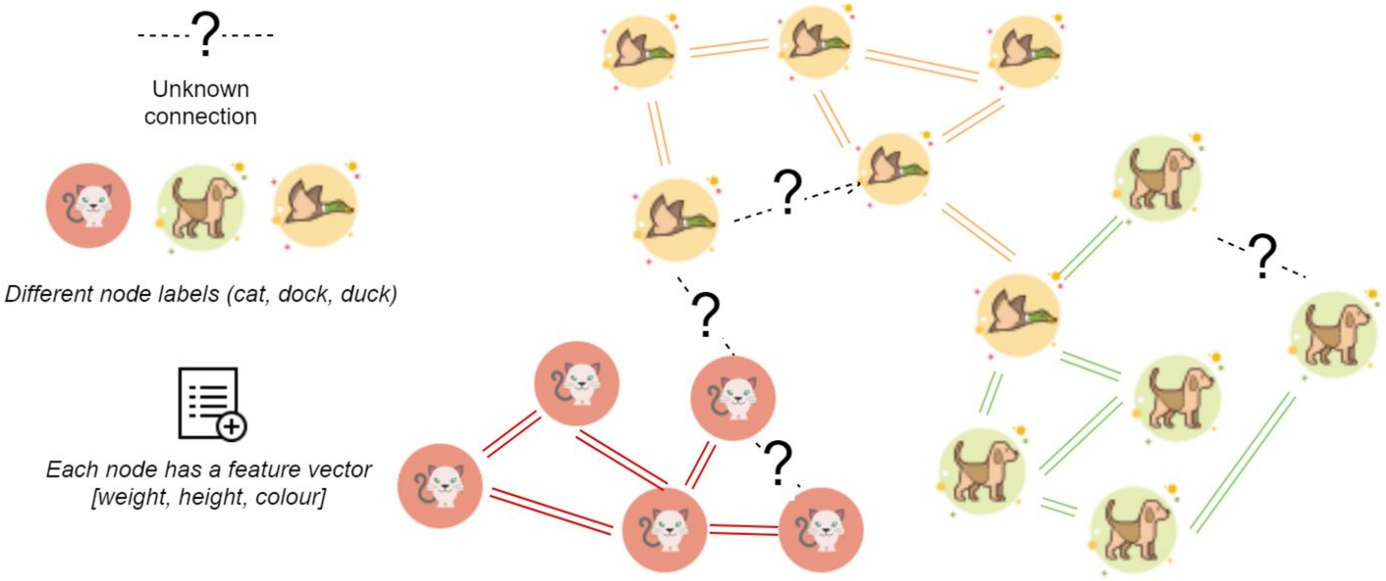
- Node classification
    - Citation networks
    - R / TW / FB / IG posts
    - Friends relationships

- **Link prediction**
    - **Recommendation system**

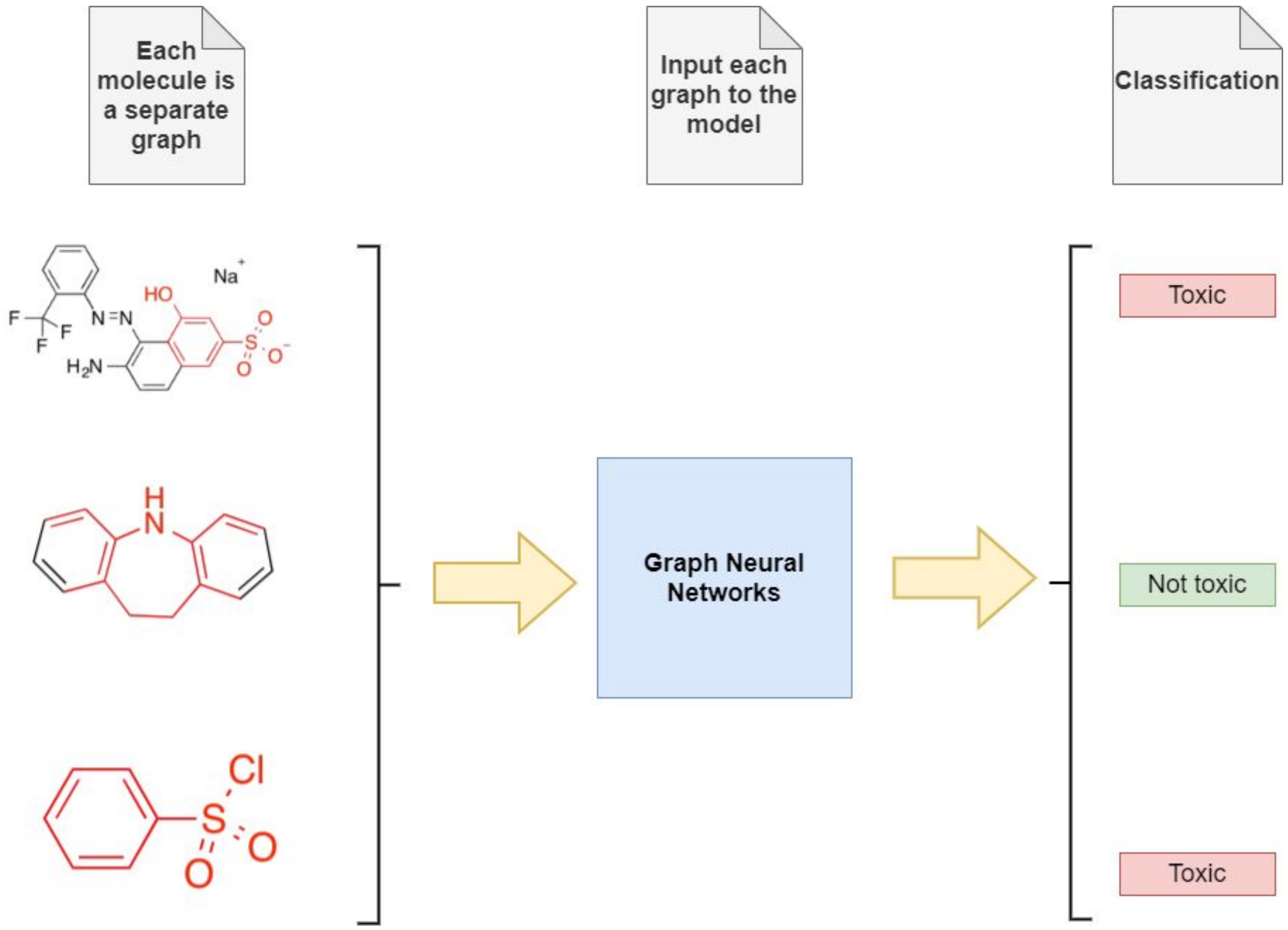
- Graph classification
    - Classify whole graph

- Community detection

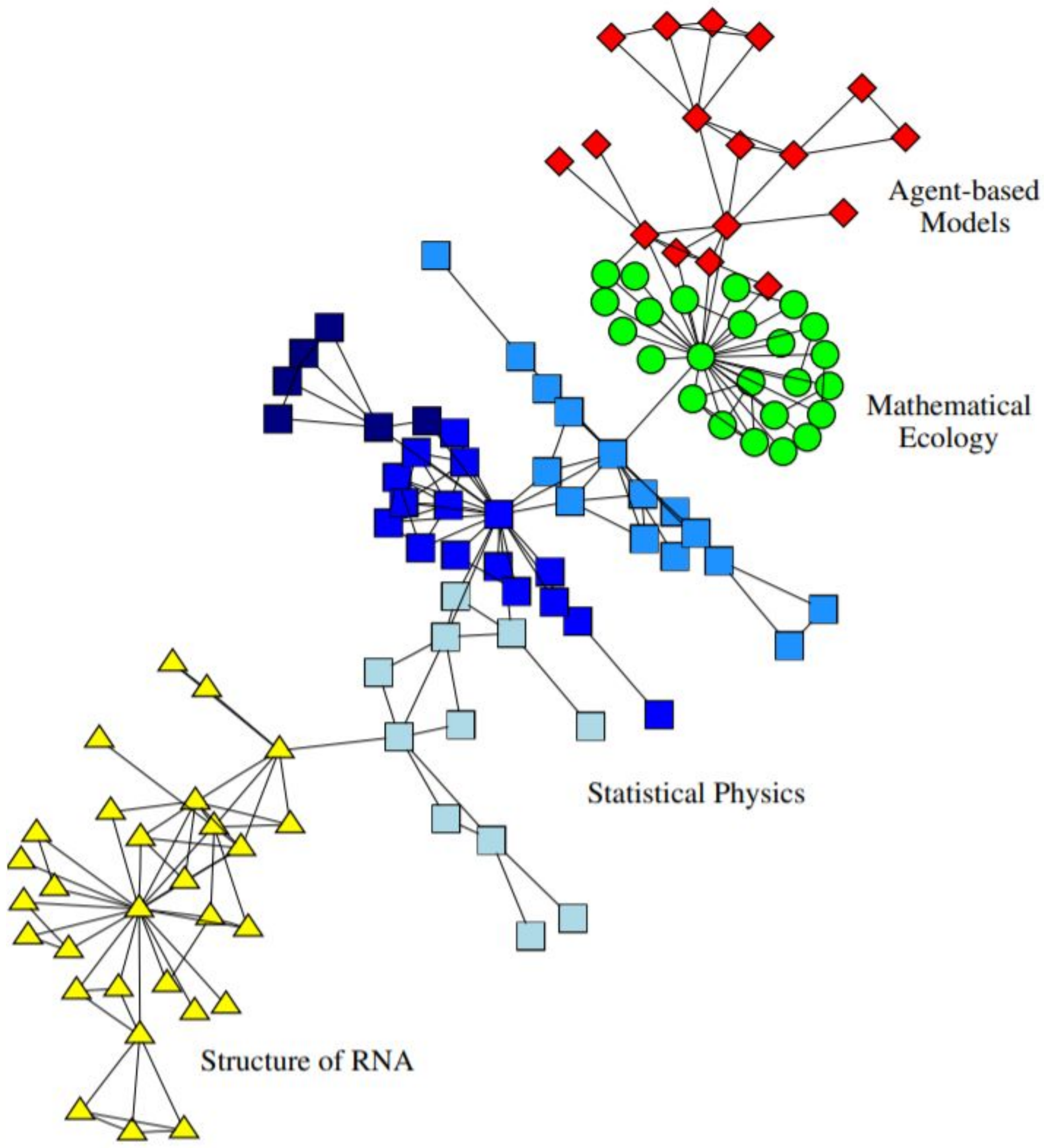
- Graph Completion/Generation



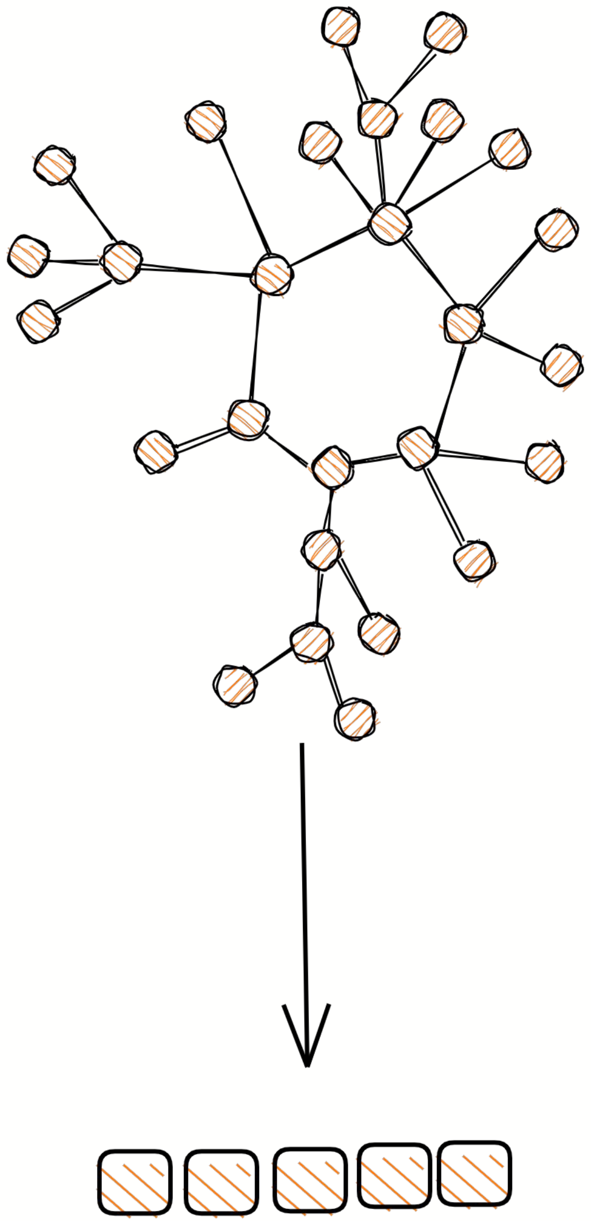
- Node classification
    - Citation networks
    - R / TW / FB / IG posts
    - Friends relationships
- Link prediction
    - Recommendation system
- **Graph classification**
    - **Classify whole graph**
- Community detection
- Graph Completion/Generation



- Node classification
    - Citation networks
    - R / TW / FB / IG posts
    - Friends relationships
- Link prediction
    - Recommendation system
- Graph classification
    - Classify whole graph
- **Community detection**
- Graph Completion/Generation



- Node classification
  - Citation networks
  - R / TW / FB / IG posts
  - Friends relationships
- Link prediction
  - Recommendation system
- Graph classification
  - Classify whole graph
- Community detection
- **Graph Learning**
  - Completion, Generation, Mining, Matching, ...



Usage scenarios

- **Structural**
  - Explicit relational structure
  - Knowledge graphs, phys. systems, …
  - Social network prediction, recommender systems, traffic prediction, …

- **Non-structural**
  - Not explicit, yet existing structure
  - Text, images, …
  - Two approaches
    - 1. Incorporate structural information from other domains to improve the performance
    - 2. Infer or assume the relational structure in the scenario

Table 3
Applications of graph neural networks.

Area	Application	References
Graph Mining	Graph Matching	(Riba et al., 2018; Li et al., 2019b)
	Graph Clustering	(Zhang et al., 2019c; Ying et al., 2018b; Tsitsulin et al., 2020)
Physics	Physical Systems Modeling	(Battaglia et al., 2016; Sukhbaatar Ferguset al., 2016; Watters et al., 2017; Hoshen, 2017; Kipf et al., 2018; Sanchez et al., 2018)
Chemistry	Molecular Fingerprints	(Duvenaud et al., 2015; Kearnes et al., 2016)
	Chemical Reaction Prediction	Do et al. (2019)
Biology	Protein Interface Prediction	Fout et al. (2017)
	Side Effects Prediction	Zitnik et al. (2018)
	Disease Classification	Rhee et al. (2018)
Knowledge Graph	KB Completion	(Hamaguchi et al., 2017; Schlichtkrull et al., 2018; Shang et al., 2019)
	KG Alignment	(Wang et al., 2018b; Zhang et al., 2019d; Xu et al., 2019c)
Generation	Graph Generation	(Shchur et al., 2018b; Nowak et al., 2018; Ma et al., 2018; You et al., 2018a, 2018b; De Cao and Kipf, 2018; Li et al., 2018d; Shi et al., 2020; Liu et al., 2019; Grover et al., 2019)
Combinatorial Optimization	Combinatorial Optimization	(Khalil et al., 2017; Nowak et al., 2018; Li et al., 2018e; Kool et al., 2019; Bello et al., 2017; Vinyals et al., 2015b; Sutton and Barto, 2018; Dai et al., 2016; Gasse et al., 2019; Zheng et al., 2020a; Selsam et al., 2019; Sato et al., 2019)
Traffic Network	Traffic State Prediction	(Cui et al., 2018b; Yu et al., 2018; Zheng et al., 2020b; Guo et al., 2019)
Recommendation Systems	User-item Interaction Prediction	(van den Berg et al., 2017; Ying et al., 2018a)
	Social Recommendation	(Wu et al., 2019c; Fan et al., 2019)
Others (Structural)	Stock Market	(Matsunaga et al., 2019; Yang et al., 2019; Chen et al., 2018c; Li et al., 2020; Kim et al., 2019)
	Software Defined Networks	Rusek et al. (2019)
	AMR Graph to Text	(Song et al., 2018a; Beck et al., 2018)
Text	Text Classification	(Peng et al., 2018; Yao et al., 2019; Zhang et al., 2018d; Tai et al., 2015)
	Sequence Labeling	(Zhang et al., 2018d; Marcheggiani and Titov, 2017)
	Neural Machine Translation	(Bastings et al., 2017; Marcheggiani et al., 2018; Beck et al., 2018)
	Relation Extraction	(Miwa and Bansal, 2016; Peng et al., 2017; Song et al., 2018b; Zhang et al., 2018f)
	Event Extraction	(Nguyen and Grishman, 2018; Liu et al., 2018)
	Fact Verification	(Zhou et al., 2019; Liu et al., 2020; Zhong et al., 2020)
	Question Answering	(Song et al., 2018c; De Cao et al., 2019; Qiu et al., 2019; Tu et al., 2019; Ding et al., 2019)
	Relational Reasoning	(Santoro et al., 2017; Palm et al., 2018; Battaglia et al., 2016)
Image	Social Relationship Understanding	Wang et al. (2018c)
	Image Classification	(Garcia and Bruna, 2018; Wang et al., 2018d; Lee et al., 2018b; Kampffmeyer et al., 2019; Marino et al., 2017)
	Visual Question Answering	(Teney et al., 2017; Wang et al., 2018c; Narasimhan et al., 2018)
	Object Detection	(Hu et al., 2018; Gu et al., 2018)
	Interaction Detection	(Qi et al., 2018; Jain et al., 2016)
	Region Classification	Chen et al. (2018d)
	Semantic Segmentation	(Liang et al., 2016, 2017; Landrieu and Simonovsky, 2018; Wang et al., 2018e; Qi et al., 2017b)
Other (Non-structural)	Program Verification	(Allamanis et al., 2018; Li et al., 2016)

Approaches

Original GNN - Intuition

- Nodes are naturally defined by their neighbors and connections

- Give every node a state (x) to represent its concept

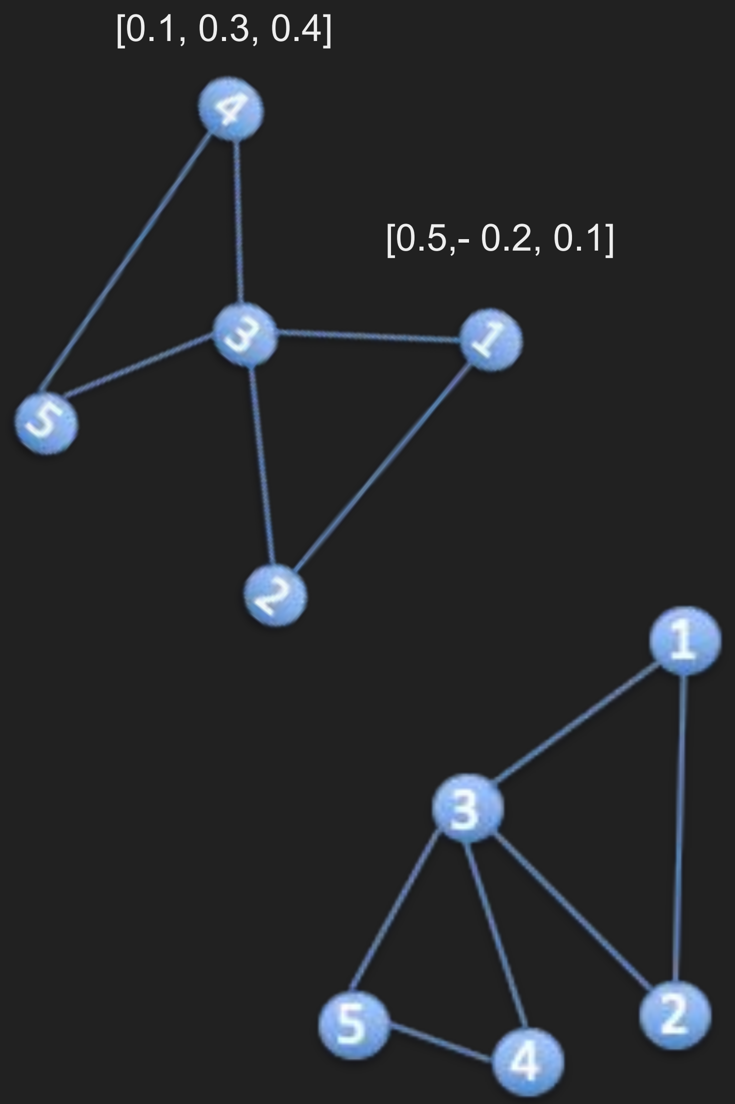
- Use the node state (x) to produce an output (o)

- Final state (x_n) of the node is normally called "node embedding"

- Task is to determine the "node embedding" of each node, by looking at the information on its neighboring nodes

Original GNN - Intuition

- Nodes are naturally defined by their neighbors and connections
- Give every node a state (x) to represent its concept
- Use the node state (x) to produce an output (o)
- Final state (x_n) of the node is normally called “node embedding”
- Task is to determine the “node embedding” of each node, by looking at the information on its neighboring nodes



Original GNN - Intuition

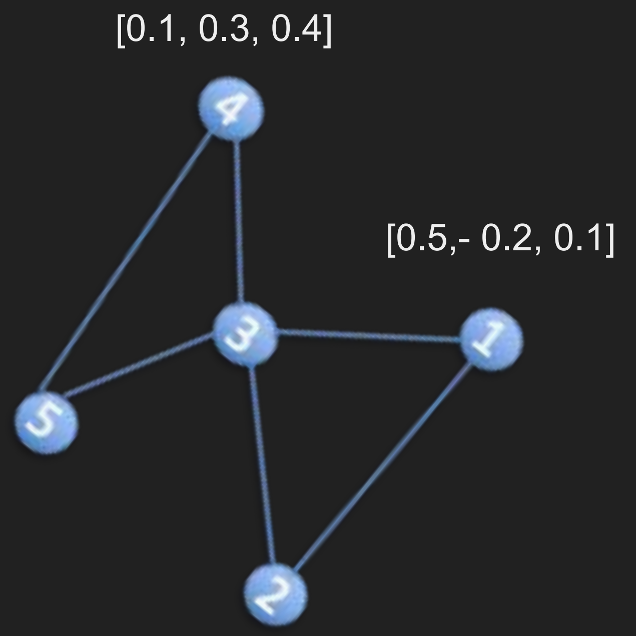
- Nodes are naturally defined by their neighbors and connections

- Give every node a state (x) to represent its concept

- Use the node state (x) to produce an output (o)

- Final state (x_n) of the node is normally called “node embedding”

- Task is to determine the “node embedding” of each node, by looking at the information on its neighboring nodes



Original GNN - Intuition

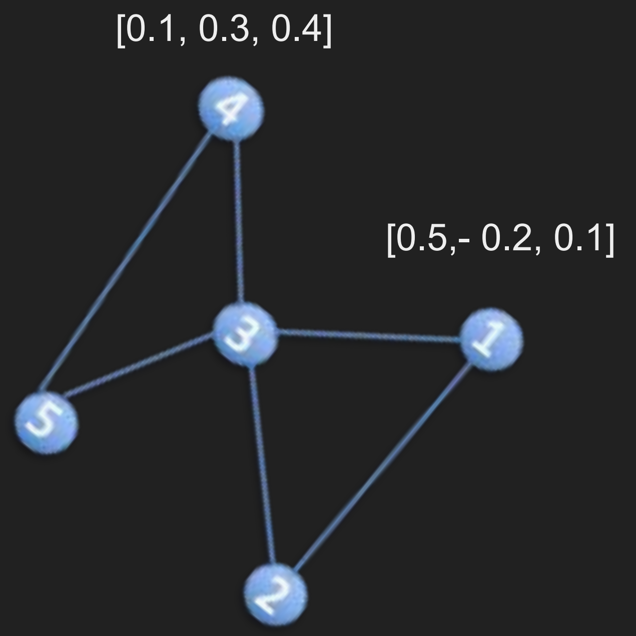
- Nodes are naturally defined by their neighbors and connections

- Give every node a state (x) to represent its concept

- Use the node state (x) to produce an output (o)

- Final state (x_n) of the node is normally called “node embedding”

- Task is to determine the “node embedding” of each node, by looking at the information on its neighboring nodes



Original GNN - Training

- TODO

Original GNN - Limitations

- Features on the edges cannot be effectively modeled in the original GNN
    - Although we can replace edge by a one new node + 2 new edges

- The input graph consist of nodes with label information and undirected edges, which is the simplest case

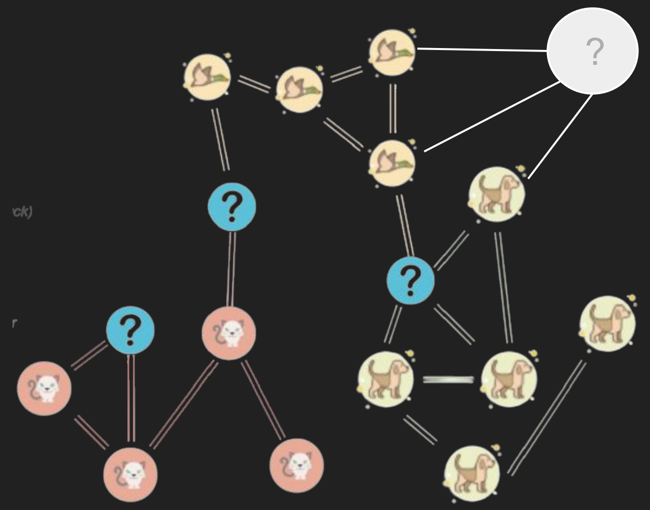
- **Transductive learning**
    - We can not use it on “during the training unseen stuff”

Original GNN - Limitations

- Features on the edges cannot be effectively modeled in the original GNN
  - Although we can replace edge by a one new node + 2 new edges

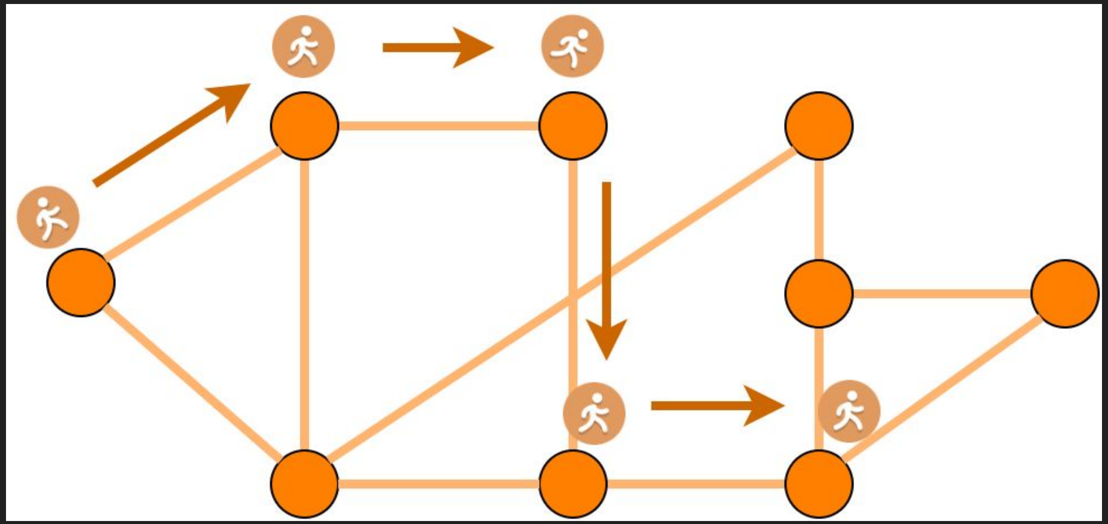
- The input graph consist of nodes with label information and undirected edges, which is the simplest case

- **Transductive learning**
  - We can not use it on “during the training unseen stuff”



NLP Inspiration

- **DeepWalk**
  - First (?) graph embedding method
  - Applies SkipGram on the generated [random walks]



- No parameters shared between nodes in the encoders
  - Number of params grows linearly with number of nodes

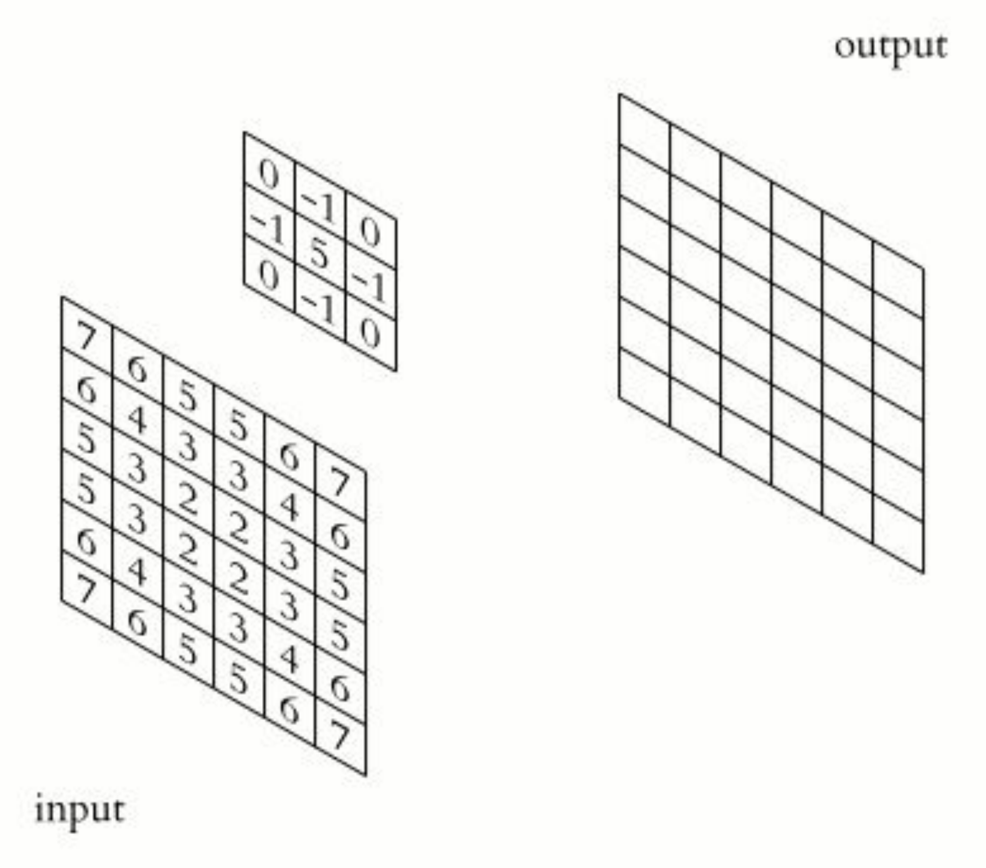
- Direct embeddings
  - Lack ability of generalization
  - (can not be applied to new graphs)

Layers

Convolutional GNNs

- The idea of convolution on an image is to sum the neighboring pixels around a center pixel, specified by a filter with parameterized size and learnable weight

- GNNs adopts the same idea by aggregate the features of neighboring nodes into the center node

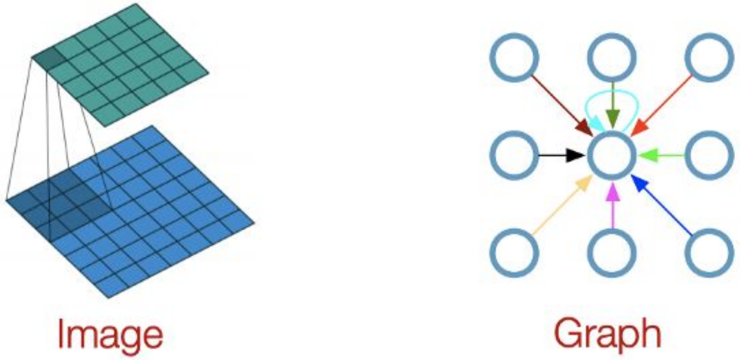


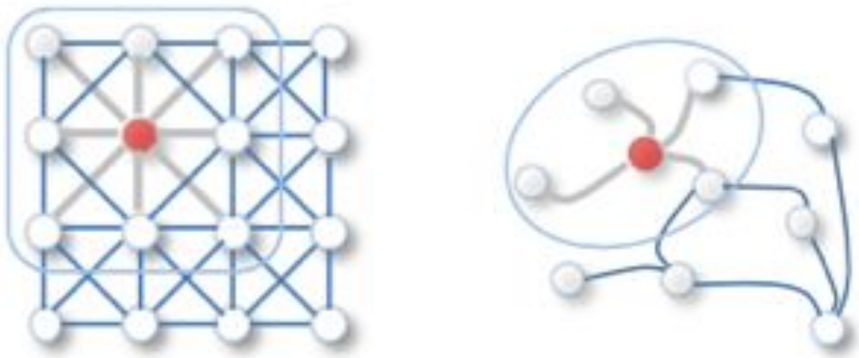
Convolutional GNNs

- The idea of convolution on an image is to sum the neighboring pixels around a center pixel, specified by a filter with parameterized size and learnable weight

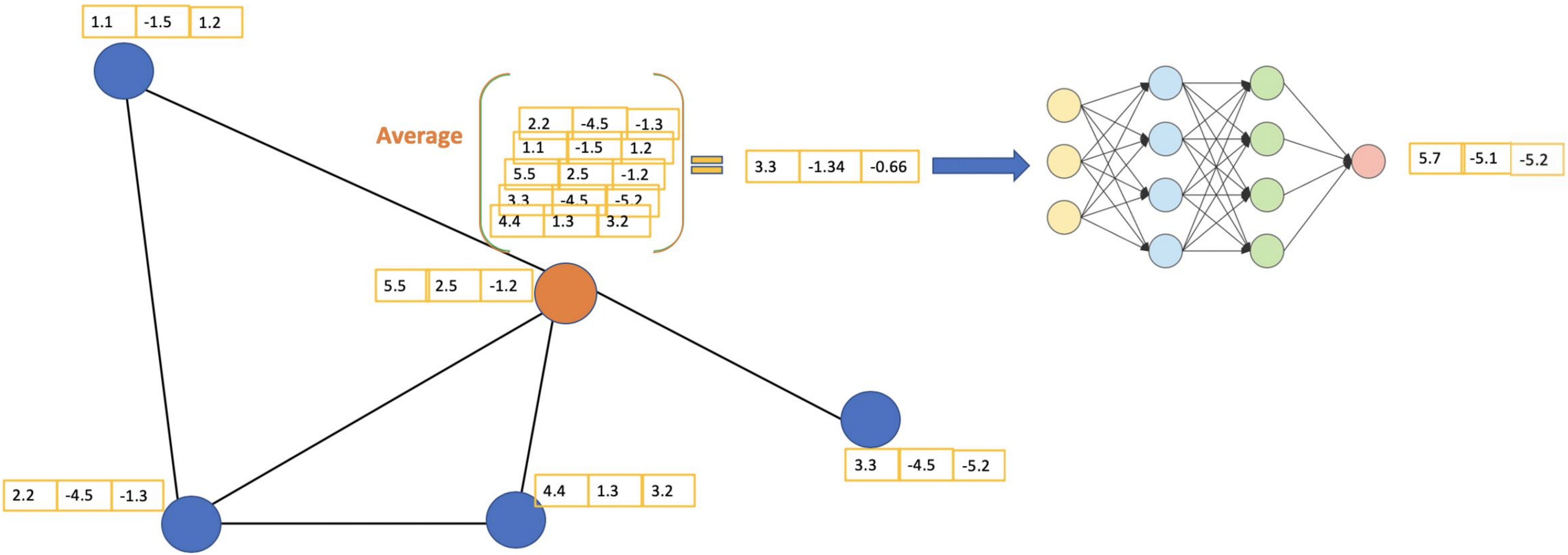
- GNNs adopts the same idea by aggregate the features of neighboring nodes into the center node

Single CNN layer with 3x3 filter:





CGNN General Idea



- **Still transductive learning**

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding. ... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)}\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k. Mean of v's neighbour's embeddings at step $k-1$. Node v's embedding at step $k-1$.

Color Codes:

- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

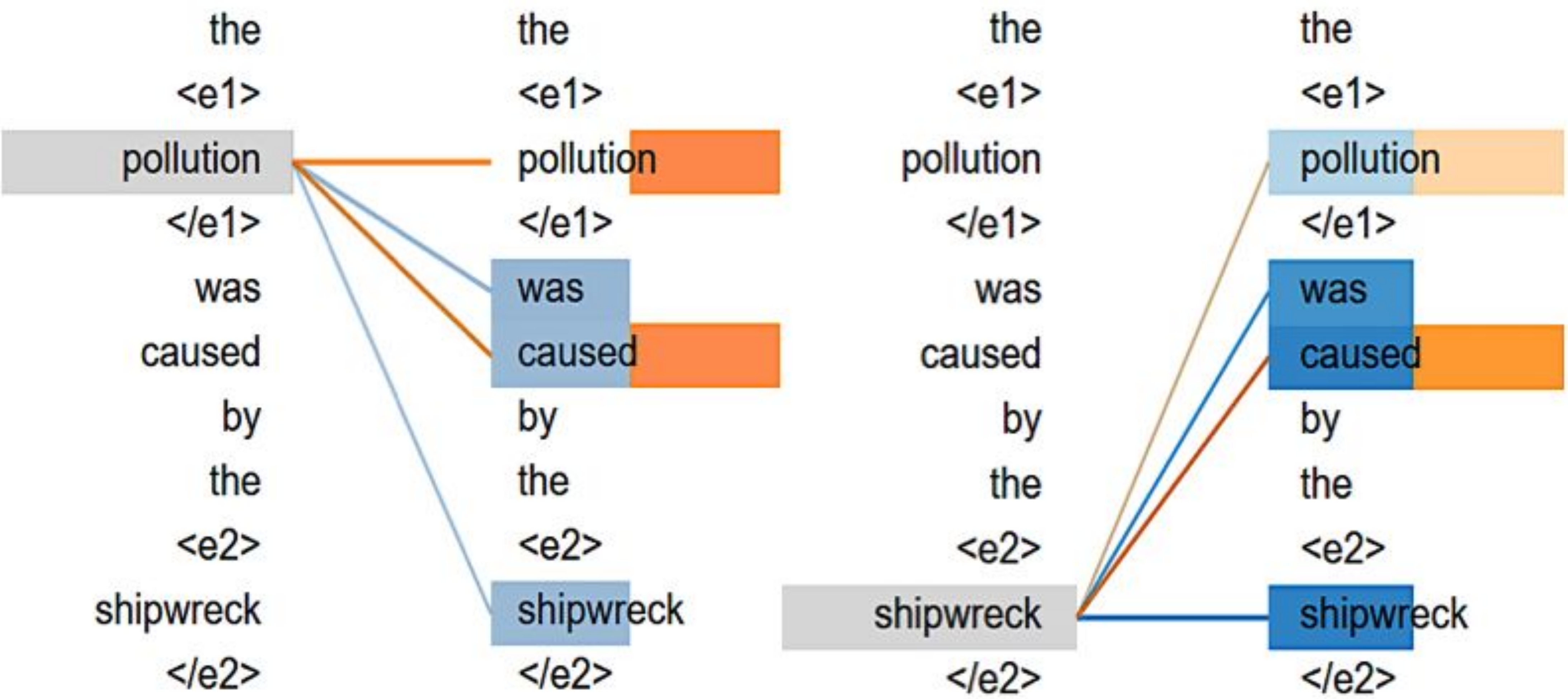
Predictions can be made at each node by using the final computed embedding:

$$\hat{y}_v = \mathrm{PREDICT}(h_v^{(K)})$$

where $\mathrm{PREDICT}$ is generally another neural network, learnt together with the GCN model.

For each step k, the function $f^{(k)}$, matrices $W^{(k)}$ and $B^{(k)}$ are shared across all nodes.

Attention



the
<e1>
pollution
</e1>
was
caused
by
the
<e2>
shipwreck
</e2>
the
<e1>
pollution
</e1>
was
caused
by
the
<e2>
shipwreck
</e2>
the
<e1>
pollution
</e1>
was
caused
by
the
<e2>
shipwreck
</e2>
the
<e1>
pollution
</e1>
was
caused
by
the
<e2>
shipwreck
</e2>

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding. ... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k-1)} h_u^{(k-1)} + \alpha_{vv}^{(k-1)} h_v^{(k-1)}\right]\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k. Weighted mean of v's neighbour's embeddings at step $k-1$. Node v's embedding at step $k-1$.

where the attention weights $\alpha^{(k)}$ are generated by an attention mechanism $A^{(k)}$, normalized such that the sum over all neighbo

$$\alpha_{vu}^{(k)} = \frac{A^{(k)}(h_v^{(k)}, h_u^{(k)})}{\sum_{w \in \mathcal{N}(v)} A^{(k)}(h_v^{(k)}, h_w^{(k)})} \quad \text{for all } (v, u) \in E.$$

Color Codes:

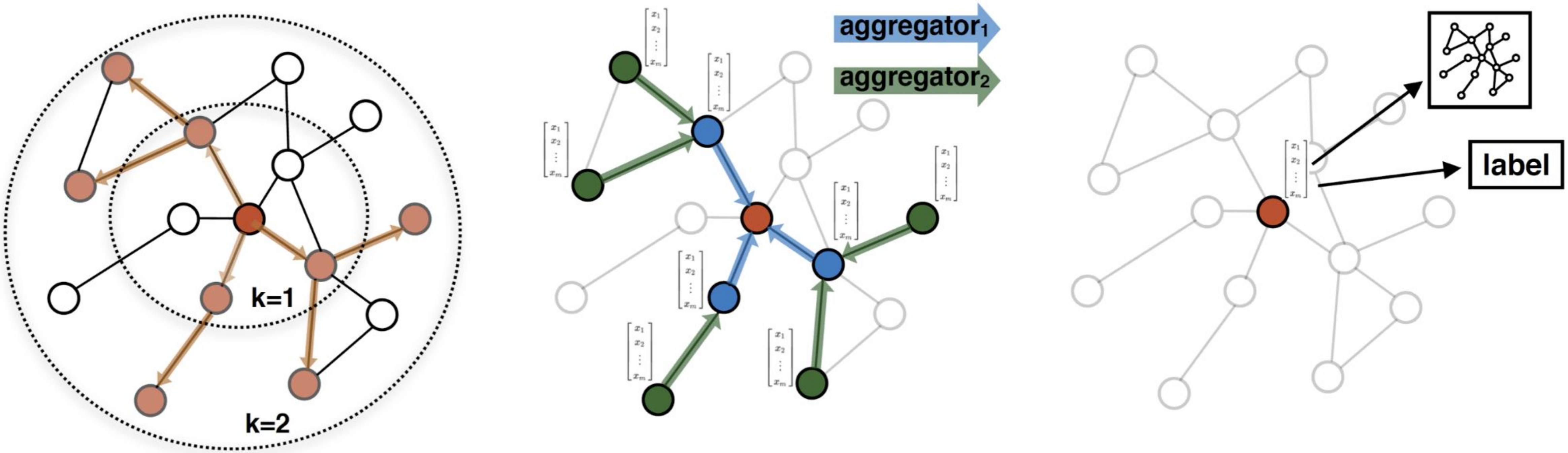
- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

GraphSAGE (Graph Sample And Aggregate)

- Introduces **SAGEConv**

- **Learns aggregation function params not node features**
    - → inductive learning
    - → **we can predict things unseen during the training**

GraphSAGE viz.



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

GraphSAGE (Graph Sample And Aggregate)

- Different aggregation options
  - Mean
  - RNNs (not permutation-invariant)
  - Avg/Max Pooling
  - **FF NN**
  - **...**

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding. ... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\underset{u \in \mathcal{N}(v)}{\mathrm{AGG}}(\{h_u^{(k-1)}\}),\ h_v^{(k-1)}\right]\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Aggregation of v's neighbour's embeddings at step $k-1$ ...

... concatenated with ...

... Node v's embedding at step $k-1$.

Color Codes:

- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

Predictions can be made at each node by using the final computed embedding:

$$\hat{y}_v = \mathrm{PREDICT}(h_v^{(K)})$$

where $\mathrm{PREDICT}$ is generally another neural network, learnt together with the GraphSAGE model.

For each step k, the function $f^{(k)}$, AGG and matrix $W^{(k)}$ are shared across all nodes.

Benchmark

So we have GNN. But is it useful?

Paper claims that

- GNNs are standard toolkit to learn from graphs

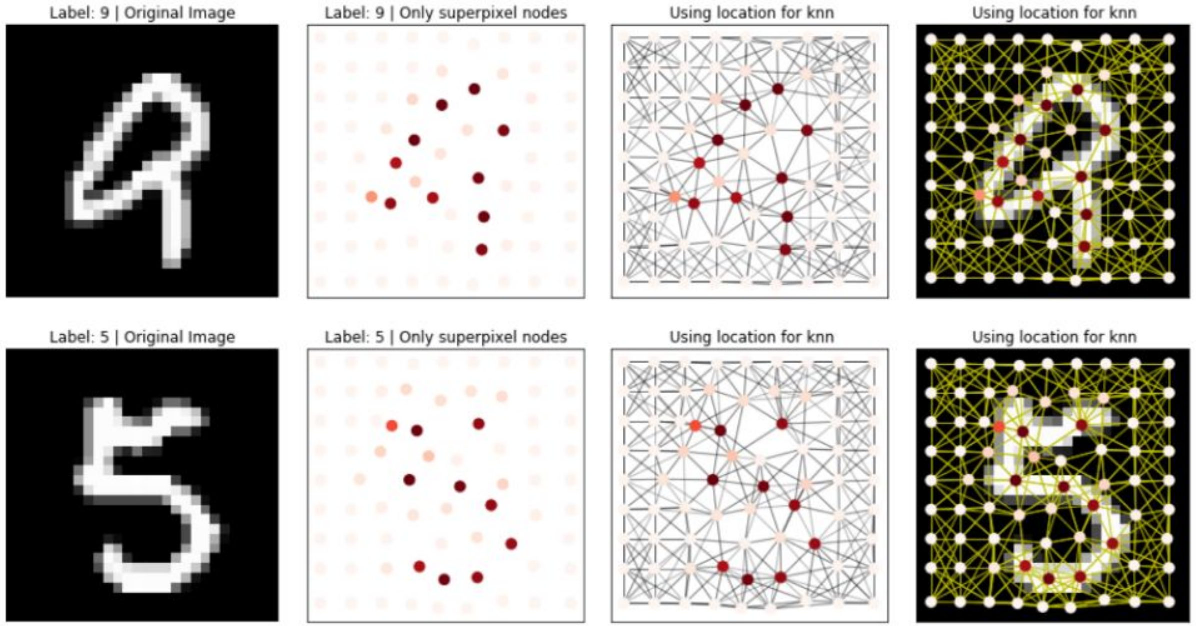
- Single benchmark suite was missing, but is very much needed
    - E.g. ImageNet in CV

- New ideas evaluated mostly on small datasets
    - *I can confirm from my experience*

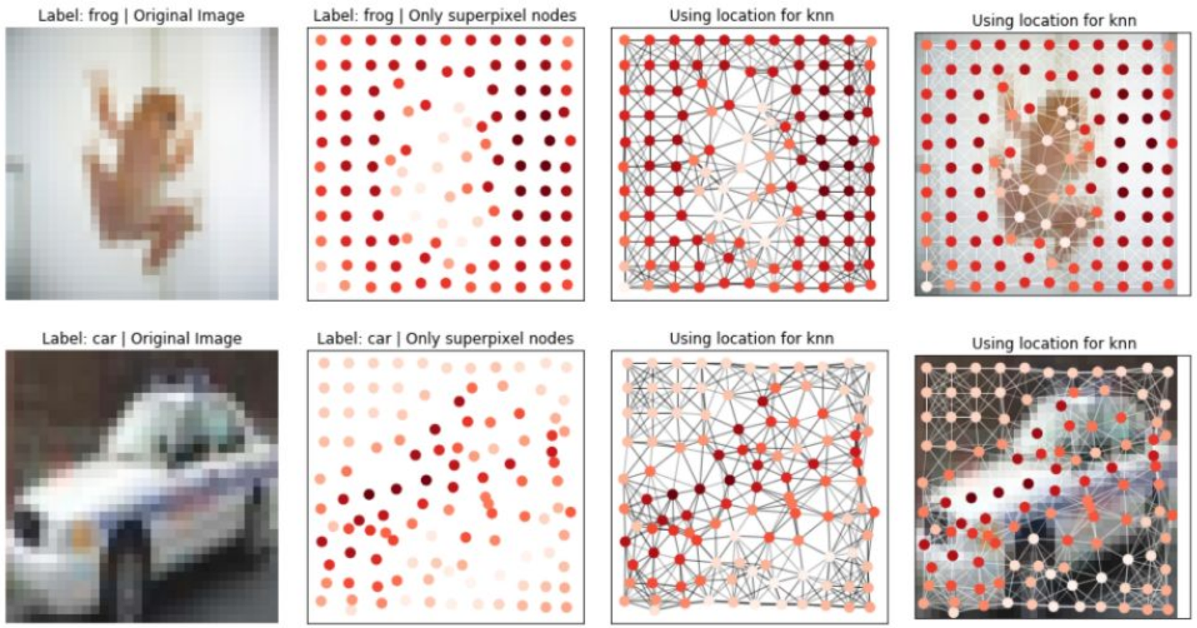
- Introduces GNN benchmarking framework
    - Almost 2k stars on Github

Table 1: Summary statistics of datasets included in the proposed benchmark.

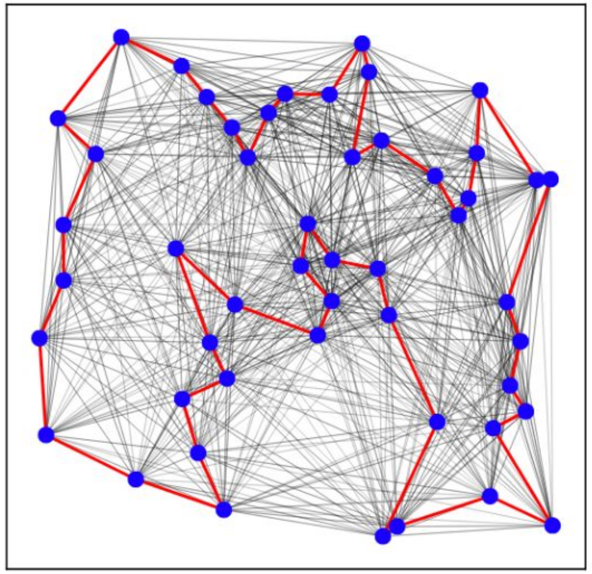
Domain & Construction	Dataset	#Graphs	#Nodes	Total #Nodes	Task
Chemistry: Real-world molecular graphs	ZINC	12K	9-37	277,864	Graph Regression
Mathematical Modelling: Artificial graphs generated from Stochastic Block Models	PATTERN CLUSTER	14K 12K	44-188 41-190	1,664,491 1,406,436	Node Classification
Computer Vision: Graphs constructed with SLIC super-pixels of images	MNIST CIFAR10	70K 60K	40-75 85-150	4,939,668 7,058,005	Graph Classification
Combinatorial Optimization: Uniformly generated artificial Euclidean graphs	TSP	12K	50-500	3,309,140	Edge Classification
Social Networks: Real-world citation graph	COLLAB	1	235,868	235,868	Edge Classification
Circular Skip Links: Isomorphic graphs with same degree	CSL	150	41	6,150	Graph Classification



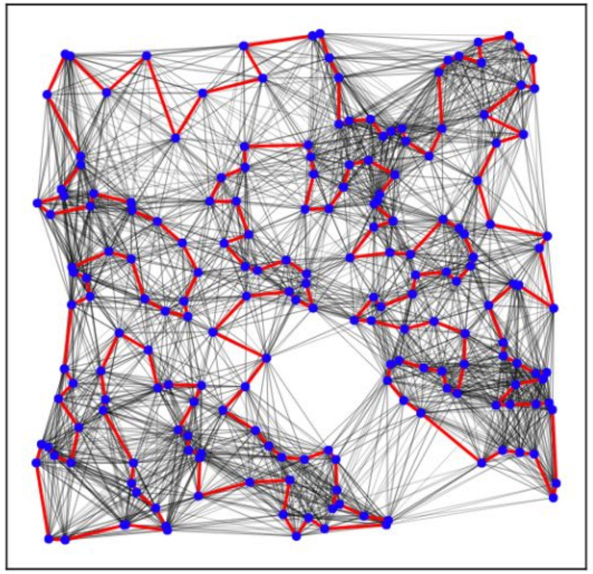
(a) MNIST



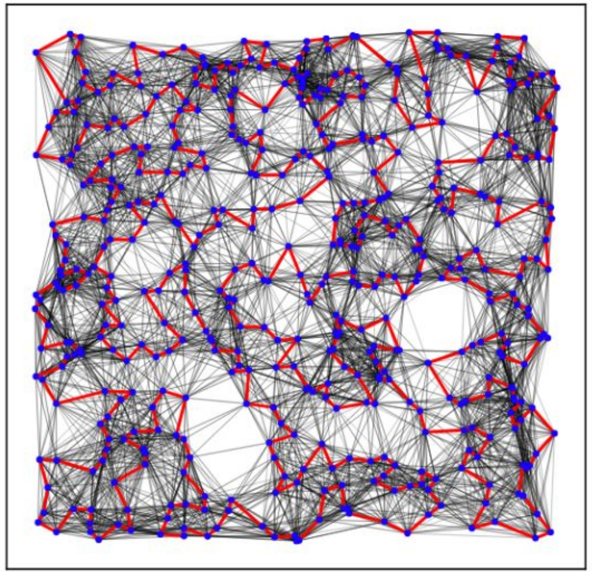
(b) CIFAR10



(a) TSP50



(b) TSP200



(c) TSP500

Graph-agnostic NNs

- MLP baseline
- Updates each node independently
- Nodes aggregated using task-specific layer

Graph-agnostic NNs

- MLP baseline
- Updates each node independently
- Nodes aggregated using task-specific layer
- **→ consistently low scores across all datasets**
- **→ shows necessity of GNNs**

Findings

- WL-GNN space and time complexity O(n * n) and O(n * n * n) respectively

- Attention based neighborhood aggregation performs generally best

- Problems with models that process adjacency matrices
  - Can not batch graphs with different sizes
  - Out of memory errors are quickly reached

- Edge representation improves link prediction
  - +10% F1 score on TSP problem
    - initializing the edge representations with euclidean distances between nodes

NLP

Every Document Owns Its Structure

Inductive Text Classification via GNN

Abstract

- Proposes TextING
    - Text classification task
    - Builds individual graph for each document
    - Use GNN to learn word representations
      Can produce embeddings for unseen words in new documents
- Claims to outperform SOTA TC methods (2020)
  and **requires less data**

TextING

- Train only on training documents

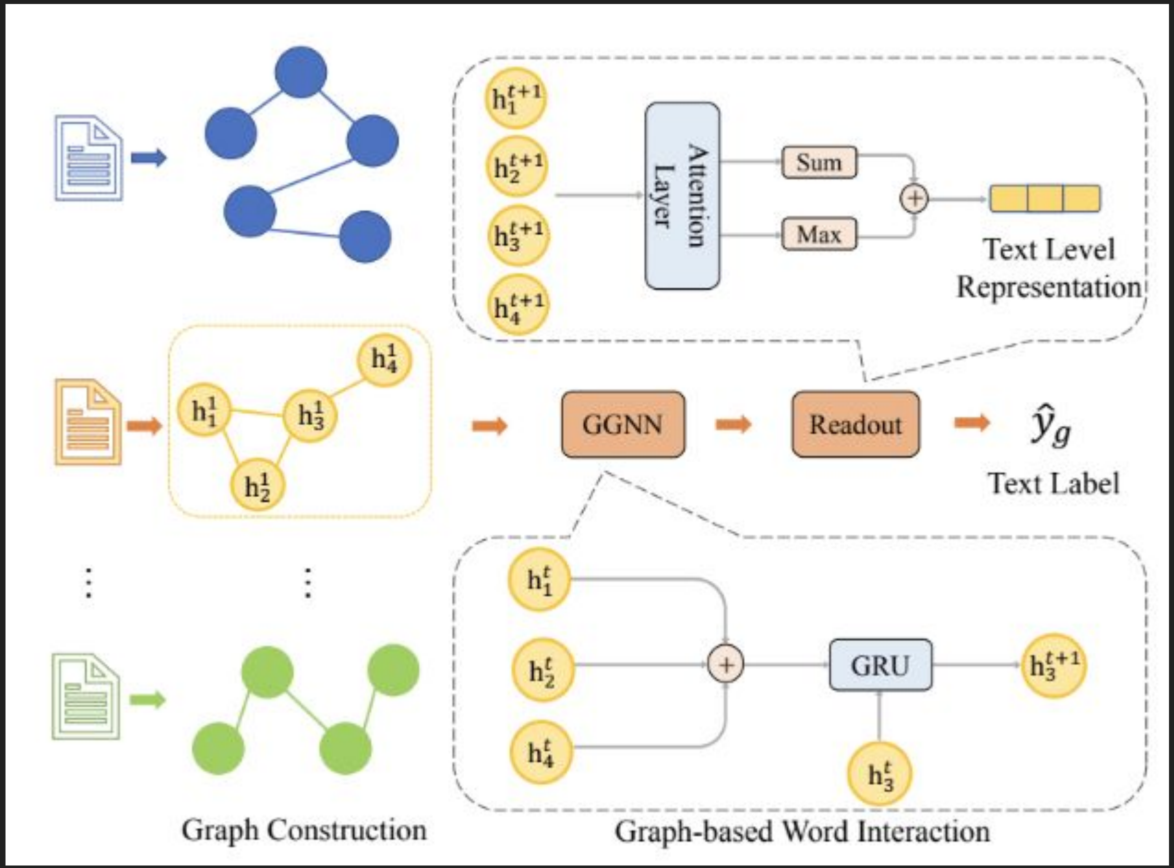
- Generalise to new documents

- Information of word nodes in propagated to their neighbours via **Gated Graph Neural Networks**

- **And nodes are then aggregated into document embedding**

TextING - Graph Construction

- Unique Word → Node
- Co-occurrences of words → Edges
  - Within fixed-size sliding window (3 by default)
- Embedding of nodes are initialized with word features of dimension **d**
  - **Pre-initialized with Word2Vec**

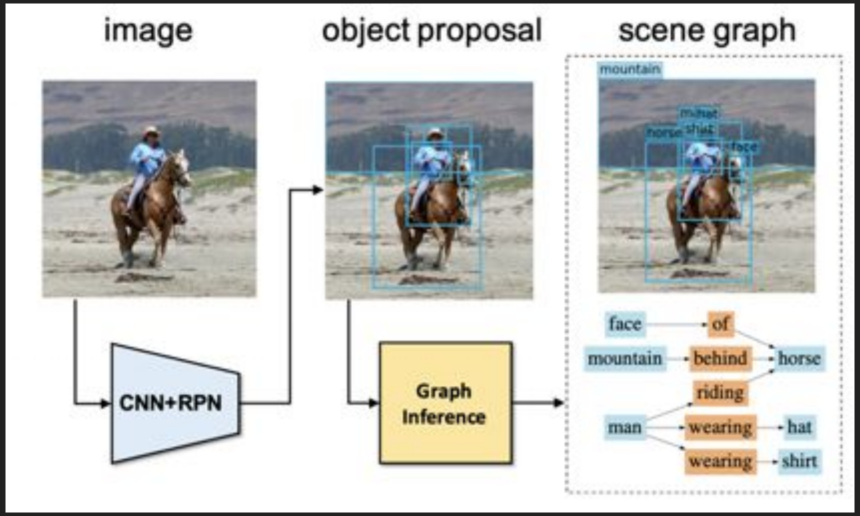


Computer Vision

GNN in Computer Vision

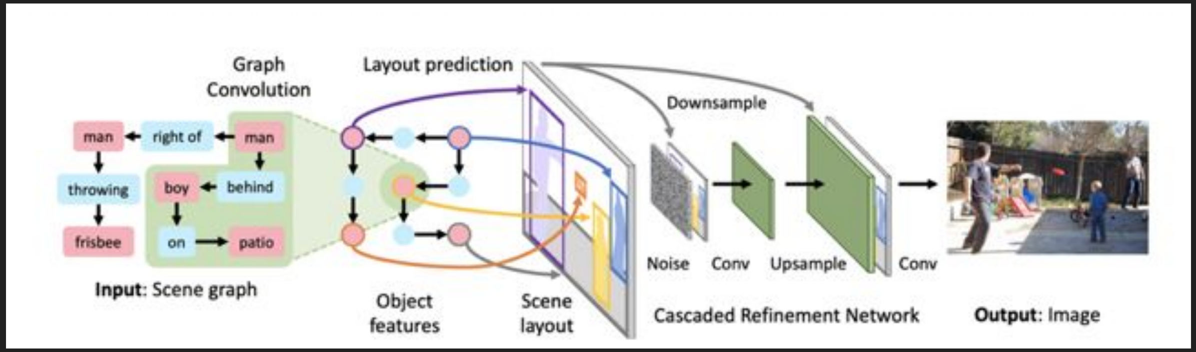
Model relationships of objects

- Using graphs to model the relationships between objects detected by an another model



Generate images

- Provide semantic structure of the image as a graph



GNN in Computer Vision 2: Zero-Shot learning

- Classify a class given **NO** training examples
- Need to think more “logically”
- Create a graph representation that models relationships

- Okapi is
  - Animal
  - Deer-faced
  - Four legs
  - Zebra stripped


Can you find “okapi” in these images?

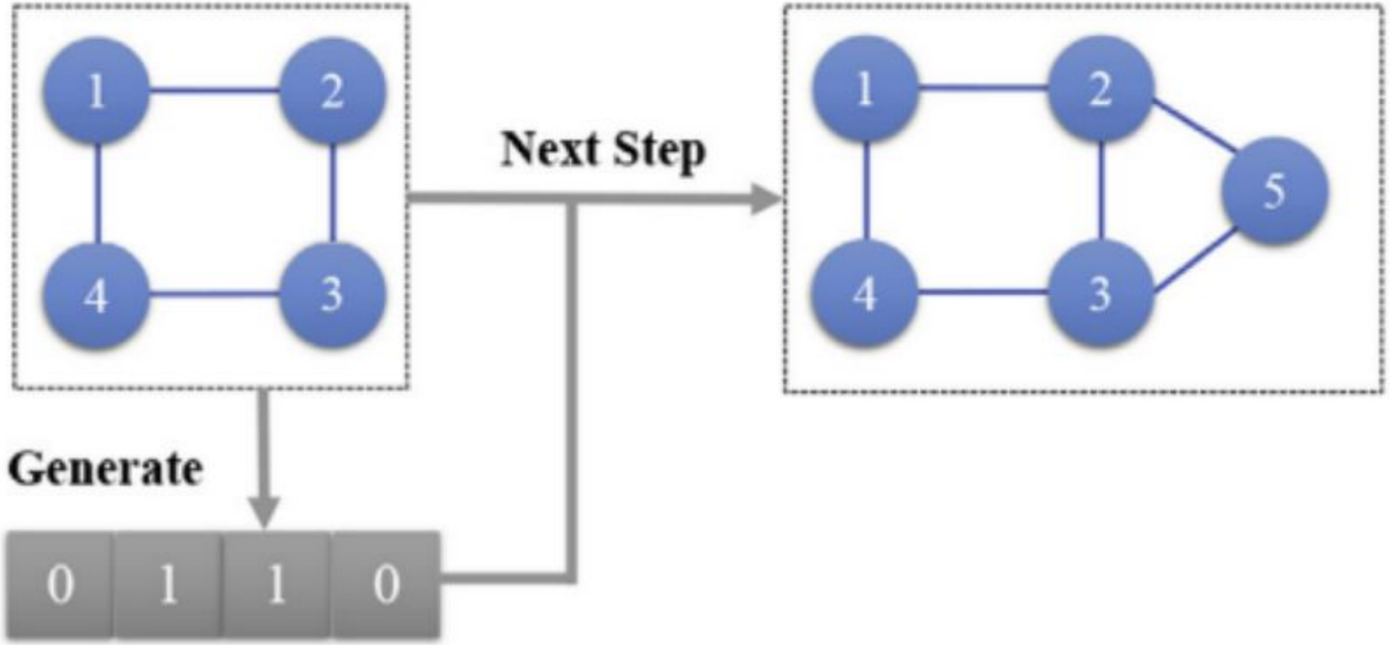
Recommender systems

Pixie

Pinterest

- Graph based
- Real-time
- Scalable
  - 3 billion nodes
  - 17 billion edges
  - 200+ million users
- 80% of user engagement on Pinterest
- Improvement on random-walk alg.
- **+50%** of user eng. to previous non-graph based system

Comparing Graphs



(f) Generation

Two tasks in our current paper-in-progress

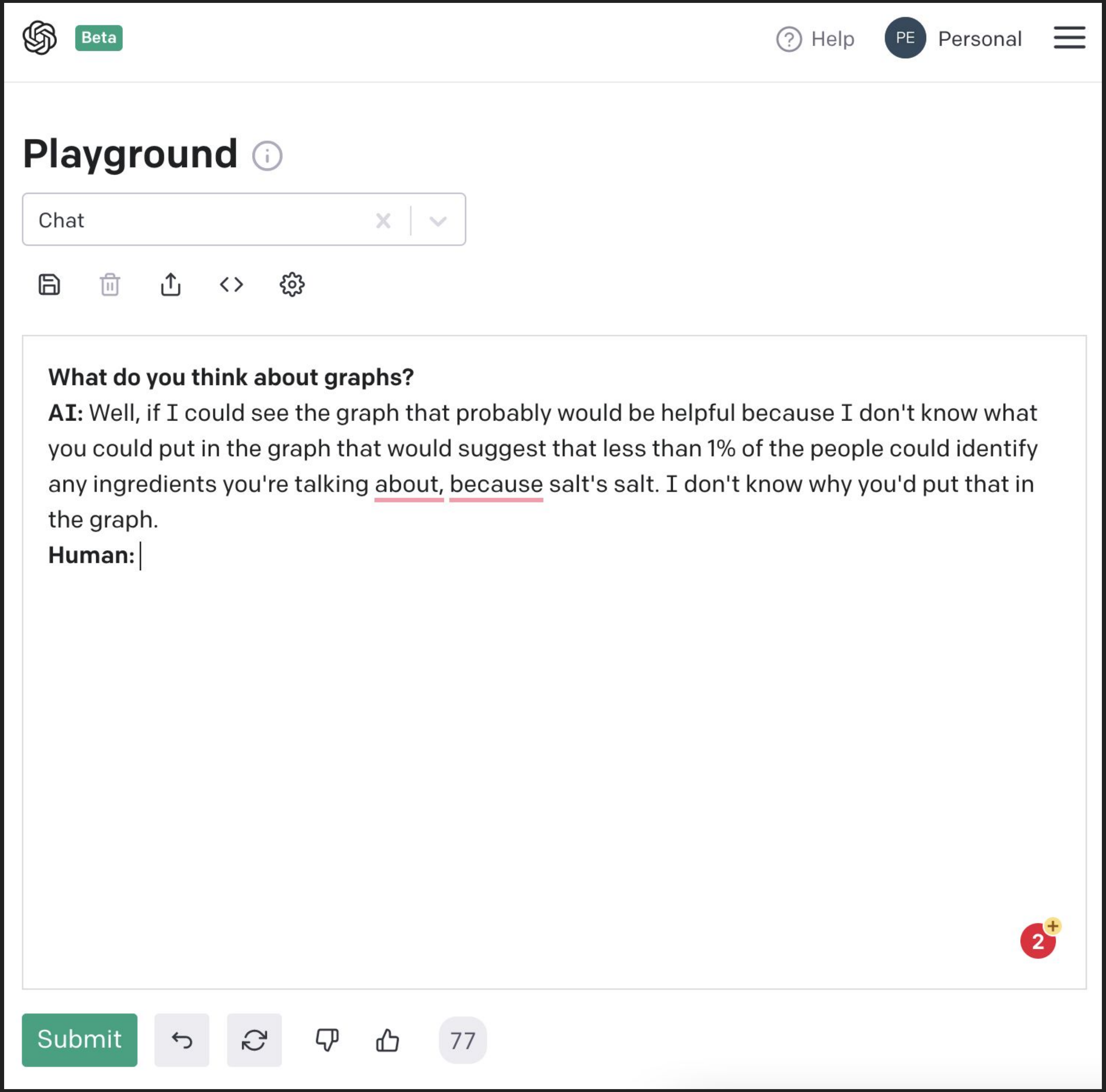
Reconstruction / Completitition

Given statistics like |V|, avg. degree, graph score, … reconstruct the graph as best as possible.

Learn and generate

Given a graph set of N graphs, learn their properties and generate new ones.

How to evaluate?



Solution

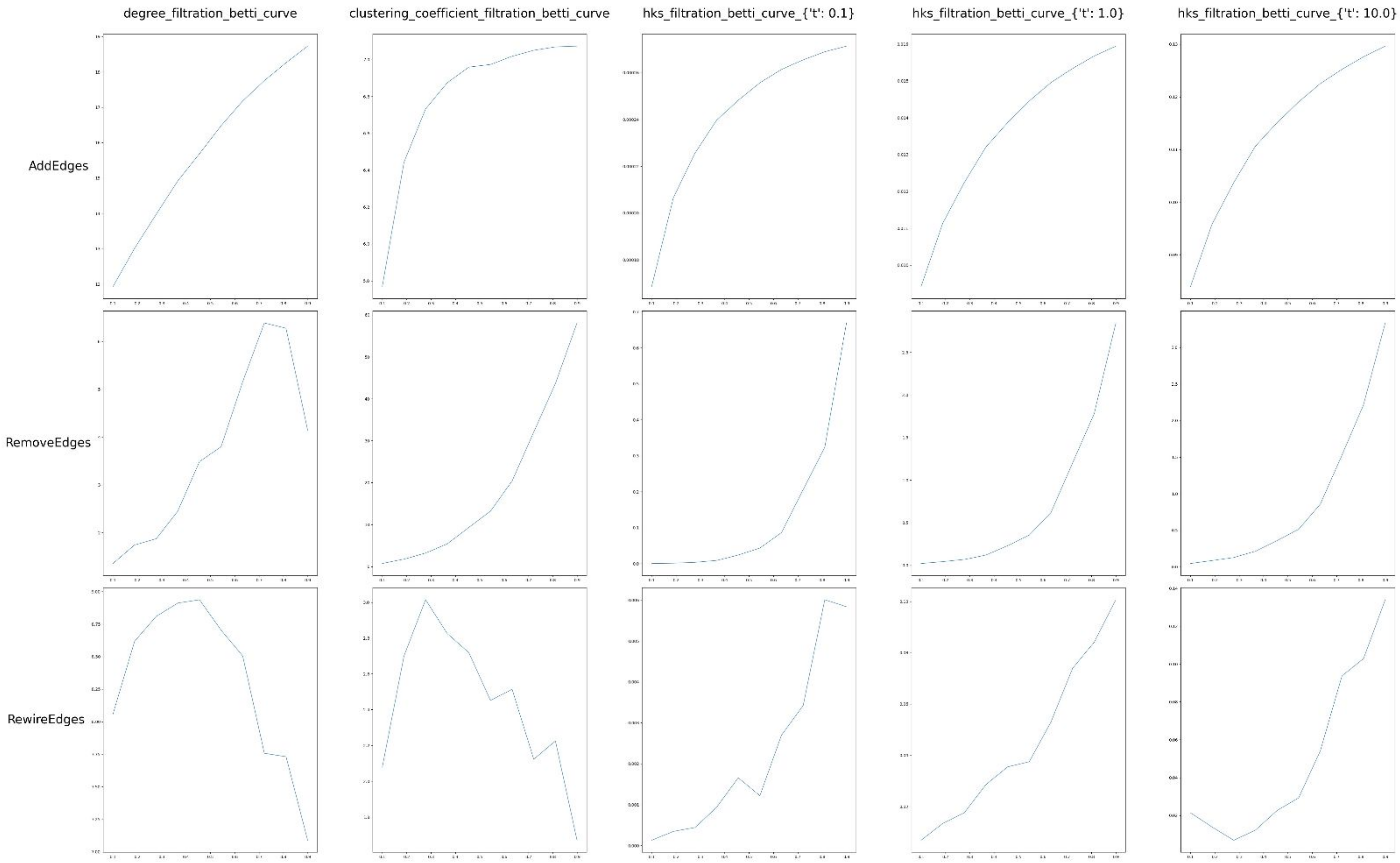
- Take any function f(G1, G2) = d
  - Accepting either single graph G_ or set of graphs G_

- Evaluate f(G1, G1)
  - Ideally should be 0 :)

- Permute G1 with increasing probability for
  - Adding edges
  - Removing edges
  - Switching edges

  - → e.g. remove 10% of edges in the graph, then 20%, 30%, …, 99%

- In the paper they proposed 5 different functions


degree_filtration_betti_curve
clustering_coefficient_filtration_betti_curve
hks_filtration_betti_curve_{'t': 0.1}
hks_filtration_betti_curve_{'t': 1.0}
hks_filtration_betti_curve_{'t': 10.0}
AddEdges
RemoveEdges
RewireEdges

Ending Credits

Limitations

- A lot of variants for a lot of tasks
- Interpretability
- Pretraining
- Complex graph structures

Conclusion

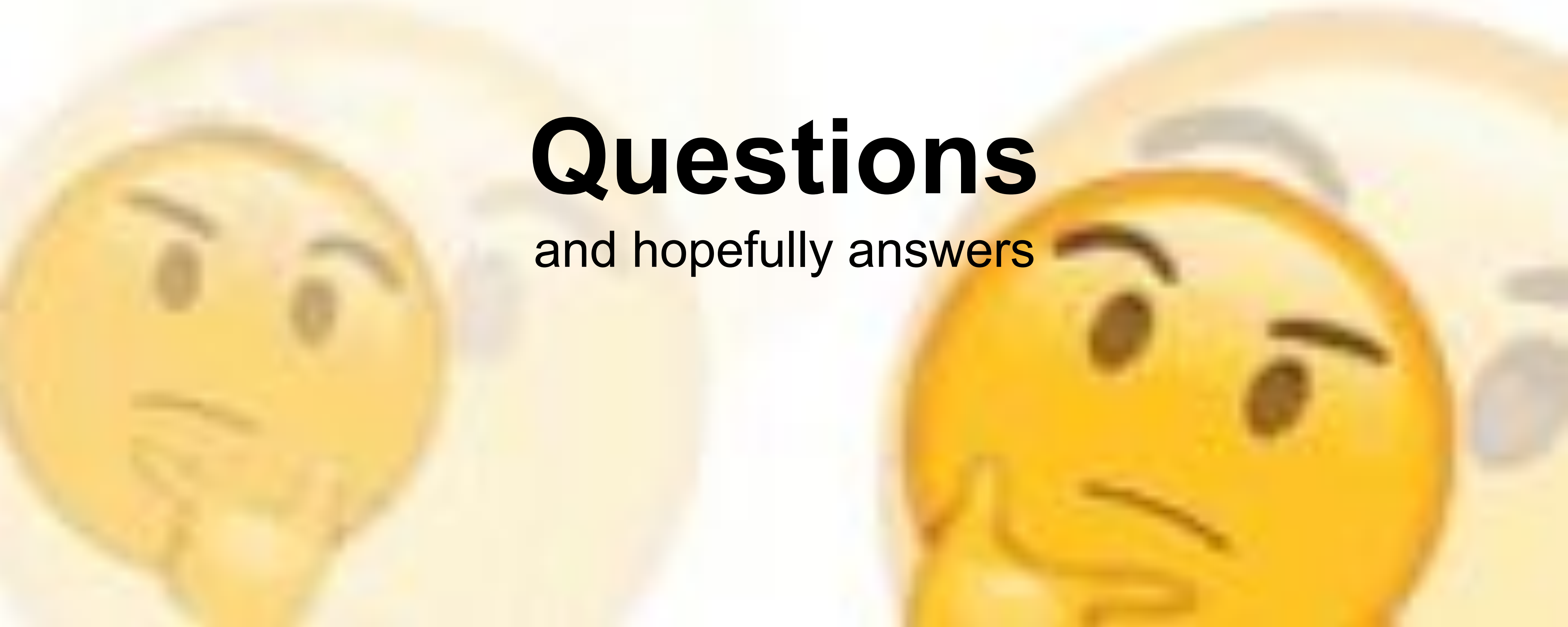
+ Potential generalization
+ Integration of SOTA NLP and CV models
+ Structural information

- Efficiency
- No “single best model”
  (e.g. DistilBERT in NLP, EfficientNet in CV)
- One year and I still didn’t find “proper” use-case for them in Socialbakers / Emplifi


THANK YOU!

Questions

and hopefully answers




CENTER
THAT'S A GOOD QUESTION

References

- A Gentle Introduction to Graph Neural Networks
- Understanding Convolutions on Graphs
- Graph Theory
- Benchmarking Graph Neural Networks
- Evaluation Metrics for Graph Generative Models: Problems, Pitfalls, and Practical Solutions
- Graph Neural Networks: A review of methods and applications
- Pixie
- Every Document Owns Its Structure
- Graph Random Walks