Graph Neural Networks: 
A Review of Methods and Applications 
Jie Zhou*, Ganqu Cui*, Zhengyan Zhang*, Cheng Yang, Zhiyuan Liu, 
Maosong Sun 
Tsinghua University 
arxiv 2019 

https://qdata.github.io/deep2Read

Presenter : Jack Lanchantin
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets
Graphs

Type of data structure which models a set of objects (vertices) and their relationships (edges).
Graphs

Type of data structure which models a set of objects (vertices) and their relationships (edges).

- **Graph** \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \)
- **Vertices** \( \mathcal{V} = \{1, \ldots, n\} \)
- **Edges** \( \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \)
- **Vertex weights** \( b_i > 0 \) for \( i \in \mathcal{V} \)
- **Edge weights** \( a_{ij} \geq 0 \) for \( (i, j) \in \mathcal{E} \)
Euclidean Data: RNNs/CNNs

- Image, volume, video: 2D, 3D, 2D+1 Euclidean domains
- Sentence, word, sound: 1D Euclidean domain
- These domains have nice regular spatial structures.
  ⇒ All ConvNet operations are math well defined and fast (convolution, pooling).
Non-Euclidean Data

- Social networks
- Regulatory networks
- Functional networks
- 3D shapes
- Graphs/Networks

- Also chemistry, NLP, physics, social science, communication networks, etc.
How to extend CNNs to graph-structured data?

- **Assumption**: Non-Euclidean data are locally stationary and manifest hierarchical structures.
- But, how to define *compositionality on graphs*? (convolution and pooling on graphs)
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets
Learning on Graphs

Naive approach: a per-node classifier

- Represent each node \( v \) as vector \( h_v \in \mathbb{R}^s \)
- Completely drop the graph structure, and classify each node individually, with a shared deep neural network classifier:

\[
o_v = f(h_v; W)
\]  

(1)
Learning on Graphs

Naive approach: a per-node classifier

- Represent each node $v$ as vector $h_v \in \mathbb{R}^s$
- Completely drop the graph structure, and classify each node individually, with a shared deep neural network classifier:

$$o_v = f(h_v; W) \quad (1)$$

Methods like DeepWalk also classify each node independently, but inject graph structure indirectly using learned embeddings
Most deep learning is done this way, even if there are relationships between training examples.
Graph Neural Networks

- Originally introduced by Scarselli et. al. (2009)
- The goal of GNNs is to learn a state embedding $h_v \in \mathbb{R}^s$ which contains information of the neighborhood for each node
- $h_v$ can be used to produce an output $o_v$ such as the node label
Graph Neural Networks

- **Local transition function** $f$: updates each node state according to its neighborhood (shared among all nodes)
- **Output function** $g$: describes how the output is produced

\[ h_v = f(x_v, x_e(v), h_n(v), x_n(v)) \]  
\[ o_v = g(h_v, x_v) \]  

where $x_v$ are the features of $v$, $x_e(v)$ the features of its edges, $h_n(v)$ the states of the nodes in the neighborhood of $v$, and $x_n(v)$ the features of the nodes in the neighborhood of $v$. 
Graph Neural Networks

- **Local transition function** $f$: updates each node state according to its neighborhood (shared among all nodes)
- **Output function** $g$: describes how the output is produced
- Then, $h_v$ and $o_v$ are defined as follows:

  $$h_v = f(x_v, x_{e(v)}, h_{n(v)}, x_{n(v)})$$  \hspace{1cm} (2)$$

  $$o_v = g(h_v, x_v)$$  \hspace{1cm} (3)$$

where $x_v$ are the features of $v$, $x_{e(v)}$ the features of its edges, $h_{n(v)}$ the states of the nodes in the neighborhood of $v$, and $x_{n(v)}$ the features of the nodes in the neighborhood of $v$
GNNs use the following iterative update to compute the state:

\[ H^{t+1} = f(H^t, X) \]  \hspace{1cm} (4)

where \( H^t \) denotes the \( t \)-th iteration of \( H \).
Graph Neural Networks
Graph Neural Network Training

- Given a GNN framework, learn the parameters of \( f \) and \( g \)
- With the target information (\( t_v \) for a specific node) for the supervision, the loss can be written as:

\[
loss = \sum_{i=1}^{p} (t_i - o_i)
\]  

where \( p \) is the number of supervised nodes.
- Weights \( W \) of \( f \) and \( g \) are updated via gradient descent
Original GNN constrains $f$ to be a contractive map

- Contractive map, on a metric space $(M, d)$ is a function $f$ from $M$ to itself, with the property that there is some real number $0 \leq k < 1$ such that for all $x$ and $y$ in $M$, $d(f(x), f(y)) \leq k d(x, y)$
Graph Neural Networks Training

- Original GNN constrains $f$ to be a contractive map
  - Contractive map, on a metric space $(M, d)$ is a function $f$ from $M$ to itself, with the property that there is some real number $0 \leq k < 1$ such that for all $x$ and $y$ in $M$, $d(f(x), f(y)) \leq k d(x, y)$
  
- This implies that the $h_v$ vectors will always converge to a unique fixed point (very restrictive)

- Impossible to inject problem-specific information into $h_v^0$ (will always converge to same value regardless of initialization)
Let $H$, $O$, $X$, and $X_N$ be the vectors constructed by stacking all the states, all the outputs, all the features, and all the node features, respectively. Then we have a compact form as:

$$H = f(H, X)$$ (6)

$$O = g(H, X_N)$$ (7)
Outline

1 Introduction

2 Graph Neural Networks

3 Graph Variants

4 Objective Variants

5 Propagation Variants
   • Convolution
     • Spectral
     • Spatial
   • Gating
   • Attention
   • Skip Connections

6 Training Variants

7 Applications and Datasets
Directed Graphs

- e.g. Knowledge graphs where there is a parent and child.
- Can use different weights for each edge type
Heterogeneous Graphs

- e.g. Multi-model biological network data
- Can group neighbors based on types or distance
Graphs with Edge Information

- e.g. Drug interaction data with different types of interactions (edges). Each edge also has its information like the weight or the type of the edge.
- Can use different types of weight matrices
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets
Objectives

- **Node-level outputs**: prediction for each node in the graph (e.g. classify each person in a social network)

- **Edge-level outputs**: prediction for each edge in the graph (e.g. classify each friendship in a social network)

- **Graph-level outputs**: prediction on the graph as a whole (e.g. classify an entire group in a social network)
Objective Frameworks

- **Supervised learning for node, edge, or graph level classification:** Given full labeled networks, predict target objective
- **Semi-supervised learning for node or edge level classification:** Given a single network with partial nodes being labeled and others remaining unlabeled
- **Unsupervised learning for graph embedding:** When no class labels are available in graphs, we can learn the graph embedding in a purely unsupervised way
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   • Convolution
     • Spectral
     • Spatial
   • Gating
   • Attention
   • Skip Connections
6. Training Variants
7. Applications and Datasets
Propagation Variants

The propagation (update) step and output step are the crucial components to obtain the hidden states of nodes (or edges).

\[ H = f(H, X) \]

Variants utilize different aggregators to gather information from each node’s neighbors and updaters to update nodes’ hidden states.
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets
Advances in this direction are often categorized into either spectral or spatial approaches.
Spectral Convolution Methods

- Define a convolutional operation by operating on the graph in the spectral domain, leveraging the convolution theorem.
- These approaches utilise the graph Laplacian matrix, \( L \), defined as \( L = D - A \), where \( D \) is the degree matrix (diagonal matrix with \( D_{ii} = \text{deg}(i) \)) and \( A \) is the adjacency matrix.

<table>
<thead>
<tr>
<th>Labeled graph</th>
<th>Degree matrix</th>
<th>Adjacency matrix</th>
<th>Laplacian matrix</th>
</tr>
</thead>
</table>
| ![Graph](image) | \[
\begin{pmatrix}
2 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\] | \[
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\] | \[
\begin{pmatrix}
2 & -1 & 0 & 0 & -1 & 0 \\
-1 & 3 & -1 & 0 & -1 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 \\
0 & 0 & -1 & 3 & -1 & -1 \\
-1 & -1 & 0 & -1 & 3 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 \\
\end{pmatrix}
\] |
Limitations of Spectral Methods

- Poor generalization to new/different graphs
- Graphs with variable size: spectral techniques work with fixed size graphs.
- Directed graphs: definition of directed graph Laplacian is unclear.
Transductive Learning
(from Petar Velickovic)

Training algorithm sees all features (including test nodes)
Inductive Learning
(from Petar Velickovic)

- Algorithm **does not have access to all nodes** upfront. Two variations:
  1. Test nodes are (incrementally) inserted into training graphs
  2. Test graphs are disjoint and completely unseen

- Requires **generalizing across arbitrary graph structures**, thus many transductive methods are useless
Spectral vs Spatial

Euclidean space/grid

Non-Euclidean space/graph

Fixed domain

Change one single edge

Variable domain

Standard ConvNets

Spectral graph ConvNets

Spectral NNs offer rich families of spectral filters

Can we still use spectral graph ConvNets?

Are spectral filters transferable?
Spatial Convolution Methods

- Spatial approaches define **convolutions directly on the graph**, operating on spatially close neighbors.
- Major challenge: defining convolution with **different sized neighborhoods** and maintaining the **local invariance** of CNNs.
Different weight matrices are used for nodes with different degrees,

$$x = h_v + \sum_{i=1}^{N_v} h_i$$

$$h'_v = \sigma(xW_{N_v}^L)$$

where $W_{N_v}^L$ is the weight matrix for nodes with degree $N_v$ at layer $L$

**Problem**: doesn’t scale to graphs with very wide degree distributions
Transition matrices are used to define the neighborhood for nodes in DCNN. For node classification, it has

\[ H = f \left( W^c \odot P^* X \right) \]  

- \( X \) is an \( N \times F \) matrix of input features (\( N \) is the number of nodes and \( F \) is the number of features)
- \( P^* \) is an \( N \times K \times N \) tensor which contains the power series \( \{P, P^2, ..., P^K\} \) of matrix \( P \) (where \( P \) is the degree-normalized transition matrix from the adjacency matrix \( A \))
  - \( P^* \) transforms each node into a \( K \times F \) diffusion convolutional representation
General inductive framework

Restricts every degree to be the same (by sampling a fixed-size set a node’s local neighborhood, during both training and inference)

\[
\begin{align*}
    h^t_{\mathcal{N}_v} &= \text{AGGREGATE}_t (\{h^{t-1}_u, \forall u \in \mathcal{N}_v\}) \\
    h^t_v &= \sigma (W^t \cdot [h^{t-1}_v \parallel h^t_{\mathcal{N}_v}])
\end{align*}
\]

Key is that \(W^t\) is not shared across time steps
Spatial Convolution Methods
GraphSAGE - Hamilton et. al.,2017

- General inductive framework
- Restricts every degree to be the same (by sampling a fixed-size set a node’s local neighborhood, during both training and inference)

\[
\begin{align*}
    h_{N_v}^t &= \text{AGGREGATE}_t \left( \{ h_{u}^{t-1}, \forall u \in N_v \} \right) \\
    h_{v}^t &= \sigma \left( W^t \cdot [h_{v}^{t-1} \| h_{N_v}^t] \right)
\end{align*}
\]  

(10)

- Key is that \( W^t \) is not shared across time steps \( T \)
### Spatial Methods: Summary

<table>
<thead>
<tr>
<th>Variant</th>
<th>Aggregator</th>
<th>Updater</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional networks in [33]</td>
<td>$h^t_{N_v} = h^{t-1}<em>v + \sum</em>{k=1}^{N_v} h^{t-1}_k$</td>
<td>$h^t_v = \sigma(h^t_{N_v} W^N_{L_v})$</td>
</tr>
<tr>
<td>DCNN</td>
<td>Node classification: $N = P^*X$</td>
<td>$H = f(W^c \odot N)$</td>
</tr>
<tr>
<td></td>
<td>Graph classification: $N = \frac{1}{N} P^*X/N$</td>
<td></td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>$h^t_{N_v} = \text{AGGREGATE}_t \left( {h^{t-1}_u, \forall u \in N_v } \right)$</td>
<td>$h^t_v = \sigma(W^t \cdot [h^{t-1}_v</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets
Gating

Gated Graph Neural Networks (GGNNs) - Li et al. 2016

- Extension of original GNN (Scarselli et. al. 2009)
- Propagate for $T$ steps, but do not restrict the propagation model to be contractive
  - Use gating in the propagation step to alleviate gradient issues
Gating

Initial idea: update $h_v^t$ using neighborhood summation and tanh

$$a_v^t = b + \sum_{j \in N_v} h_j^{t-1}$$  \hspace{1cm} (11)

$$h_v^t = \tanh(Wa_v^t)$$  \hspace{1cm} (12)

Now, extend this to incorporate gating mechanisms, to prevent full overwrite of $h_v^{t-1}$ by $h_v^t$
Gating

Gated Graph Neural Networks (GGNNs) - Li et al. 2016

Gating mechanism defined similar to LSTM:

\[
a_v^t = b + \sum_{j \in \mathcal{N}_v} h_j^{t-1}
\]

\[
z_v^t = \sigma (W^z a_v^t + U^z h_v^{t-1})
\]

\[
r_v^t = \sigma (W^r a_v^t + U^r h_v^{t-1})
\]

\[
\tilde{h}_v^t = \tanh (W a_v^t + U (r_v^t \odot h_v^{t-1}))
\]

\[
h_v^t = (1 - z_v^t) \odot h_v^{t-1} + z_v^t \odot \tilde{h}_v^t
\]

Where \(\odot\) is element-wise multiplication, and \(z_v\) and \(r_v\) are the update and reset vectors, respectively.

Where \(\odot\) is element-wise multiplication, and \(z_v\) and \(r_v\) are the update and reset vectors, respectively.
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets
- Gating mechanisms are designed for data that changes sequentially; however, our graphs have static features
- GAT incorporates attention mechanisms into the propagation step
Attention

The agreement on the European Economic Area was signed in August 1992.
GAT uses a *self-attention* mechanism to compute the hidden states of each node by attending over itself and its neighbors.
Graph attention network (GAT) - Velickovic et. al., 2017

\[ e_{vj} = a(W^a h_v, W^b h_j) \] (18)

\[ \alpha_{vj} = \frac{\exp(e_{vj})}{\sum_{k \in N_v} \exp(e_{ik})} \] (19)

\[ h_v = \sigma \left( \sum_{j \in N_v} \alpha_{vj} W^z h_j \right) \] (20)
(a) Graph Convolution Networks [14] explicitly assign a non-parametric weight $a_{ij} = \frac{1}{\sqrt{\text{deg}(v_i) \text{deg}(v_j)}}$ to the neighbor $v_j$ of $v_i$ during the aggregation process.

(b) Graph Attention Networks [15] implicitly capture the weight $a_{ij}$ via an end to end neural network architecture, so that more important nodes receive larger weights.
## Gating and Attention

<table>
<thead>
<tr>
<th>Name</th>
<th>Variant</th>
<th>Aggregator</th>
<th>Updater</th>
</tr>
</thead>
</table>
| **Graph Attention Networks** | GAT     | $\alpha_{vk} = \frac{\exp(\text{LeakyReLU}(a^T[Wh_v \parallel Wh_k]))}{\sum_{j \in N_v} \exp(\text{LeakyReLU}(a^T[Wh_v \parallel Wh_j]))}$  
$ h^t_{\mathcal{N}_v} = \sigma \left( \sum_{k \in \mathcal{N}_v} \alpha_{vk} Wh_k \right)$  
Multi-head concatenation:  
$ h^t_{\mathcal{N}_v} = \|_{m=1}^M \sigma \left( \sum_{k \in \mathcal{N}_v} \alpha^m_{vk} W^m h_k \right)$  
Multi-head average:  
$ h^t_{\mathcal{N}_v} = \sigma \left( \frac{1}{M} \sum_{m=1}^M \sum_{k \in \mathcal{N}_v} \alpha^m_{vk} W^m h_k \right) $ | $ h^t_v = h^t_{\mathcal{N}_v} $ |
| **Gated Graph Neural Networks** | GGNN    | $ h^t_{\mathcal{N}_v} = \sum_{k \in \mathcal{N}_v} h^t_{k} - 1 + b $                                                                                                                                 | $ z^t_v = \sigma(W^z h^t_{\mathcal{N}_v} + U^z h^t_{v} - 1) $  
$ r^t_v = \sigma(W^r h^t_{\mathcal{N}_v} + U^r h^t_{v} - 1) $  
$ h^t_v = \tanh(W h^t_{\mathcal{N}_v} + U (r^t_v \odot h^t_{v} - 1)) $  
$ h^t_v = (1 - z^t_v) \odot h^t_{v} - 1 + z^t_v \odot \tilde{h}^t_v $ |
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets
Skip Connections
Highway GCN - Rahimi et. al., 2018

Uses layer-wise gates. The output of a layer is summed with its input with gating weights (inspired by Highway nets)

\[
T(h^t) = \sigma \left( W^t h^t + b^t \right)
\]

\[
h^{t+1} = h^{t+1} \odot T(h^t) + h^t \odot (1 - T(h^t))
\]  

(21)
Skip Connections
Jump Knowledge Network (JKN) - Xu et. al., 2018

- With neighborhood aggregation, the receptive field of each node grows exponentially w.r.t. the number of layers (steps) $T$
- The Jump Knowledge Network selects from all of the intermediate representations for each node at the last layer
  - Allows the model adapt the effective neighborhood size for each node as needed
Propagation Methods: Summary

- Convolutional Aggregator
- Attention Aggregator
- Gate Updater
- Skip connection

- Graph Convolutional Networks
  - Spectral Methods
  - Spatial
  - Non-spectral Methods

- Graph Attention Networks
  - GRU
  - LSTM

- Gated Graph Neural Networks
  - Graph LSTM

- Spectral Network
- GCN
- ChebNet
- Molecular Convolutional Network
- MoNet
- DCNN
- GraphSAGE

Highway GNN
Jump Knowledge Network
1 Introduction
2 Graph Neural Networks
3 Graph Variants
4 Objective Variants
5 Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6 Training Variants
7 Applications and Datasets
GraphSAGE solved the problems of the original GCN by replacing full graph Laplacian with learnable aggregation functions, which are key to generalize to unseen nodes.

In addition, GraphSAGE uses neighbor sampling to alleviate receptive field expansion.
Chen et. al (2018) proposed a control-variate based stochastic approximation algorithms by utilizing the historical activations of nodes as a control variate.

This limits the receptive field in the 1-hop neighborhood, but is efficient.
Li et. al. (2018) note that GNNs requires many additional labeled data for validation and also suffers from the localized nature of the convolutional filter.

To solve the limitations, the authors propose a method to find the nearest neighbors of training data and a boosting-like method.
Training Methods

- Neighborhood Sampling
  - GraphSAGE
  - FastGCN
  - Adaptive
- Receptive Field Control
  - Control Variate
  - Co-training GCN
  - Self-training GCN
- Boosting
Application Areas

(a) physics

(b) molecule

(c) image

(d) text
## Commonly Used Datasets

<table>
<thead>
<tr>
<th>Category</th>
<th>Dataset</th>
<th>Source</th>
<th># Graphs</th>
<th># Nodes</th>
<th># Edges</th>
<th># Features</th>
<th># Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Citation Networks</td>
<td>Cora</td>
<td>[103]</td>
<td>1</td>
<td>2708</td>
<td>5429</td>
<td>1433</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Citeseer</td>
<td>[103]</td>
<td>1</td>
<td>3327</td>
<td>4732</td>
<td>3703</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Pubmed</td>
<td>[103]</td>
<td>1</td>
<td>19717</td>
<td>44338</td>
<td>500</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>DBLP</td>
<td>dblp.uni-trier.de<a href="aminer.org/citation">105</a></td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Social Networks</td>
<td>BlogCatalog</td>
<td>[107]</td>
<td>1</td>
<td>10312</td>
<td>333983</td>
<td>-</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Reddit</td>
<td>[24]</td>
<td>1</td>
<td>232965</td>
<td>11606919</td>
<td>602</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Epinions</td>
<td><a href="http://www.epinions.com">www.epinions.com</a></td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Chemical/Biological Graphs</td>
<td>PPI</td>
<td>[109]</td>
<td>24</td>
<td>56944</td>
<td>818716</td>
<td>50</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>NCI-1</td>
<td>[110]</td>
<td>4100</td>
<td>-</td>
<td>-</td>
<td>37</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>NCI-109</td>
<td>[110]</td>
<td>4127</td>
<td>-</td>
<td>-</td>
<td>38</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>MUTAG</td>
<td>[111]</td>
<td>188</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>D&amp;D</td>
<td>[112]</td>
<td>1178</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>QM9</td>
<td>[113]</td>
<td>133885</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>tox21</td>
<td>tripod.nih.gov/tox21/challenge/</td>
<td>12707</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>Unstructured Graphs</td>
<td>MNIST</td>
<td>yann.lecun.com/exdb/mnist/</td>
<td>70000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Wikipedia</td>
<td><a href="http://www.mattmahoney.net/dc/textdata">www.mattmahoney.net/dc/textdata</a></td>
<td>1</td>
<td>4777</td>
<td>184812</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>20NEWS</td>
<td>[114]</td>
<td>1</td>
<td>18846</td>
<td>-</td>
<td>-</td>
<td>20</td>
</tr>
</tbody>
</table>