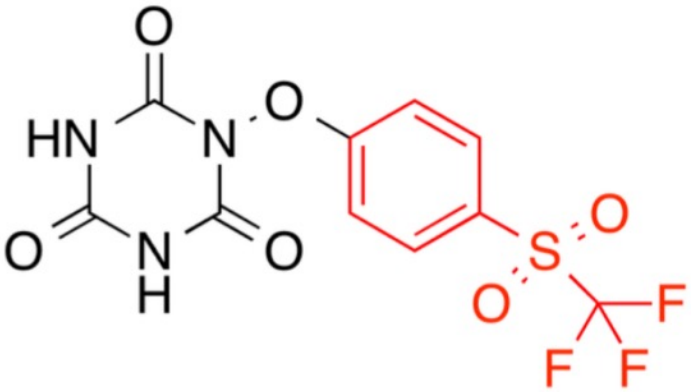
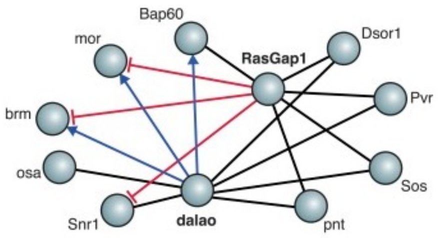
Neural networks for graph-structured data

Master in Computer Vision Barcelona

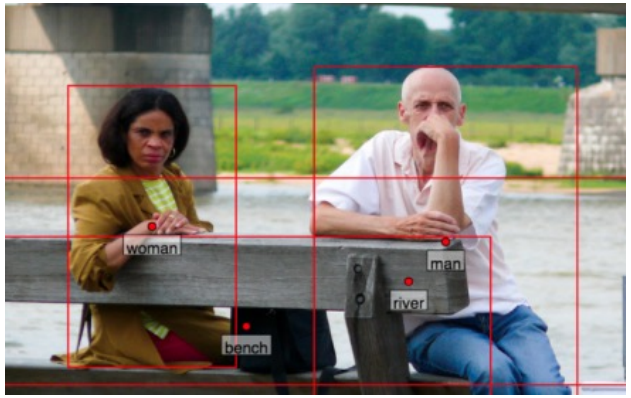
Adriana Romero
adrianars@fb.com

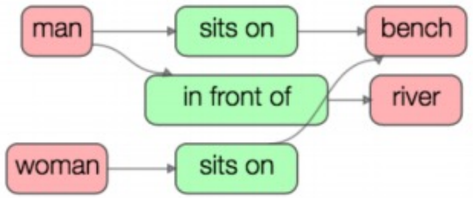
Motivation

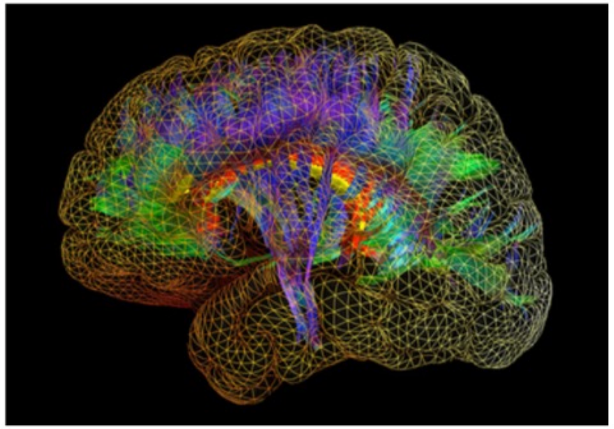




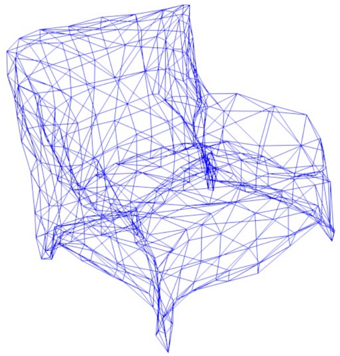
from Wikipedia

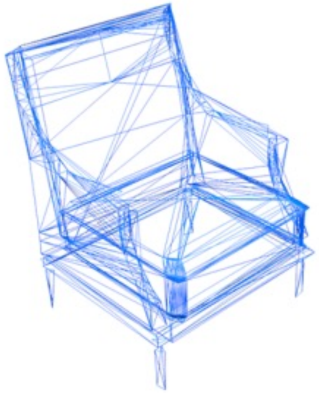






Graphs are everywhere!





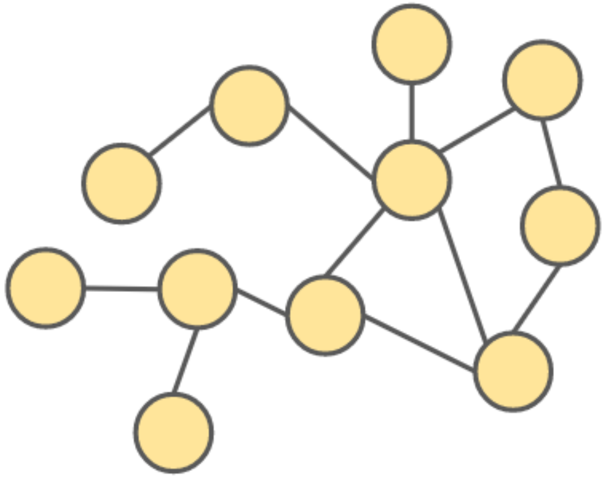
Outline

- Notation & problem formulation
- Node Embeddings:
  - Per-node classifier
  - Indirectly injecting structural information
- Graph Neural Networks
  - Neural Message Passing
  - Basic GNN
  - GCN
  - GAT
  - Updates and over-smoothing
- Computer vision applications
- Wrap Up

Notation &
problem formulation

Notation (1)

A graph $\mathcal{G}$ is defined as a set of nodes (vertices) $\mathcal{V}$ and a set of edges $\mathcal{E}$ between nodes: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

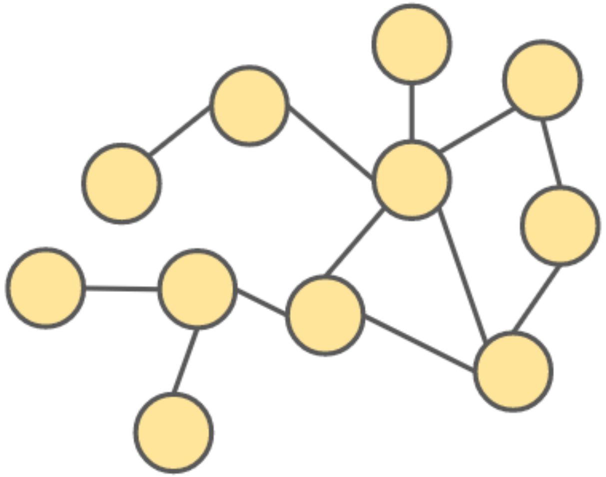


Graphs are often represented by means of an adjacency matrix $\boldsymbol{A}$:

- $\boldsymbol{A}$ can represent directed and undirected graphs (symmetric).
- $\boldsymbol{A} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ to represent presence/absence of edges.
- $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ to represent graphs with weighted edges.
- $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times |\mathcal{R}|}$ to represent multi-relation graphs (with edges representing types of interactions)

Notation (2)

A graph $\mathcal{G}$ is defined as a set of nodes (vertices) $\mathcal{V}$ and a set of edges $\mathcal{E}$ between nodes: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

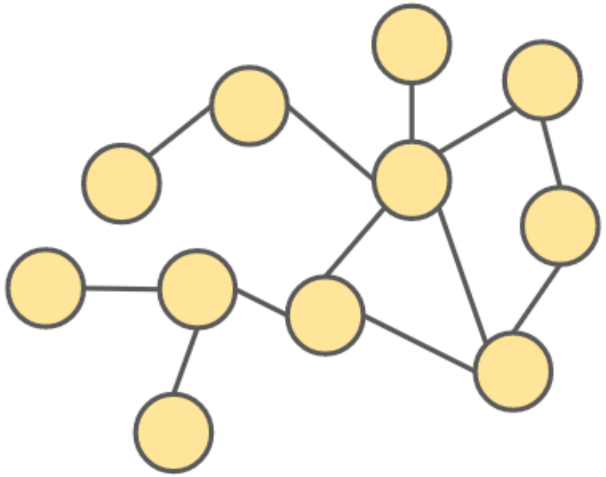


Graphs are often represented by means of an adjacency matrix $\boldsymbol{A}$.

Nodes may be characterized by a matrix of node features $\boldsymbol{F} \in \mathbb{R}^{|\mathcal{V}| \times m}$, representing e.g. attribute information.

Problem formulation

Machine learning with graphs can also follow supervised, unsupervised or reinforcement learning paradigms.

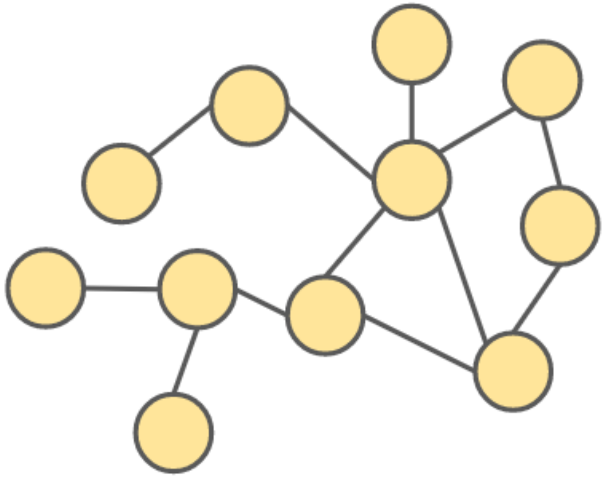


Among the most common tasks, we can find:

- Node classification
- Relation prediction
- Community detection
- Graph classification

Problem formulation

Machine learning with graphs can also follow supervised, unsupervised or reinforcement learning paradigms.



Among the most common tasks, we can find:

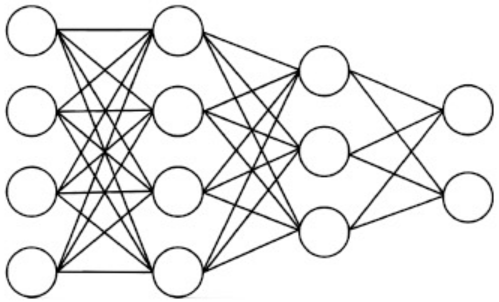
- **Node classification (transductive vs inductive)**
- Relation prediction
- Community detection
- Graph classification

Node embeddings

Per-node classifier

Classify single node features, ignoring graph structure.

$\boldsymbol{F}_u$

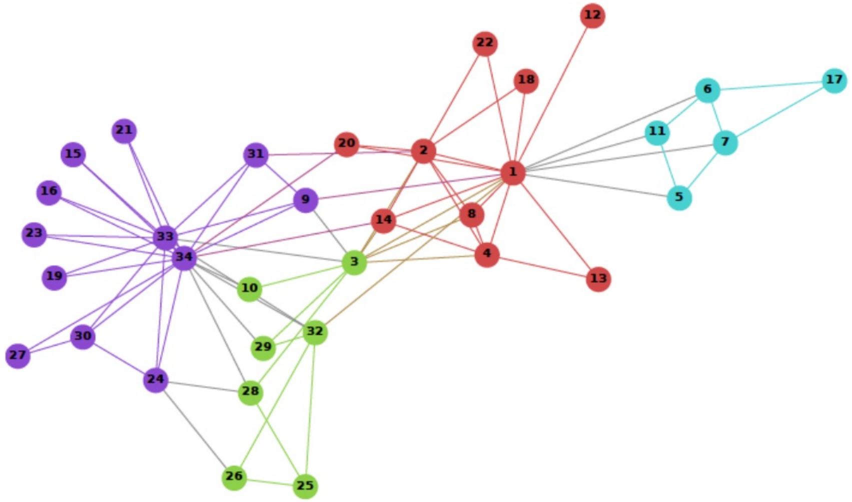


$\hat{\boldsymbol{y}}_{\boldsymbol{u}}$

How can we leverage structural information?

Node embeddings (1)

Represent each node with a feature vector that preserves the structure of the graph.



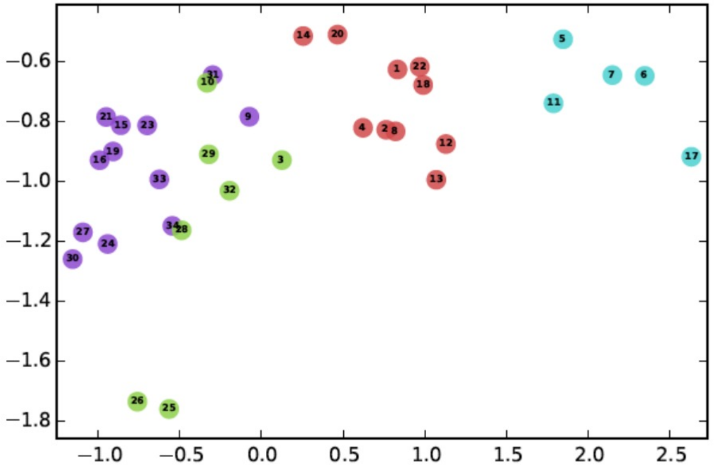
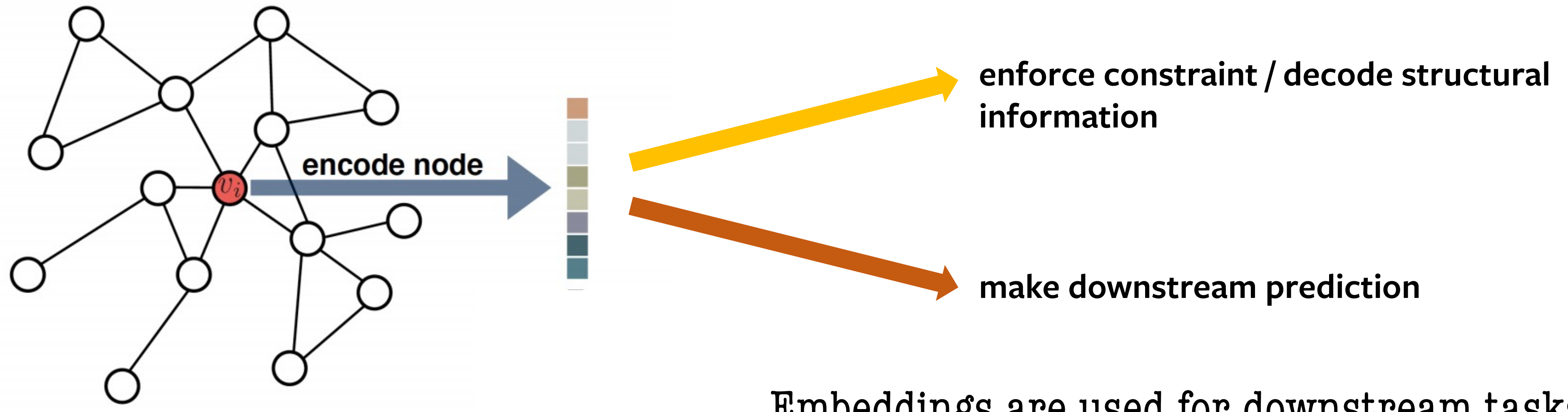


Image from https://cs.mcgill.ca/~wlh/comp766/files/chapter2_draft_jan22.pdf

Node embeddings (2)

Node embedding architectures:

- encode nodes into an embedding vector (feature vector).
- enforce some structural constraint in the embedding space or decode node embeddings into information about node and/or its neighborhood

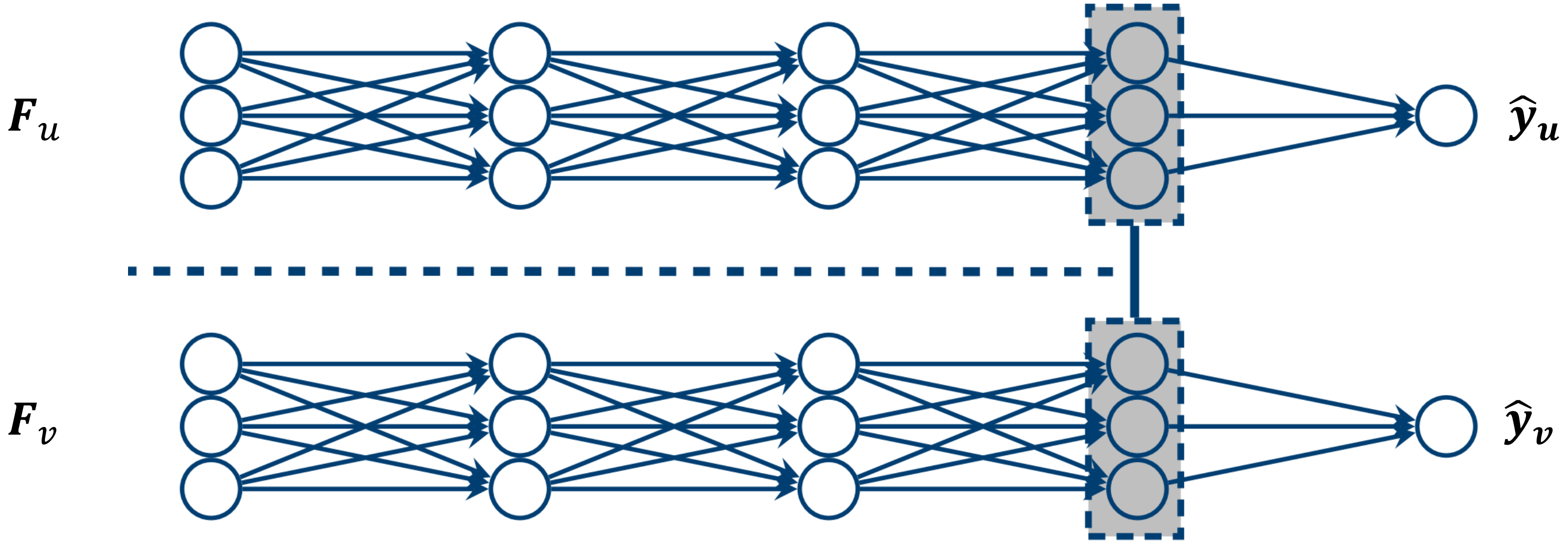


Embeddings are used for downstream tasks.

Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter2_draft_jan22.pdf

Node embeddings (3)

Constraining the learnt node embeddings to be close/distant depending on the presence of edges
(Weston et al., 2008, extending Belkin and Niyogi, 2006 to neural networks)



Training: node classification loss + similarity constraint

Image based on *https://www.cl.cam.ac.uk/~pv273/slides/UCLGraph.pdf*

Node embeddings (4)

Decoding structural information (LINE – Tang et al. 2015)

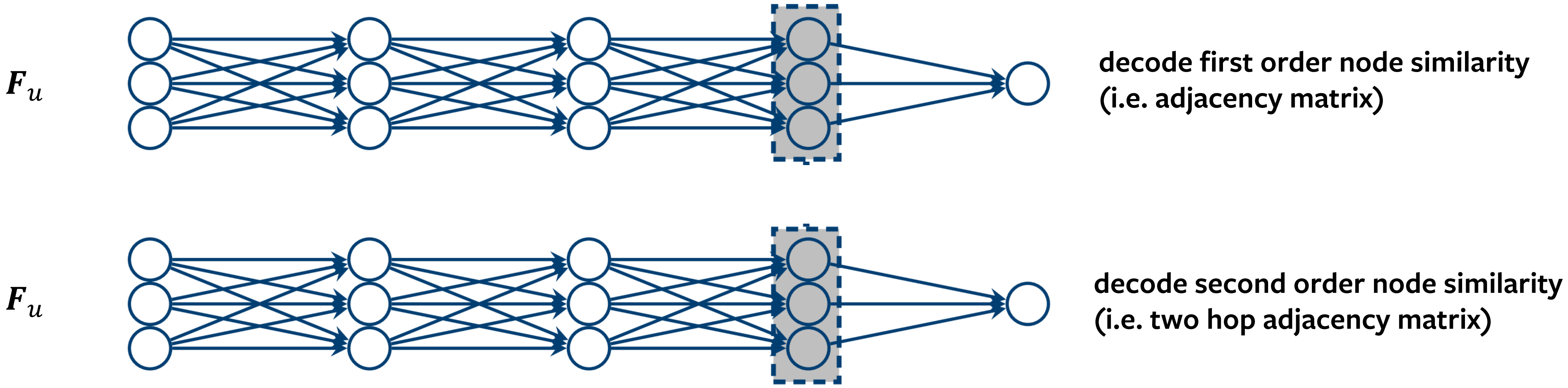
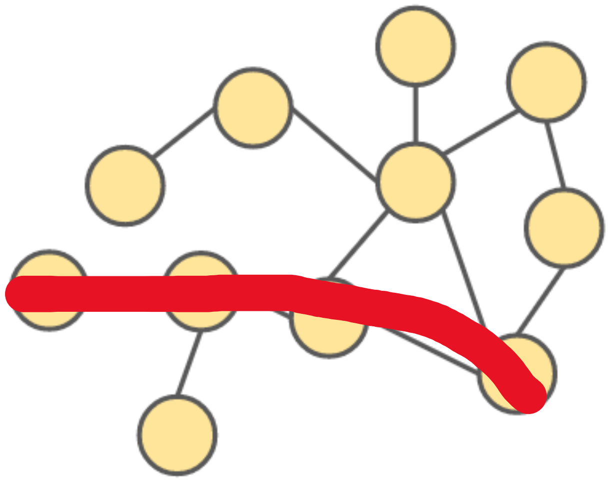


Image based on https://www.cl.cam.ac.uk/~pv273/slides/UCLGraph.pdf

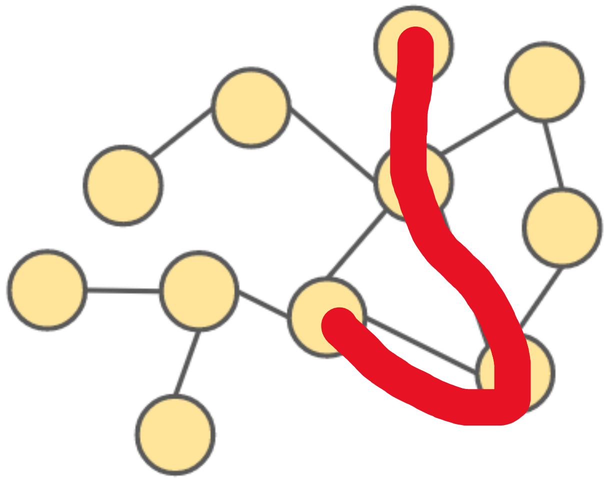
Node embeddings (5)

Alternatively to augmenting the loss with structural constraints, learn node structural features from random walks (DeepWalk – Perozzi et al. 2014, Node2Vec – Grover et al. 2016)



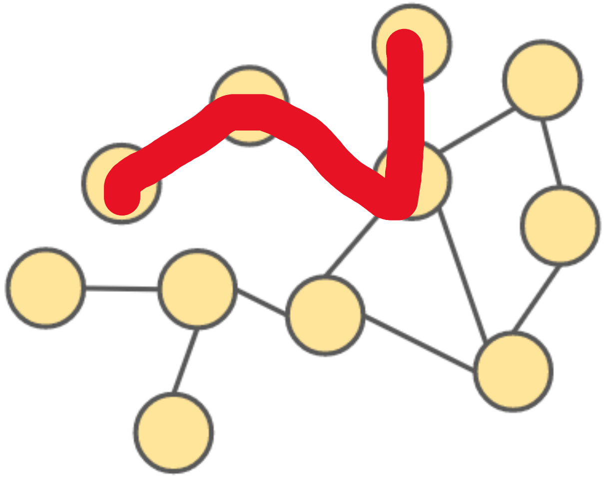
Node embeddings (5)

Alternatively to augmenting the loss with structural constraints, learn node structural features from random walks (DeepWalk – Perozzi et al. 2014, Node2Vec – Grover et al. 2016)



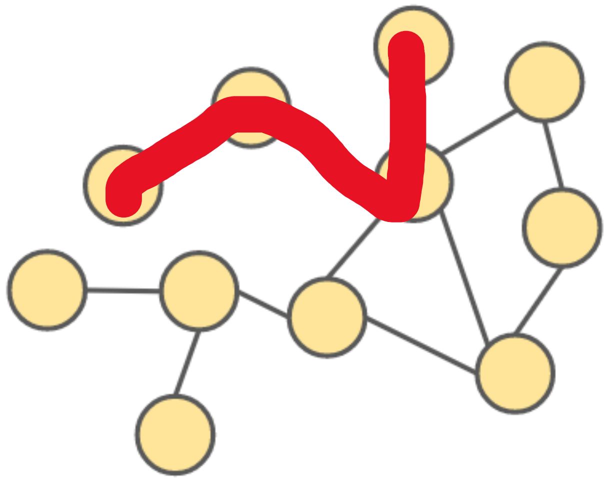
Node embeddings (5)

Alternatively to augmenting the loss with structural constraints, learn node structural features from random walks (DeepWalk – Perozzi et al. 2014, Node2Vec – Grover et al. 2016)



Node embeddings (5)

Alternatively to augmenting the loss with structural constraints, learn node structural features from random walks (DeepWalk – Perozzi et al. 2014, Node2Vec – Grover et al. 2016)

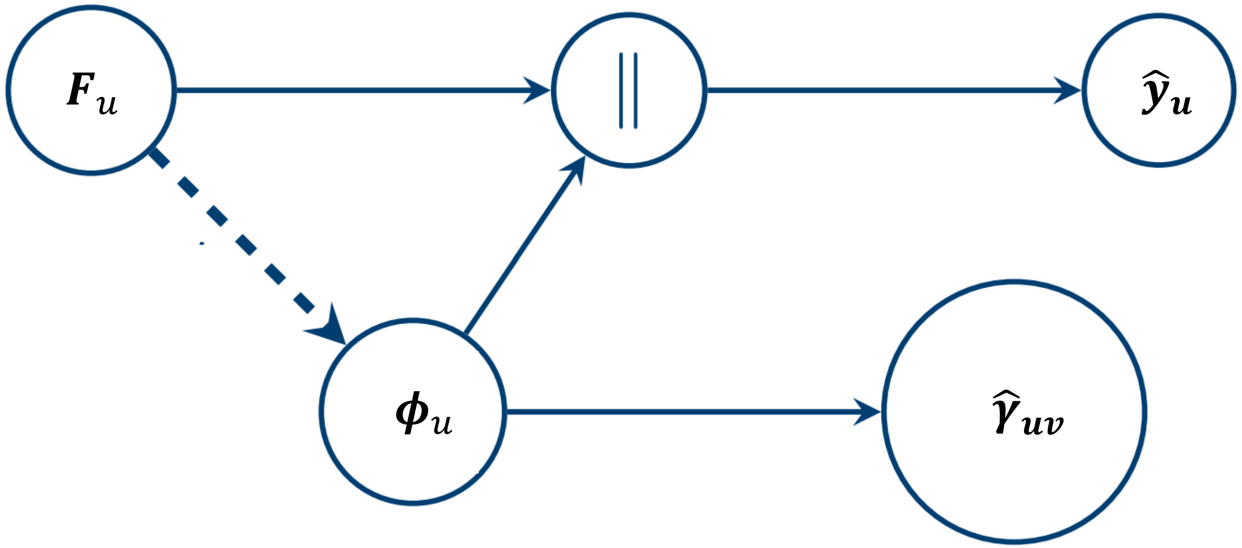


1. Start with random node features $\boldsymbol{\phi}_u$
2. Sample a random walk $\mathcal{W}_u$ from u
3. Update $\boldsymbol{\phi}_u$ to maximize $P(v|\boldsymbol{\phi}_u)$, where v $\in$ $\mathcal{W}_u$.

A good representation of a node should allow us to easily predict the nodes that *surround* it.

Node embeddings (6)

What if we had access to node features and labels? (Planetoid – Yang et al. 2016)



$\hat{\gamma}_{uv}$ predicts whether node u and v have the same label or different label.

Image based on https://www.cl.cam.ac.uk/~pv273/slides/UCLGraph.pdf

Node embeddings (7)

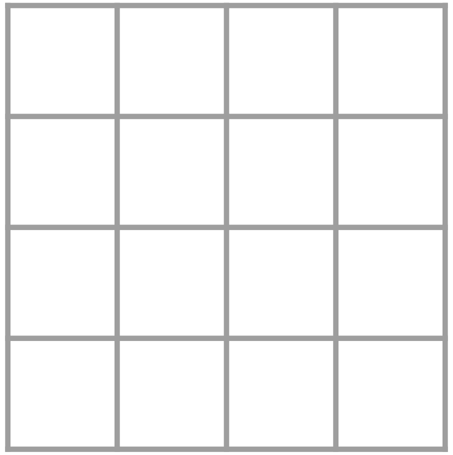
All methods covered so far used a classifier that classifies each node independently, with graph structure injected only indirectly, through the learnt embeddings.

How can we directly leverage the graph structure when computing node features?

Deep Learning toolbox (1)

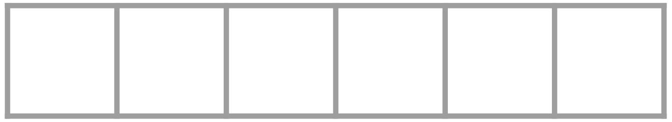
CNNs ? They work well on data defined in n-D grids

2D grid



1D grid

"I like cats and dogs."



Deep Learning toolbox (2)

RNNs? They are well-defined over sequences.

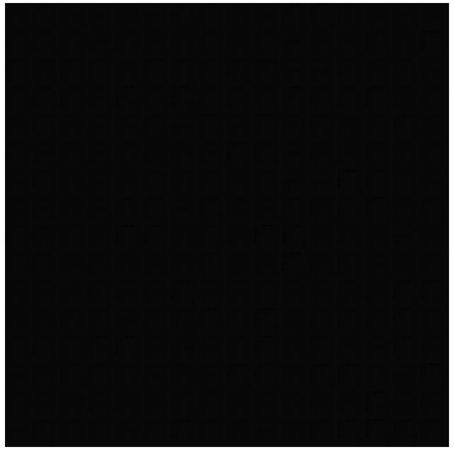
image captioning

from http://kelvinxu.github.io/projects/capgen.html

video frame classification

from http://cs.stanford.edu/people/karpathy/deepvideo/

image generation
(sequential processing of output)



from https://github.com/jbornschein/draw

Deep Learning toolbox (3)

MLPs ?

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

flatten

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

Depends on ordering of nodes in the adjacency matrix, the model would not be permutation equivariant.

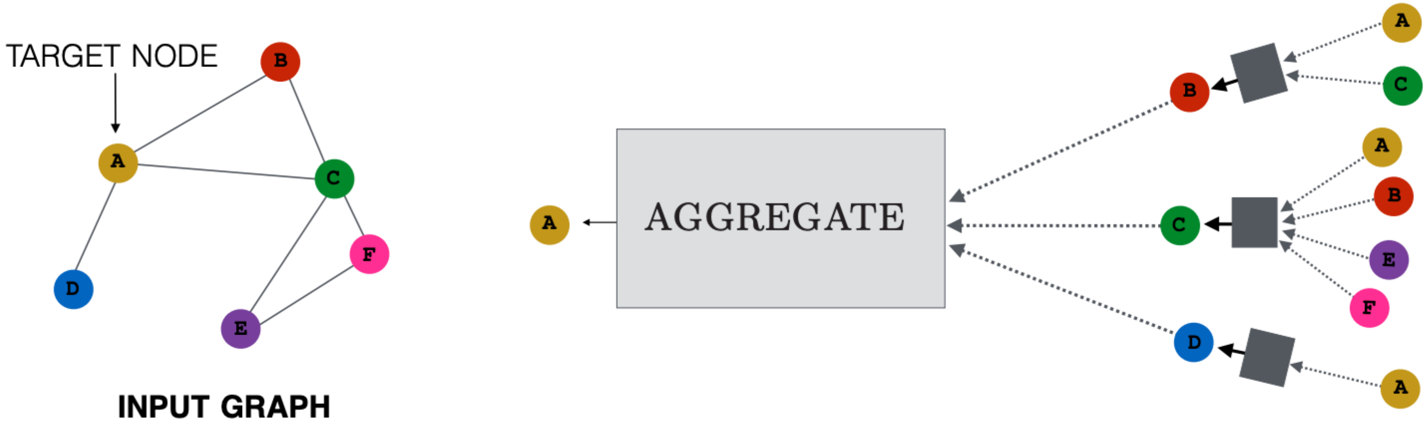
Graph Neural Networks (GNN)

GNN (1)

Key idea: compute node representations that depend on the structure of the graph as well as node features. (Gori et al. 2005, Scarselli et al. 2009)

> *Note: The same model has been derived as a generalization of convolutions to non-Euclidean data* (Bruna et al., 2014).

Neural Message Passing



Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf

GNN (2)

Note that we can reformulate with self loops by considering $\boldsymbol{H}_u^{(k-1)}$ within $AGGREGATE^{(k)}$.

Formally, message passing updates are expressed as

$$\boldsymbol{H}_u^{(k)} = UPDATE^{(k)}(\boldsymbol{H}_u^{(k-1)}, \underbrace{AGGREGATE^{(k)}(\{\boldsymbol{H}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\})}_{\text{message}})$$

where $UPDATE$ and $AGGREGATE$ are arbitrary differentiable functions (e.g. neural nets).

As iterations progress, each node contain more information from further reaches of the graph.

Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf

GNN (3)

Formally, message passing updates are expressed as

$$\boldsymbol{H}_u^{(k)} = UPDATE^{(k)}\Big(\boldsymbol{H}_u^{(k-1)}, \underbrace{AGGREGATE^{(k)}\big(\{\boldsymbol{H}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\}}_{\text{message}}\big)\Big)$$

where $UPDATE$ and $AGGREGATE$ are arbitrary differentiable functions (e.g. neural nets).

For basic GNNs: $$\boldsymbol{H}_u^{(k)} = \sigma\Big(\boldsymbol{W}_{self}^{(k)} \boldsymbol{H}_u^{(k-1)} + \boldsymbol{W}_{neigh}^{(k)} \underbrace{\textstyle\sum_{v \in \mathcal{N}(u)} \boldsymbol{H}_v^{(k-1)}}_{\text{Sum of neighbor features}} + \boldsymbol{b}^{(k)}\Big)$$

Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf

GNN (3)

Formally, message passing updates are expressed as

$$\boldsymbol{H}_u^{(k)} = UPDATE^{(k)}\Big(\boldsymbol{H}_u^{(k-1)}, \underbrace{AGGREGATE^{(k)}\big(\big\{\boldsymbol{H}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\big\}}_{\text{message}}\big)\Big)$$

where $UPDATE$ and $AGGREGATE$ are arbitrary differentiable functions (e.g. neural nets).

For basic GNNs: $$\boldsymbol{H}^{(k)} = \sigma((\boldsymbol{A}+\boldsymbol{I})\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)})$$

Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf

GNN (3)

Formally, message passing updates are expressed as

$$\boldsymbol{H}_u^{(k)} = UPDATE^{(k)}(\boldsymbol{H}_u^{(k-1)}, \underbrace{AGGREGATE^{(k)}(\{\boldsymbol{H}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\})}_{\text{message}})$$

where $UPDATE$ and $AGGREGATE$ are arbitrary differentiable functions (e.g. neural nets).

For basic GNNs: $$\boldsymbol{H}_u^{(k)} = \sigma(\boldsymbol{W}_{self}^{(k)} \boldsymbol{H}_u^{(k-1)} + \boldsymbol{W}_{neigh}^{(k)} \underbrace{\sum_{v \in \mathcal{N}(u)} \boldsymbol{H}_v^{(k-1)}}_{\text{Sum of neighbor features}} + \boldsymbol{b}^{(k)})$$

highly sensitive to node degree

Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf

Graph Convolutional Networks (GCN)

Key idea: neighborhood normalization motivated by spectral graph theory, resulting in first order approximation of a spectral graph convolution. (Kipf and Welling, 2017)

Message defined as: $$\sum_{v \in \mathcal{N}(u)} \frac{\boldsymbol{H}_v^{(k-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}$$

and so, $$\boldsymbol{H}_u^{(k)} = \sigma\left(\boldsymbol{W}^{(k)}\left(\sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\boldsymbol{H}_v^{(k-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)\right)$$

This formulation gives less weight to messages from neighbors with higher degrees.

Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf and https://tkipf.github.io/graph-convolutional-networks/

Other aggregate functions

Key idea: use any permutation invariant function as aggregator (e.g. set aggregators).

In particular, the message could be defined by means of a set pooling operation (Zaheer et al., 2017)

$$MLP_\theta\left(\sum_{v\in\mathcal{N}(u)} MLP_\phi\left(\boldsymbol{H}_v^{(k-1)}\right)\right)$$

by combining a set of node features into a single embedding.

Note that the sum could be replaced by max or min operation. (Qi et al., 2017)

Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf

Graph Attention Networks - GAT (1)

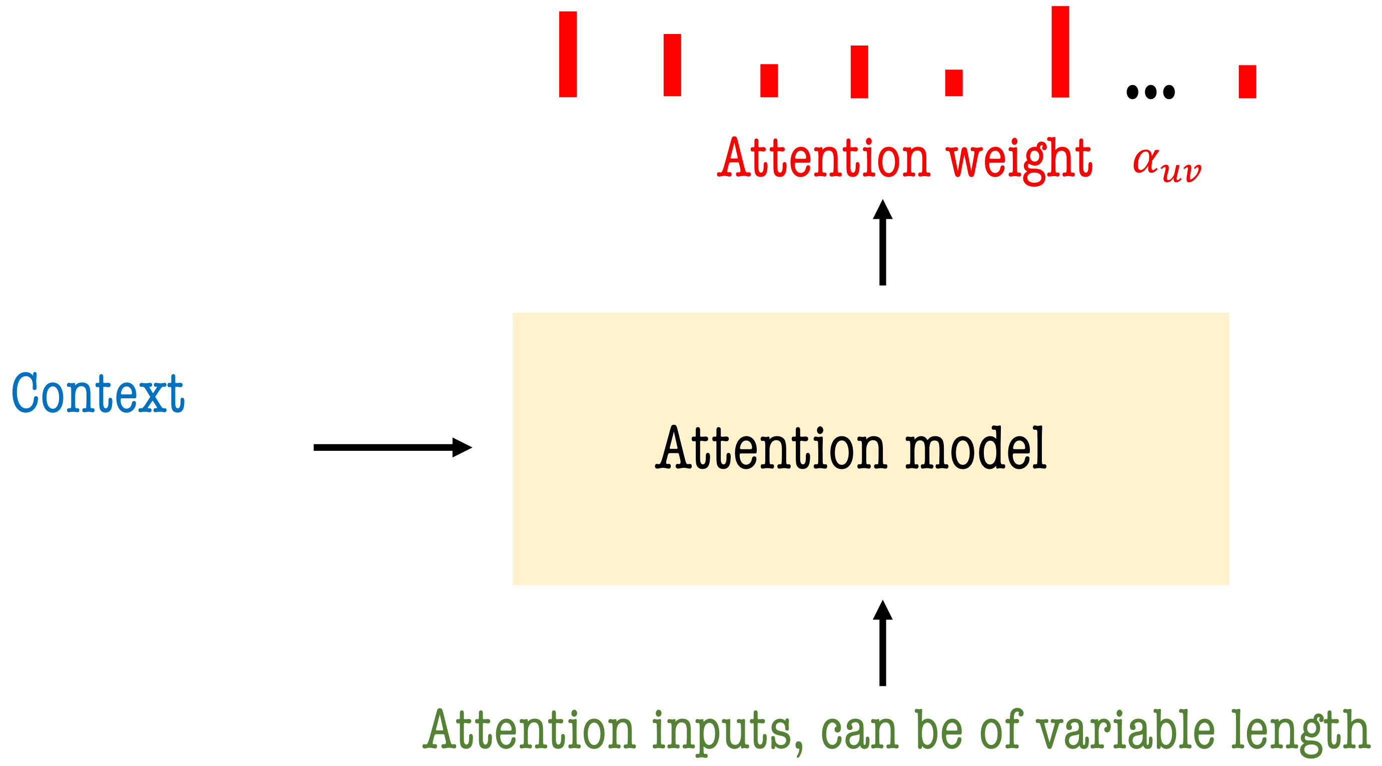
Key idea: weigh the neighbors influence at aggregation time, by learning attention weights. (Velickovic et al., 2018)

Message defined as: $$\sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv} \boldsymbol{H}_v^{(k-1)}$$

How do we compute α_{uv} ?

GAT (2)

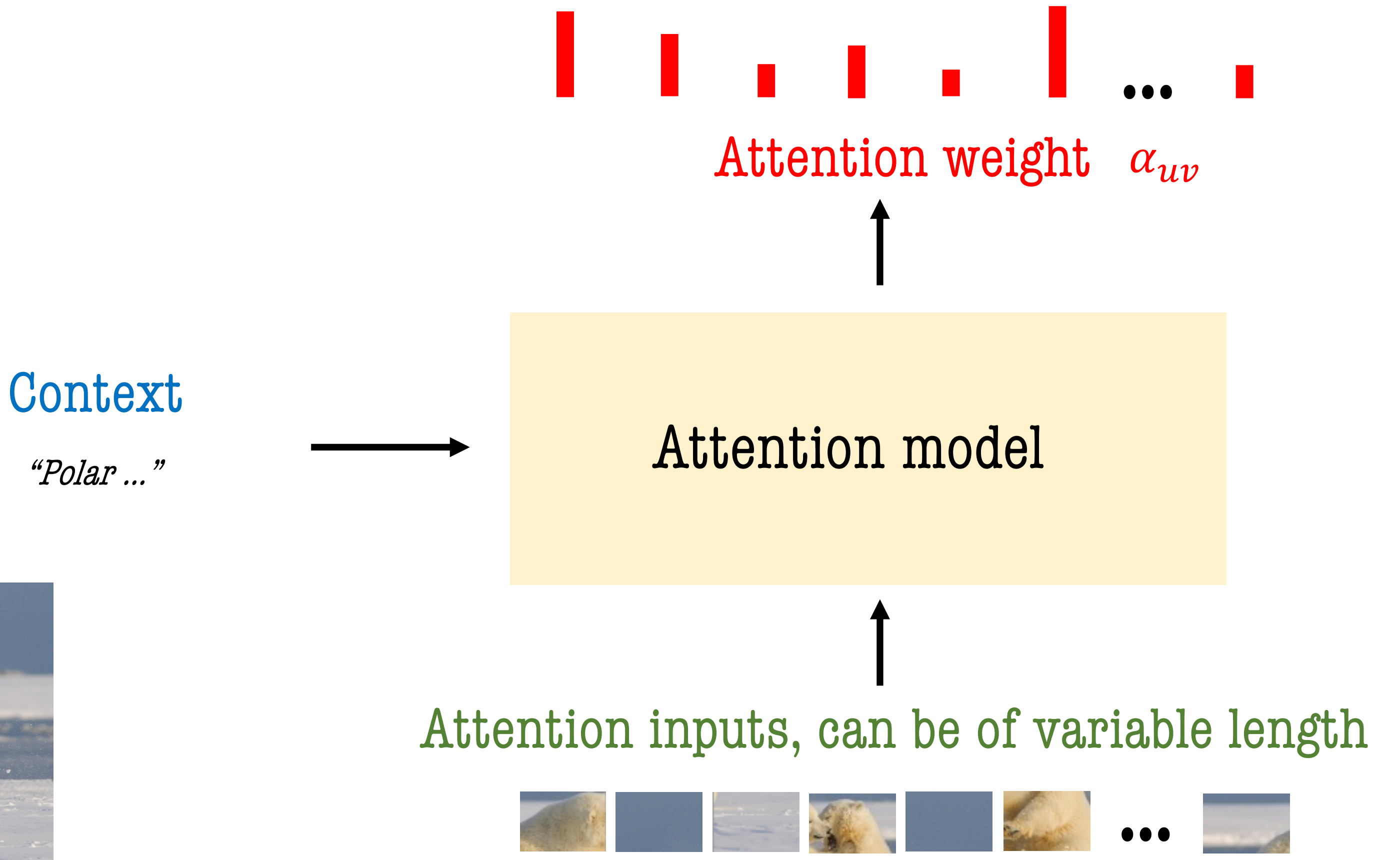
Attention allows us to focus and select the most pertinent pieces of information.



(Bahdanau et al., 2015; Vaswani et al., 2017)

GAT (2)

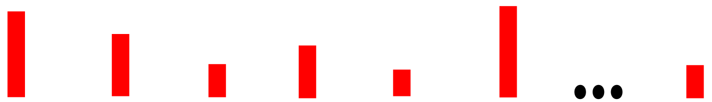
Attention allows us to focus and select the most pertinent pieces of information.



(Bahdanau et al., 2015; Vaswani et al., 2017)

GAT (2)

Attention allows us to focus and select the most pertinent pieces of information.



Attention weight α_{uv}

Context

Features of node u

$\boldsymbol{H}_u^{(k-1)}$

Attention model

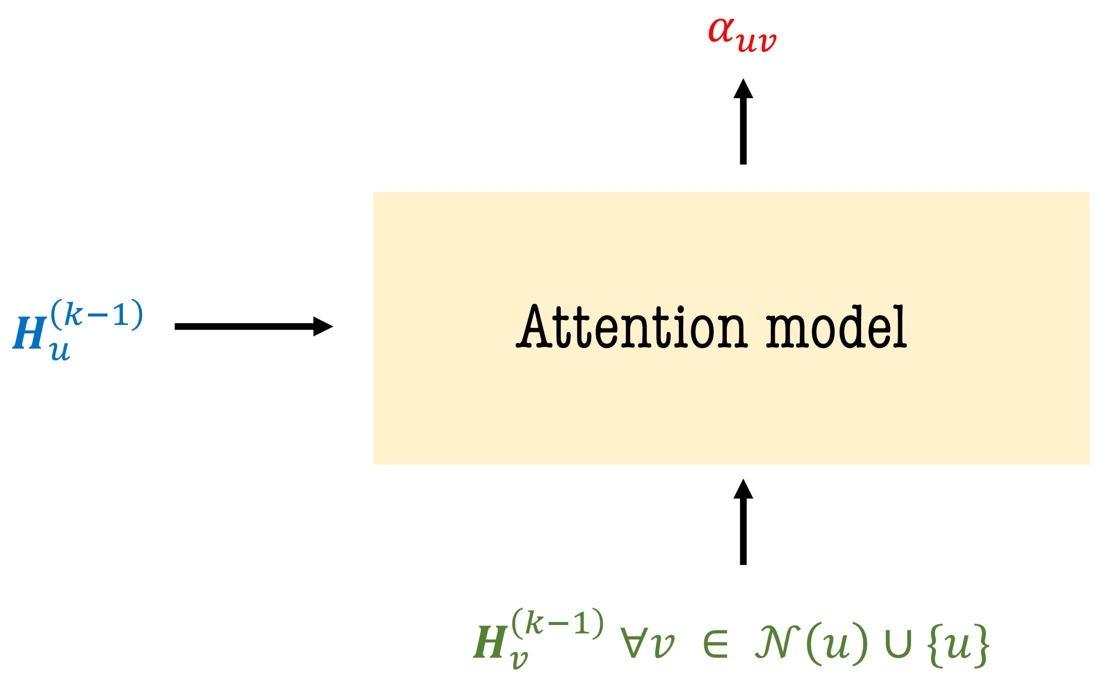
Attention inputs, can be of variable length

Features of neighboring nodes v

$\boldsymbol{H}_v^{(k-1)} \; \forall v \in \mathcal{N}(u) \cup \{u\}$

(Velickovic et al., 2018)

GAT (3)



1. Combine attention input and context information

$$s_{uv} = f_\theta\left(H_u^{(k-1)}, H_v^{(k-1)}\right)$$

2. Apply softmax to obtain attention weights α_{uv}

$$\alpha_{uv} = \frac{\exp(s_{uv})}{\sum_{v' \in \mathcal{N}(u)\cup\{u\}} \exp(s_{uv'})}$$

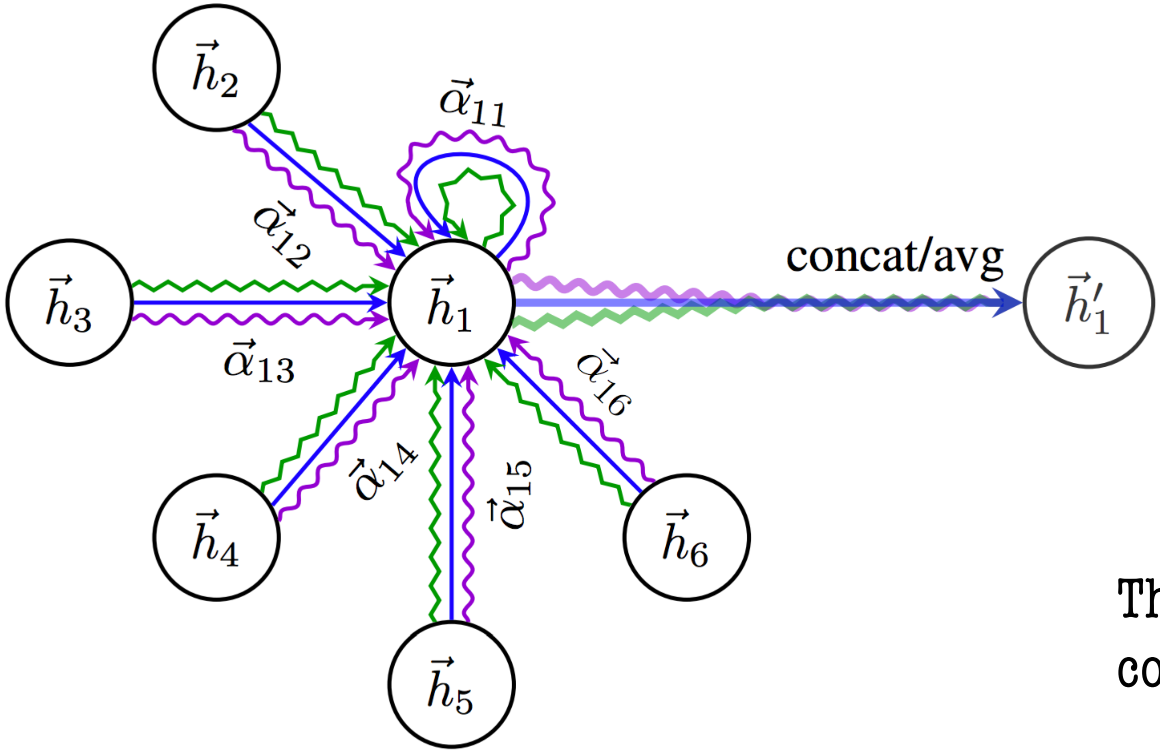
3. Apply attention coefficients to input and aggregate

$$\sum_{v \in \mathcal{N}(u)\cup\{u\}} \alpha_{uv} H_v^{(k-1)}$$

(Velickovic et al., 2018)

GAT (4)

Following the Transformer (Vaswani et al., 2017), GAT employs multiple attention heads:



$$\boldsymbol{H}_u^{(k)} = \sigma\left(\sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv} \boldsymbol{H}_v^{(k-1)}\right)$$

For one attention head.

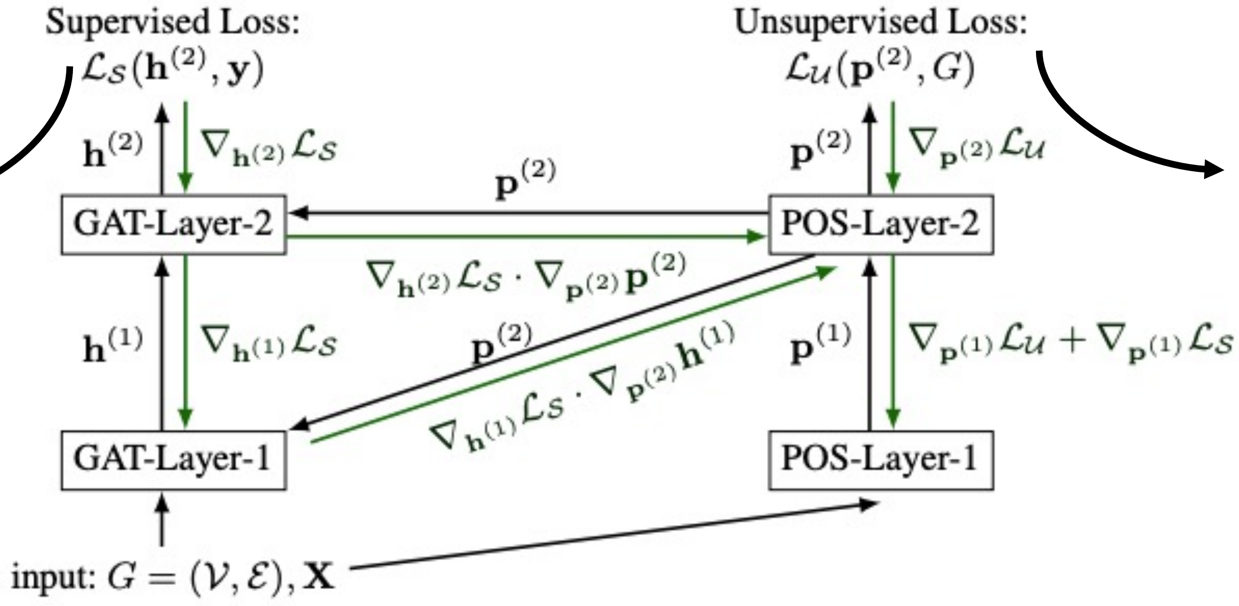
The output of multiple heads can then be combined via e.g. concatenation.

Image from https://arxiv.org/pdf/1710.10903.pdf

GAT-POS

Note that GAT formulation treats nodes in a neighborhood as a set, and so structural information of the nodes is lost. GAT-POS augments GAT with positional information.

Predictive of node classes



Predictive of structural information

(Ma et al., 2021)

UPDATE & over-smoothing

Recall the basic GNN update:

$$\boldsymbol{H}_u^{(k)} = \sigma(\underbrace{\boldsymbol{W}_{self}^{(k)} \boldsymbol{H}_u^{(k-1)} + \boldsymbol{W}_{neigh}^{(k)} \sum_{v \in \mathcal{N}(u)} \boldsymbol{H}_v^{(k-1)} + \boldsymbol{b}^{(k)}}_{\text{Linear combination of the aggregated neighboring information and the previous representation of the node}})$$

May result in over-smoothing.

How can we address over-smoothing?

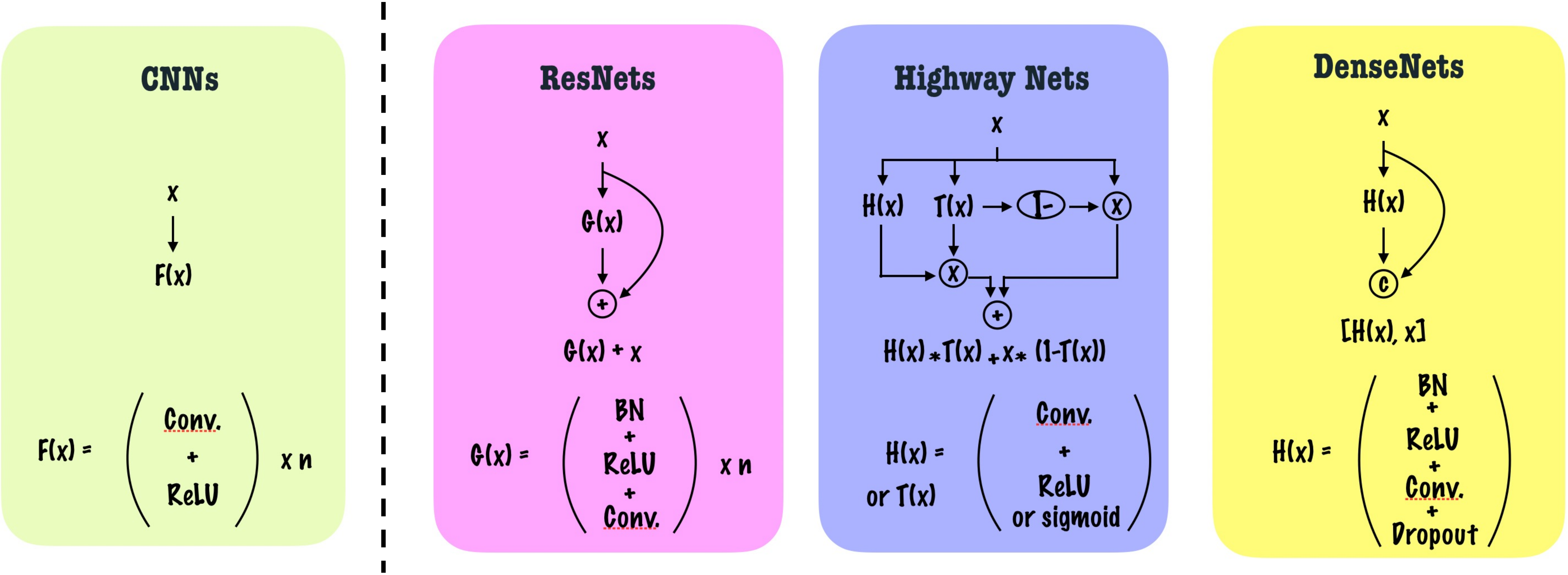
Based on https://cs.mcgill.ca/~wlh/comp766/files/chapter4_draft_partial_feb12.pdf

Skip connections (1)

To preserve information from previous message passing iterations in the update function.

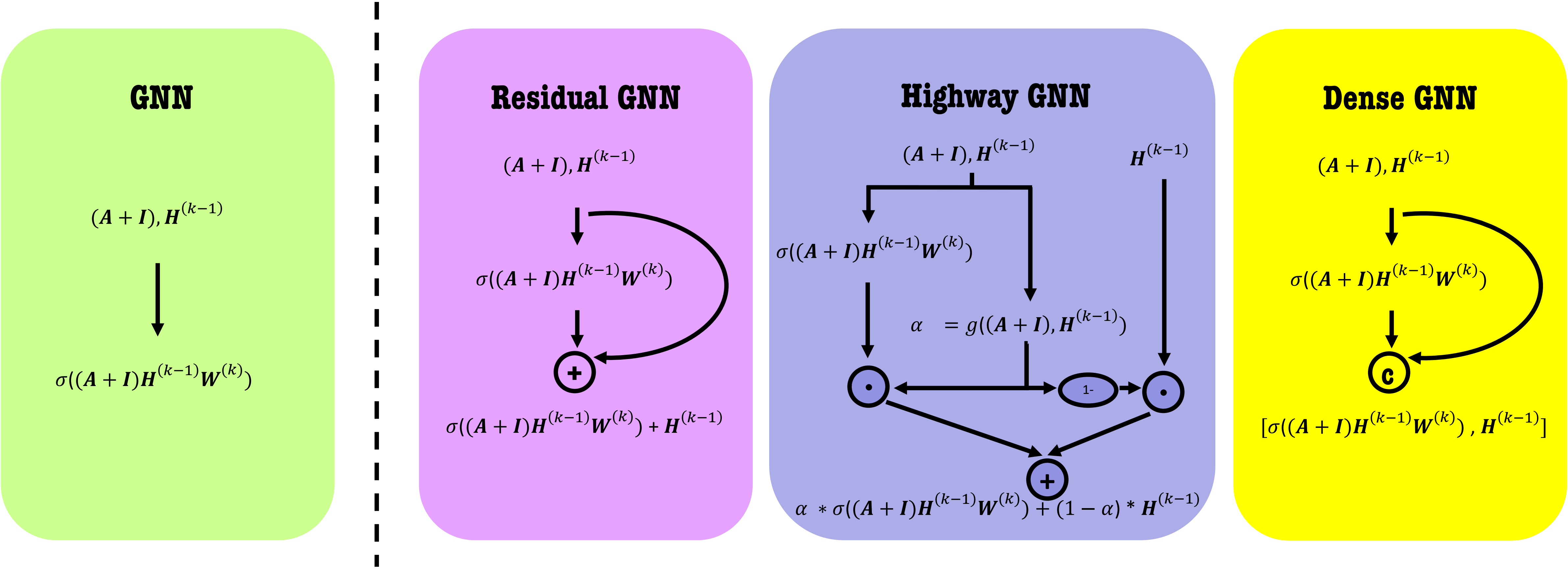
Skip connections (2)

Review of image classification architectures with skip connections



Skip connections (3)

GNN also have their skip connection-based counterparts.



(Pham et al., 2017) (Hamilton et al., 2017 a& Xu et al., 2018)

Zero-neighbor update

Key idea: Update only a part of the node's features with the neighbors' information.

Update defined as: $\boldsymbol{H}^k = \sigma([\boldsymbol{A}\boldsymbol{H}'^{k}_{0:i} \,||\, \boldsymbol{I}\boldsymbol{H}'^{k}_{i:}] + \boldsymbol{b}^k)$

where $\boldsymbol{H}'^k = \boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$

Providing a soft middle ground between full exchange of information and no communication

(Smith et al., 2019)

Graph Isomorphism Networks (GIN)

Key idea: maps different node neighborhoods to different feature vectors to obtain high discriminative/representational power.

Update definied as: $\boldsymbol{H}_u^{(k)} = \mathrm{MLP}^{(k)}((1+\epsilon^{(k)})\boldsymbol{H}_u^{(k-1)} + \sum_{v\in\mathcal{N}(u)} \boldsymbol{H}_v^{(k-1)})$

Learnable parameter.

(Xu et al., 2019)

RNN-based updates

Key idea: GNN updates formulated in terms of RNN-based updates, where
- the RNN hidden state h_t becomes the node's hidden state
- the RNN observation (input) x_t becomes the aggregated message

(Li et al., 2016 and Selsam et al., 2018)

$$\boldsymbol{H}_u^{(k)} = RNN^{(k)}(\boldsymbol{H}_u^{(k-1)}, AGGREGATE^{(k)}(\{\boldsymbol{H}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\}))$$

- RNN update function – eg LSTM/GRU update
- Parameters are shared across nodes (same RNN cell used to update each node)
- Update function shared across (message passing) layers

Notes

- All GNNs presented thus far iteratively update node features that can be used for any downstream task.
- In the case of node classification:
  - the non-linearity σ used in the last update (layer) is the softmax.
  - the loss function to train the model is analogous to the semantic segmentation one: per node cross-entropy loss.
- In the case of graph classification,
  - a pooling operation is often used to aggregate the information across all nodes.
  - the pooled features are fed to a classifier.
  - the loss function to train the model is analogous to the one of image classification: per graph cross-entropy loss.
- Most of the presented models can be augmented to handle multi-relational graphs, by separately aggregating information across different edge types.

Computer Vision Applications

3D vision (1)

Key idea: Exploit mesh representations in the context of 3D visual understanding.

Why meshes?

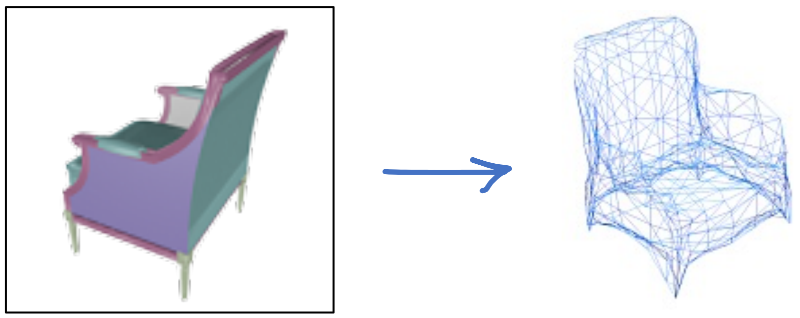
(a) Voxels (262, 144 units) (b) Point cloud (30, 000 points) (c) Uniform mesh (2416 vertices) (d) Adaptive mesh (120 vertices)

GNNs for mesh reconstructions: see e.g. Wang et al. 2018, Smith et al. 2019, Gkioxari et al. 2019, Smith et al. 2021.

Image from https://arxiv.org/pdf/1901.11461.pdf

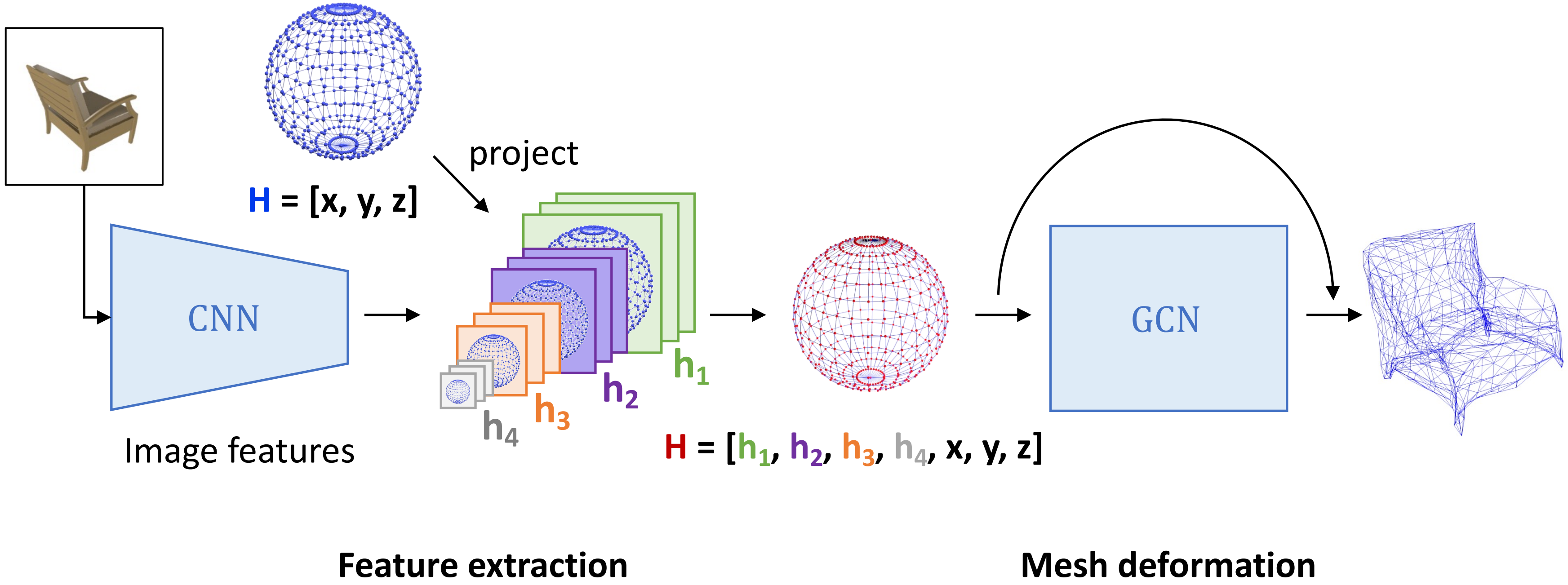
3D vision (2)

Often posed as shape reconstruction from a **single view** image:



Goal: Inferring 3D shape from a single view image of an object.

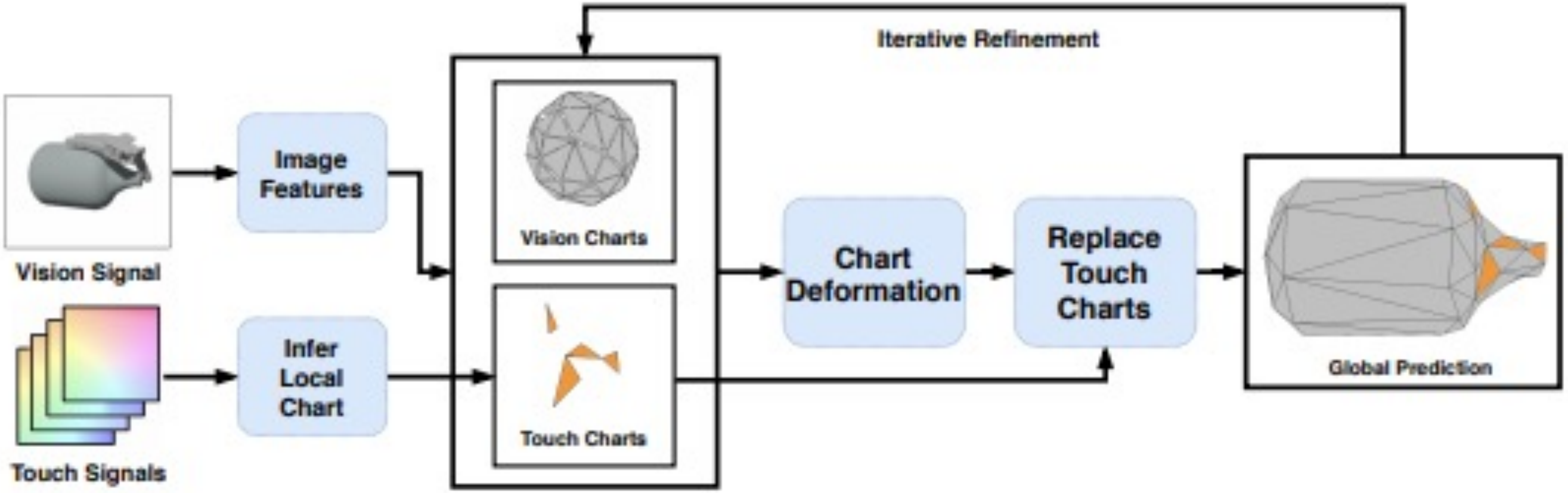
3D vision (3)



Trained to minimize the Chamfer loss: $\sum_{p \in S} \min_{q \in \hat{S}} \|p - q\|_2^2 + \sum_{q \in \hat{S}} \min_{p \in S} \|p - q\|_2^2$

Figure credit: Edward Smith

3D vision (4)

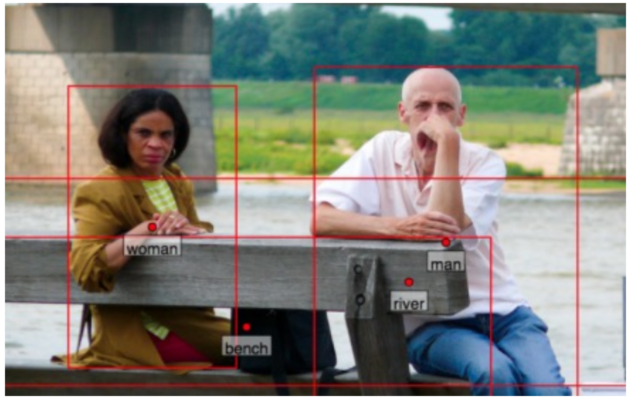


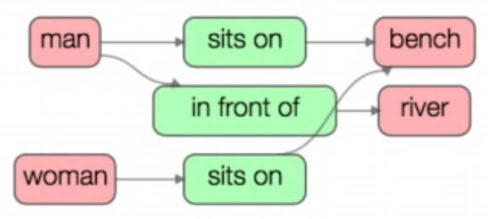
Goal: Inferring 3D shape from a single view image of an object and tactile information.

(Smith et al., 2020)

Relational representations

Key idea: Represent the visual world relations (e.g. positional, conceptual or from attributes) by means of graph.





Tasks: Infer relations (graph) from images or videos, exploit relations to improve model generalization, and generate images from graphs, among others.

(Xu et al. 2017, Johnson et al. 2018, Ashual and Wolf 2019, Casanova et al. 2020)

Wrap Up

Wrap Up

Problem formulation
- What are graphs?
- How do we represent them?
- What kind of problems can we tackle?

Node embeddings
- Per-node classifier (ignoring graph structure)
- Indirect injection of structural information

Graph Neural Networks
- Neural Message Passing
    - Basic GNN
    - GCN
    - GAT

Different aggregate functions.
    - Skip connections
    - 0-neighbor updates
    - GIN
    - RNN updates

Different update functions.
- How to train those models?
- Multi-relational graphs

Computer Vision Applications
- 3D Vision
- Relational representations
    - Inferring visual knowledge graph from data
    - Generating visual content from scene graphs

References

(Weston et al., 2008) J. Weston, F. Ratle & R. Collobert; **Deep Learning via Semi-Supervised Embedding**, ICML, 2008.
(Perozzi et al., 2014) B. Perozzi, R. Al-Rfou & S. Skiena; **DeepWalk: Online Learning of Social Representations**, KDD, 2014.
(Tang et al., 2015) J. Tang, M. Qu, M. Wang, J. Yan & Q. Mei; **LINE: Large-scale Information Network Embedding**, WWW, 2015.
(Grover et al., 2016) A. Grover & J. Leskovek; **node2vec: Scalable Feature Learning for Networks**, KDD, 2016.
(Yang et al., 2016) Z. Yang, W. W. Cohen & R. Salakhutdinov; **Revisiting Semi-Supervised Learning with Graph Embeddings**, ICML, 2016.
(Gori et al., 2005) M. Gori, G. Monfardini & F., Scarselli; **A new model for learning in graph domains**, IJCNN, 2005.
(Scarselli et al., 2009) F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner & G. Monfardini; **The Graph Neural Network Model**, TNNLS, 2009.
(Li et al., 2016) Y. Li, D. Tarlow, M. Brockschmidt & R. Zemmel; **Gated Graph Neural Networks**, ICLR, 2016.
(Bruna et al., 2014) J. Bruna, W. Zaremba, A. Szlam & Y. LeCun; **Spectral Networks and Locally Connected Networks on Graphs**, ICLR, 2014.
(Kipf & Welling, 2017) T. Kipft and M. Welling; **Semi-Supervised Classification with Graph Convolutional Networks**, ICLR, 2017.
(Belkin and Niyogi, 2002) M. Belkin and P. Niyogi. **Laplacian eigenmaps and spectral techniques for embedding and clustering**, NIPS, 2002

(Hamilton et al., 2017) W. L. Hamilton, R. Ying & J. Leskovec; **Inductive Representation Learning on Large Graphs**, NIPS, 2017.
(Bahdanau et al., 2015) D. Bahdanau, K. Cho & Y. Bengio; **Neural Machine Translation by Jointly Learning to Align and Translate**, ICLR, 2015.
(Vaswani et al., 2017) A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser & I. Polosukhin; **Attention is all you need**, NIPS, 2017.
(Velickovic et al., 2018) P. Velickovic, G. Cucurull, A. Casanova, A. Romero. P. Lio & Y. Bengio; **Graph Attention Networks**, ICLR, 2018.
(Qi et al., 2017) C.R. Qi, H. Su, K. Mo, and L .J. Guibas; **Pointnet: Deep learning on point sets for 3d classification and segmentation**, In CVPR, 2017.
(Zaheer et al., 2017) M. Zaheer, S. Kottur, S.. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. J. Smola; **Deep sets**, NIPS, 2017.
(Pham et al., 2017) T. Pham, T. Tran, D. Phung, and S. Venkatesh; **Column networks for collective classification**, AAAI, 2017.
(Xu et al, 2018) K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka; **Representation learning on graphs with jumping knowledge networks**, arXiv preprint arXiv:1806.03536, 2018.
(Defferrard et al., 2016) M. Defferrard, X. Bresson & P. Vandergheynst; **Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering**, NIPS, 2016.
(Duvenaud et al., 2015) D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik & R. P. Adams; **Convolutional Networks on Graphs for Learning Molecular Fingerprints**, NIPS, 2016.

(Smith et al., 2019) E. J. Smith, S. Fujimoto, A. Romero, and D. Meger; **GEOMetrics: Exploiting Geometric Structure for Graph-Encoded Objects,** ICML, 2019.
(Selsam et al., 2018) D. Selsam, M. Lamm, B. Bunz, P. Liang, L. de Moura, and D. L. Dill; **Learning a sat solver from single-bit supervision,** arXiv preprint arXiv:1802.03685, 2018.
(Wang et al., 2018) N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang; **Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images**, ECCV, 2018.
(Gkioxari et al., 2019) G. Gkioxari, J. Malik, and J. Johnson; **Mesh R-CNN**, ICCV, 2019.
(Ashual and Wolf, 2019) O. Ashual and Lior Wolf; **Specifying Object Attributes and Relations in Interactive Scene Generation**, ICCV, 2019.
(Johnson et al., 2018) J. Johnson, A. Gupta, and L. Fei-Fei; **Image Generation from Scene Graphs**, CVPR, 2018.
(Xu et al., 2017) D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei; **Scene Graph Generation by Iterative Message Passing**, CVPR, 2017.
(Smith et al., 2020) E. Smith, R. Calandra, A. Romero, G. Gkioxari, D. Meger, J. Malik, M. Drozdzal; **3D Shape Reconstruction from Vision and Touch**; NeurIPS, 2020.
(Casanova et al., 2020) A. Casanova, M. Drozdzal, A. Romero-Soriano; **Generating unseen complex scenes: are we there yet?**; arXiv preprint arXiv:2012.04027, 2020.
(Ma et al., 2021) L. Ma, R. Rabbany, and A. Romero-Soriano; **Graph Attention Networks with Positional Embeddings**, PAKDD, 2021.
(Xu et al., 2019) K. Xu, W. Hu, J. Leskovec, and S. Jegelka; **How powerful are graph neural networks?**, ICLR, 2019.

Neural networks for graph-structured data

Master in Computer Vision Barcelona
Adriana Romero
adrianars@fb.com