

# How Powerful are Graph Neural Networks?

(EE531 Final Project - Graph)

K. Xu<sup>1</sup>   W. Hu<sup>2</sup>   J. Leskovec<sup>2</sup>   S. Jegelka<sup>1</sup>

<sup>1</sup>MIT

<sup>2</sup>Stanford University

Appeared at ICLR 2019

# Table of Contents

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

- Since GNN(Graph Neural Network) has come out, it has revolutionized the field of representation learning, especially with graph datas.
- But why GNNs work? Why are they so powerful?
- Most interestingly, *how powerful are they?*

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

- There hasn't been much work regarding this topic.
- Scarselli et al.<sup>1</sup> showed that the (probably) earliest GNN model<sup>2</sup> can approximate measurable functions in probability.
- Lei et al.<sup>3</sup> showed that their architecture lies in the RKHS of graph kernels, *but do not study explicitly which graph it can distinguish*.
- Above works focus on a specific architecture and **do not easily generalize to other architectures**.
- This paper presents a **general framework** for analyzing/characterizing the expressive power of a **broad class of GNNs**!

---

<sup>1</sup>Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80.

<sup>2</sup>Franco Scarselli et al. "Computational Capabilities of Graph Neural Networks". In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 81–102.

<sup>3</sup>Tao Lei et al. "Deriving Neural Architectures from Sequence and Graph Kernels". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 2024–2033.

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References



- Let  $G = (V, E)$  be a graph, or data with graph structure.
- Each  $v \in V$  has a *node feature vector*,  $X_v$
- There are two tasks of interest where GNN is commonly used:

## Node Classification Problem

Each  $v \in V$  has an associated label  $y_v$ .

**Goal:** Learn a representation vector  $h_v$  of  $v$  such that  $y_v = f(h_v)$  i.e. such that  $v$ 's label can be predicted.

## Graph Classification Problem

A set of graphs  $\{G_1, \dots, G_N\} \subset \mathcal{G}$  is given, along with their labels  $\{y_1, \dots, y_N\} \subset \mathcal{Y}$ .

**Goal:** Learn a representation vector  $h_G$  of  $G$  such that  $y_G = f(h_G)$  i.e. such that  $G$ 's label can be predicted.

- Modern GNNs follow a **neighborhood aggregation strategy** (message passing strategy)
- Iteratively update the representation of a nodes by aggregating representations of its neighbors!
- Let  $h_v^{(k)}$  is the feature vector of node  $v$  at the  $k$ -th iteration/layer, and let us initialize it as  $h_v^{(0)} = X_v$ .
- $k$ -th layer of a GNN is

$$a_v^{(k)} = \text{AGGREGATE}^{(k)} \left( \left\{ h_u^{(k-1)} : u \in \mathcal{N}_G(v) \right\} \right)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)} \left( h_v^{(k-1)}, a_v^{(k)} \right)$$

- Different choices of  $\text{AGGREGATE}^{(k)}$  and  $\text{COMBINE}^{(k)}$  have led to different GNN variants/architectures.
- GraphSAGE<sup>4</sup>:

$$a_v^{(k)} = \text{MAX} \left( \left\{ \text{ReLU} \left( W h_u^{(k-1)} \right) : u \in \mathcal{N}_G(v) \right\} \right)$$

$$h_v^{(k)} = W \left[ h_v^{(k-1)}, a_v^{(k)} \right]$$

- Graph Convolutional Networks, or GCN<sup>5</sup>:

$$h_v^{(k)} = \text{ReLU} \left( W \text{MEAN} \left\{ h_u^{(k-1)} : u \in \mathcal{N}_G(v) \cup \{v\} \right\} \right)$$

---

<sup>4</sup>William L. Hamilton, Zitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 1024–1034.

<sup>5</sup>Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

- In case of node classification, the final node representation  $h_v^{(K)}$  is used for prediction.
- In case of graph classification, the final node representations are aggregated by READOUT function to obtain the entire graph's representation:

$$h_G = \text{READOUT} \left( \left\{ h_v^{(K)} : v \in V \right\} \right)$$

- READOUT can be a simple permutation invariant function, or something more sophisticated<sup>67</sup>

---

<sup>6</sup>Zhitao Ying et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 2018, pp. 4805–4815.

<sup>7</sup>Muhan Zhang et al. "An End-to-End Deep Learning Architecture for Graph Classification". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 4438–4445.

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - **WL test**
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

# Graph Isomorphism Problem

- Consider the following problem:

## GRAPH ISOMORPHISM (GI)

**Input:** Two finite graphs  $G_1$  and  $G_2$

**Question:**  $G_1 \cong G_2$ ?

- Appears in: discrete mathematics, mathematical logic, theory of computation, machine learning, computer vision...etc.
- This seemingly harmless problem has harassed researchers for decades!

# Graph Isomorphism Problem

Here are some facts related to GI:

- Not known to be of class NP-complete nor tractable!  
(Researchers have actually defined a new complexity class **GI**)
- It is currently known that GI can be solved in quasipolynomial time *i.e.* in  $O\left(2^{O((\log n)^c)}\right)$  ( $c > 0$ ) time<sup>8</sup>:

## Theorem (Babai, 2015)

The Graph Isomorphism problem ... can be solved in quasipolynomial time.

(Confirmed by Harald Andrés Helfgott, probably correct)

- But it is not practical!
- Some practical algorithms: McKay (1981), Schmidt & Druffel (1976), Ullman (1976)...etc.

---

<sup>8</sup>László Babai. "Graph isomorphism in quasipolynomial time [extended abstract]". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 2016, pp. 684–697.

# Weisfeiler-Lehman test

- Weisfeiler-Lehman test of graph isomorphism<sup>9</sup>, or simply WL test, is a combinatorial algorithm for GI.
- WL test is proved to be successful (and computationally efficient) in isomorphism testing for a broad class of graphs<sup>10</sup>
- There are some cases (ex. regular graphs) when the WL test fails<sup>11</sup>

---

<sup>9</sup>Boris Weisfeiler and Andrei A. Lehman. "A reduction of a graph to a canonical form and an algebra arising during this reduction". In: *Nauchno-Tekhnicheskaya Informatsia* 2.9 (1968), pp. 12–16.

<sup>10</sup>László Babai and Ludek Kucera. "Canonical Labelling of Graphs in Linear Average Time". In: *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*. 1979, pp. 39–46.

<sup>11</sup>Jin-yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identifications". In: *Combinatorica* 12.4 (1992), pp. 389–410.



# Weisfeiler-Lehman test

- Why are we interested in this WL test?
- 1-dimensional form of the WL test ("naïve vertex refinement") is *based on neighbor aggregations*, analogous to the GNNs!
- Overview of the algorithm:
  - Aggregate the labels of nodes and their neighborhoods
  - Hashes the aggregated label into *unique* new labels
  - If at some iteration the labels of the nodes between the two graphs differ, then the two graphs are non-isomorphic.

# 1-dim WL test

Let  $(G, I)$  be a labeled graph i.e. a graph  $G$  with an endowed node coloring  $I : V(G) \rightarrow \Sigma$ . ( $\Sigma$ : arbitrary codomain)

- At  $t$ -th iteration ( $t \geq 0$ ), the 1-WL computes a node coloring  $c_I^{(t)} : V(G) \rightarrow \Sigma$ , which depends on the previous node coloring:

$$c_I^{(0)} = I, \quad c_I^{(t)}(v) = \text{HASH} \left( \left( c_I^{(t-1)}(v), \{ \{ c_I^{(t-1)}(u) \mid u \in \mathcal{N}(v) \} \} \right) \right)$$

(HASH bijectively maps the above pair to a unique value in  $\Sigma$  that hasn't been used in previous iterations)

- Run above algorithm in parallel for the two input graphs.
- If at some iteration, the two graphs have a different number of nodes colored  $\sigma \in \Sigma$ , conclude that the graphs are not isomorphic.  
(This why this 1-dim version is commonly called the *color refinement algorithm*)

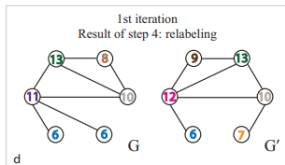
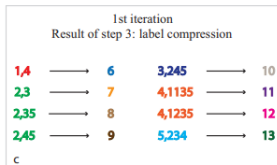
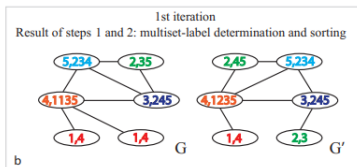
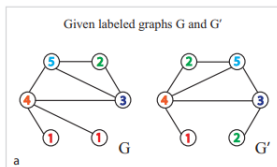
- **Graph kernel**: kernel function that defines *inner product on graphs*<sup>12</sup> (Function measuring the similarity of a pair of two given graphs)
- Some examples: *random walk* (Gärtner *et al.*, 2003; Borgwardt *et al.*, 2005), *marginalized* (Kashima *et al.*, 2003, 2004; Mahé *et al.*, 2004)
- **Weisfeiler-Lehman subtree kernel**<sup>13</sup>: counts common *original and compressed labels* (resulting from 1-dim WL test) in two graphs.

---

<sup>12</sup>S. V. N. Vishwanathan *et al.* "Graph Kernels". In: *J. Mach. Learn. Res.* 11 (2010), pp. 1201–1242.

<sup>13</sup>Nino Shervashidze *et al.* "Weisfeiler-Lehman Graph Kernels". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2539–2561.

# WL subtree kernel



End of the 1st iteration  
Feature vector representations of  $G$  and  $G'$

$$\phi_{WLsubtree}^{(1)}(G) = (2, 1, 1, 1, 1, 2, 0, 1, 0, 1, 1, 0, 1)$$

$$\phi_{WLsubtree}^{(1)}(G') = (1, 2, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1)$$

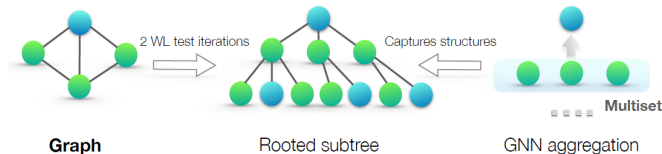
Counts of original node labels      Counts of compressed node labels

$$k_{WLsubtree}^{(1)}(G, G') = \langle \phi_{WLsubtree}^{(1)}(G), \phi_{WLsubtree}^{(1)}(G') \rangle = 11.$$

e

# WL subtree kernel

- Is it related to GNN? Yes!
- The kernel uses the *counts of node labels* at different iterations of the WL test as the *feature vector* of a graph.
- Intuitively, a node's label at the  $k$ -th iteration of the 1-dim WL test represents a subtree structure of height  $k$  rooted at the node.



- Thus, the graph features considered by the WL subtree kernel are essentially counts of different rooted subtrees in the graph!

- k-dim WL test is a generalization of the 1-dim WL test; it colors tuples from  $V(G)^k$  instead of nodes.
- Why would we want to do that?
- By increasing  $k$ , the algorithm gets more powerful in terms of distinguishing non-isomorphic graphs!
- It was shown that for each  $k \geq 2$ , there are non-isomorphic graphs which can be distinguished by the  $(k + 1)$ -dim WL test, but not by the  $k$ -dim WL test<sup>14</sup>
- In this work, we only focus on 1-dim WL test.

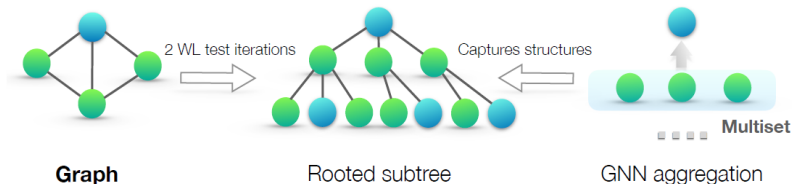
---

<sup>14</sup>Jin-yi Cai, Martin Fürer, and Neil Immerman. “An optimal lower bound on the number of variables for graph identifications”. In: *Combinatorica* 12.4 (1992), pp. 389–410.

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

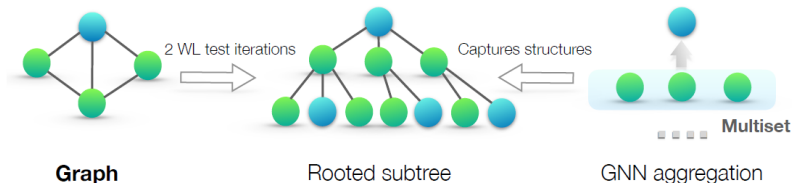
# (Overview of) Theoretical Framework



- GNN: recursive update of each node's feature vector *i.e.* its rooted subtree structure!
- (1-dim) WL test: also results in rooted subtree structure!
- Assign each feature vector a unique label from a countable universe.
- Then, feature vectors of a set of neighboring nodes form a **multiset**.



# (Overview of) Theoretical Framework



- Representational power of a GNN: when a GNN maps two nodes to the same location (in the embedding space)?
- Maximally powerful GNN: its aggregation scheme must be **injective**!
- Closely related to GRAPH ISOMORPHISM.
- GNN's aggregation scheme: *class of functions over multisets that their neural networks can represent*

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs**
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

# Representational capacity of GNNs

- Recall: maximally powerful GNN has injective aggregation scheme.
- Two (non)isomorphic graphs are mapped to the (different)same representation(s).
- Characterized by GRAPH ISOMORPHISM!

## Lemma 2

Let  $G_1$  and  $G_2$  be any two non-isomorphic graphs.

If a graph neural network  $\mathcal{A} : \mathcal{G} \rightarrow \mathbb{R}^d$  maps  $G_1$  and  $G_2$  to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides  $G_1$  and  $G_2$  are not isomorphic.

- It says that **any aggregation-based GNN is at most as powerful as the WL test** in distinguishing different graphs.

# Representational capacity of GNNs

- Is that "bound" tight?
- In other words, does there exist GNN that is, in principle, as powerful as the WL test in distinguishing different graphs?

## Theorem 3

Let  $\mathcal{A} : \mathcal{G} \rightarrow \mathbb{R}^d$  be a GNN.

With a sufficient number of GNN layers,  $\mathcal{A}$  maps any  $G_1$  and  $G_2$  that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:

- $\mathcal{A}$  aggregates and updates node features iteratively with

$$h_v^{(k)} = \phi \left( h_v^{(k-1)}, f \left( \left\{ h_u^{(k-1)} : u \in \mathcal{N}_G(v) \right\} \right) \right)$$

where the functions  $f$ , which operates on multisets, and  $\phi$  are *injective*.

- $\mathcal{A}$ 's graph-level readout, which operates on the multiset of node features  $\{h_v^{(k)}\}$ , is *injective*.

# Representational capacity of GNNs

- Node feature vectors in the WL test are essentially one-hot encodings, and thus cannot capture similarity between subtrees!
- GNN (satisfying the criteria in Theorem 3) generalizes the WL test by *learning to embed* the subtrees to low-dimensional space.
- GNNs can not only discriminate different structures, but can also *learn to map similar graph structures to similar embeddings and capture dependencies between graph structures*.
- Especially useful when co-occurrence of subtrees is sparse across different graphs, or there are noisy edges and node features.<sup>15</sup>

---

<sup>15</sup>Pinar Yanardag and S. V. N. Vishwanathan. “Deep Graph Kernels”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 2015, pp. 1365–1374.

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)**
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

# Graph Isomorphism Network (GIN)

- Well we've proved (or more accurately, seen) that GNNs under certain conditions is maximally powerful.
- Let us develop a simple architecture, GIN!
- Idea: **deep multisets**<sup>16</sup> i.e. parametrizing universal multiset functions with neural networks.

---

<sup>16</sup>Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 3391–3401.

## Lemma 5

Assume  $\mathcal{X}$  is countable. There exists a function  $f : \mathcal{X} \rightarrow \mathbb{R}^n$  so that  $h(X) = \sum_{x \in X} f(x)$  is unique for each multiset  $X \subset \mathcal{X}$  of bounded size. Moreover, any multiset function  $g$  can be decomposed as  $g(X) = \phi(\sum_{x \in X} f(x))$  for some function  $\phi$ .

- Observe that certain popular injective set functions, such as the mean aggregator, are *not* injective multiset functions!
- This lemma tells us that sum aggregators can represent injective, in fact, *universal* functions over multisets.
- Thus, we can conceive aggregation schemes that can **represent universal functions over a node and the multiset of its neighbors**, satisfying the *injectiveness condition* (a) in Theorem 3!



# Graph Isomorphism Network (GIN)

- Here is a simple and concrete formulation of the previous discussion:

## Corollary 6

Assume  $\mathcal{X}$  is countable. There exists a function  $f : \mathcal{X} \rightarrow \mathbb{R}^n$  so that for infinitely many choices of  $\epsilon$ , including all irrational numbers,  $h(c, X) = (1 + \epsilon)f(c) + \sum_{x \in X} f(x)$  is unique for each pair  $(c, X)$ , where  $c \in \mathcal{X}$  and  $X \subset \mathcal{X}$  is a multiset of bounded size.

Moreover, any function  $g$  over such pairs can be decomposed as  $g(c, X) = \varphi((1 + \epsilon)f(c) + \sum_{x \in X} f(x))$  for some function  $\varphi$ .

- We can use MLPs to model and learn  $f$  and  $\varphi$ , thanks to the Universal Approximation Theorem<sup>1718</sup>.

---

<sup>17</sup>Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

<sup>18</sup>Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257.

# How powerful is MLP?

## Universal Approximation Theorem (Hornik, 1991)

Define

$$\mathcal{N}_k^{(n)}(\psi) = \left\{ h : \mathbb{R}^k \rightarrow \mathbb{R} \mid h(x) = \sum_{j=1}^n \beta_j \psi(a'_j x - \theta_j) \right\}$$

as the set of all functions implemented by such a network with  $n$  hidden units, where  $\psi$  is the common activation function of the hidden units.

*If  $\psi$  is continuous, bounded and nonconstant, then  $\mathcal{N}_k^{(n)}(\psi)$  is dense in  $\mathcal{C}(X)$  for all compact subsets  $X$  of  $\mathbb{R}^k$ .*

- Every continuous function can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer.
- The choice of the activation function doesn't matter; it's the multilayer feedforward architecture that gives neural networks the potential of being universal approximators.

# Graph Isomorphism Network (GIN)

- In practice,  $f^{(k+1)} \circ \varphi^{(k)}$  is modeled with one MLP.
- We may make  $\epsilon$  as a learnable parameter, or a fixed scalar.
- Then, GIN updates node representations as:

$$h_v^{(k)} = \text{MLP}^{(k)} \left( \left(1 + \epsilon^{(k)}\right) h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right)$$

# Graph Isomorphism Network (GIN)

- Node embeddings, learned by the GIN, can be directly used for node classification and link prediction.
- For graph classification tasks, we need a READOUT function.
- We want to consider all structural information, considering that features from earlier iterations may sometimes generalize better.
- Use information from *all* depths/iterations of the model<sup>19</sup>!

$$h_G = \text{CONCAT} \left( \text{READOUT} \left( \left\{ h_v^k \mid v \in V(G) \right\} \right) \mid k = 0, 1, \dots, K \right)$$

- Note that if GIN replaces READOUT with summing all node features from the same iteration, it provably generalizes the WL test and the WL subtree kernel.

---

<sup>19</sup>Keyulu Xu et al. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2018, pp. 5449–5458.

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs**
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

- Now we consider GNNs that do not satisfy the conditions as described in Theorem 3 and/or GNNs with different choice of AGGREGATE (Max-pooling, Mean)
- What if 1-layer perceptron is used instead of MLPs?
- What if the sum  $h(X) = \sum_{x \in X} f(x)$  is replaced by mean/max pooling?

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs**
  - **1-Layer Perceptrons**
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

- The function  $f$  in Lemma 5 helps map distinct multisets to unique embeddings.
- $f$  can be parametrized by MLPs, as shown by the Universal Approximation Theorem<sup>20</sup>

---

<sup>20</sup>Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257.



# Without MLP?

- Many modern GNNs, however, use a *1-layer perceptron*  $\sigma \circ W$ : a linear mapping followed by a non-linear activation function.
- Is 1-layer perceptron enough for graph learning?

## Lemma 7

There exist finite multisets  $X_1 \neq X_2$  so that for any linear mapping  $W$ ,  $\sum_{x \in X_1} \text{ReLU}(Wx) \neq \sum_{x \in X_2} \text{ReLU}(Wx)$

- Unlike models using MLPs, 1-layer perceptron (even with the bias term) is **not a universal approximator of multiset functions**.

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs**
  - 1-Layer Perceptrons
  - **GCN**
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

- As described previously, Graph Convolutional Network<sup>21</sup> takes the form:

$$h_v^{(k)} = \text{ReLU} \left( W \text{MEAN} \left\{ h_u^{(k-1)} : u \in \mathcal{N}_G(v) \cup \{v\} \right\} \right)$$

- GCN utilizes mean aggregator.
- How can we characterize the structures that GCN can or cannot capture?

---

<sup>21</sup>Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

# Mean aggregator

- Consider two multisets  $X_1 = (S, m)$  and  $X_2 = (S, km)$
- Observation: Any mean aggregator maps  $X_1$  and  $X_2$  to the *same* embeddings!
- Mean aggregator captures the **distribution of elements in a multiset**.

## Corollary 8

Assume  $\mathcal{X}$  is countable. There exists a function  $f : \mathcal{X} \rightarrow \mathbb{R}^n$  so that  $h(X) = \frac{1}{|X|} \sum_{x \in X} f(x)$ ,  $h(X_1) = h(X_2)$  if and only if multisets  $X_1$  and  $X_2$  have the same distribution. That is, assuming  $|X_2| \geq |X_1|$ , we have  $X_1 = (S, m)$  and  $X_2 = (S, km)$  for some  $k \in \mathbb{N}$ .

- This is as powerful as the sum aggregator if the node features are diverse and rarely repeat, and thus *effective for node classification*.

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs**
  - 1-Layer Perceptrons
  - GCN
  - **GraphSAGE**
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

- As described previously, GraphSAGE<sup>22</sup> takes the form:

$$a_v^{(k)} = \text{MAX} \left( \left\{ \text{ReLU} \left( W h_u^{(k-1)} \right) : u \in \mathcal{N}_G(v) \right\} \right)$$

$$h_v^{(k)} = W \left[ h_v^{(k-1)}, a_v^{(k)} \right]$$

- GraphSAGE utilizes max-pooling aggregator.
- How can we characterize the structures that GraphSAGE can or cannot capture?

---

<sup>22</sup>William L. Hamilton, Zitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 1024–1034.

# Max-pooling aggregator

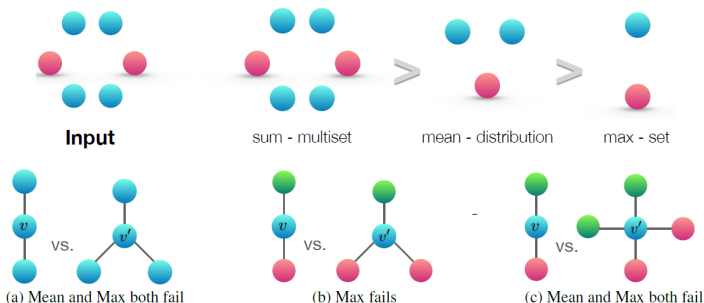
- Unlike previous aggregators, max-pooling can't capture exact structure nor the distribution!
- But it can capture the **underlying set of multiset** i.e.  $S$  in  $X = (S, m)$

## Corollary 9

Assume  $\mathcal{X}$  is countable. There exists a function  $f : \mathcal{X} \rightarrow \mathbb{R}^\infty$  so that  $h(X) = \max_{x \in X} f(x)$ ,  $h(X_1) = h(X_2)$  if and only if multisets  $X_1$  and  $X_2$  have the same underlying set.

# Summary

- Let us rank the three aggregators by their representational power:



- Sum over multiset aggregator (as in GIN) completely captures the exact structure of graph.
- Mean aggregator (as in GCN) captures the statistical and distributional information of the graph.
- Max-pooling aggregator (as in GraphSAGE) captures the representative elements of the graph, or its skeleton



# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments**
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References

The goal of the experiment is to compare the training and test performance of GIN and less powerful GNN variants.

- Training set performance: compare different GNN models based on their *representational power*
- Test set performance: quantifies *generalization ability*

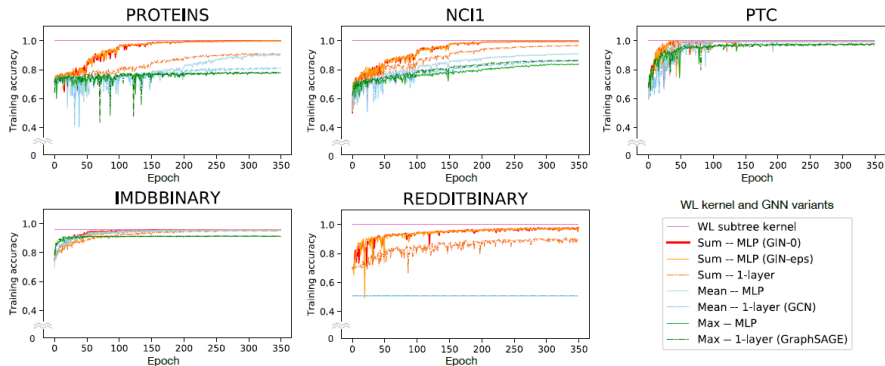
# Experiment Design

- 9 graph classification benchmarks were used<sup>23</sup>:
  - 4 bioinformatics datasets (*MUTAG*, *PTC*, *NCI1*, *PROTEINS*)
  - 5 social network datasets (*COLLAB*, *IMDB-BINARY*, *IMDB-MULTI*, *REDDIT-BINARY*, *REDDIT-MULTI5K*)
- Several models were used:
  - GIN  $-\epsilon$ : GIN that *learns*  $\epsilon$  by gradient descent
  - GIN  $-0$ : GIN that fixes  $\epsilon$  to 0.
  - Architectures that replace the sum in the GIN  $-0$  aggregation with mean or max-pooling, or replace MLPs with 1-layer perceptrons
  - GCN
  - GraphSAGE
- The baselines used were:
  - WL subtree kernel with C-SVM used as a classifier
  - Deep learning architectures i.e. DCNN, PATCHY-SAN, DGCNN
  - AWL

---

<sup>23</sup>Pinar Yanardag and S. V. N. Vishwanathan. “Deep Graph Kernels”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 2015, pp. 1365–1374.

# Results



# Results

|              |                            |                                  |                                  |                                  |                                  |                                  |                                    |                                  |                                  |                                    |
|--------------|----------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|------------------------------------|----------------------------------|----------------------------------|------------------------------------|
| Datasets     | Datasets                   | IMDB-B                           | IMDB-M                           | RD1-B                            | RD1-M5K                          | COLLAB                           | MUTAG                              | PROTEINS                         | PTC                              | NC11                               |
|              | # graphs                   | 1000                             | 1500                             | 2000                             | 5000                             | 5000                             | 188                                | 1113                             | 344                              | 4110                               |
|              | # classes                  | 2                                | 3                                | 2                                | 5                                | 3                                | 2                                  | 2                                | 2                                | 2                                  |
|              | Avg # nodes                | 19.8                             | 13.0                             | 429.6                            | 508.5                            | 74.5                             | 17.9                               | 39.1                             | 25.5                             | 29.8                               |
| Baselines    | WL subtree                 | 73.8 $\pm$ 3.9                   | 50.9 $\pm$ 3.8                   | 81.0 $\pm$ 3.1                   | 52.5 $\pm$ 2.1                   | 78.9 $\pm$ 1.9                   | 90.4 $\pm$ 5.7                     | 75.0 $\pm$ 3.1                   | 59.9 $\pm$ 4.3                   | <b>86.0 <math>\pm</math> 1.8 *</b> |
|              | DCNN                       | 49.1                             | 33.5                             | –                                | –                                | 52.1                             | 67.0                               | 61.3                             | 56.6                             | 62.6                               |
|              | PATCHYSAN                  | 71.0 $\pm$ 2.2                   | 45.2 $\pm$ 2.8                   | 86.3 $\pm$ 1.6                   | 49.1 $\pm$ 0.7                   | 72.6 $\pm$ 2.2                   | <b>92.6 <math>\pm</math> 4.2 *</b> | 75.9 $\pm$ 2.8                   | 60.0 $\pm$ 4.8                   | 78.6 $\pm$ 1.9                     |
|              | DGCNN                      | 70.0                             | 47.8                             | –                                | –                                | 73.7                             | 85.8                               | 75.5                             | 58.6                             | 74.4                               |
|              | AWL                        | 74.5 $\pm$ 5.9                   | 51.5 $\pm$ 3.6                   | 87.9 $\pm$ 2.5                   | 54.7 $\pm$ 2.9                   | 73.9 $\pm$ 1.9                   | 87.9 $\pm$ 9.8                     | –                                | –                                | –                                  |
| GIN variants | SUM-MLP (GIN-0)            | <b>75.1 <math>\pm</math> 5.1</b> | <b>52.3 <math>\pm</math> 2.8</b> | <b>92.4 <math>\pm</math> 2.5</b> | <b>57.5 <math>\pm</math> 1.5</b> | <b>80.2 <math>\pm</math> 1.9</b> | <b>89.4 <math>\pm</math> 5.6</b>   | <b>76.2 <math>\pm</math> 2.8</b> | <b>64.6 <math>\pm</math> 7.0</b> | <b>82.7 <math>\pm</math> 1.7</b>   |
|              | SUM-MLP (GIN- $\epsilon$ ) | <b>74.3 <math>\pm</math> 5.1</b> | <b>52.1 <math>\pm</math> 3.6</b> | <b>92.2 <math>\pm</math> 2.3</b> | <b>57.0 <math>\pm</math> 1.7</b> | <b>80.1 <math>\pm</math> 1.9</b> | <b>89.0 <math>\pm</math> 6.0</b>   | <b>75.9 <math>\pm</math> 3.8</b> | 63.7 $\pm$ 8.2                   | <b>82.7 <math>\pm</math> 1.6</b>   |
|              | SUM-1-LAYER                | 74.1 $\pm$ 5.0                   | <b>52.2 <math>\pm</math> 2.4</b> | 90.0 $\pm$ 2.7                   | 55.1 $\pm$ 1.6                   | <b>80.6 <math>\pm</math> 1.9</b> | <b>90.0 <math>\pm</math> 8.8</b>   | <b>76.2 <math>\pm</math> 2.6</b> | 63.1 $\pm$ 5.7                   | 82.0 $\pm$ 1.5                     |
|              | MEAN-MLP                   | 73.7 $\pm$ 3.7                   | <b>52.3 <math>\pm</math> 3.1</b> | 50.0 $\pm$ 0.0                   | 20.0 $\pm$ 0.0                   | 79.2 $\pm$ 2.3                   | 83.5 $\pm$ 6.3                     | 75.5 $\pm$ 3.4                   | <b>66.6 <math>\pm</math> 6.9</b> | 80.9 $\pm$ 1.8                     |
|              | MEAN-1-LAYER (GCN)         | 74.0 $\pm$ 3.4                   | 51.9 $\pm$ 3.8                   | 50.0 $\pm$ 0.0                   | 20.0 $\pm$ 0.0                   | 79.0 $\pm$ 1.8                   | 85.6 $\pm$ 5.8                     | 76.0 $\pm$ 3.2                   | 64.2 $\pm$ 4.3                   | 80.2 $\pm$ 2.0                     |
|              | MAX-MLP                    | 73.2 $\pm$ 5.8                   | 51.1 $\pm$ 3.6                   | –                                | –                                | –                                | 84.0 $\pm$ 6.1                     | 76.0 $\pm$ 3.2                   | 64.6 $\pm$ 10.2                  | 77.8 $\pm$ 1.3                     |
|              | MAX-1-LAYER (GraphSAGE)    | 72.3 $\pm$ 5.3                   | 50.9 $\pm$ 2.2                   | –                                | –                                | –                                | 85.1 $\pm$ 7.6                     | 75.9 $\pm$ 3.2                   | 63.9 $\pm$ 7.7                   | 77.7 $\pm$ 1.5                     |

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research**
- 9 Summary of GNNs
- 10 References

In summary,

- Theoretical foundations for reasoning about the expressive power of GNNS
- Tight bounds on the representational capacity of popular GNN variants. (cf. WL test)
- Designed a provably maximally powerful GNN under the neighborhood aggregation framework (*Graph Isomorphism Network*)

- Different aggregators?
- Go beyond neighborhood aggregation
- Understand/improve the generalization properties of GNNs
- What if the node features are continuous (uncountable)?
- Better understanding of the optimization landscape



# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs**
- 10 References

# Summary of GNNs

Here is a summary of "major" GNNs<sup>24</sup>: (next page)

---

<sup>24</sup>Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *arXiv e-prints* (Jan. 2019). arXiv: 1901.00596 [cs.LG].

# Summary of GNNs

(1: RecGNN, 2: Spectral-based ConvGNN, 3: Spatial-based ConvGNN)

| Method          | Category | Time Complexity | Features  |
|-----------------|----------|-----------------|---|
| GNN[16]         | 1        | $O(m)$          | Information diffusion mechanism, updates nodes' states until a stable equilibrium is reached.             |
| Spectral CNN[4] | 2        | $O(n^3)$        | Treats the filters as a set of learnable parameters.  |
| ChebNet[6]      | 2        | $O(m)$          | Approximates the filter by Chebyshev polynomials of the diagonal matrix of eigenvalues.                   |
| GCN[11]         | 2        | $O(m)$          | First-order approximation of ChebNet  |
| AGCN[13]        | 2        | $O(n^2)$        | Learns hidden structural relations by using the residual graph adjacency matrix through learnable metric. |
| DualGCN[27]     | 2        | $O(m)$          | Introduces dual graph convolutional architecture with two graph convolutional layers in parallel.         |
| NN4G[14]        | 3        | $O(m)$          | Performs graph convolutions by summing up a node's neighborhood information directly.                     |
| DCNN[1]         | 3        | $O(n^2)$        | Treats graph convolutions as a diffusion process  |
| MPNN[7]         | 3        | $O(m)$          | Treats graph convolutions as a message passing process.   |
| GraphSAGE[8]    | 3        | -               | Sampling of fixed number of neighbors for each node.  |
| GIN[21]         | 3        | $O(m)$          | This paper!   |

# Outline

- 1 Introduction
- 2 Previous / Related Works
- 3 Preliminaries
  - GNNs
  - WL test
  - (Overview of) Theoretical Framework
- 4 Representational capacity of GNNs
- 5 Graph Isomorphism Network (GIN)
- 6 Less powerful, but still interesting GNNs
  - 1-Layer Perceptrons
  - GCN
  - GraphSAGE
- 7 Experiments
- 8 Summary / Future Research
- 9 Summary of GNNs
- 10 References**

# References I

- [1] James Atwood and Don Towsley. “Diffusion-Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 1993–2001.
- [2] László Babai. “Graph isomorphism in quasipolynomial time [extended abstract]”. In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 2016, pp. 684–697.
- [3] László Babai and Ludek Kucera. “Canonical Labelling of Graphs in Linear Average Time”. In: *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*. 1979, pp. 39–46.
- [4] Joan Bruna et al. “Spectral Networks and Locally Connected Networks on Graphs”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.
- [5] Jin-yi Cai, Martin Fürer, and Neil Immerman. “An optimal lower bound on the number of variables for graph identifications”. In: *Combinatorica* 12.4 (1992), pp. 389–410.
- [6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 3837–3845.
- [7] Justin Gilmer et al. “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 1263–1272.
- [8] William L. Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 1024–1034.
- [9] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257.
- [10] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.

# References II

- [11] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.
- [12] Tao Lei et al. "Deriving Neural Architectures from Sequence and Graph Kernels". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 2024–2033.
- [13] Ruoyu Li et al. "Adaptive Graph Convolutional Neural Networks". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 3546–3553.
- [14] Alessio Micheli. "Neural Network for Graphs: A Contextual Constructive Approach". In: *IEEE Trans. Neural Networks* 20.3 (2009), pp. 498–511.
- [15] Franco Scarselli et al. "Computational Capabilities of Graph Neural Networks". In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 81–102.
- [16] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80.
- [17] Nino Shervashidze et al. "Weisfeiler-Lehman Graph Kernels". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2539–2561.
- [18] S. V. N. Vishwanathan et al. "Graph Kernels". In: *J. Mach. Learn. Res.* 11 (2010), pp. 1201–1242.
- [19] Boris Weisfeiler and Andrei A. Lehman. "A reduction of a graph to a canonical form and an algebra arising during this reduction". In: *Nauchno-Tekhnicheskaya Informatsia* 2.9 (1968), pp. 12–16.
- [20] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *arXiv e-prints* (Jan. 2019). arXiv: 1901.00596 [cs.LG].
- [21] Keyulu Xu et al. "How Powerful are Graph Neural Networks?". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. 2019.

# References III

- [22] Keyulu Xu et al. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2018, pp. 5449–5458.
- [23] Pinar Yanardag and S. V. N. Vishwanathan. "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 2015, pp. 1365–1374.
- [24] Zhitao Ying et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 2018, pp. 4805–4815.
- [25] Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 3391–3401.
- [26] Muhan Zhang et al. "An End-to-End Deep Learning Architecture for Graph Classification". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 4438–4445.
- [27] Chenyi Zhuang and Qiang Ma. "Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification". In: *Proceedings of the 2018 World Wide Web Conference. WWW '18, Lyon, France: International World Wide Web Conferences Steering Committee, 2018*, pp. 499–508. ISBN: 978-1-4503-5639-8.

*Thank you for your attention! Any questions?*