Graph Machine Learning

Artificial Intelligence Summer Camp - 2025

Inteligencia Artificial PUCP (IA-PUCP)

January 14, 2025

Graphs

Graph is a data structure and is defined by $G = (V, E)$, where V is the set of nodes and E is the set of edges.

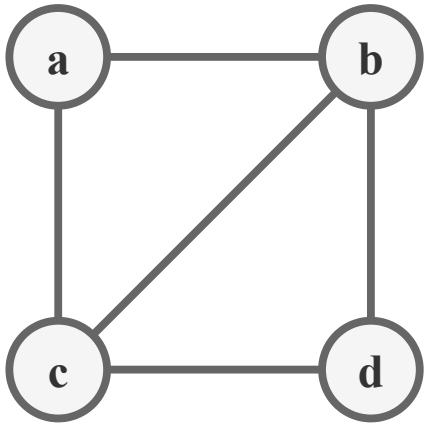
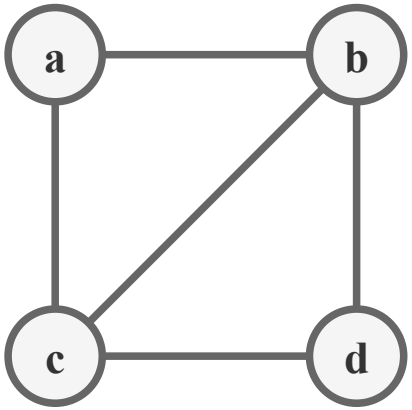


Figure 1: Graph example.

Graph representations



(a) Input graph.

	a	b	c	d
a	0	1	1	0
b	1	0	1	1
c	1	1	0	1
d	0	1	1	0

(b) Adjacency matrix.

a	b	c	
b	a	c	d
c	a	b	d
d	b	c	

(c) Adjacency list.

a	b
a	c
b	c
b	d
c	d

(d) Edge list.

Figure 2: Graph representations.

Other definitions/concepts

- Homogeneous graph
- Heterogeneous graph
- Directed graph
- Undirected graph
- Weighted graph
- Bipartite graph
- Complete graph
- Degree
- Density
- Diameter

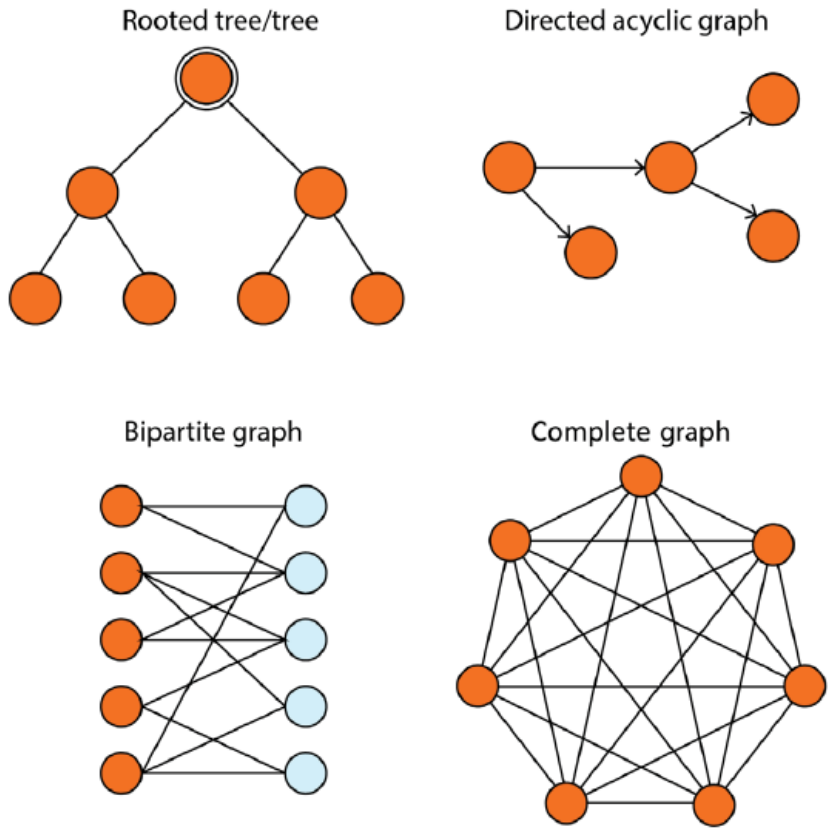
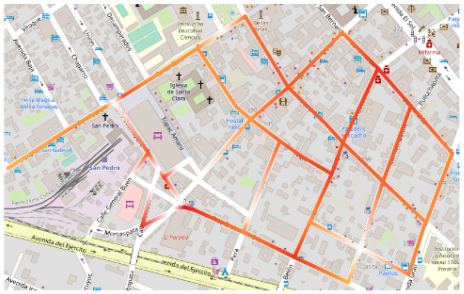


Figure 3: Common type of graphs. Adapted from [1].

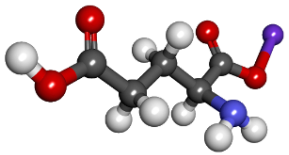
Graphs are everywhere!

Social networks

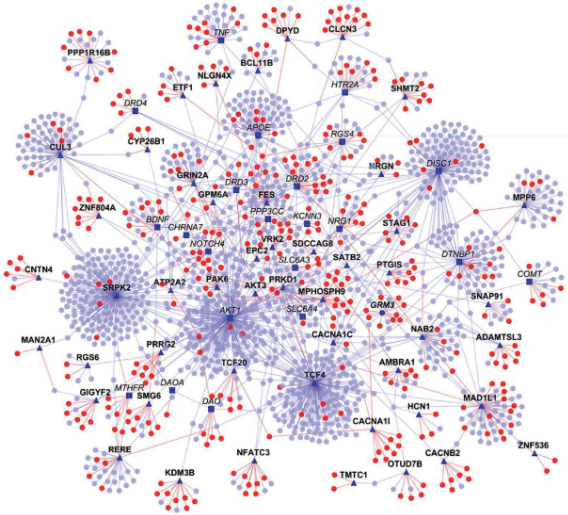


Road map

Pose estimation



Chemical compound



Protein interaction networks

Figure 4: Graphs are everywhere.

Graph embedding

It is a function $f : A \rightarrow Z$ that maps each node of the graph $G = (V, E)$ into a d-dimensional vector. Where, $Z = \{z_1, z_2, ..., z_N\}$, $z_i \in \mathbb{R}^d$ with $d \ll N$.

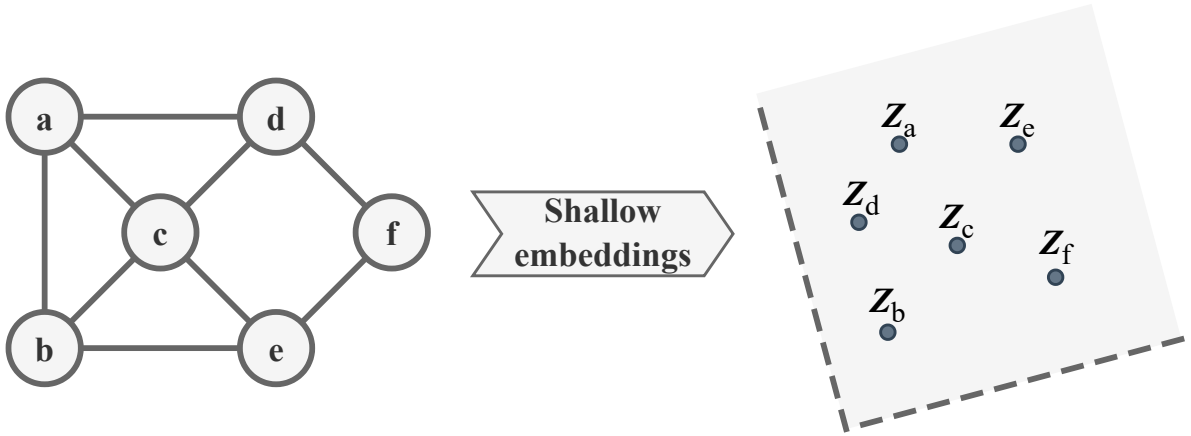


Figure 5: Shallow embeddings scheme.

More about embeddings

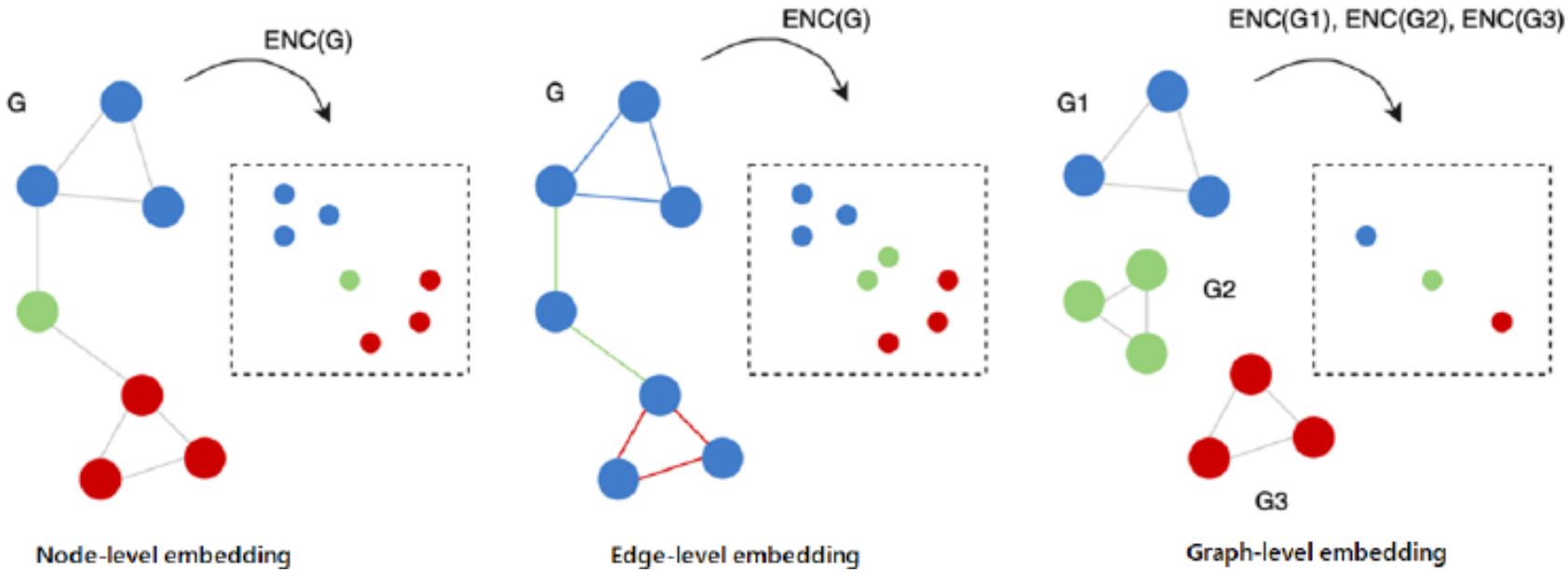


Figure 6: Visual representation of the three different levels of granularity in graphs. Adapted from [2].

Tasks

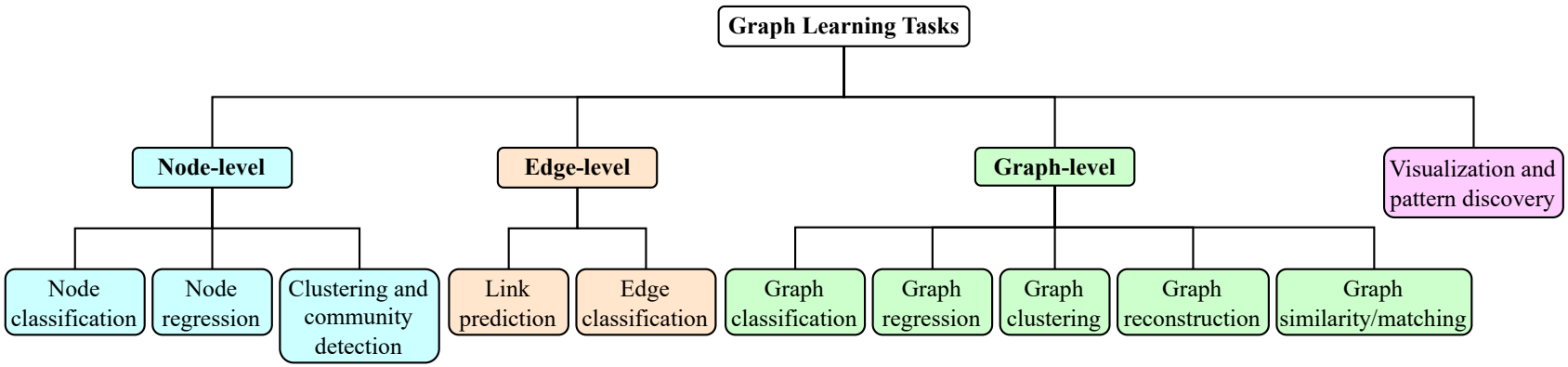


Figure 7: Overview of graph learning tasks.

Tasks

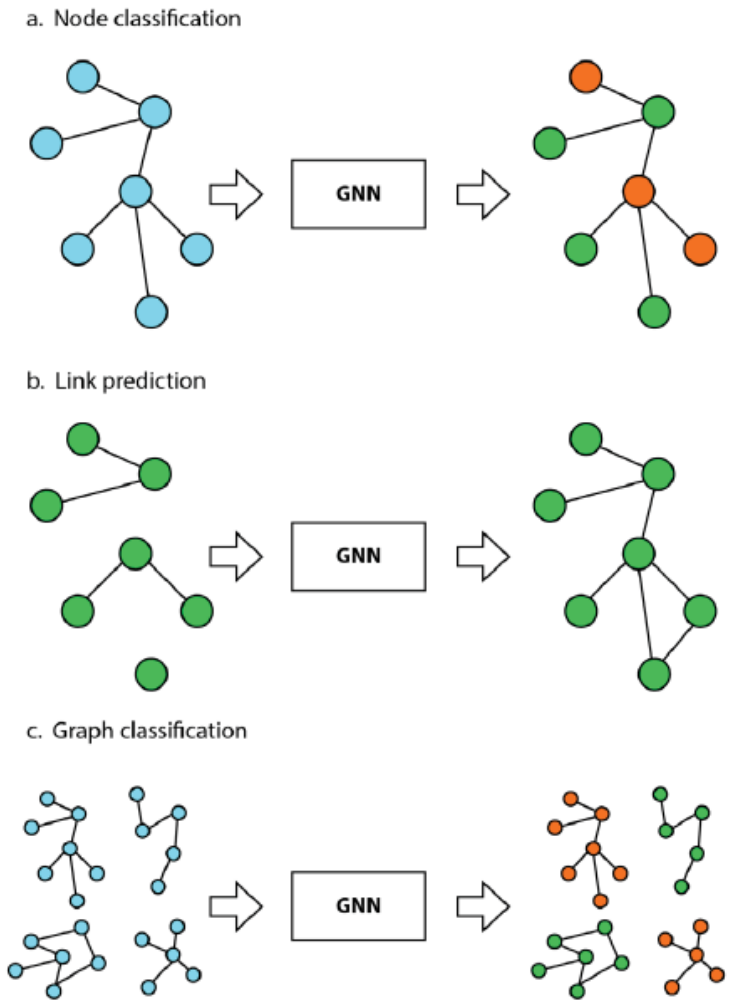


Figure 8: Examples of graph learning tasks. Adapted from [1].

Applications

- Computer vision
- Natural language processing
- Recommender systems
- Program analysis
- Knowledge graphs
- Bioinformatics
- Chemistry
- Biology
- Physic
- Anomaly detection
- Urban intelligence
- Financial transaction
- ...

Taxonomy of graph machine learning methods.

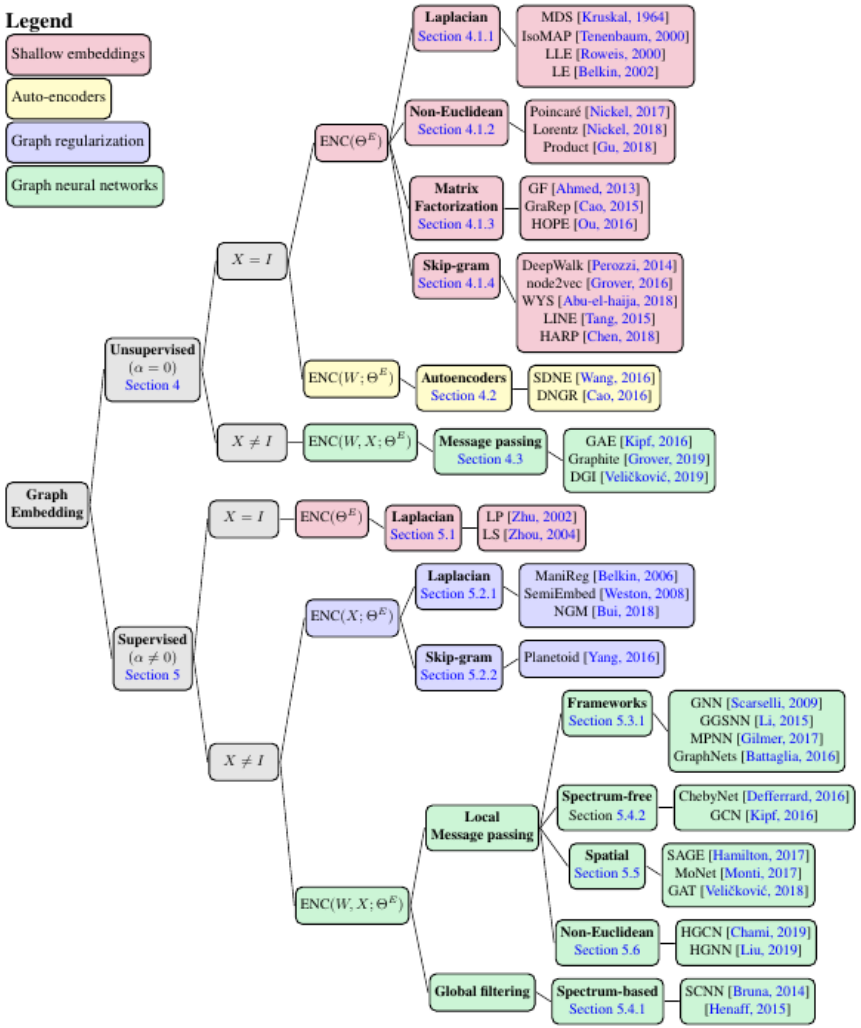
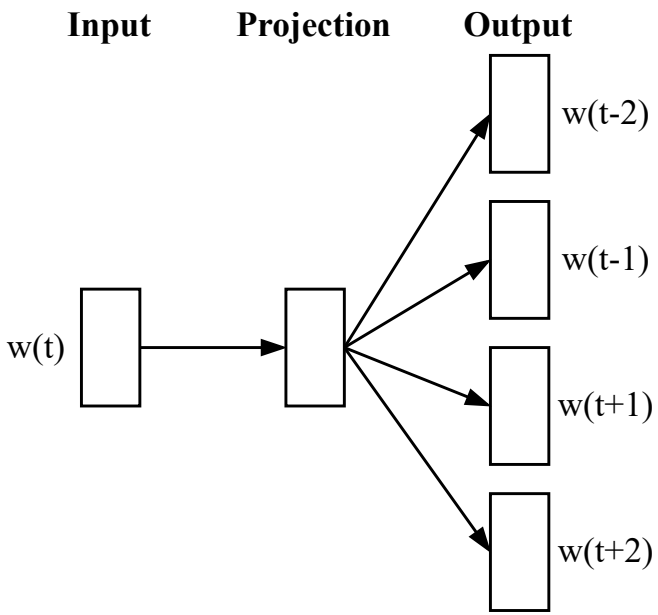
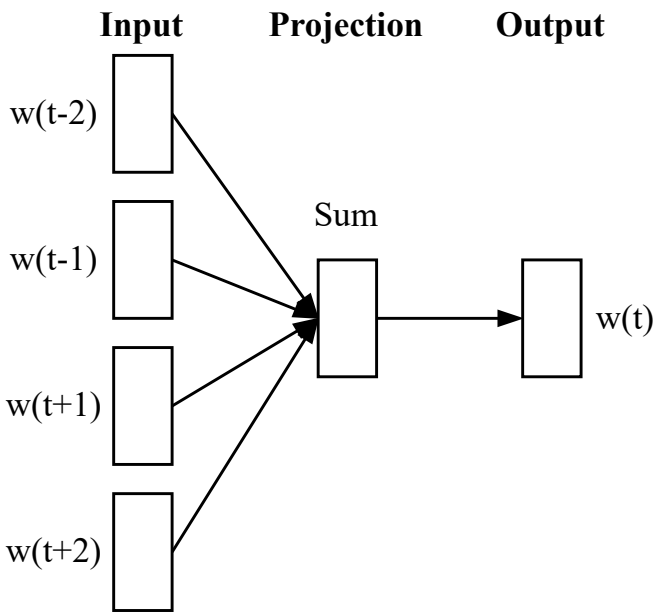


Figure 9: Taxonomy of graph machine learning. Adapted from [3].

Word2vec [4]



(a) Skip-gram.



(b) CBOW.

Figure 10: Word embedding architectures. Adapted from [4].

Word2vec

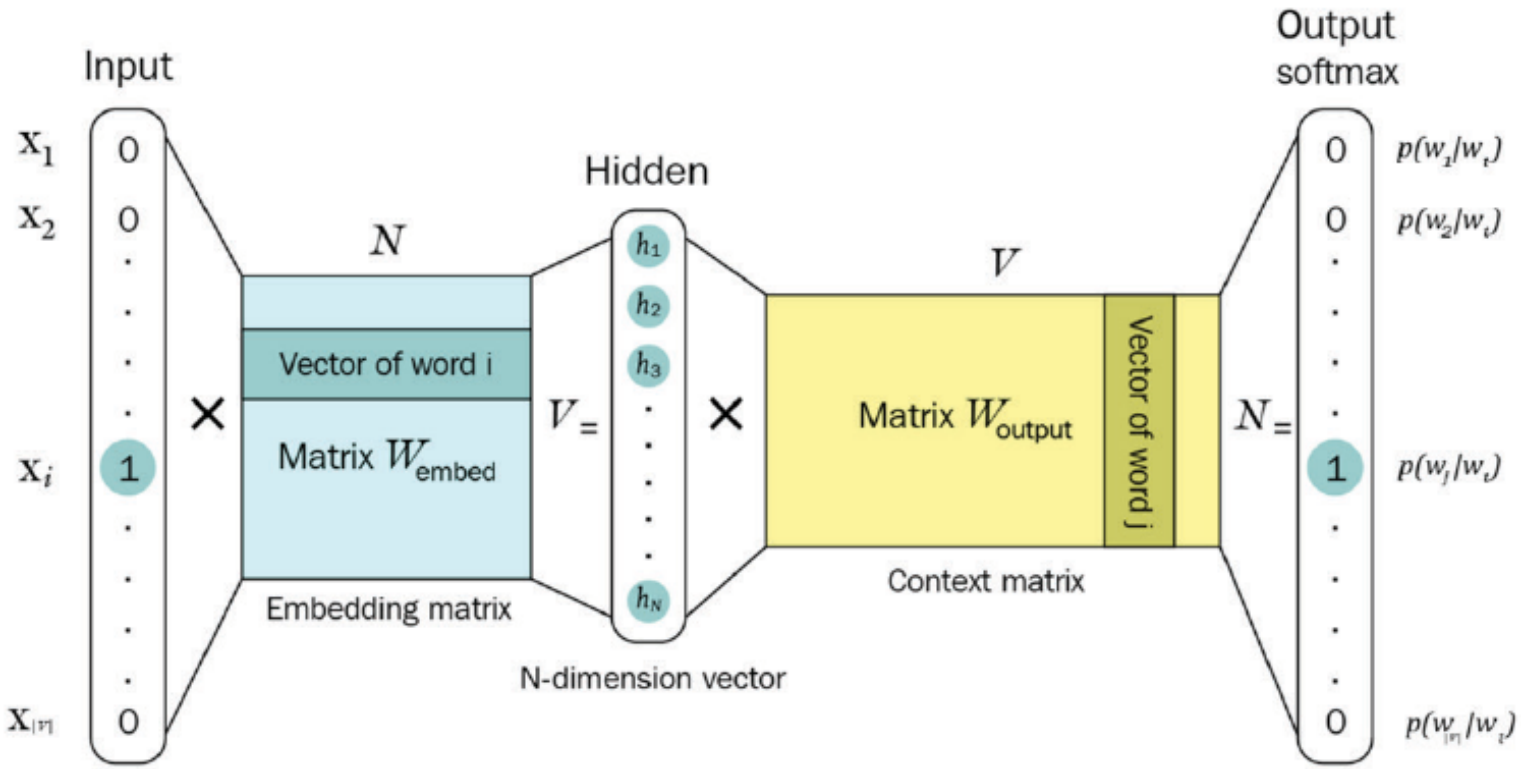


Figure 11: Word2vec architecture. Adapted from [1].

Word2vec in action

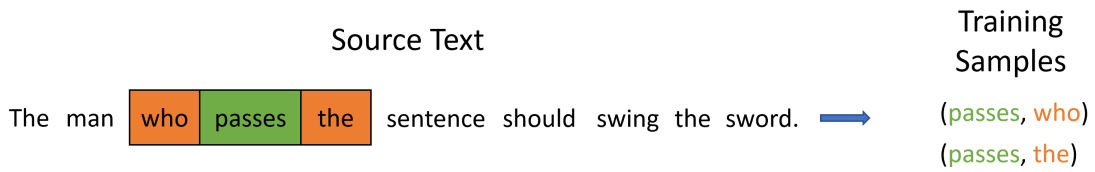


Figure 12: Train window.

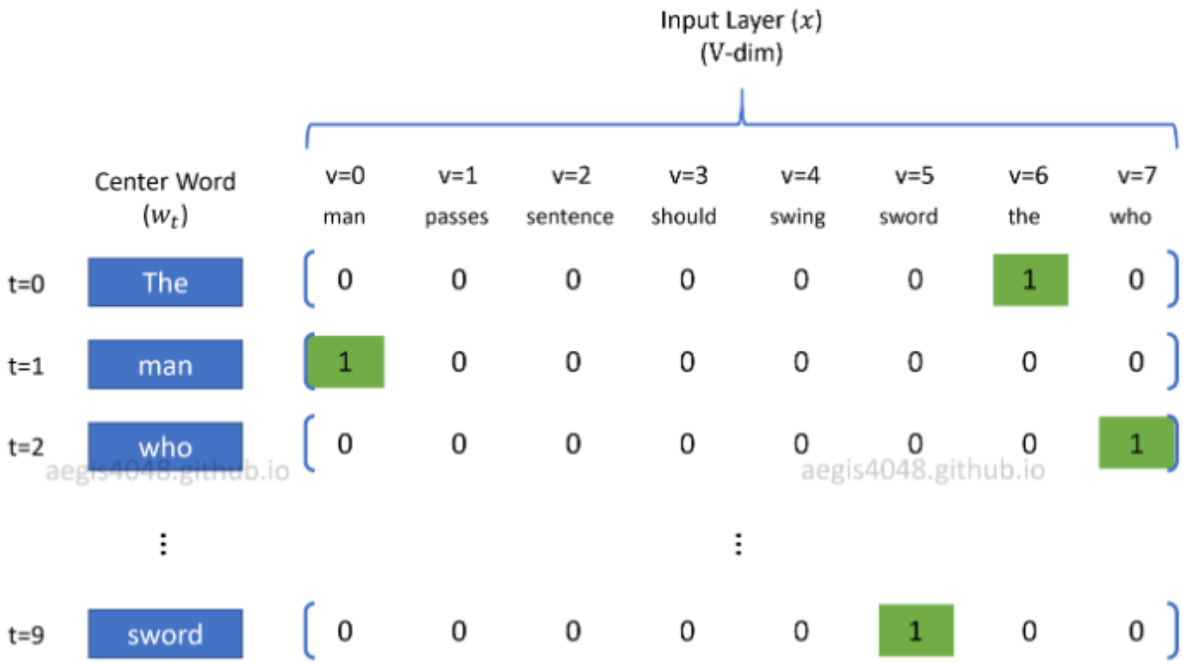


Figure 13: One-hot encoded input vector.

Word2vec in action

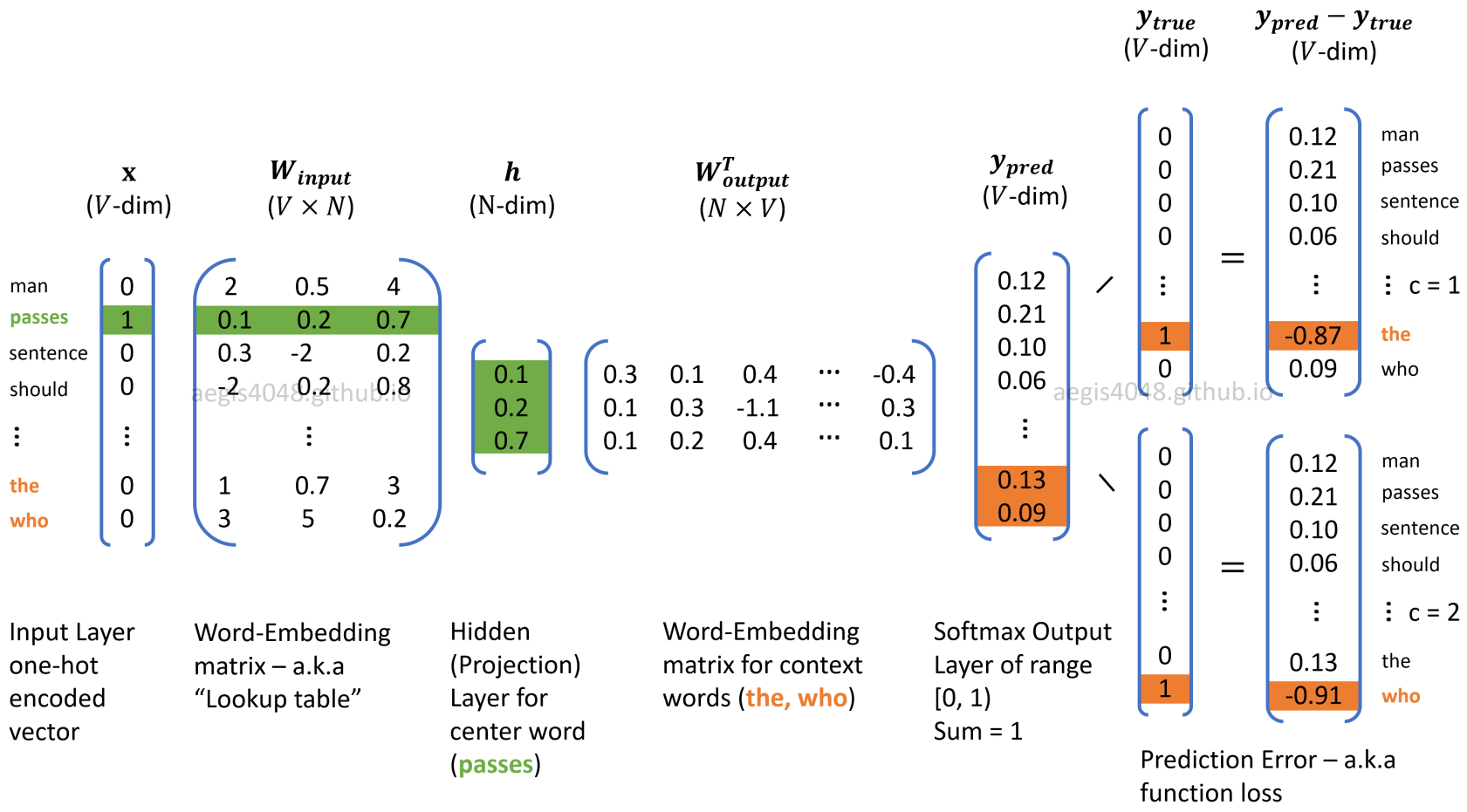


Figure 14: Skip-Gram model structure. Current center word is "passes". From `https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling`.

Word2vec in action

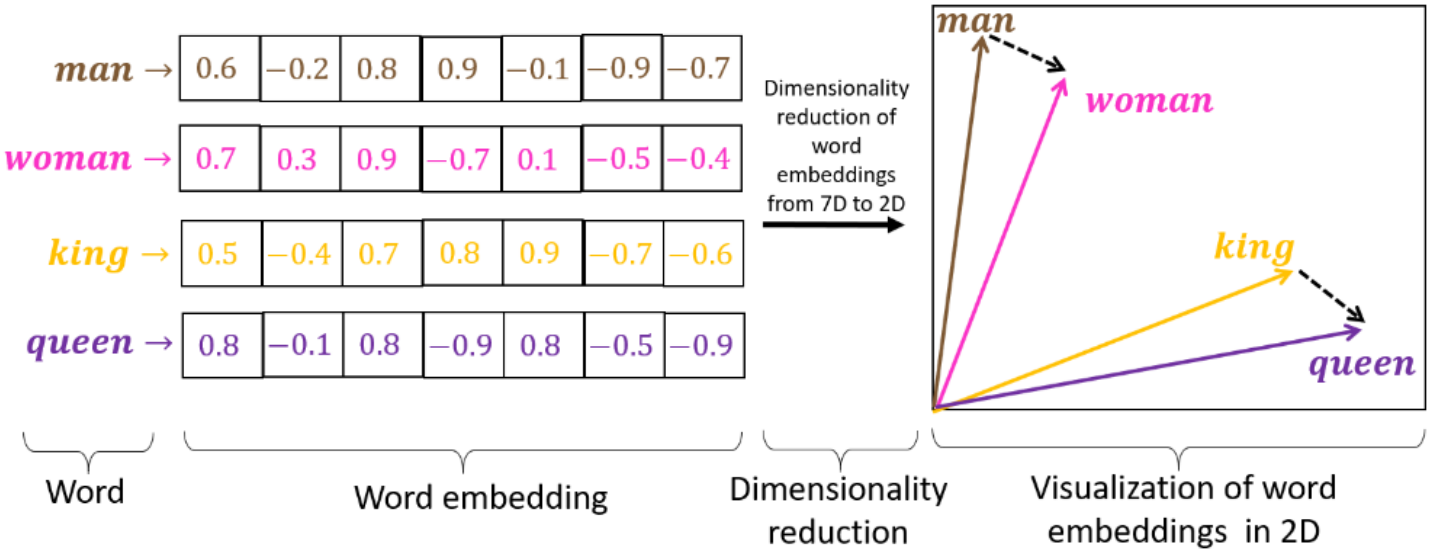


Figure 15: Word embeddings.
From `https://doi.org/10.1371/journal.pone.0231189.g008`.

Cosine similarity

$$cos(\theta) = \text{sim}(z_i, z_j) = \frac{z_i . z_j}{\|z_i\| \|z_j\|} \quad (1)$$

DeepWalk [5]

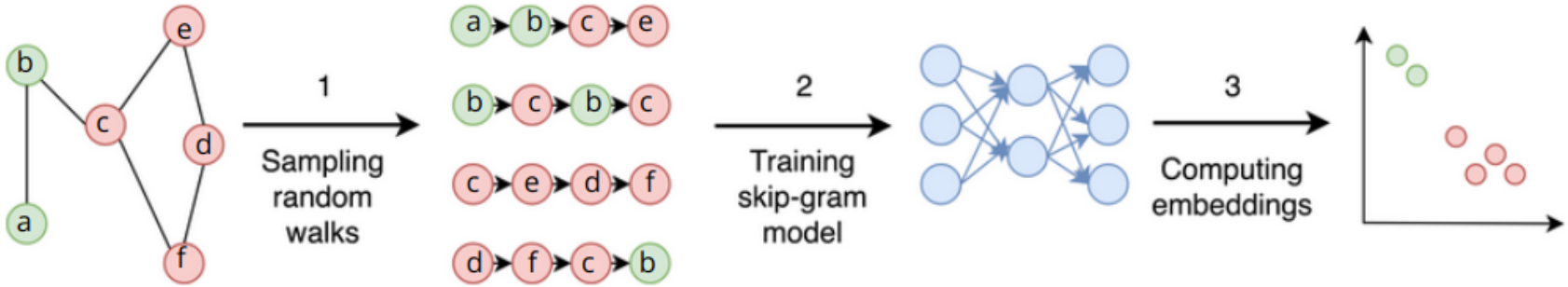


Figure 16: DeepWalk workflow.

Parameters:

- length $= 4$
- n $= 1$

Node2vec [6]

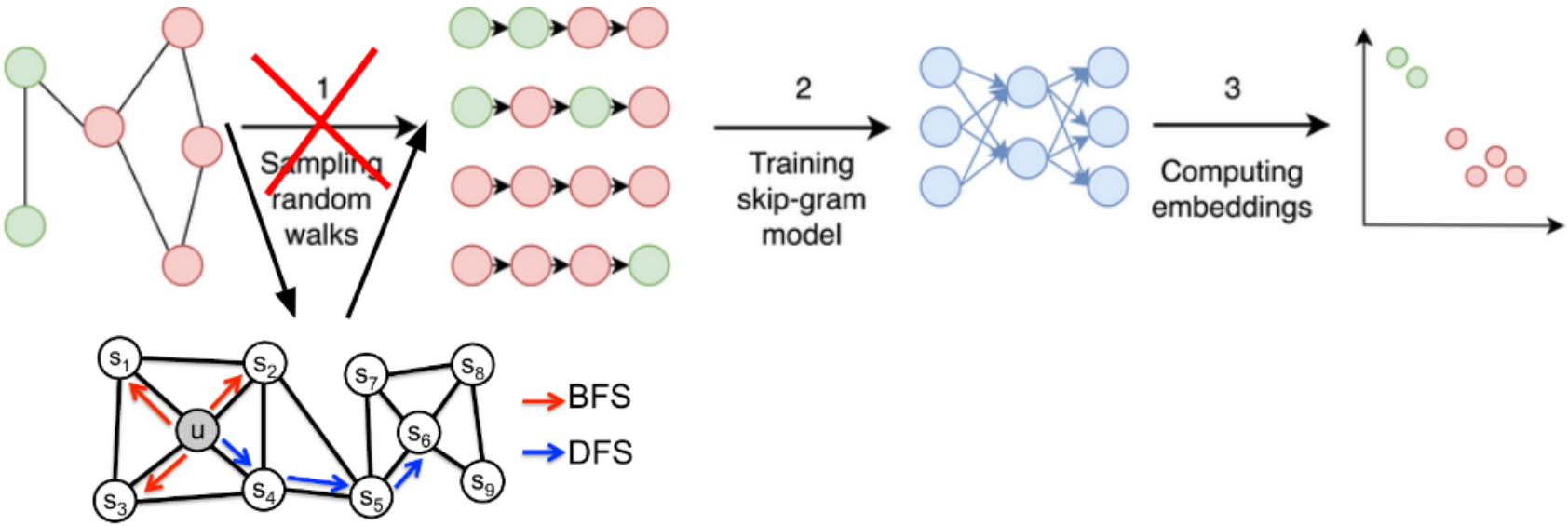


Figure 17: Node2vec workflow.

Parameters:

- length = 4
- n = 1
- p = 1, q = 2 (BFS: micro-view) or
- p = 1, q = 0.5 (DFS: macro-view)

Application on biomedical data

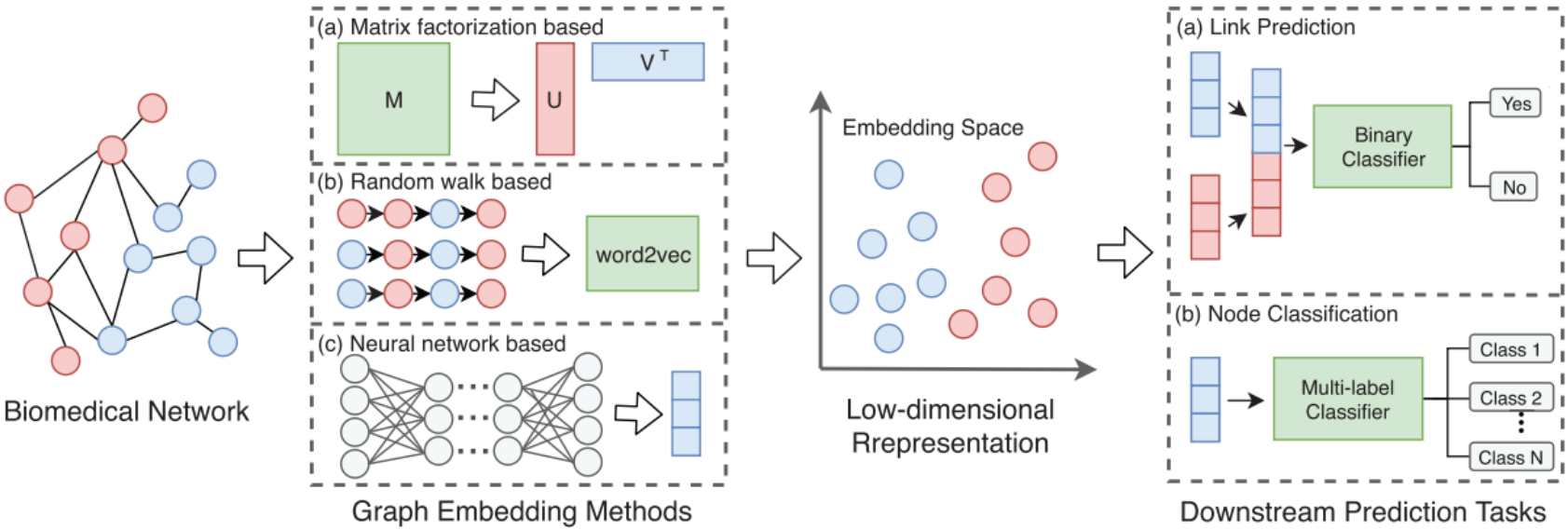


Figure 18: Applying graph embedding methods to biomedical tasks. Adapted from [7].

Practical example: Node2vec

https://github.com/win7/Presentations-Hub/blob/main/SummerCamp-2025_IA-PUCP/Community_Detection.ipynb

Graph Neural Network (GNN)

$$\text{GNN} = \text{Neural networks} + \text{Graph theory}$$

Basics of Neural networks (NN)

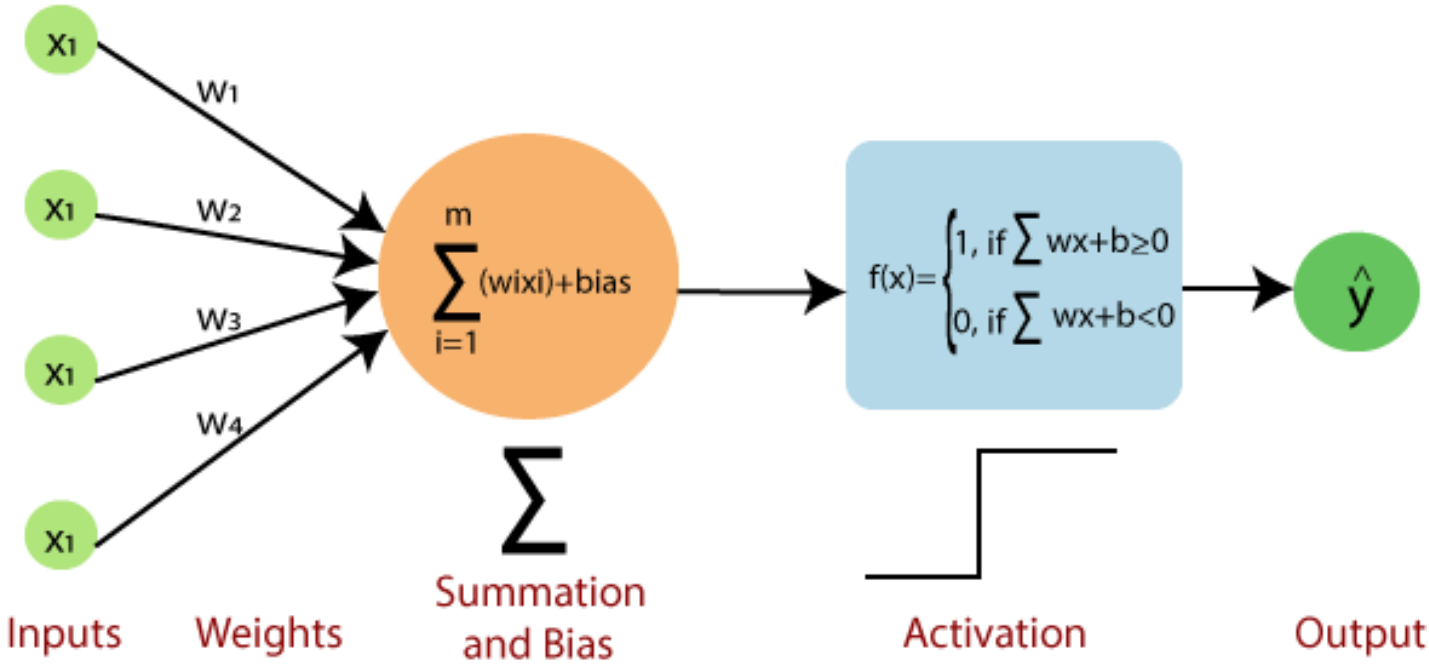


Figure 19: Basic NN.
From `https://www.javatpoint.com/single-layer-perceptron-in-tensorflow`.

Neural networks (NN)

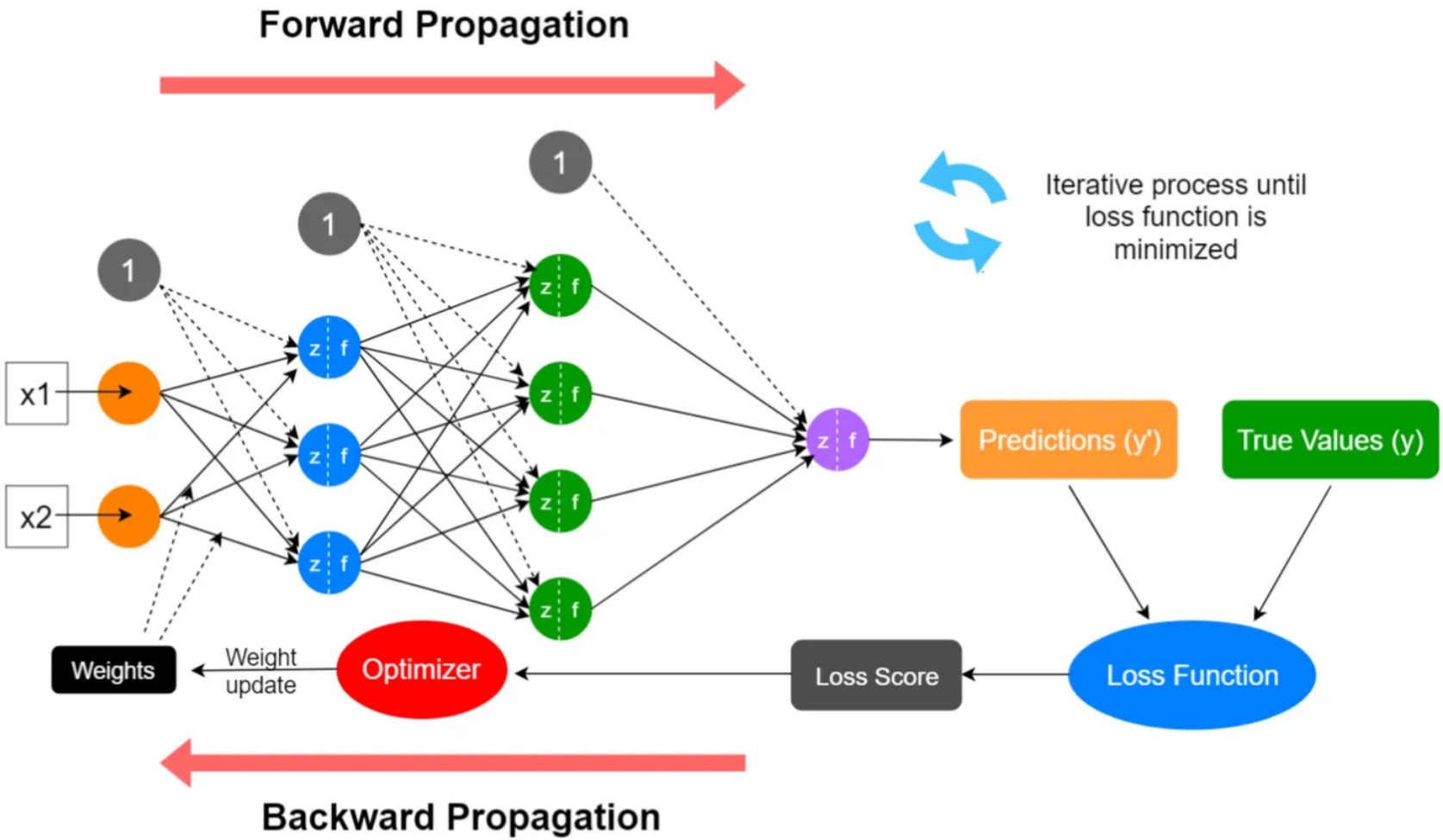


Figure 20: Basic NN. From `https://medium.com/data-science-365/overview-of-a-neural-networks-learning-process-61`.

Example of NN

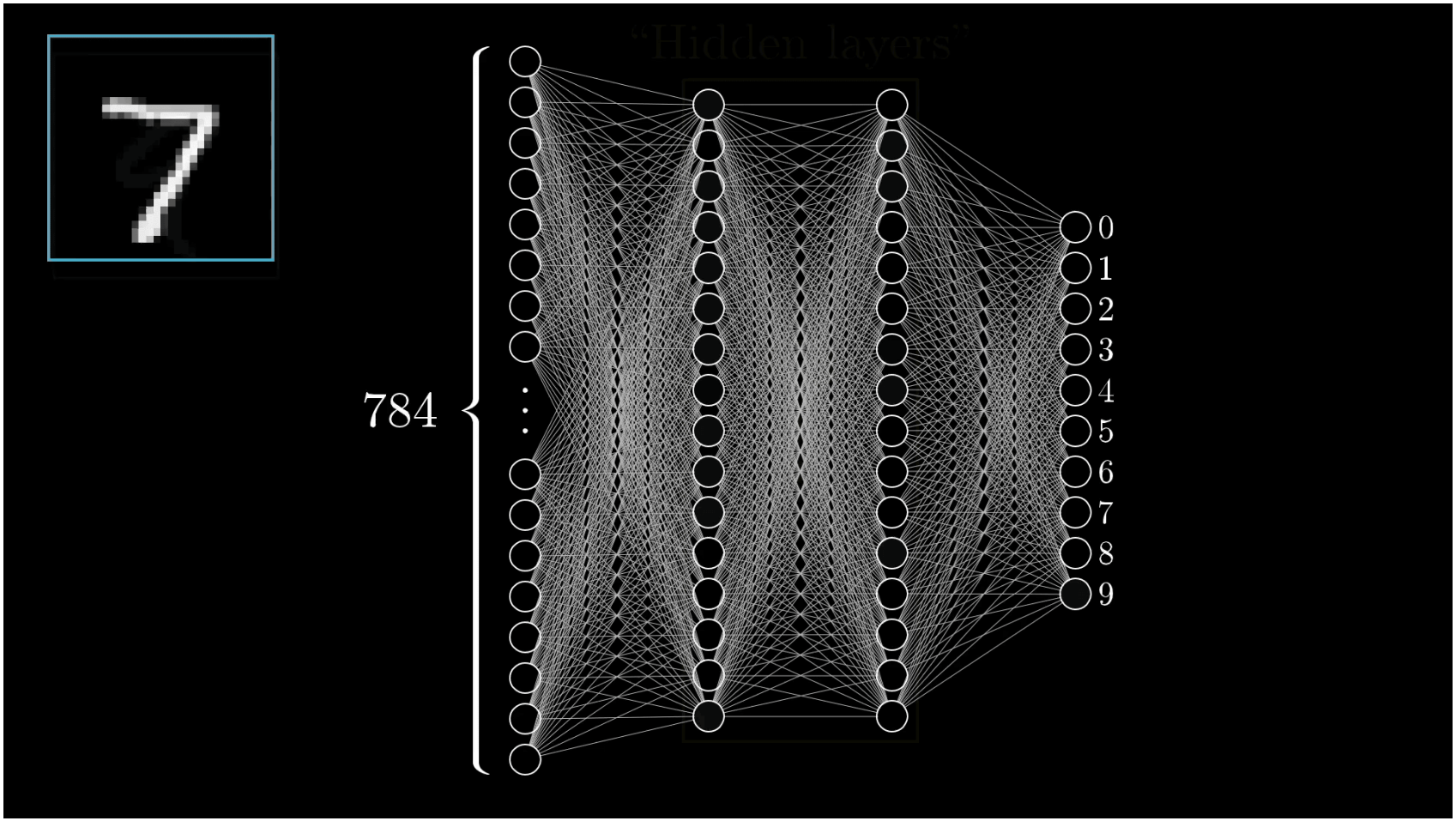


Figure 21: Image classification. From `https://www.3blue1brown.com/lessons/neural-networks`.

Graph Neural Networks (GNN)

It is a function $f: (A, X) \rightarrow H$ that maps each node of the graph $G = (V, E)$ into a F-dimensional vector.
Here, $A \in \mathbb{R}^{N \times N}$, $X \in \mathbb{R}^{N \times C}$, $H^l \in \mathbb{R}^{N \times F}$ with $F \ll N$, $l \in \{1, 2, ..., L\}$.

$$H = \text{GNN}(A, X)$$

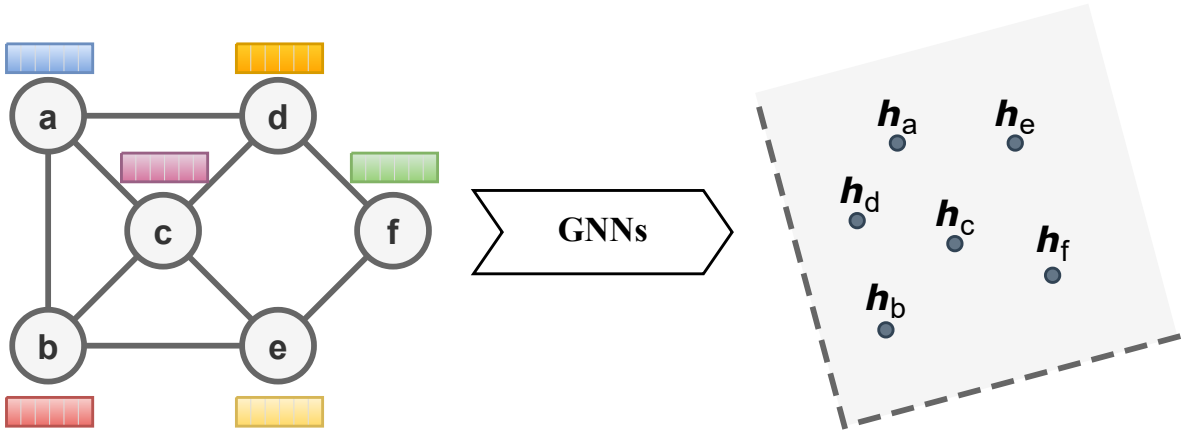


Figure 22: GNNs scheme.

How GNNs works?

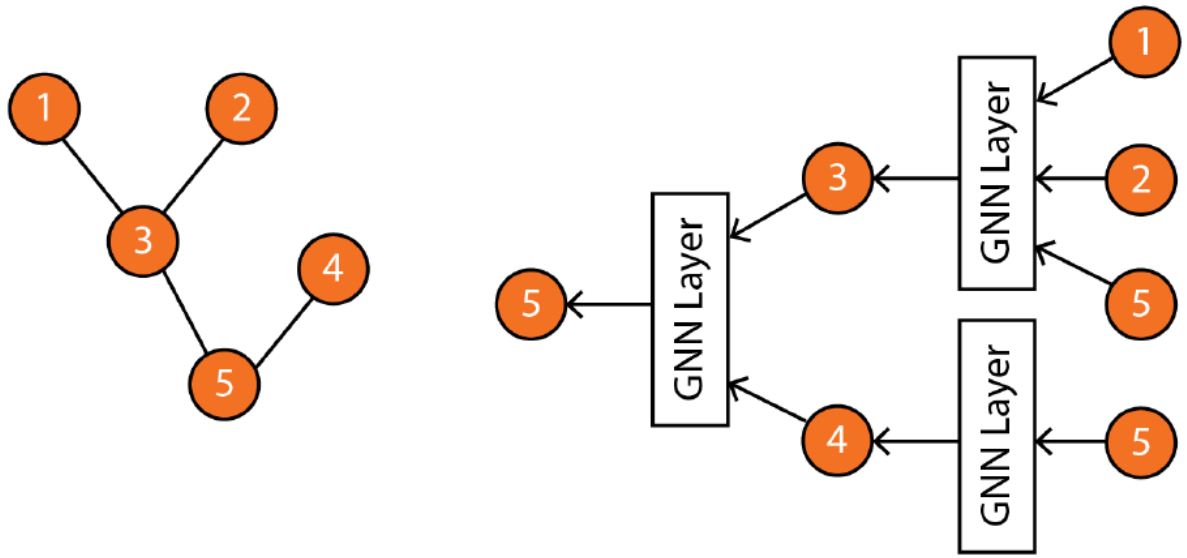


Figure 23: Computation graph representation. Adapted from [1].

Tasks

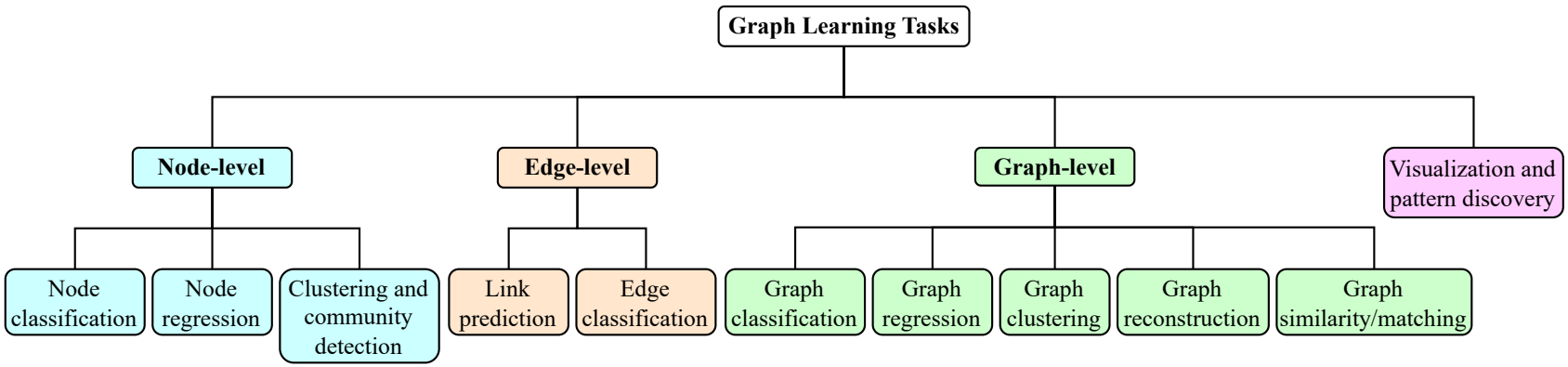


Figure 24: Overview of graph learning tasks.

Taxonomy of graph machine learning methods.

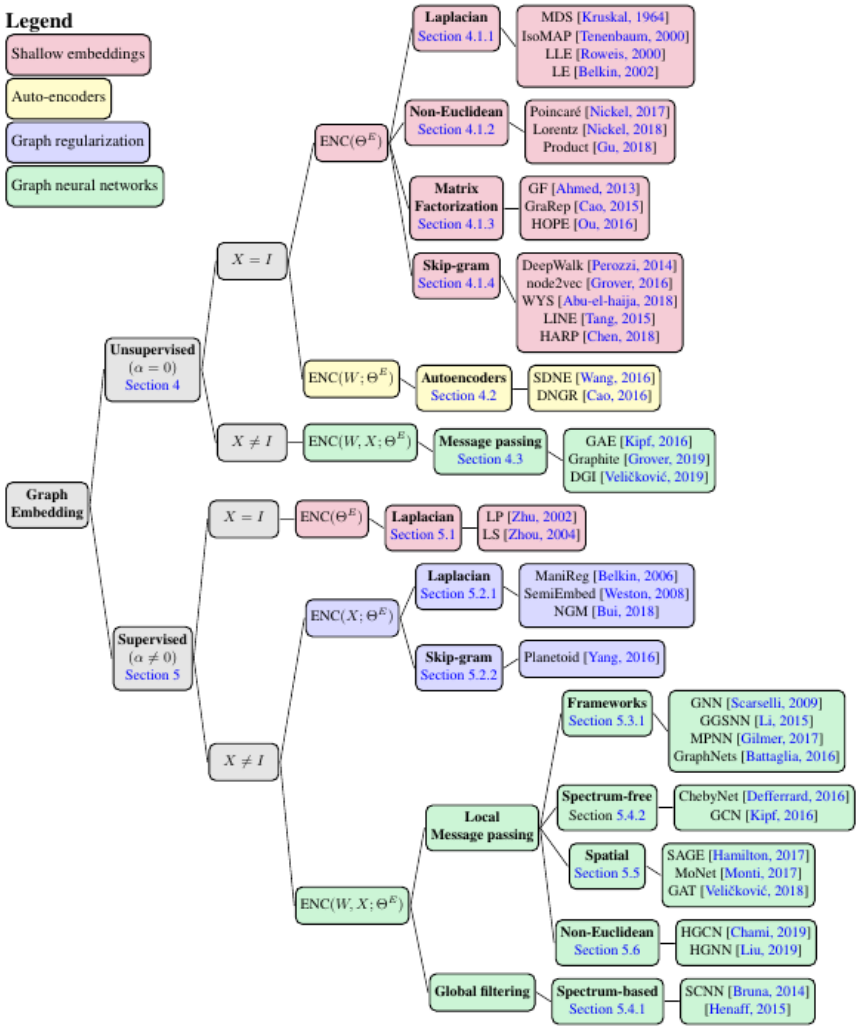


Figure 25: Taxonomy of graph machine learning. Adapted from [3].

Variants of GNNs

- Graph Convolutional Networks (GCN) [8]
- Graph Attention Networks (GAT) [9]
- Graph Isomorphism Networks (GIN) [10]
- Variational Graph Auto-Eencoders (VGAE) [11]
- Deep Graph Infomax (DGI) [12]
- Adversarially Regularized Variational Graph Autoencoder (ARVGA) [13]
- Linear Graph Variational Autoencoder (LGVAE) [14]
- ...

Graph Convolutional Network (GCN)

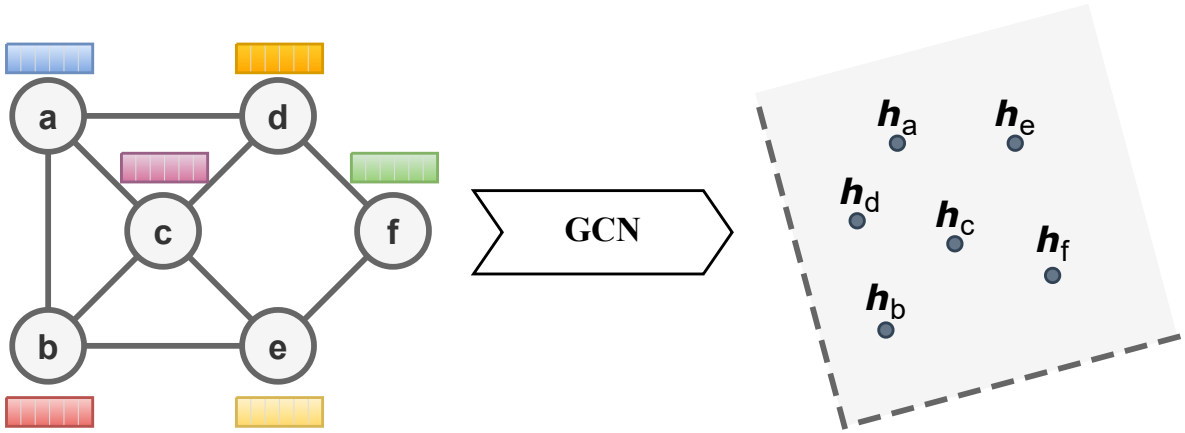


Figure 26: GCN scheme.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2}$$

Here, $\tilde{A} = A + I_N$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^{(l)}$ is trainable weight matrix, $\sigma(.)$ is an activation function.

Application in computer vision

SuperGlue: Learning Feature Matching with Graph Neural Networks [15].

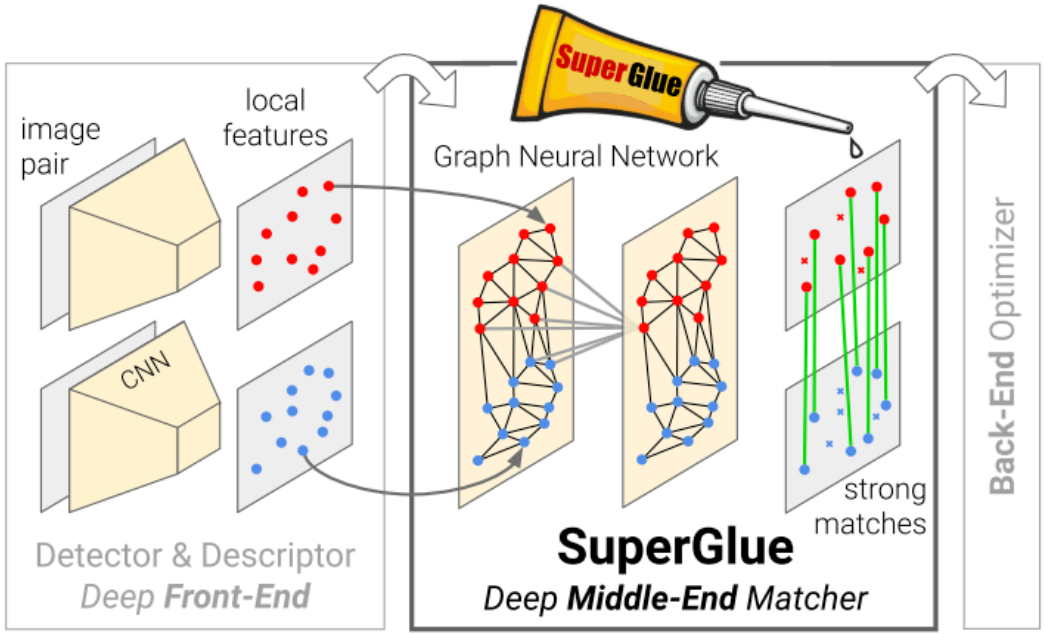


Figure 27: SuperGlue scheme.

SuperGlue

Figure 28: Matching example (indoor).

SuperGlue

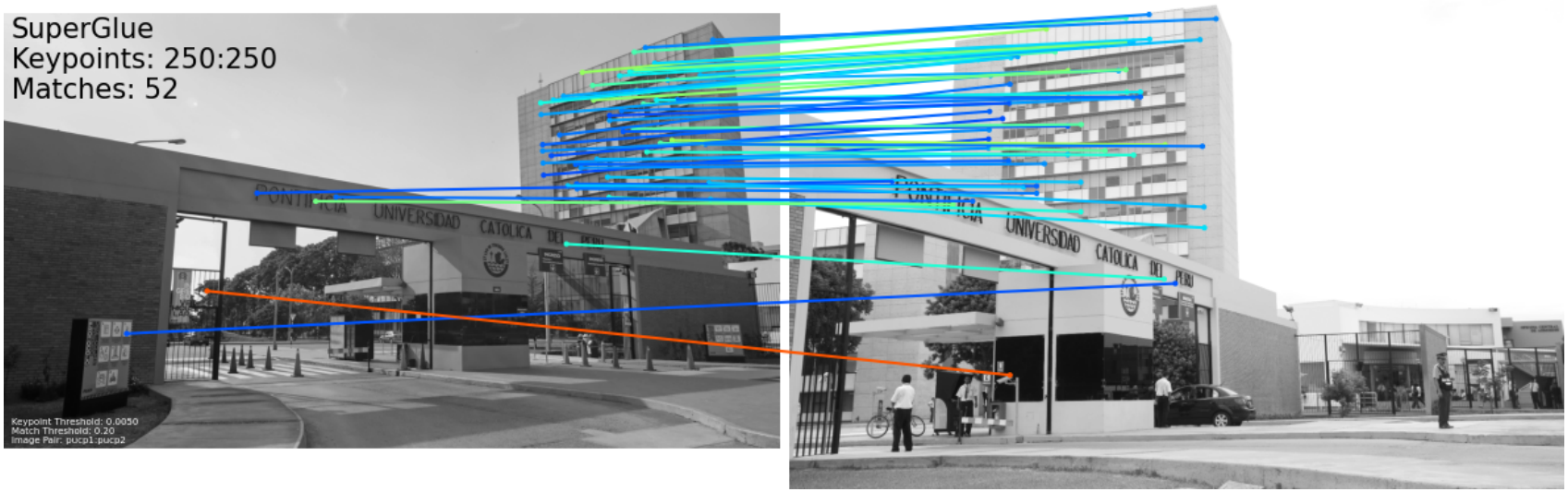


Figure 29: Matching example (outdoor).

Application in recommender systems

Uber Eats

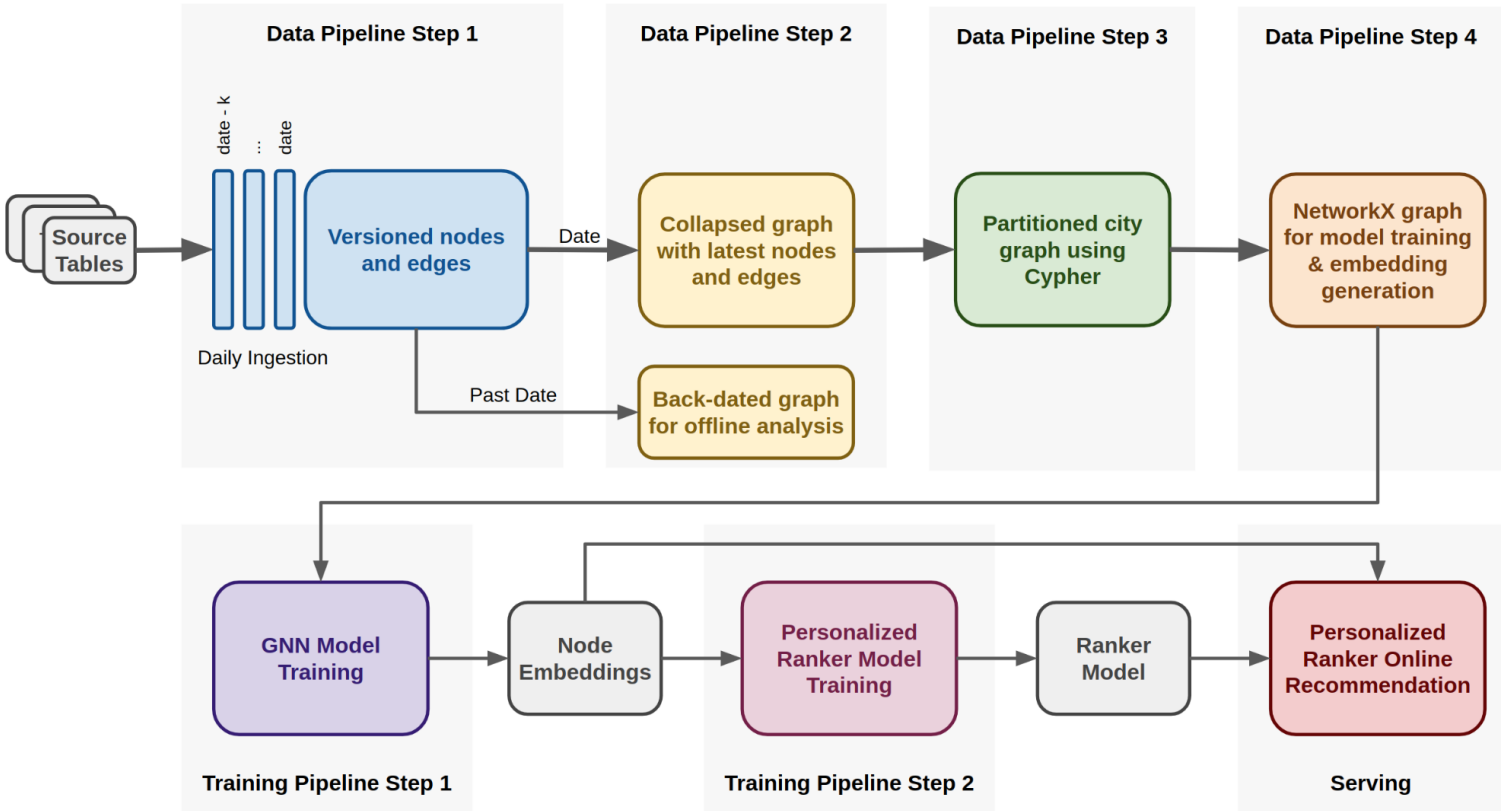


Figure 30: Pipeline Uber eats. From `https://www.uber.com/en-CO/blog/uber-eats-graph-learning/`.

Uber Eats

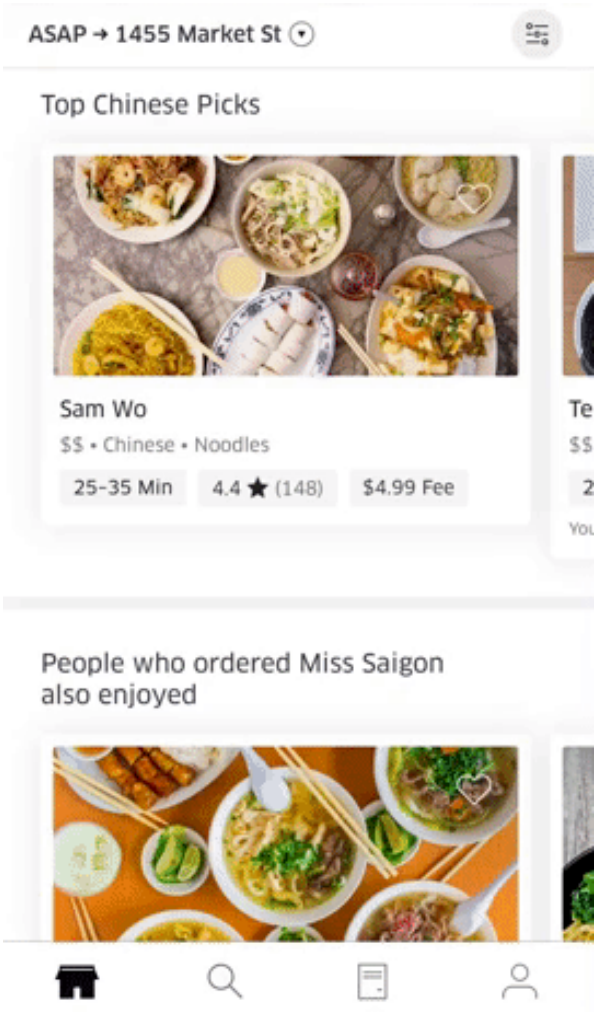


Figure 31: Recommender systems example.

Practical example: GCN to node classification

Dataset: Karate club

```
Number of graphs: 1
Number of nodes: 34
Number of edges: 156
Number of features: 34
Number of classes: 4
```

```
Data(x=[34, 34], edge_index=[2, 156], y=[34], train_mask=[34])
```

x=[34, 34]

```
tensor([[1., 0., 0.,  ..., 0., 0., 0.],
        [0., 1., 0.,  ..., 0., 0., 0.],
        [0., 0., 1.,  ..., 0., 0., 0.],
        ...,
        [0., 0., 0.,  ..., 1., 0., 0.],
        [0., 0., 0.,  ..., 0., 1., 0.],
        [0., 0., 0.,  ..., 0., 0., 1.]])
```

y=[34]

```
tensor([1, 1, 1, 1, 3, 3, 3, 1, 0, 1, 3, 1, 1, 1, 0, 0, 3, 1, 0, 1, 0, 1, 0, 0,
        2, 2, 0, 0, 2, 0, 0, 2, 0, 0])
```

train_mask=[34]

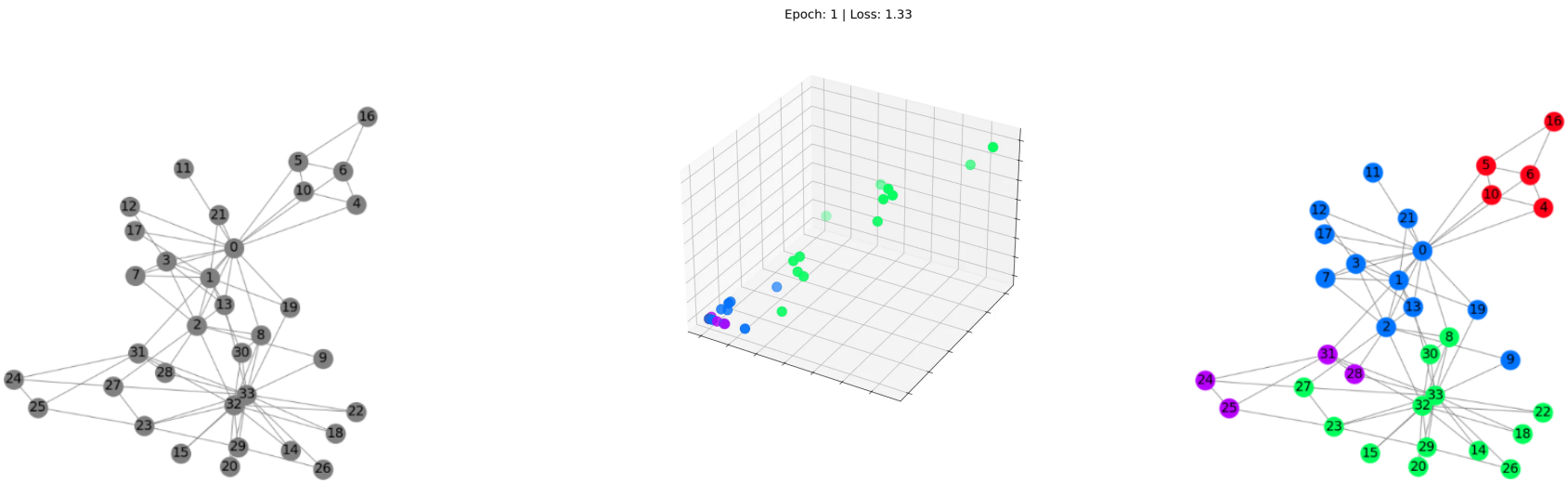
```
tensor([ True, False, False, False,  True, False, False, False,  True, False,
        False, False, False, False, False, False, False, False, False, False,
        False, False, False, False,  True, False, False, False, False, False,
        False, False, False, False])
```

edge_index=[2, 156]

```
tensor([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,
          1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  3,
          3,  3,  3,  3,  4,  4,  4,  5,  5,  5,  5,  6,  6,  6,  6,  7,  7,
          7,  7,  8,  8,  8,  8,  9,  9, 10, 10, 10, 11, 12, 12, 13, 13, 13,
         13, 13, 14, 14, 15, 15, 16, 16, 17, 17, 18, 18, 19, 19, 19, 20, 20, 21,
         21, 22, 22, 23, 23, 23, 23, 23, 24, 24, 24, 25, 25, 25, 26, 26, 27, 27,
         27, 27, 28, 28, 28, 29, 29, 29, 29, 30, 30, 30, 30, 31, 31, 31, 31, 31,
         31, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33, 33, 33, 33,
         33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33],
        [ 1,  2,  3,  4,  5,  6,  7,  8, 10, 11, 12, 13, 17, 19, 21, 31,  0,  2,
          3,  7, 13, 17, 19, 21, 30,  0,  1,  3,  7,  8,  9, 13, 27, 28, 32,  0,
          1,  2,  7, 12, 13,  0,  6, 10,  0,  6, 10, 16,  0,  4,  5, 16,  0,  1,
          2,  3,  0,  2, 30, 32, 33,  2, 33,  0,  4,  5,  0,  0,  3,  0,  1,  2,
          3, 33, 32, 33, 32, 33,  5,  6,  0,  1, 32, 33,  0,  1, 33, 32, 33,  0,
          1, 32, 33, 25, 27, 29, 32, 33, 25, 27, 31, 23, 24, 31, 29, 33,  2, 23,
         24, 33,  2, 31, 33, 23, 26, 32, 33,  1,  8, 32, 33,  0, 24, 25, 28, 32,
         33,  2,  8, 14, 15, 18, 20, 22, 23, 29, 30, 31, 33,  8,  9, 13, 14, 15,
         18, 19, 20, 22, 23, 26, 27, 28, 29, 30, 31, 32]])
```

Figure 32: Karate club dataset.

Practical example: GCN to node classification



(a) Input graph. (b) Train GCN. (c) Output graph.

Figure 33: GCN to node classification.

Practical example: GCN to node classification

https://github.com/win7/Presentations-Hub/blob/main/SummerCamp-2025_IA-PUCP/Node_Classification.ipynb

Package, library, and framework

Figure 34: Package, library, and framework.

Books

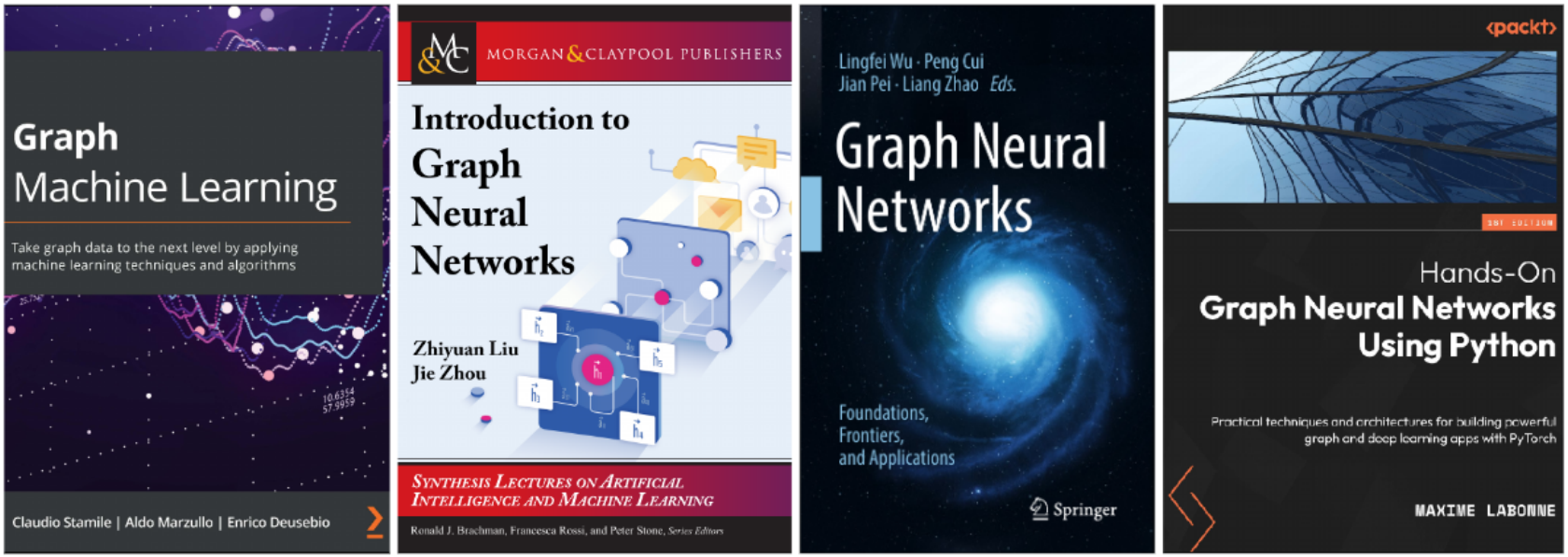


Figure 35: GNNs books.

More learning

- Courses
  - CS224W: Machine Learning with Graphs (`http://web.stanford.edu/class/cs224w/`)
- Tutorials
  - PyTorch Geometric: Introduction by Example (`https://pytorch-geometric.readthedocs.io/en/latest/notes/introduction.html`)
- Conferences
  - Learning on Graphs (`https://logconference.org/`)

Thank you for your attention!

∞

edwin.alvarez@pucp.edu.pe
https://github.com/win7

[1] Maxime Labonne.
Hands-On Graph Neural Networks Using Python: Practical techniques and architectures for building powerful graph and deep learning apps with PyTorch.
Packt Publishing Ltd, 2023.

[2] Claudio Stamile, Aldo Marzullo, and Enrico Deusebio.
Graph Machine Learning: Take graph data to the next level by applying machine learning techniques and algorithms.
Packt Publishing Ltd, 2021.

[3] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy.
Machine learning on graphs: A model and comprehensive taxonomy.
Journal of Machine Learning Research, 23(89):1–64, 2022.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space.
arXiv preprint arXiv:1301.3781, 2013.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.
Distributed representations of words and phrases and their compositionality.
Advances in neural information processing systems, 26, 2013.

[6] Aditya Grover and Jure Leskovec.
node2vec: Scalable feature learning for networks.
In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[7] Xiang Yue, Zhen Wang, Jingong Huang, Srinivasan Parthasarathy, Soheil Moosavinasab, Yungui Huang, Simon M Lin, Wen Zhang, Ping Zhang, and Huan Sun.
Graph embedding on biomedical networks: methods, applications and evaluations.
Bioinformatics, 36(4):1241–1251, 2020.

[8] Thomas N Kipf and Max Welling.
Semi-supervised classification with graph convolutional networks.
arXiv preprint arXiv:1609.02907, 2016.

[9] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al.
Graph attention networks.
stat, 1050(20):10–48550, 2017.

[10] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka.
How powerful are graph neural networks?
arXiv preprint arXiv:1810.00826, 2018.

[11] Thomas N Kipf and Max Welling.
Variational graph auto-encoders.
arXiv preprint arXiv:1611.07308, 2016.

[12] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm.
Deep graph infomax.
arXiv preprint arXiv:1809.10341, 2018.

[13] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang.
Adversarially regularized graph autoencoder for graph embedding.
arXiv preprint arXiv:1802.04407, 2018.

[14] Guillaume Salha, Romain Hennequin, and Michalis Vazirgiannis.
Simple and effective graph autoencoders with one-hop linear models.
In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, pages 319–334. Springer, 2021.

[15] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich.
Superglue: Learning feature matching with graph neural networks.
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.