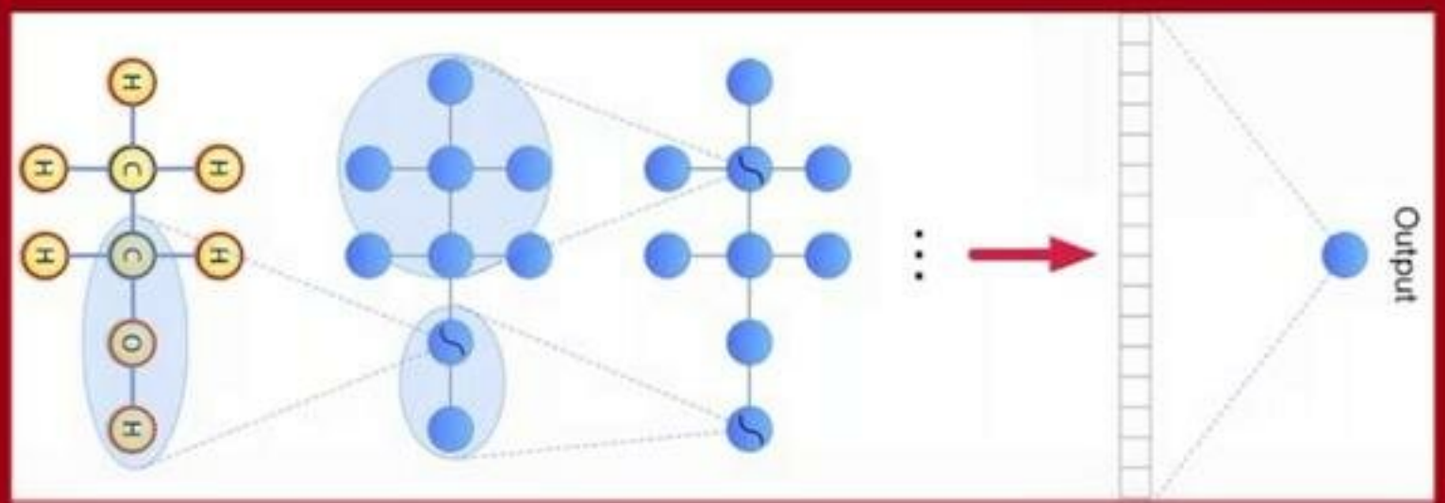
# Learning effective representations with Graph Neural Networks




*Nicolò Navarin*

*Assistant Professor*
*Department of Mathematics*
*University of Padua*

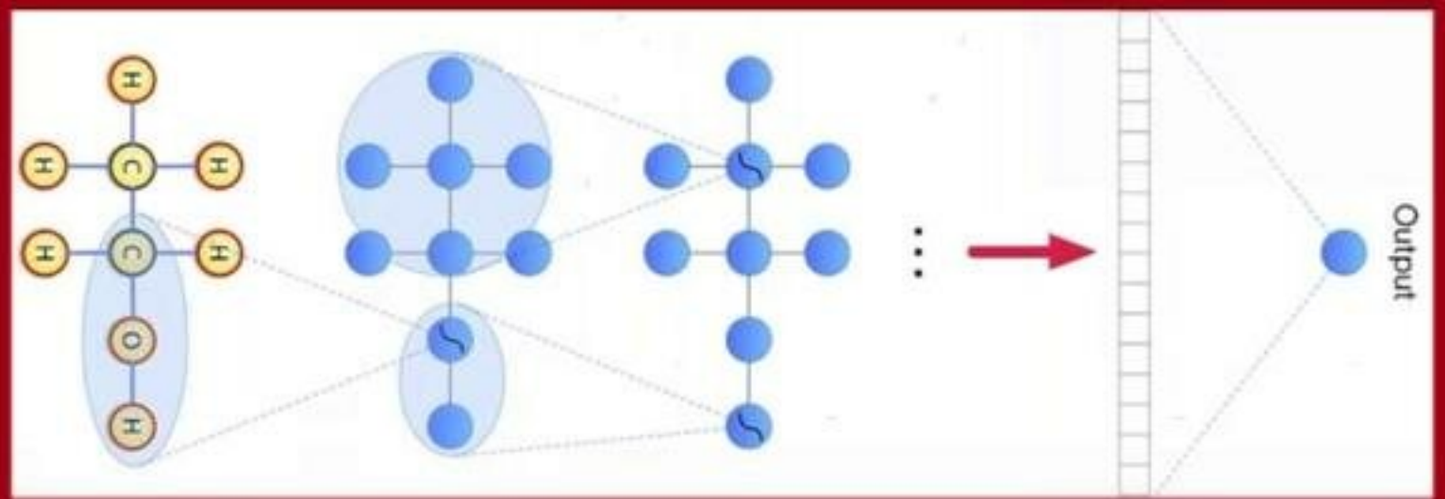
Talk @ TU Wien
24/11/2020

Università degli Studi di Padova

Dipartimento Matematica

# Learning kernel-based embeddings in Graph Neural Networks

# Learning effective representations with Graph Neural Networks




*Nicolò Navarin*

*Assistant Professor*
*Department of Mathematics*
*University of Padua*

Talk @ TU Wien
24/11/2020

Università degli Studi di Padova

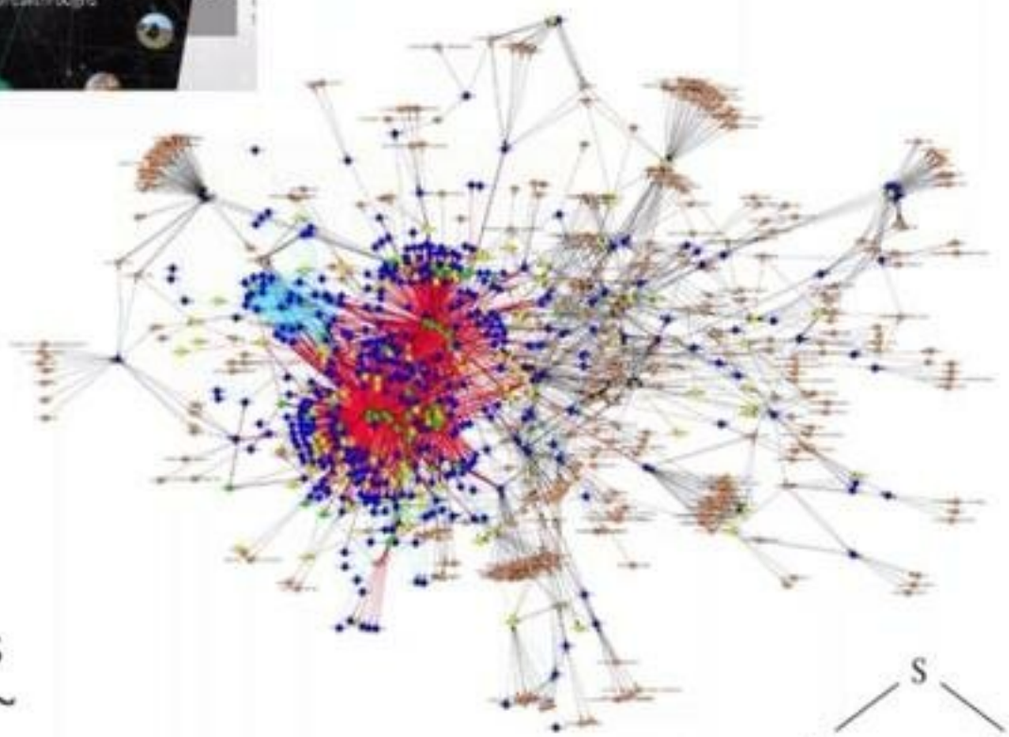
Dipartimento Matematica

# Graphs are everywhere..



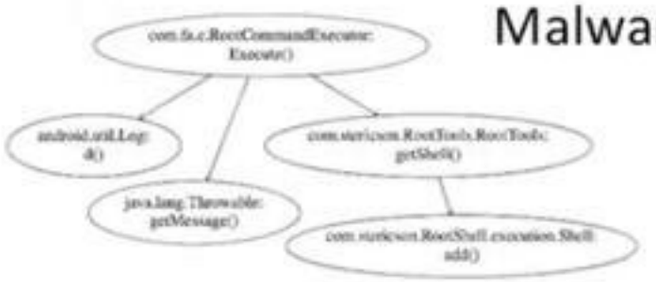
Knowledge graphs


Gene networks



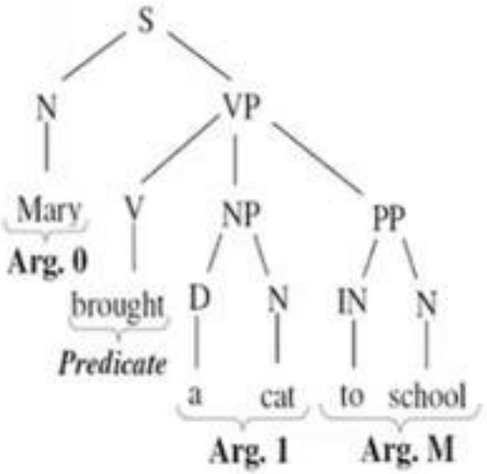
- In several settings it is natural to represent data as graphs.
- We may want to predict some property:
  - **over Graphs** or
  - **over Graph Nodes**

Malware API calls



Chemical compounds


Parse trees




Images

# Classification/Regression on Graphs

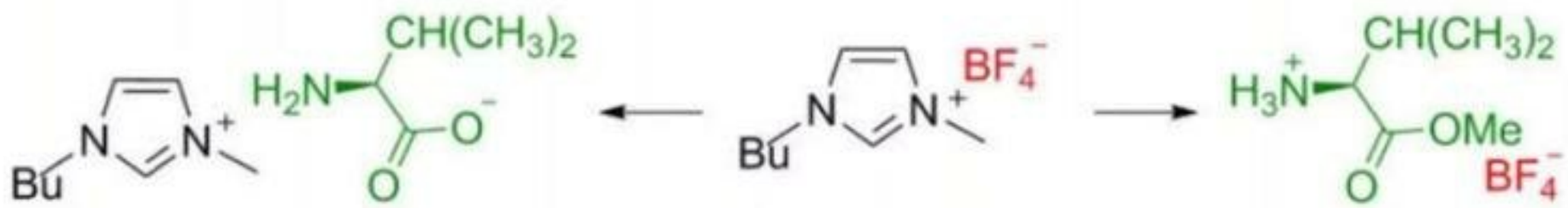


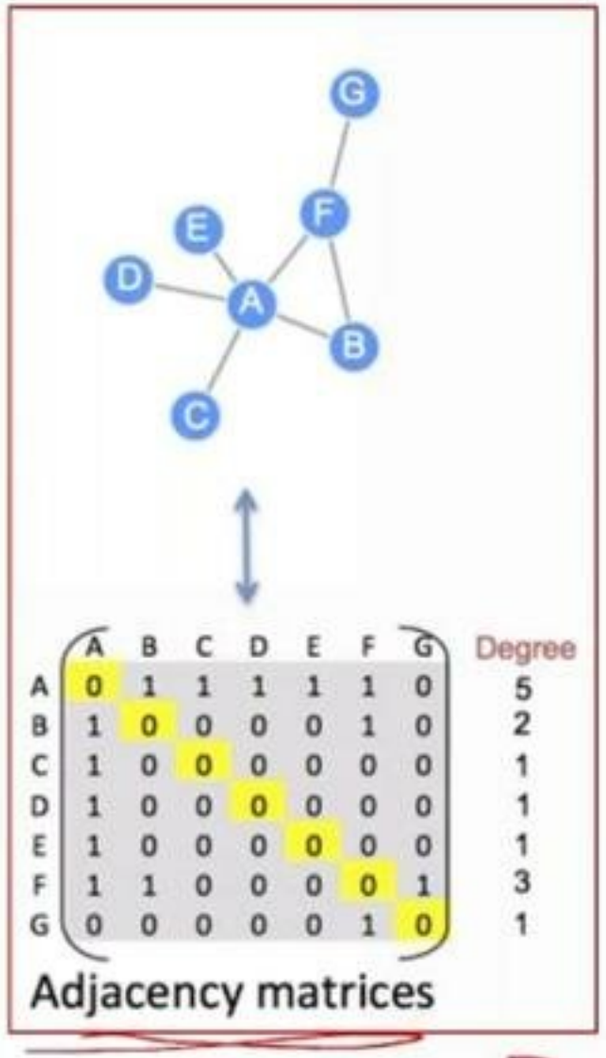
In this talk, classification/regression on graphs

Low toxicity
IC50=25.2 mM

Moderate toxicity
IC50=11.3 mM

High toxicity
IC50=6.6 mM



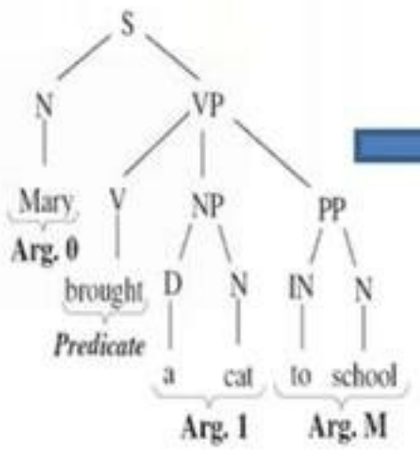




A D

- Dataset composed of N pairs $\{(G_i, y_i), 1 \leq i \leq N\}$
  - Each graph:
    - $n_i$ vertices
    - (possibly) discrete label associated to each node: l(v)
    - $d$ vectorial **attributes** associated to each node: a(v) or $\boxed{X} \in \mathbb{R}^{n_i \times d}$
- Given an unseen graph $G$, the task is to predict the correct target $y$

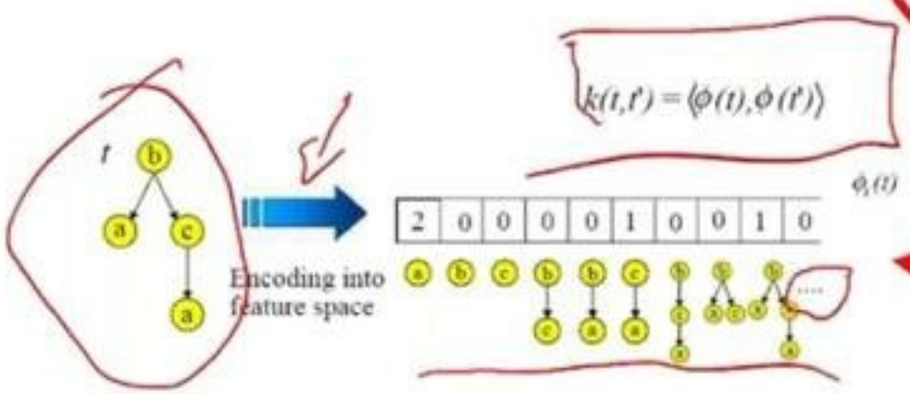
# Machine learning on Graphs



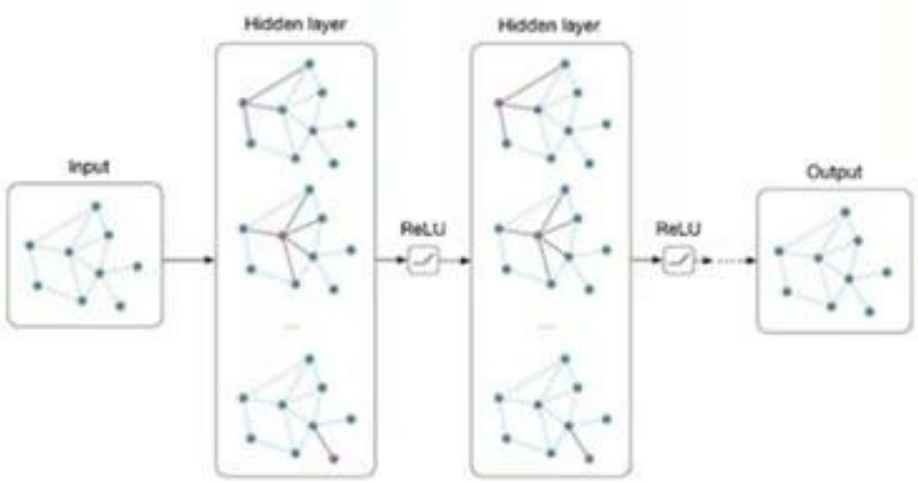



$$\phi : \mathfrak{X} \to N^s$$

$$[\underbrace{1}_{\#S}, \underbrace{3}_{\#N}, \underbrace{1}_{\dots}, 1, 1, 1, 1, \underbrace{1}_{\#IN}]$$




$$k(t,t') = \langle \phi(t), \phi(t') \rangle$$

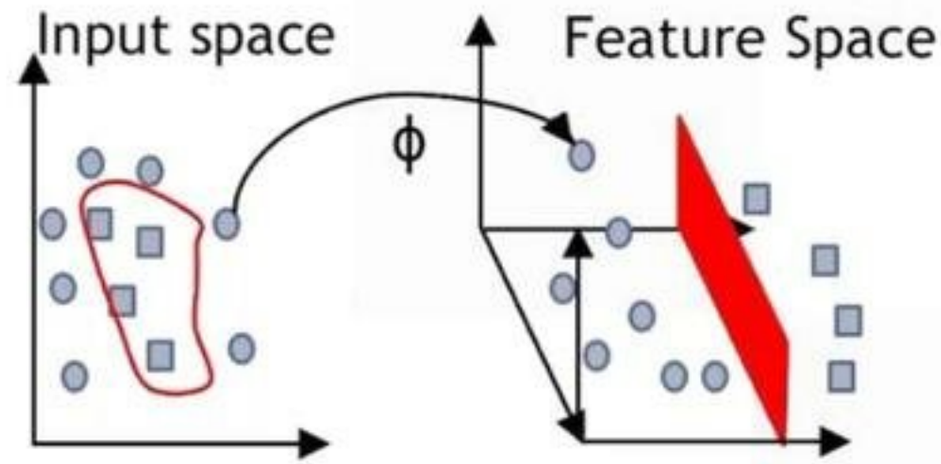


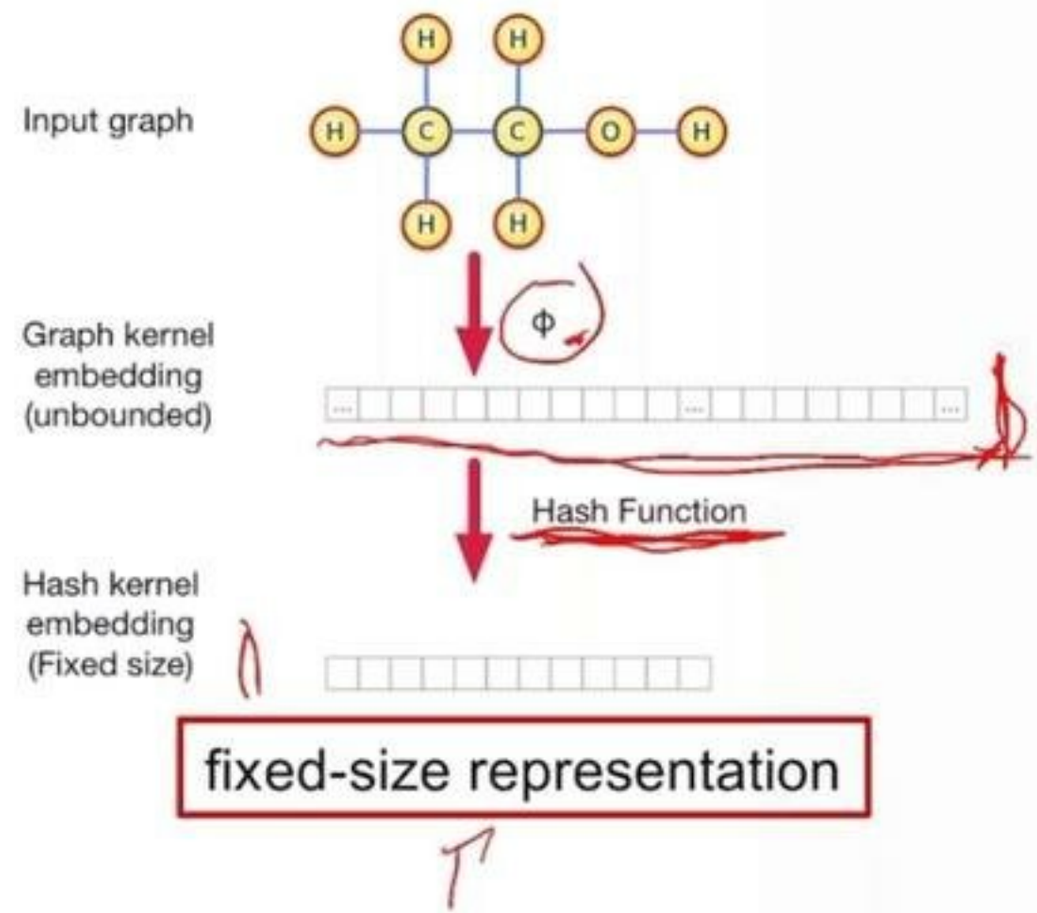

- It is difficult to cast "mainstream" learning algorithms to work on graphs.
- 3 possible ways:
  1. Define a ad-hoc vector representation for graphs
  2. Kernel methods: 2 components
     - kernel (similarity) function on graphs
     - "kernelized" learning algorithms
       - access examples only via dot products.
     - Fixed, infinite-dimensional representation
  3. Define new learning algorithms able to deal directly with graphs (e.g. Graph Neural Networks, Graph Attention Networks..)

# Graph Kernels

$$k(X,Y) = \phi(X)^T \phi(Y)$$

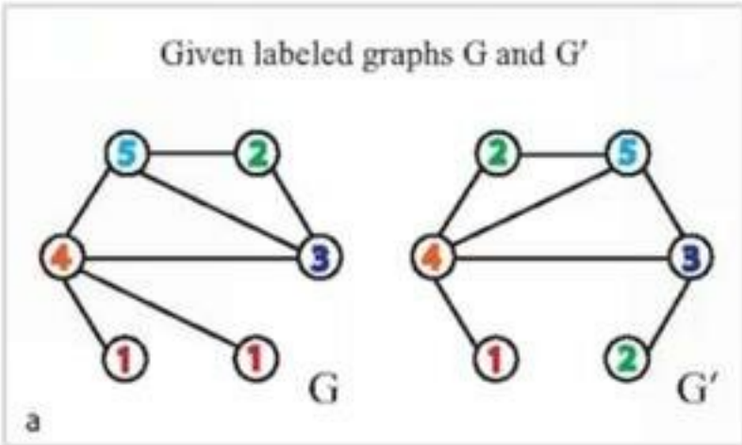
- Implicitly define a mapping from input space (the space of graphs) to feature space.
- Several proposals in literature:
  - Global measures: Random walks, Shortest paths, Graphlets
  - Local measures:
    - Weisfeiler-Lehman subtree kernel
    - Ordered Decomposition DAGs kernel
    - Explicit feature space, hashing to obtain a fixed-size representation

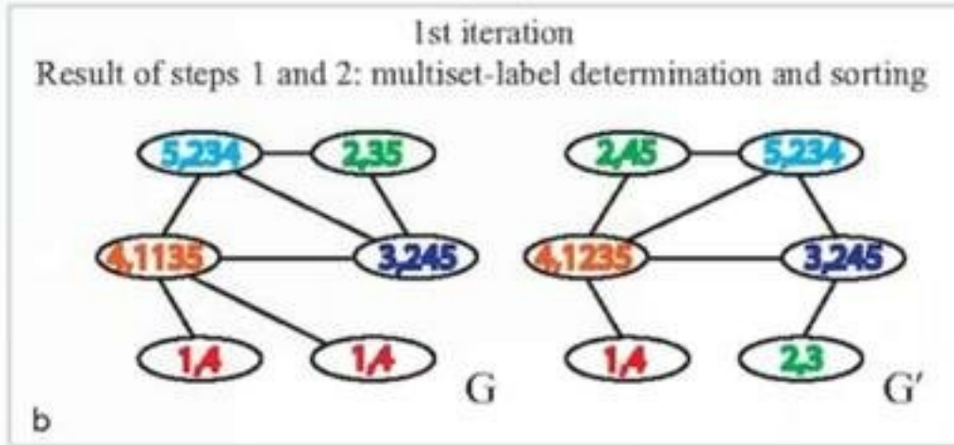


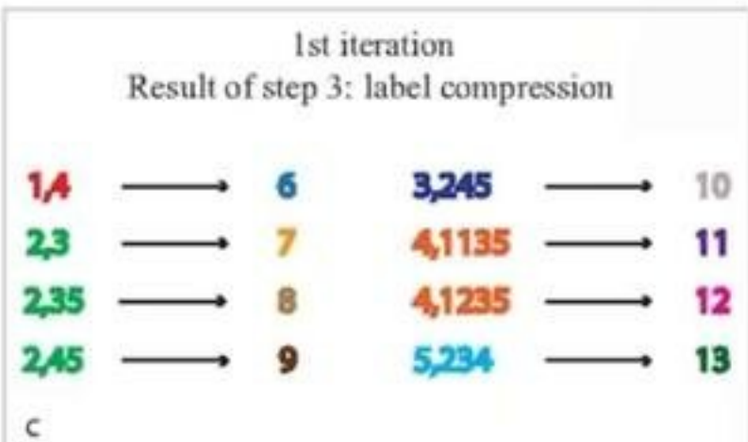



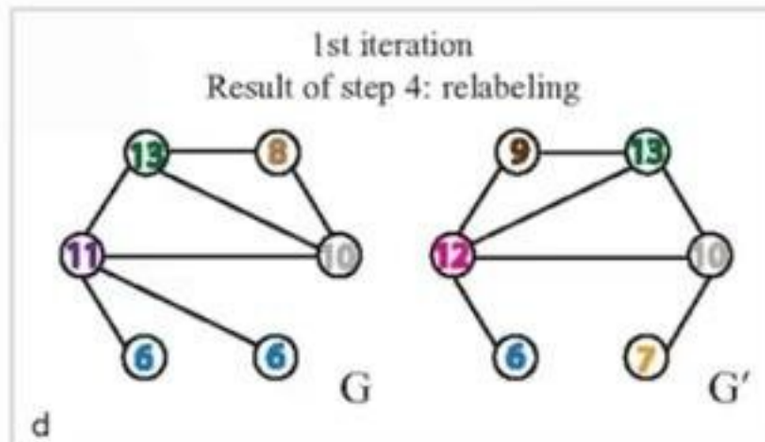
# Weisfeiler-Lehman Subtree kernel

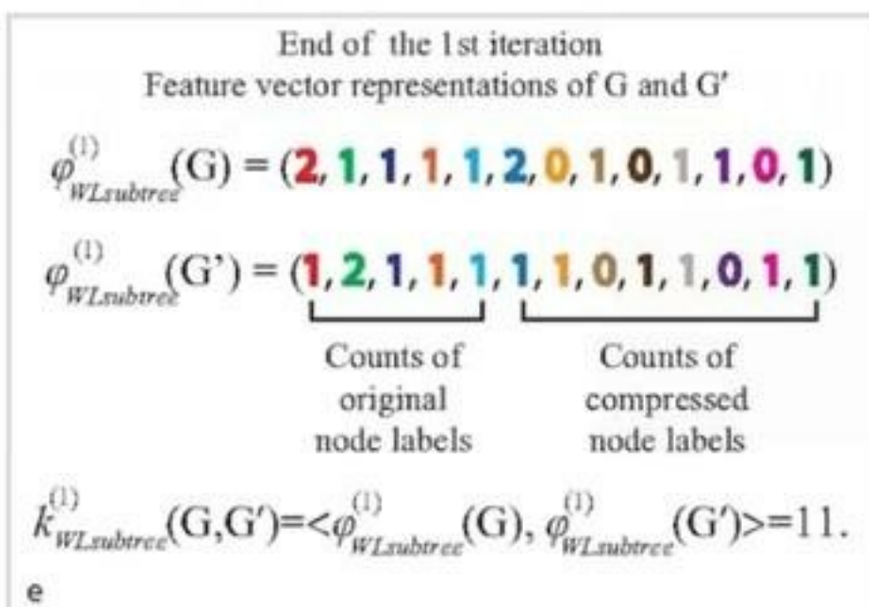


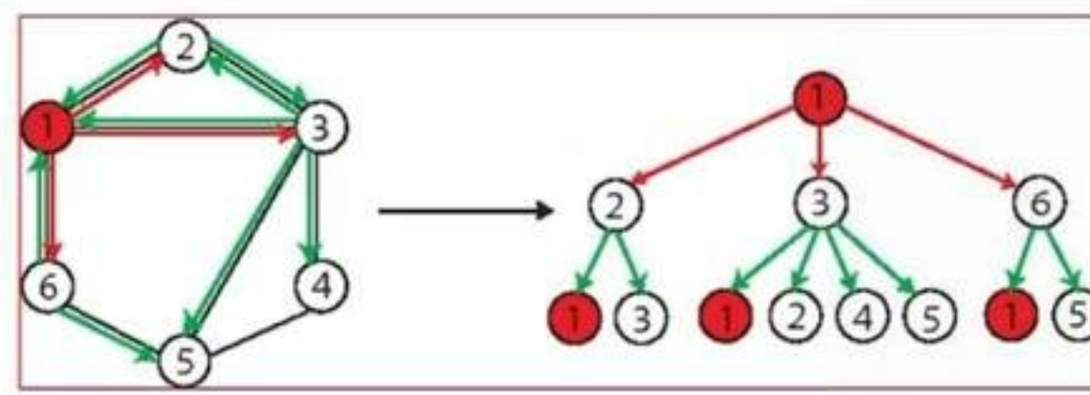


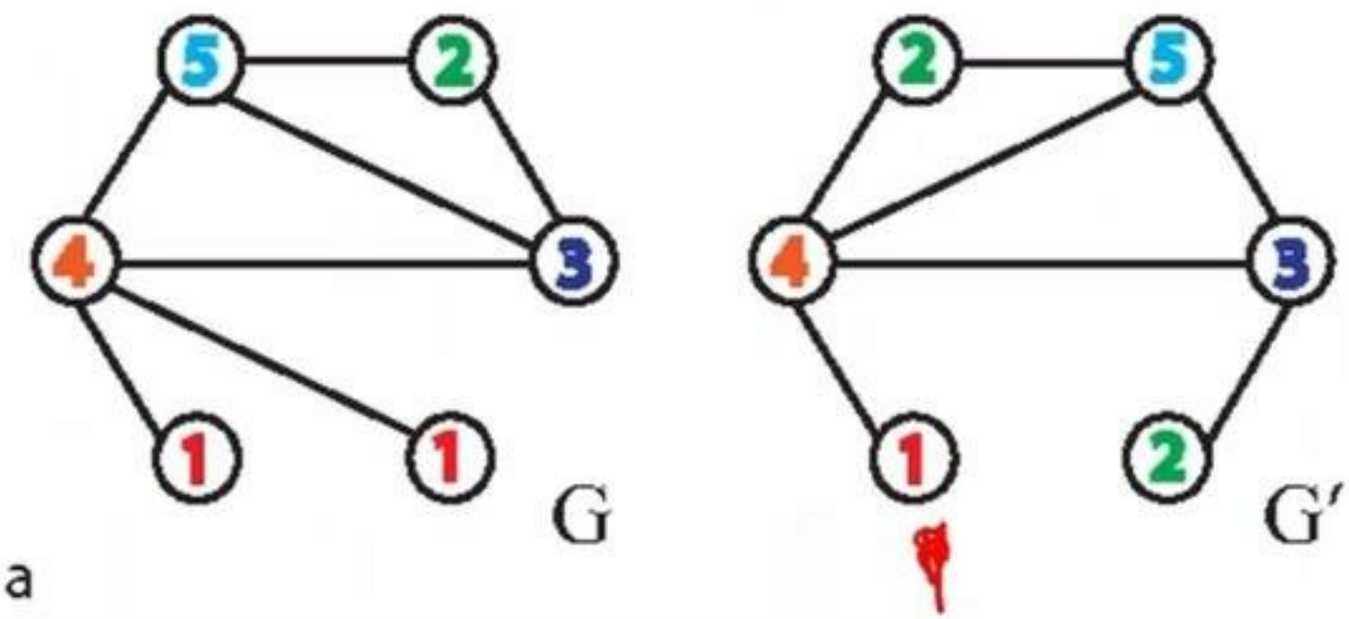



Figures from: Shervashidze at al. Weisfeiler-Lehman Graph Kernels, JMLR 2011

# Weisfeiler-Lehman Subtre

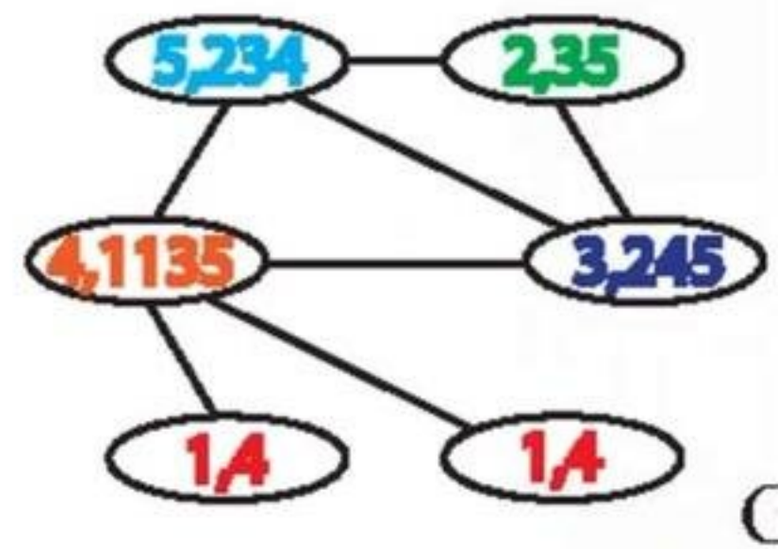
Given labeled graphs G and G′




a

1st ite

Result of steps 1 and 2: multiset




b

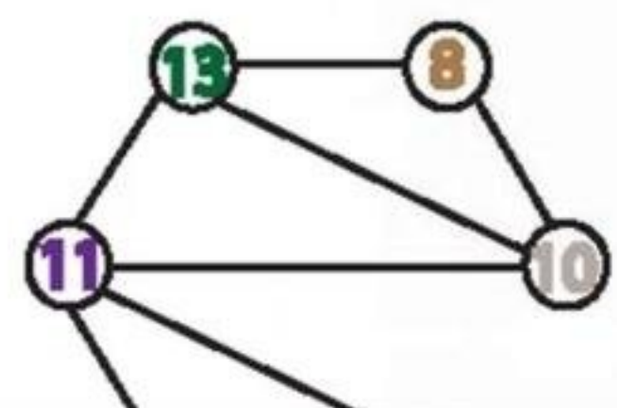
1st iteration

Result of step 3: label compression

| | | | |
|---|---|---|---|
| 1,4 → | 6 | 3,245 → | 10 |
| 2,3 → | 7 | 4,1135 → | 11 |
| 2,35 → | 8 | 4,1235 → | 12 |

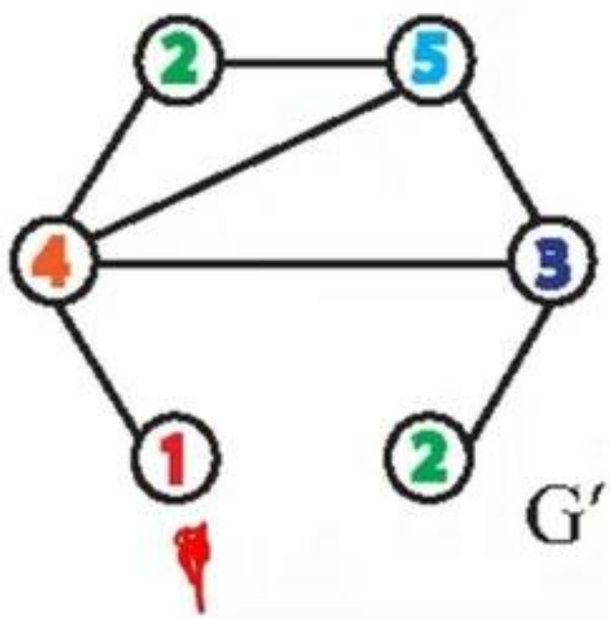
1st ite

Result of step

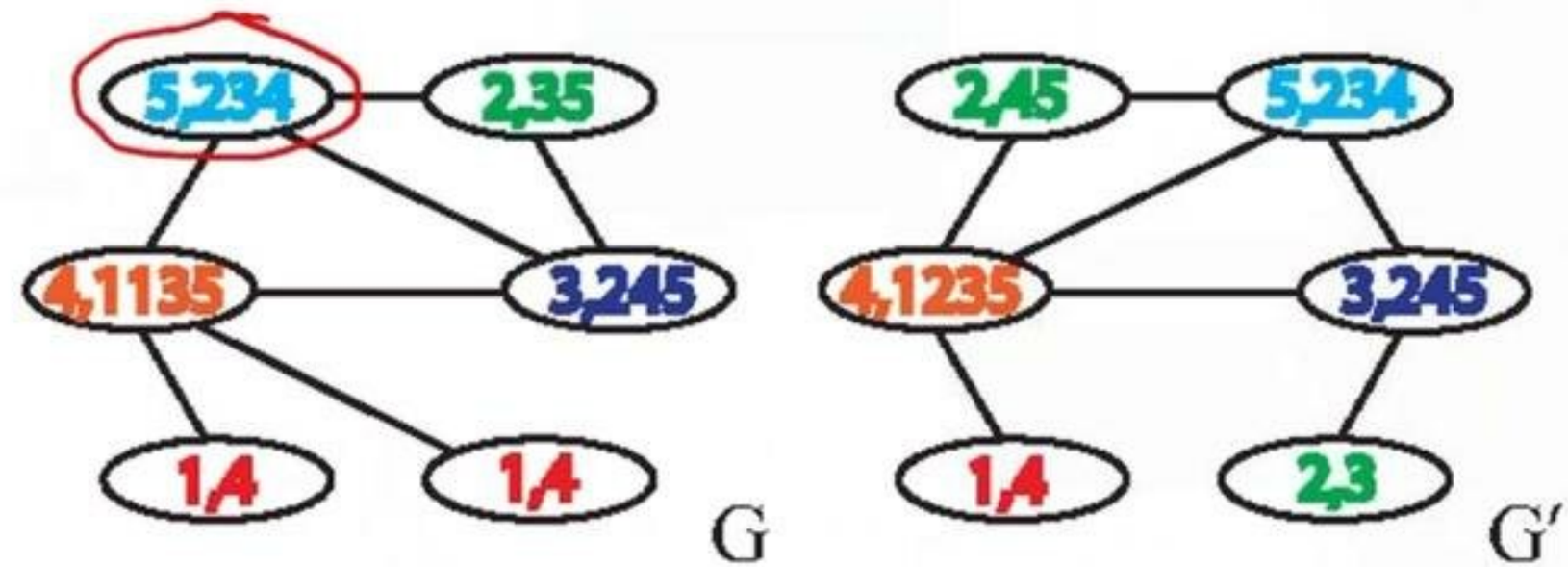


# er-Lehman Subtree kernel

aphs G and G′




1st iteration
Result of steps 1 and 2: multiset-label determination and sorting

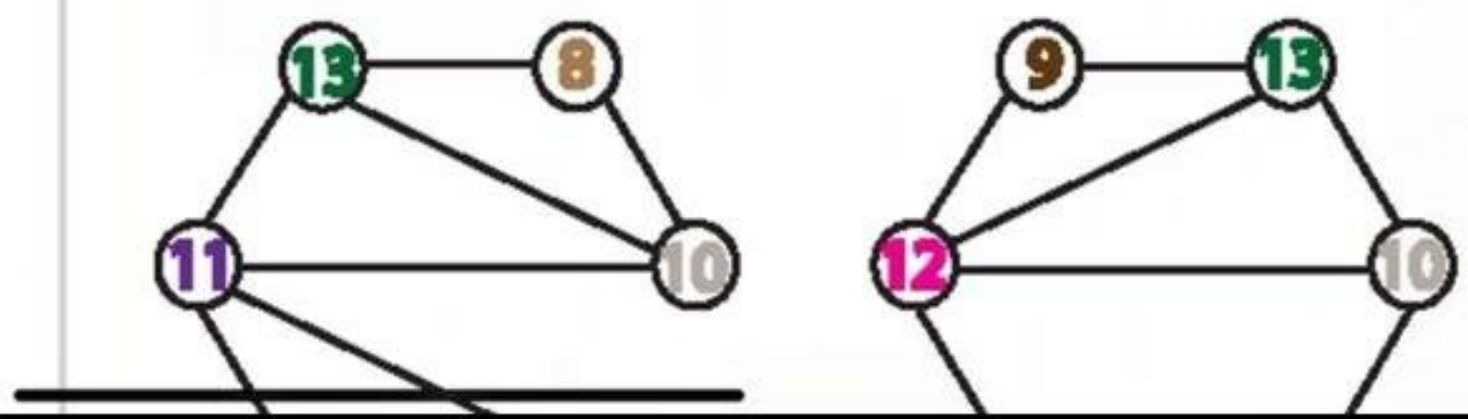



b

ation
bel compression

| | | |
|---|---|---|
| 3,245 | → | 10 |
| 4,1135 | → | 11 |
| 4,1235 | → | 12 |

1st iteration
Result of step 4: relabeling



# Brief History of GNNs

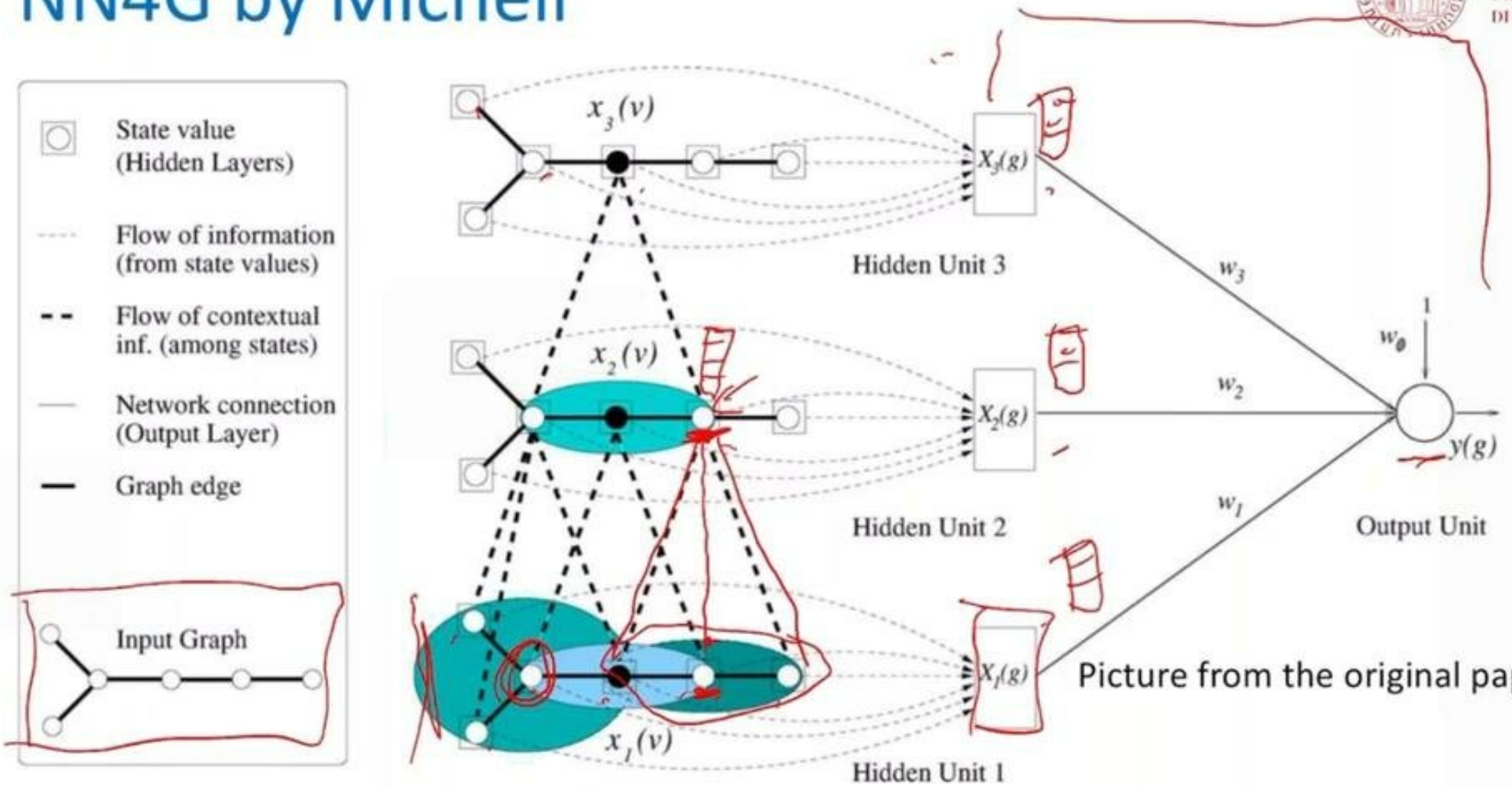


- The idea of Neural Networks for structured data dates back to '97 [Sperduti & Starita, 1997]
- In the '00s, two proposals:
  - Graph Neural Network Model [Scarselli, Gori et al., 2009]
    - Recurrent model, contraction mapping
  - Neural Networks for Graphs [Micheli, 2009]
    - Convolutional model, layer-wise training
- Recently, many works proposing slight modifications, e.g. :
  - [Tarlow et al., 2016] Extends [Scarselli, Gori et al., 2009]
    - no contraction mapping, GRUs
  - [Kipf & Welling, 2017] proposes an approach similar to [Micheli, 2009] for node classification
    - end-to-end

Sperduti & Starita (1997). *Supervised neural networks for the classification of structures. IEEE TNNs.*
Scarselli, Gori et al. (2009). *The Graph Neural Network Model. IEEE TNNs.*
Micheli (2009). *Neural network for graphs: A contextual constructive approach. IEEE TNNs.*
Kipf & Welling(2017). *Semi-Supervised Classification with Graph Convolutional Networks.* In *ICLR.*
Tarlow et al. (2016). *Gated Graph Sequence Neural Networks.* In *ICLR.*

# NN4G by Micheli

Università degli Studi di Padova

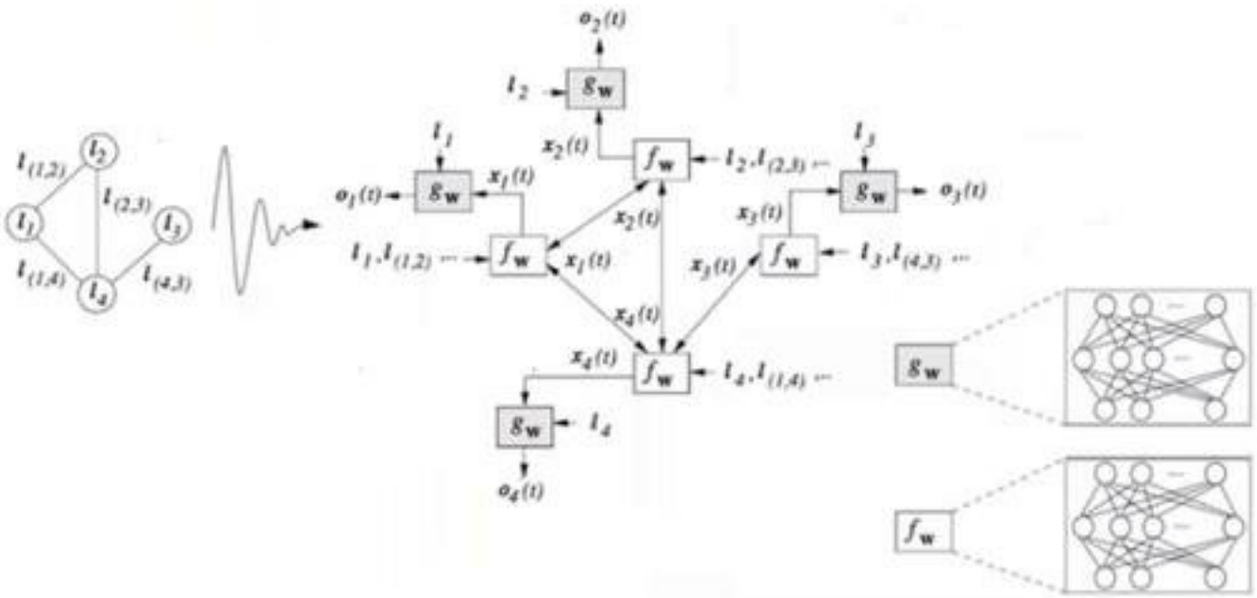


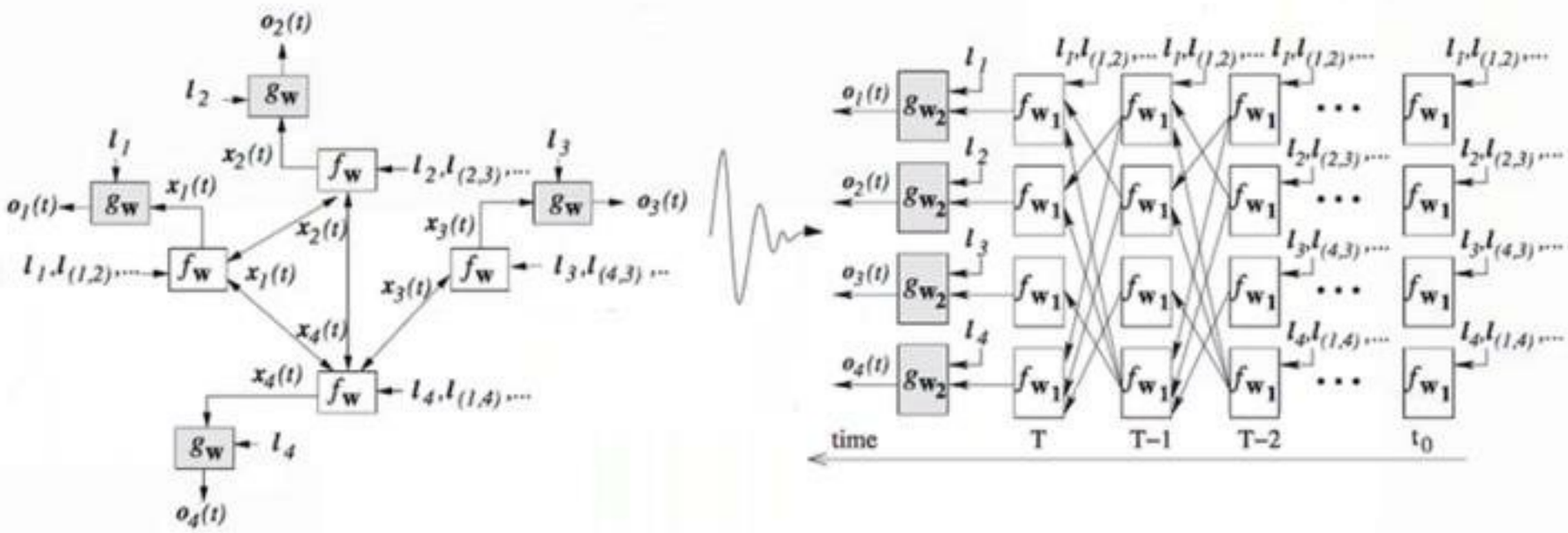

Picture from the original paper.

- Each convolution takes as input the representation of all previous layers
- Trained layer-wise (cascade correlation)
- Readout: a representation per-graph per-layer is computed using the sum

# GNN by Scarselli, Gori et al.

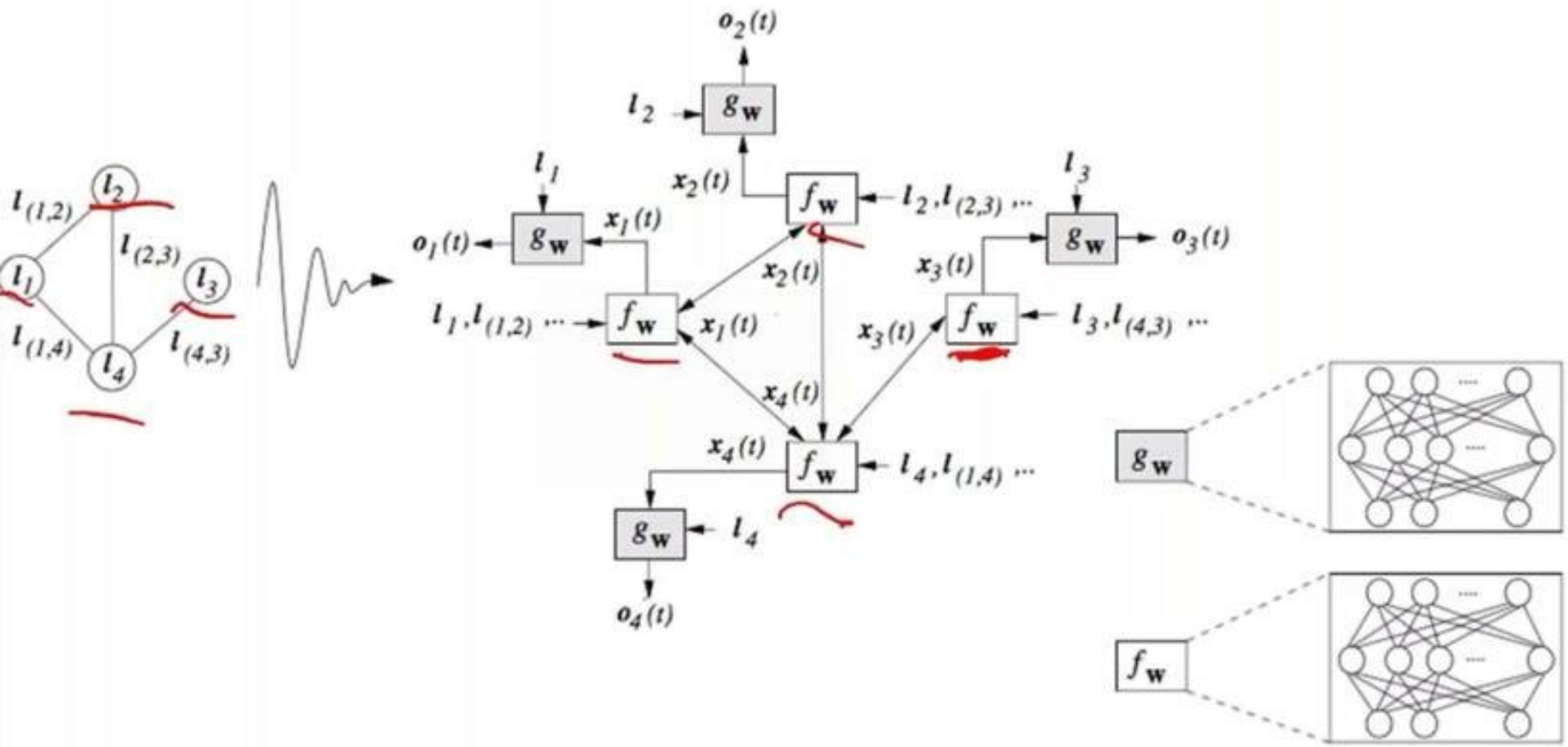


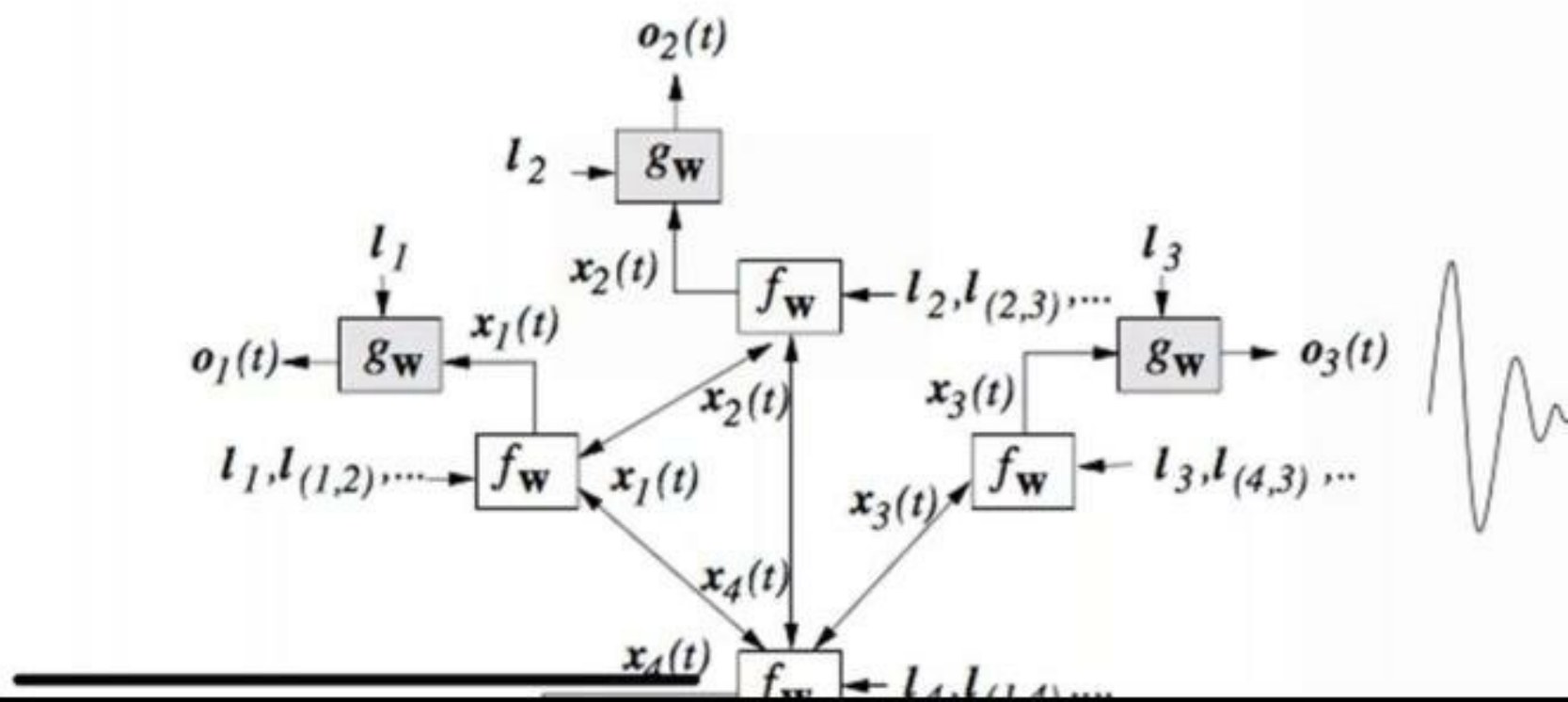





- Recurrent neurons, but similar in spirit to NN4G.
- The unrolled network is close to NN4G, but:
  - tied parameters
  - defined as a contraction mapping

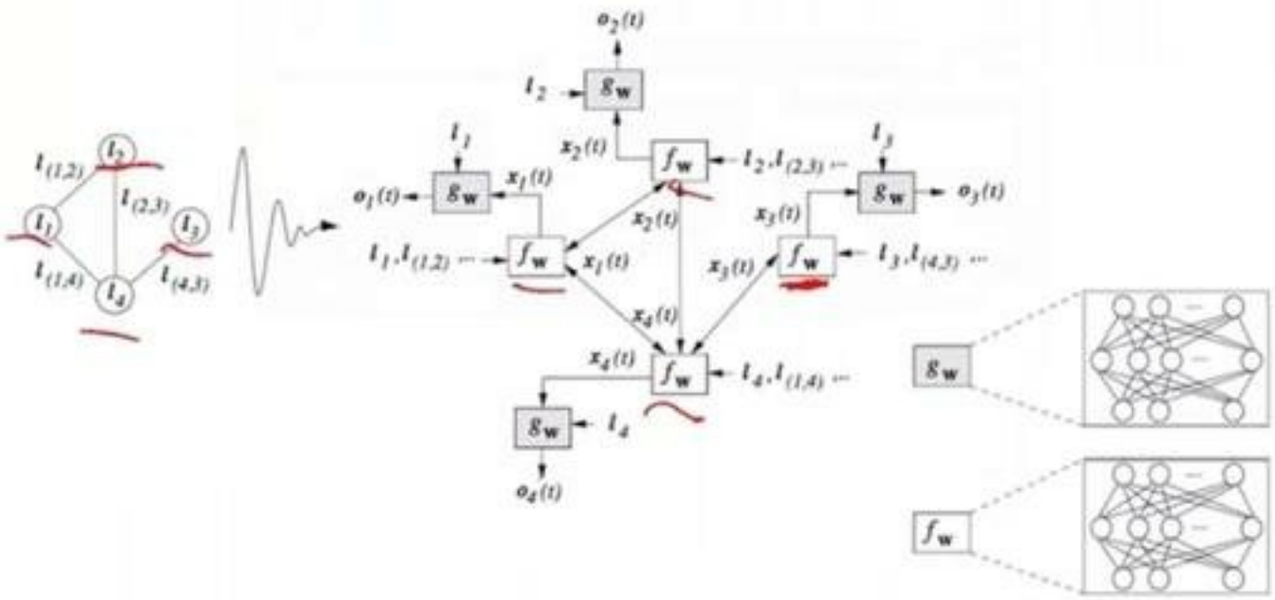
# GNN by Scarselli, Gori et al.

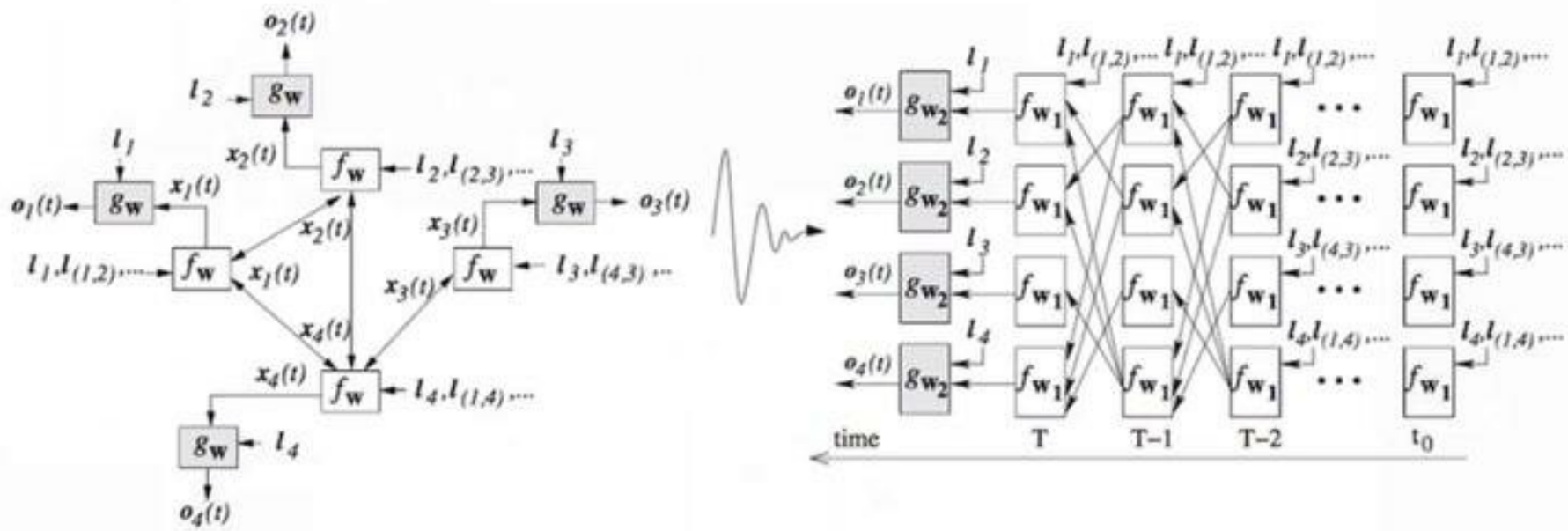


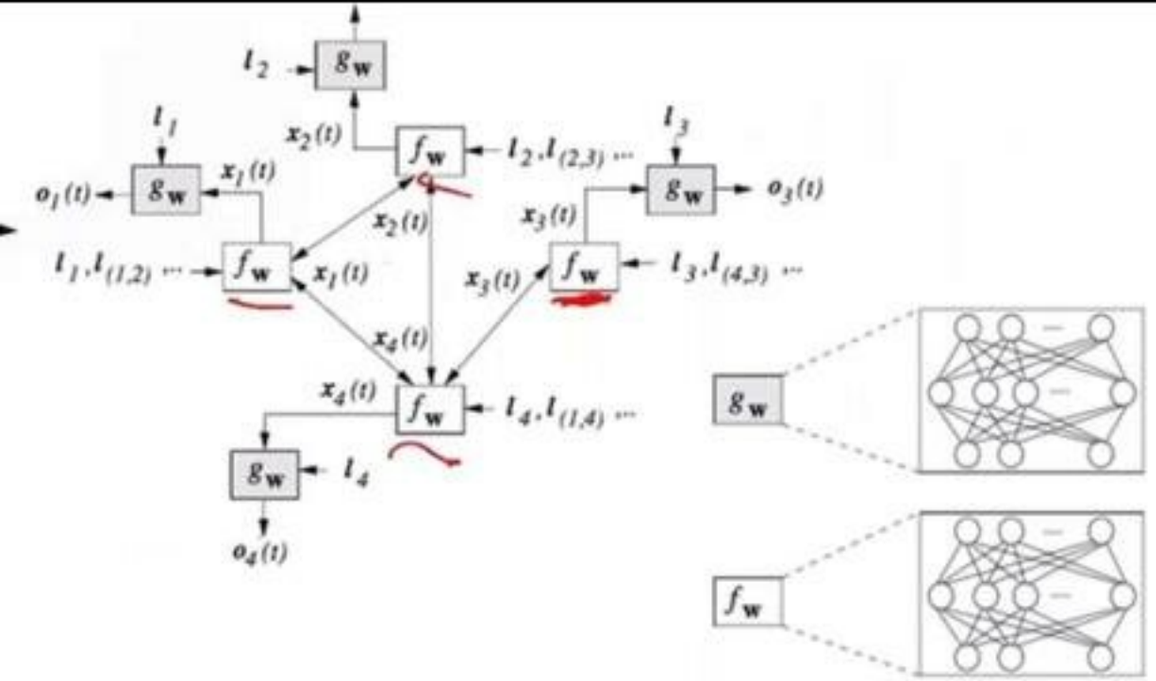
# GNN by Scarselli, Gori et al.

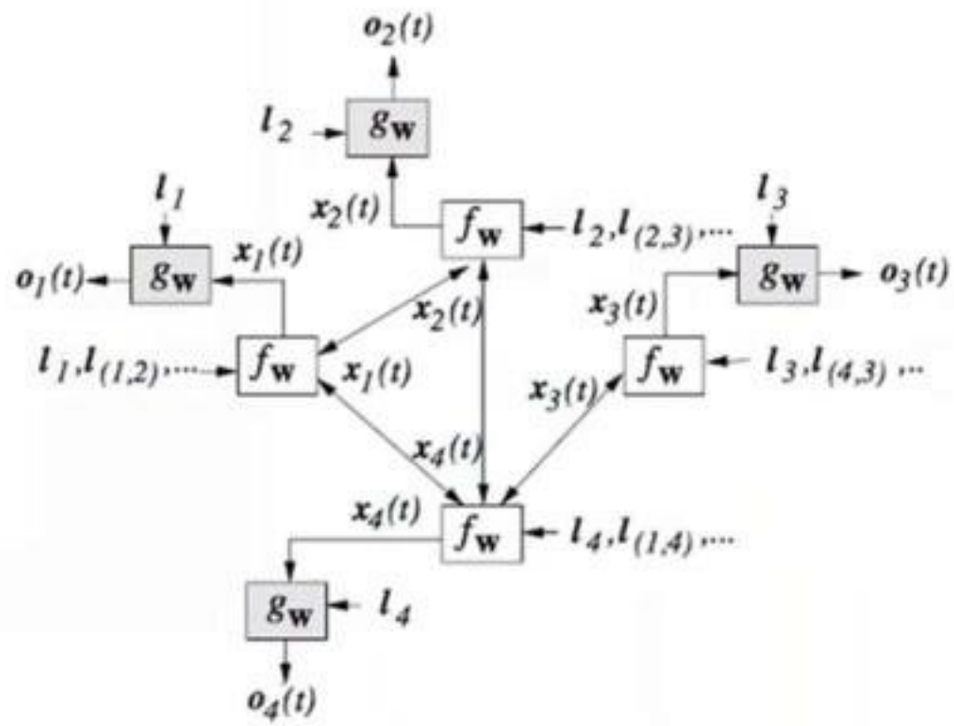


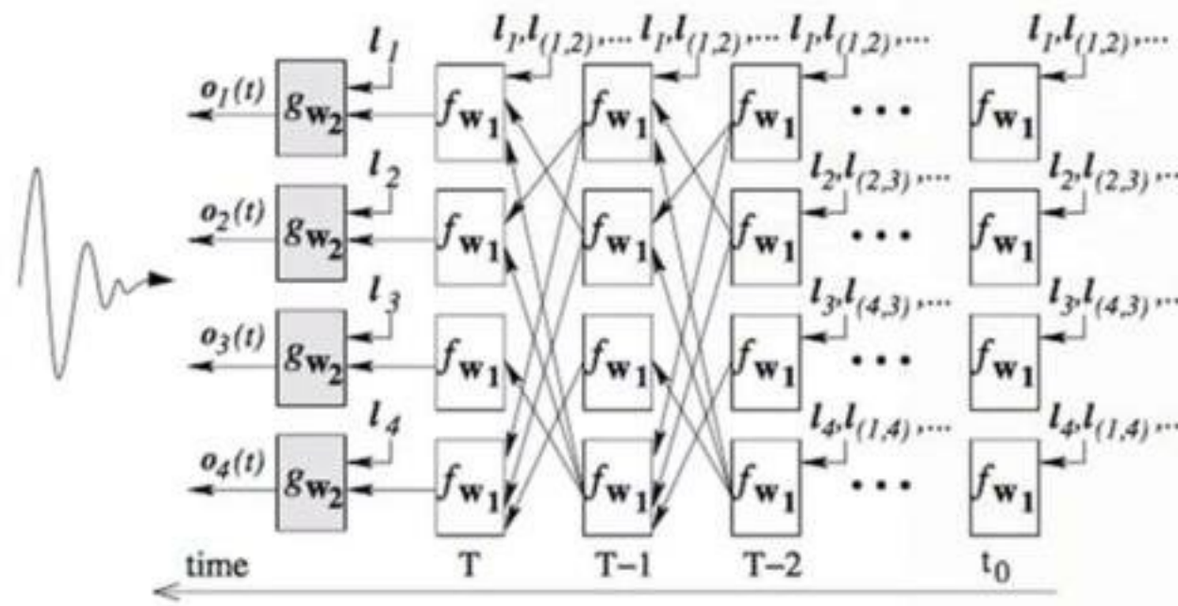





- Recurrent neurons, but similar in spirit to NN4G.
- The unrolled network is close to NN4G, but:
  - tied parameters
  - defined as a contraction mapping



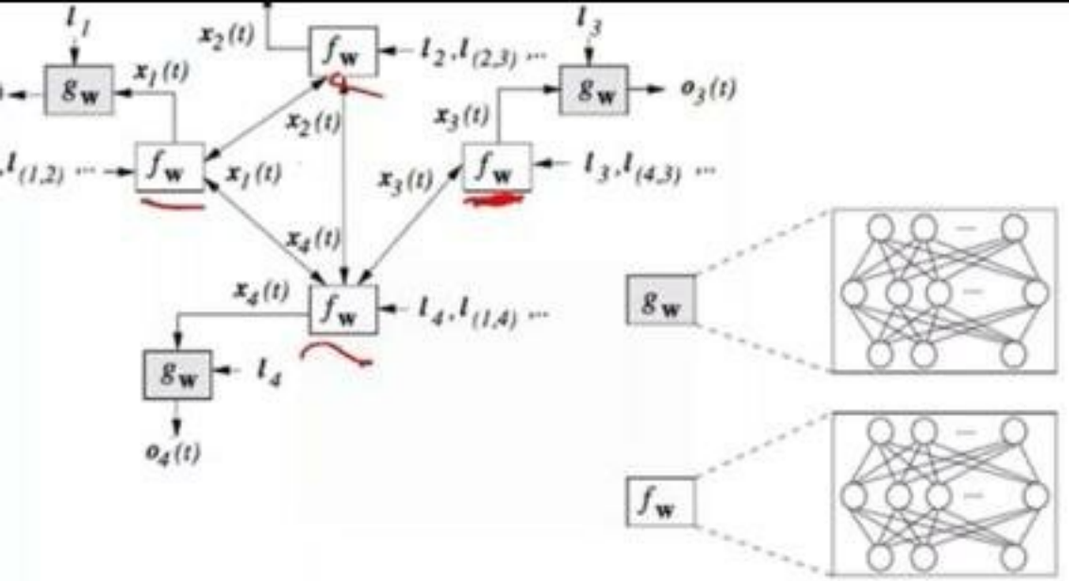


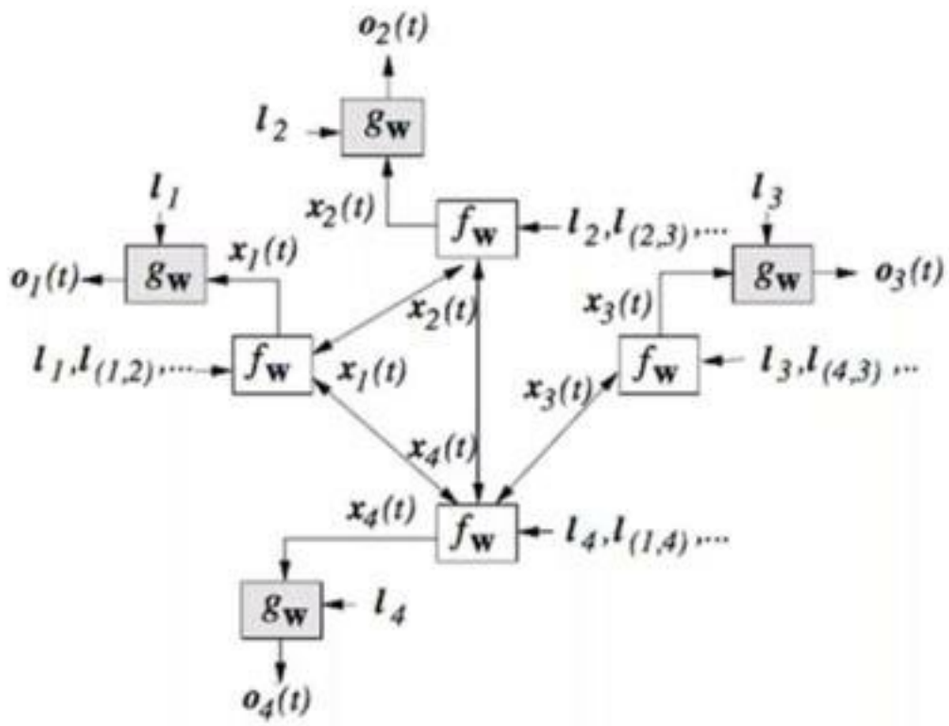


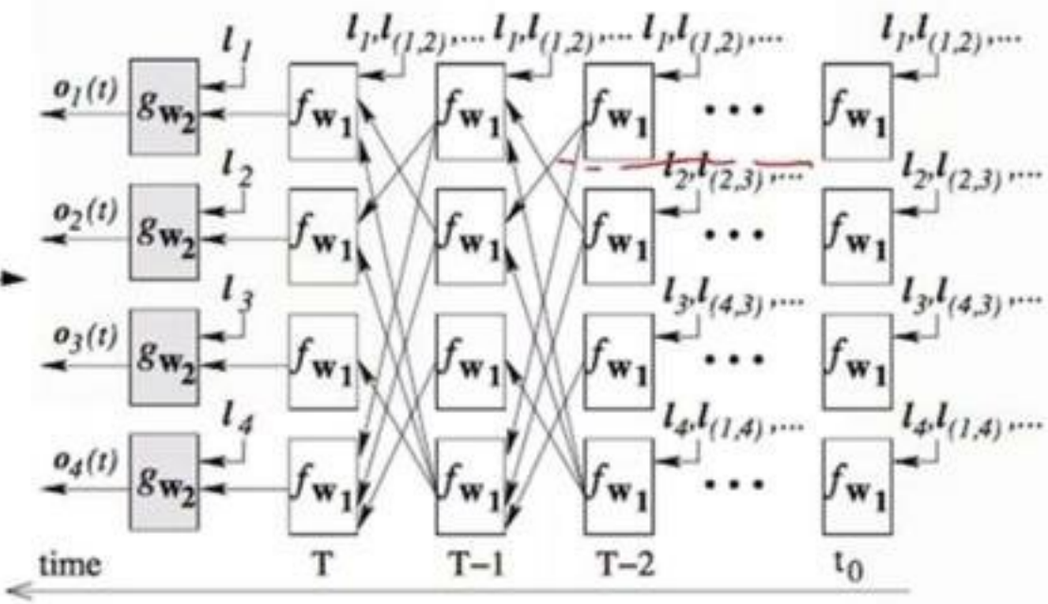

urrent neurons, but similar in spirit to NN4G.

unrolled network is close to NN4G, but:

- tied parameters
- defined as a contraction mapping








rent neurons, but similar in spirit to NN4G.

nrolled network is close to NN4G, but:

tied parameters

defined as a contraction mapping

# Convolution operator (on images)



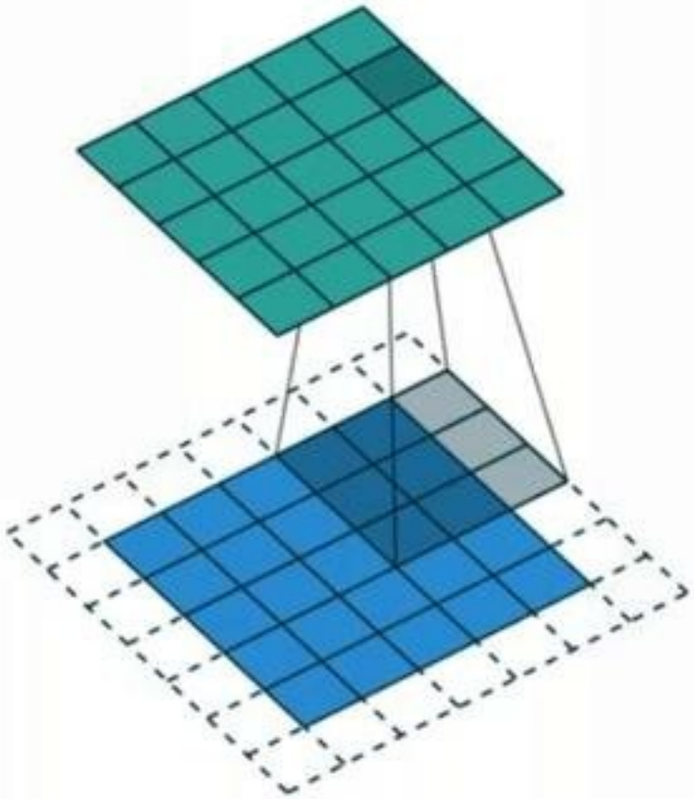
General definition for filter $f$ and signal $x$:

$$(f * x)(t) = \int_{-\infty}^{\infty} f(\tau)x(t-\tau)\, d\tau$$

In images, it corresponds to

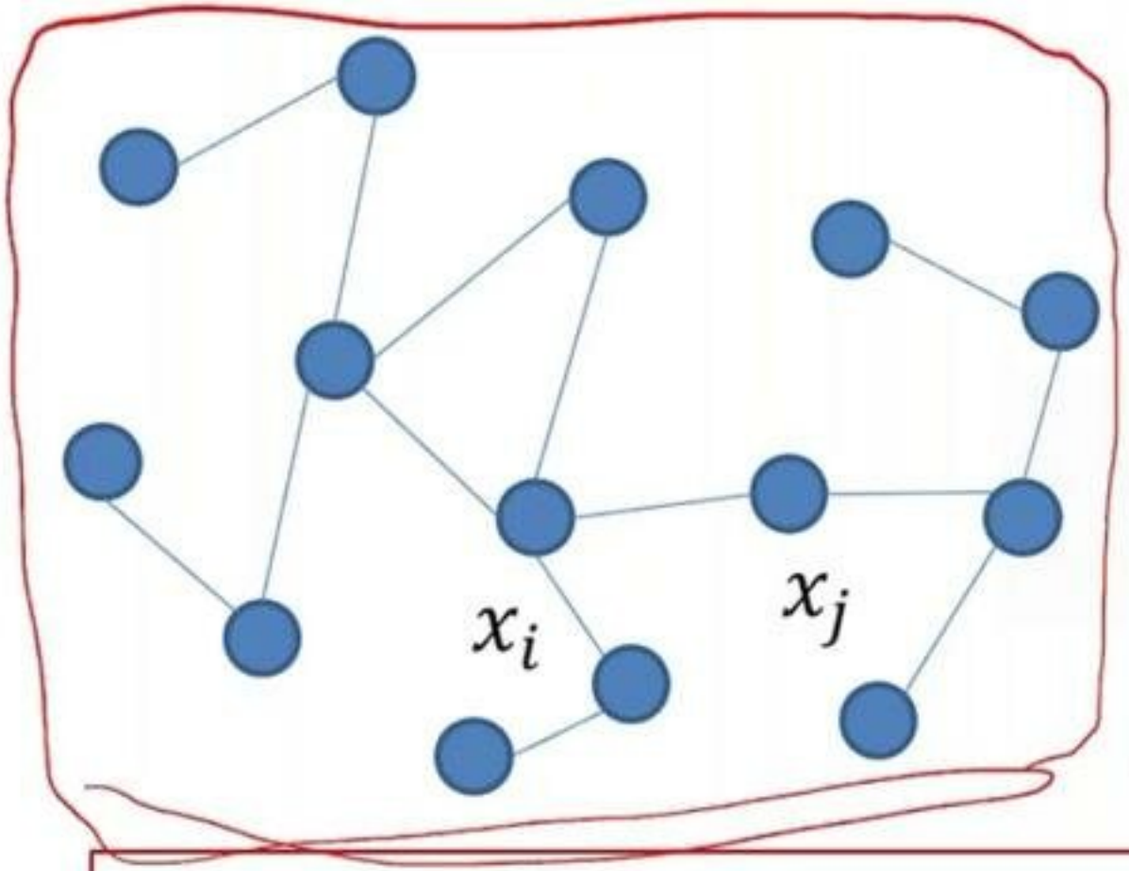
$$(f * x)\,(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(s,t)x(i-s, j-t)$$

Where $f$ is a $2a \times 2b$ filter and $x$ an image

# How to define convolution on graphs?






Consider a simple setting:

- single undirected graph
- $x$: $V \to \mathbb{R}$: a signal on the nodes of a graph
  - Represented as vector $x \in \mathbb{R}^n$

The convolution operator is difficult to define in the vertex domain

Convolution Theorem:

- Convolution in one domain (time, space)
- corresponds to pointwise multiplication in frequency domain

$$\widehat{f * g} = \hat{f} \odot \hat{g}$$

$\hat{f}$ : Fourier transform of $f$

- $\odot$ Hadamard (element-wise) product

# How to define convolution on graphs?



Main steps:

- Graph Fourier Transform
  - Fourier Basis are eigenvectors of normalized Graph Laplacian

$$L = U \Lambda U^T$$

- We can then define the graph convolution in the frequency domain

$$f *_G x = U \hat{F} U^T x$$

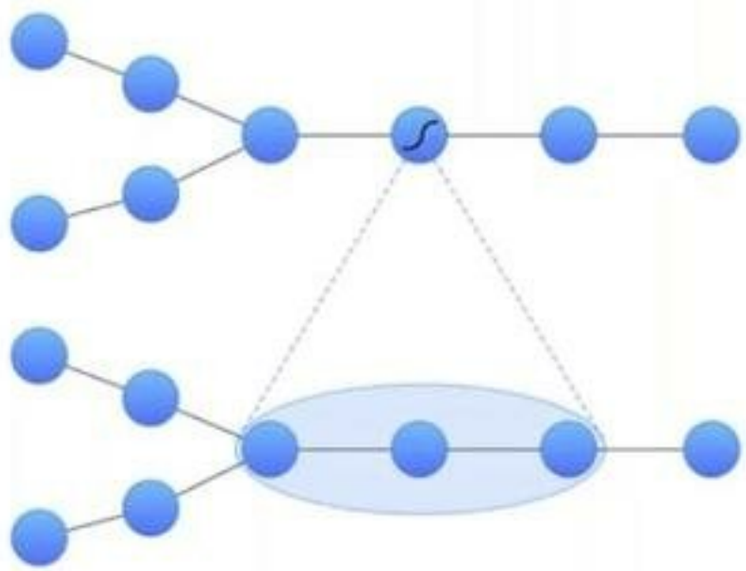
where $\hat{F} = diag(\hat{f})$

- For some choice of filters, e.g. polynomials of the spectral matrix
  - The convolution can be computed in the node space directly

$$\hat{F}_\Theta = \sum_{k=0}^{K} \theta_k \Lambda^k \rightarrow f *_G x = \sum_{k=0}^{K} \theta_k L^k x$$

# Summary of Graph Convolution





- **1-localized** GCN maps multisets of representations (node and neighbours at the previous layer) to a new one:

$$H_v^{l+1} = f\left(\{H_v^l, H_u^l, \forall u \in ne(v)\}\right)$$

  - where $H^0 = X$

- $f$: linear mapping & non-linear activation function, e.g.

$$H^{l+1} = \sigma\left(\breve{D}^{-1/2}\breve{A}\breve{D}^{-1/2}H^l\Theta^l\right)$$

The convolution operator can be generalized to be more expressive than 1-WL [6]

If $f$ is expressive enough (and with an injective readout), a multilayer 1-localized GCN is as expressive ad the 1-dim WL isomorphism test

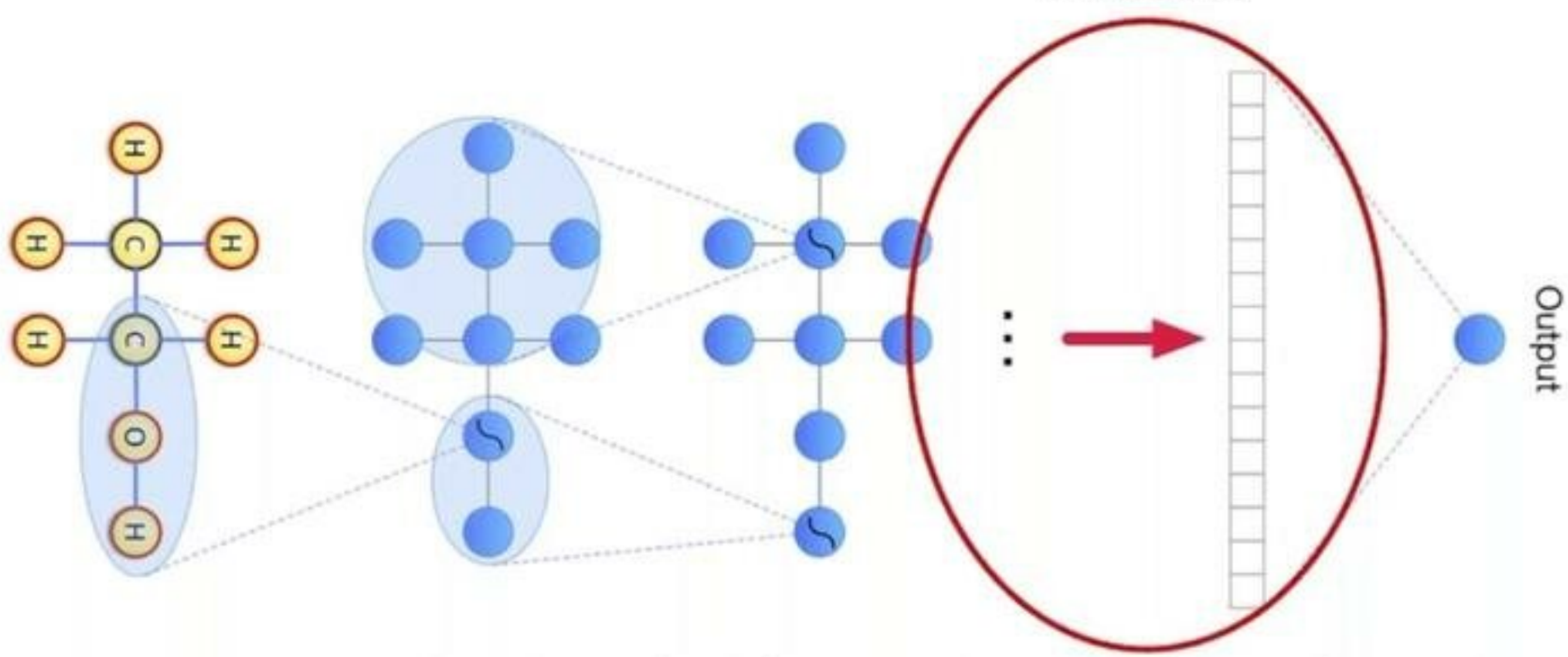
[6] Tran, D. V., Navarin, N., & Sperduti, A. (2018). On Filter Size in Graph Convolutional Networks. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1534–1541.

# Readout Layer (back to graph classification)



With GC we have a representation for each graph node.
How can we map node representations to a graph-level representation?

- **Readout** function:
  - Maps a (multi) **set** of node representations to a **graph-level representation**
  - Differentiable
- Naïve solutions:
  - **sum** (or average) of node representations
  - More complex alternatives: Universal readout (DeepSets) [7]




[7] Navarin, N., Tran, D. Van, & Sperduti, A. (2019). Universal Readout for Graph Convolutional Neural Networks. *International Joint Conference on Neural Networks*. Budapest, Hungary.

# Kernel information in GNNs



- In many cases, **graph kernels perform better** (or comparably) to GCNs
- In particular in the domain of molecules

Our idea is to integrate the knowledge carried by the kernel in the representation learned by the GCN

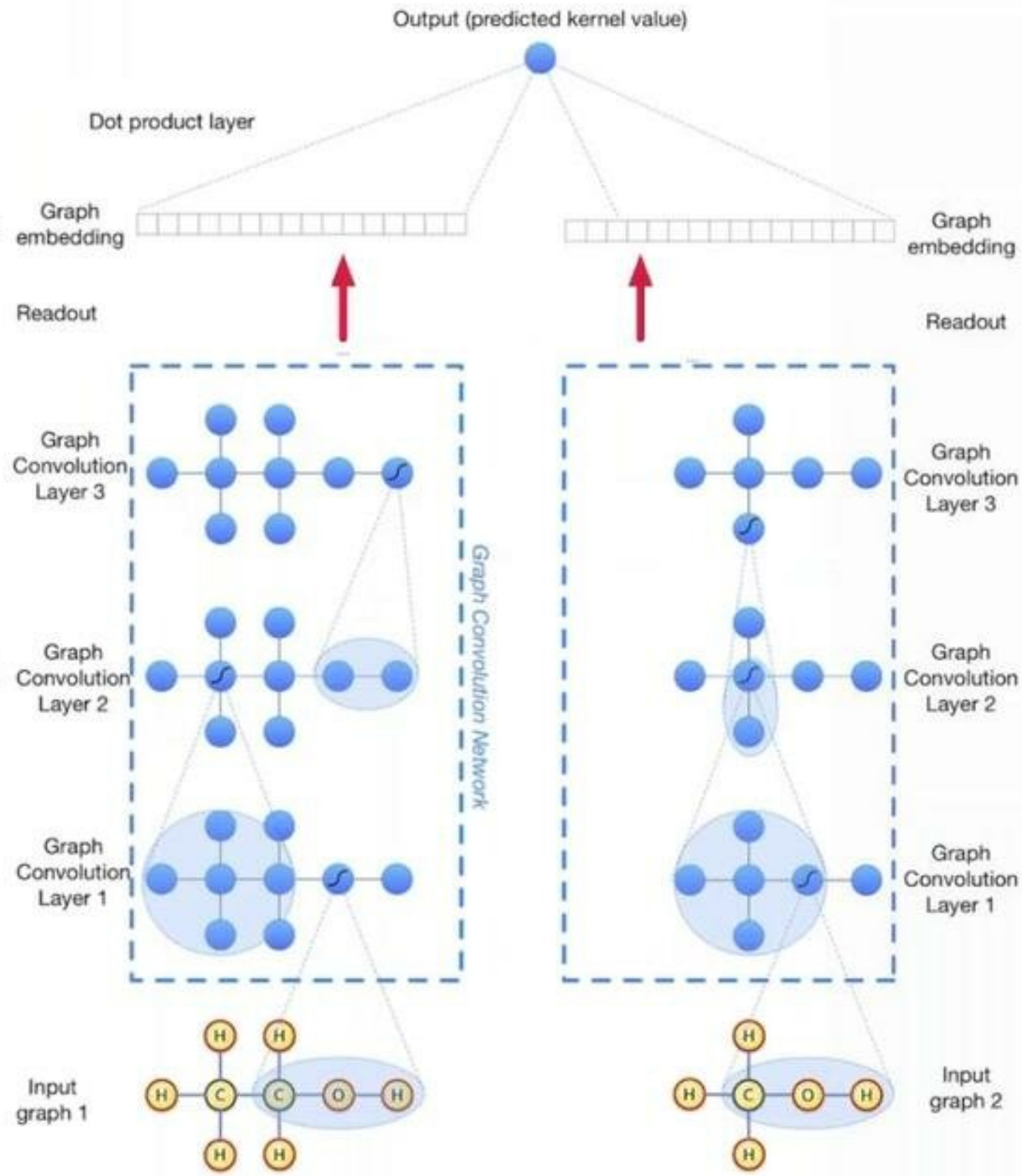
| | Method/Dataset | MUTAG | PTC | NCI1 | PROTEINS | D&D |
|---|---|---|---|---|---|---|
| Graph kernels | RW | 79.17±2.07 | 55.91±0.32 | >3 days | 59.57±0.09 | >3 days |
| | PK | 76.00±2.69 | 59.50±2.44 | 82.54±0.47 | 73.68±0.68 | 78.25±0.51 |
| | WL | 84.11±1.91 | 57.97±2.49 | 84.46±0.45 | 74.68±0.49 | 78.34±0.62 |
| Graph Neural Networks | PSCN | - | - | 76.34±1.68 | 75.00±2.51 | 76.27±2.64 |
| | CapsGCN | 86.67±6.88 | - | 78.35±1.55 | 76.28±3.63 | 75.38±4.17 |
| | DGCNN | 82.48±1.49 | 57.14±2.19 | 72.97±0.92 | 73.96±0.41 | 78.09±0.72 |

# Pre-training GNNs with kernels



First attempt: **pre-training**

- Kernels are defined over **pairs** of graphs, so we define a siamese GCN architecture
- The target is the kernel value
- After pre-training, we can train a single network using supervised labels as usual

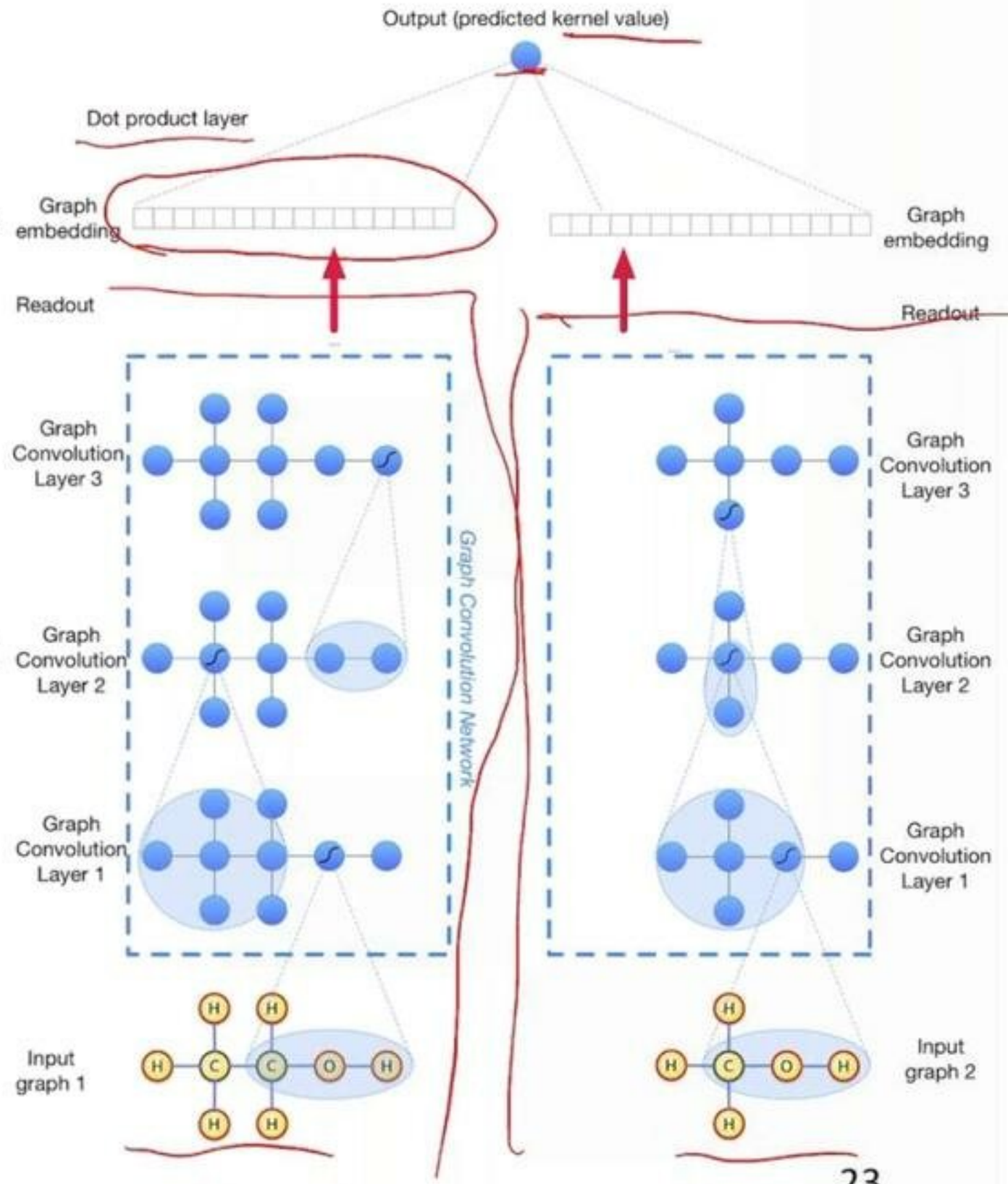


# Pre-training GNNs with kernels



First attempt: **pre-training**

- Kernels are defined over **pairs** of graphs, so we define a siamese GCN architecture
- The target is the kernel value
- After pre-training, we can train a single network using supervised labels as usual



# Pre-training GNNs with kernels



Pros:

- Promising results
- Unsupervised
- Pre training acts as a bias toward good representations
- bound on (training) error depending on reconstruction loss

Cons:

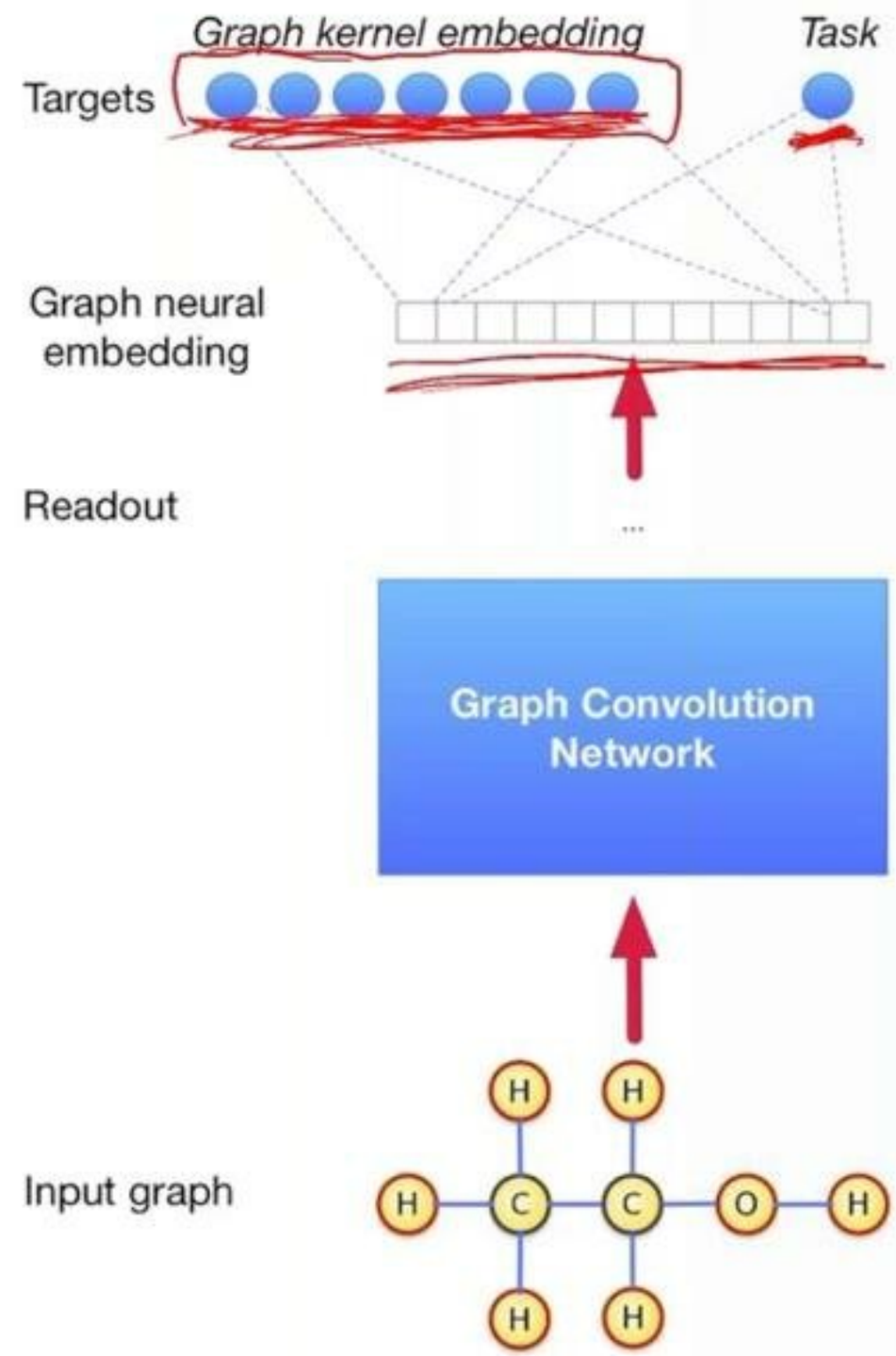
- Computational time: quadratic in the number of examples

| Method / Dataset | MUTAG | PTC | NCI1 |
|---|---|---|---|
| GK | 81.39±1.74 | 55.65±0.46 | 62.49±0.27 |
| RW | 79.17±2.07 | 55.91±0.32 | >3 days |
| PK | 76.00±2.69 | 59.50±2.44 | 82.54±0.47 |
| WL | 84.11±1.91 | 57.97±2.49 | **84.46±0.45** |
| DGCNN | 85.83±1.66 | 58.59±2.47 | 74.44±0.47 |
| Pre-trained DGCNN | **88.10±1.05** | **61.03±2.86** | 77.13±0.45 |

# Multi-task training approach

Second approach: **Multi-task training**

- Single network, two outputs
- **Hashing** to map the kernel embedding to a fixed, small vector
- We try to reconstruct the kernel embedding as a secondary output




Pros:

- No added complexity (pre-processing of kernel computation)
- Any graph embedding can be used (i.e. domain knowledge)

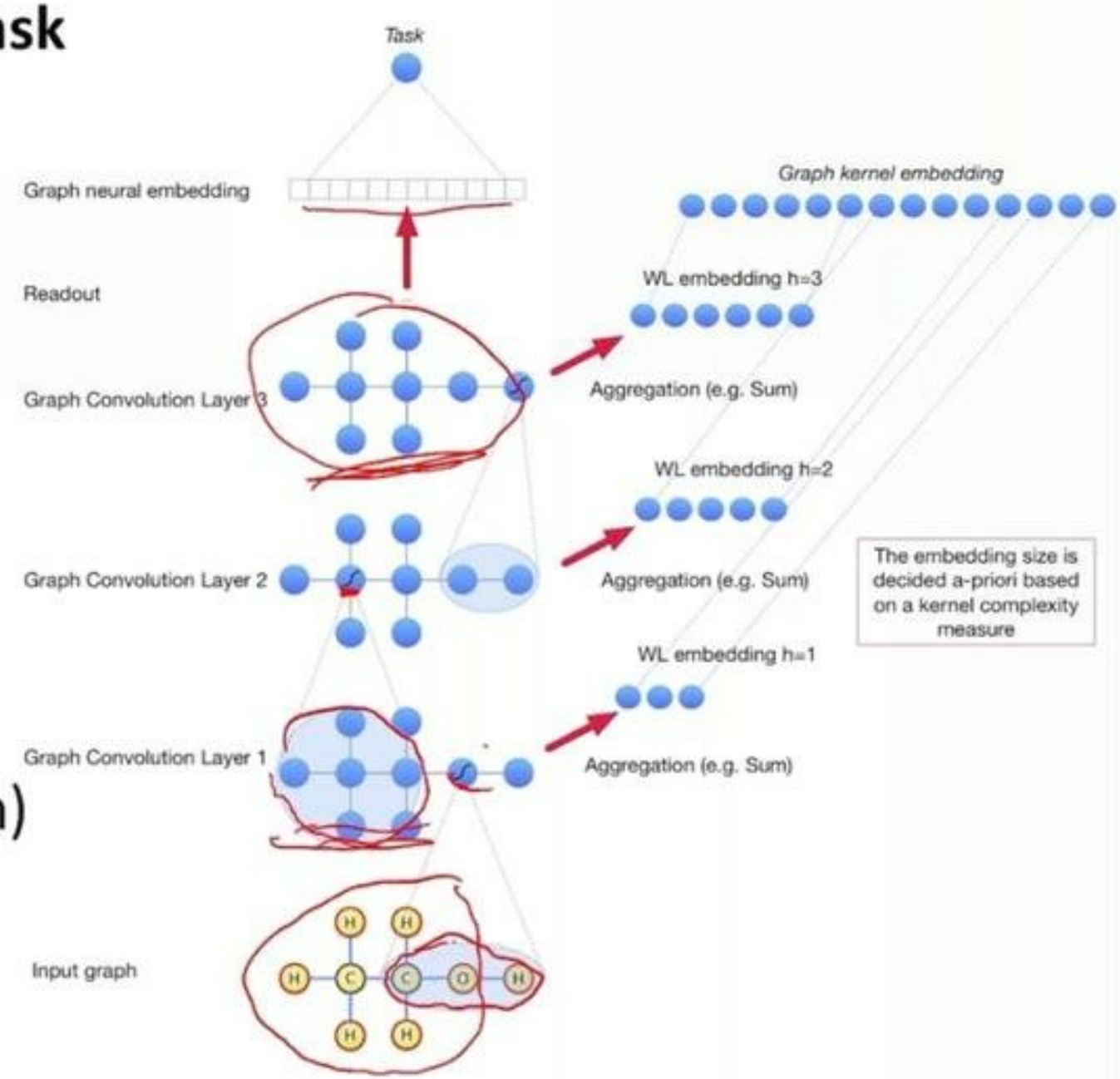
Cons:

- We don't have control in where the network will store information from the embedding

# Layer-wise kernel supervision

Third approach: **Layer-wise Multi-task training**

- Single network, multiple outputs
- **Split features** in the embedding according to their **complexity**
- Deeper layers reconstruct more complex features




Pros:

- No added complexity (pre-processing of kernel computation)

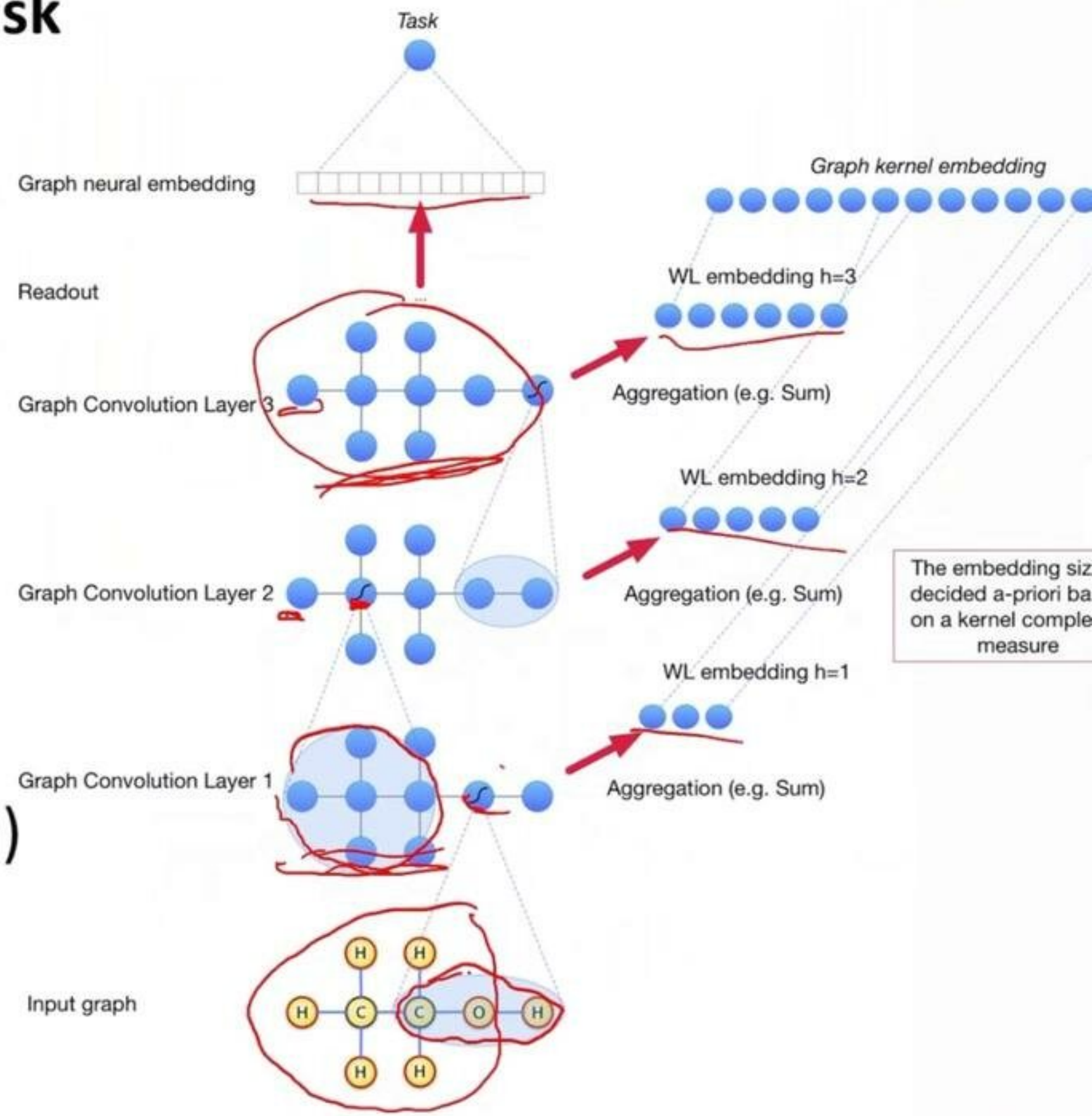
Cons (future works):

- Even lower-grained supervision possible?

# Layer-wise Multi-task

k, multiple outputs
in the embedding
heir **complexity**
reconstruct more
res

nplexity (pre-
kernel computation)
rks):
ained supervision



# Results for Kernel-based embeddings



- All kernel-based training approaches improve over standard training
- In general, $MT < PT < LMT$
- The techniques are applicable to all GNNs
  - The improvement may vary depending on the architecture

| | Method/Dataset | MUTAG | PTC | NCI1 | PROTEINS | D&D |
|---|---|---|---|---|---|---|
| | WL (h=3) | 76.79*±3.17 | 57.48*±1.36 | 82.13±2.17 | 69.63*±1.22 | 73.64*±2.56 |
| | DGCNN | 82.48*±1.49 | 57.14*±2.19 | 72.97*±0.87 | 73.96*±0.41 | 78.09*±0.72 |
| | LMT-FGCNN | **86.81**±1.75 | 59.04±0.94 | **82.20**±0.54 | **76.03**±0.68 | **80.14**±0.76 |
| Ablation study | FGCNN | 84.49±1.90 | 58.82± 1.80 | 81.50±0.39 | 74.57*±0.80 | 77.47*±0.86 |
| Ablation study | LMT-DGCNN | 85.00±1.15 | **59.39**±0.51 | 77.02*±0.48 | 74.61*±0.89 | 78.11*±0.61 |
| Ablation study | MT-DGCNN | 83.68±1.29 | 58.39±1.11 | 76.55*±0.40 | 74.42*±0.36 | 78.17*±0.57 |
| Ablation study | MT-FGCNN | 85.81±1.62 | 59.23±2.35 | 81.86±0.41 | 75.18±0.66 | 79.90±0.39 |
| Ablation study | PT-DGCNN | 85.38±1.47 | 58.48±1.92 | 75.20* ±0.87 | 75.19±0.42 | 78.38*±0.55 |

# Results for Kernel-based embeddings



- All kernel-based training approaches improve over standard training
- In general, $MT < PT < LMT$
- The techniques are applicable to all GNNs
  - The improvement may vary depending on the architecture

| | Method/Dataset | MUTAG | PTC | NCI1 | PROTEINS | D&D |
|---|---|---|---|---|---|---|
| | WL (h=3) | 76.79*±3.17 | 57.48*±1.36 | 82.13±2.17 | 69.63*±1.22 | 73.64*±2.56 |
| BASE | DGCNN | 82.48*±1.49 | 57.14*±2.19 | 72.97*±0.87 | 73.96*±0.41 | 78.09*±0.72 |
| | LMT-FGCNN | **86.81**±1.75 | 59.04±0.94 | **82.20**±0.54 | **76.03**±0.68 | **80.14**±0.76 |
| Ablation study | FGCNN | 84.49±1.90 | 58.82± 1.80 | 81.50±0.39 | 74.57*±0.80 | 77.47*±0.86 |
| | LMT-DGCNN | 85.00±1.15 | **59.39**±0.51 | 77.02*±0.48 | 74.61*±0.89 | 78.11*±0.61 |
| | MT-DGCNN | 83.68±1.29 | 58.39±1.11 | 76.55*±0.40 | 74.42*±0.36 | 78.17*±0.57 |
| | MT-FGCNN | 85.81±1.62 | 59.23±2.35 | 81.86±0.41 | 75.18±0.66 | 79.90±0.39 |
| | PT-DGCNN | 85.38±1.47 | 58.48±1.92 | 75.20* ±0.87 | 75.19±0.42 | 78.38*±0.55 |

# Results for Kernel-based embeddings



- All kernel-based training approaches improve over standard training
- In general, $MT < PT < LMT$
- The techniques are applicable to all GNNs
  - The improvement may vary depending on the architecture

| | Method/Dataset | MUTAG | PTC | NCI1 | PROTEINS | D&D |
|---|---|---|---|---|---|---|
| | WL (h=3) | 76.79*±3.17 | 57.48*±1.36 | 82.13±2.17 | 69.63*±1.22 | 73.64*±2.56 |
| BASE | DGCNN | 82.48*±1.49 | 57.14*±2.19 | 72.97*±0.87 | 73.96*±0.41 | 78.09*±0.72 |
| | LMT-FGCNN | **86.81**±1.75 | 59.04±0.94 | **82.20**±0.54 | **76.03**±0.68 | **80.14**±0.76 |
| Ablation study | FGCNN | 84.49±1.90 | 58.82± 1.80 | 81.50±0.39 | 74.57*±0.80 | 77.47*±0.86 |
| | LMT-DGCNN | 85.00±1.15 | **59.39**±0.51 | 77.02*±0.48 | 74.61*±0.89 | 78.11*±0.61 |
| | MT-DGCNN | 83.68±1.29 | 58.39±1.11 | 76.55*±0.40 | 74.42*±0.36 | 78.17*±0.57 |
| | MT-FGCNN | 85.81±1.62 | 59.23±2.35 | 81.86±0.41 | 75.18±0.66 | 79.90±0.39 |
| | PT-DGCNN | 85.38±1.47 | 58.48±1.92 | 75.20* ±0.87 | 75.19±0.42 | 78.38*±0.55 |

# Results for Kernel-based embeddings



- All kernel-based training approaches improve over standard training
- In general, $MT < PT < LMT$
- The techniques are applicable to all GNNs
  - The improvement may vary depending on the architecture

| | Method/Dataset | MUTAG | PTC | NCI1 | PROTEINS | D&D |
|---|---|---|---|---|---|---|
| | WL (h=3) | 76.79*±3.17 | 57.48*±1.36 | 82.13±2.17 | 69.63*±1.22 | 73.64*±2.56 |
| BASE | DGCNN | 82.48*±1.49 | 57.14*±2.19 | 72.97*±0.87 | 73.96*±0.41 | 78.09*±0.72 |
| | LMT-FGCNN | **86.81**±1.75 | 59.04±0.94 | **82.20**±0.54 | **76.03**±0.68 | **80.14**±0.76 |
| Ablation study | FGCNN | 84.49±1.90 | 58.82± 1.80 | 81.50±0.39 | 74.57*±0.80 | 77.47*±0.86 |
| | LMT-DGCNN | 85.00±1.15 | **59.39**±0.51 | 77.02*±0.48 | 74.61*±0.89 | 78.11*±0.61 |
| | MT-DGCNN | 83.68±1.29 | 58.39±1.11 | 76.55*±0.40 | 74.42*±0.36 | 78.17*±0.57 |
| | MT-FGCNN | 85.81±1.62 | 59.23±2.35 | 81.86±0.41 | 75.18±0.66 | 79.90±0.39 |
| | PT-DGCNN | 85.38±1.47 | 58.48±1.92 | 75.20* ±0.87 | 75.19±0.42 | 78.38*±0.55 |

# Semi-supervised learning



- We can add more (unlabelled) data in the training
- Semi-supervised learning
- The more data we add, the higher the performance

| Dataset/ Method | DGCNN | FGCNN | LMT-FGCNN | | |
|---|---|---|---|---|---|
| | | | +0 | +1 | +2 |
| NCI1B | 72.92 ±0.56 | 79.27 ±0.70 | 81.01 ±0.56 | 81.19 ±0.46 | 82.07 ±0.21 |
| NCI33B | 75.00 ±0.42 | 81.75 ±0.67 | 81.81 ±0.20 | 82.60 ±0.39 | 82.69 ±0.56 |
| NCI41B | 70.94 ±0.53 | 78.30 ±0.67 | 79.02 ±0.17 | 79.10 ±0.40 | 79.54 ±0.15 |

# Semi-supervised learning



- We can add more (unlabelled) data in the training
- Semi-supervised learning
- The more data we add, the higher the performance

| Dataset/ Method | DGCNN | FGCNN | LMT-FGCNN | | |
|---|---|---|---|---|---|
| | | | +0 | +1 | +2 |
| NCI1B | 72.92 ±0.56 | 79.27 ±0.70 | 81.01 ±0.56 | 81.19 ±0.46 | 82.07 ±0.21 |
| NCI33B | 75.00 ±0.42 | 81.75 ±0.67 | 81.81 ±0.20 | 82.60 ±0.39 | 82.69 ±0.56 |
| NCI41B | 70.94 ±0.53 | 78.30 ±0.67 | 79.02 ±0.17 | 79.10 ±0.40 | 79.54 ±0.15 |

# Conclusions

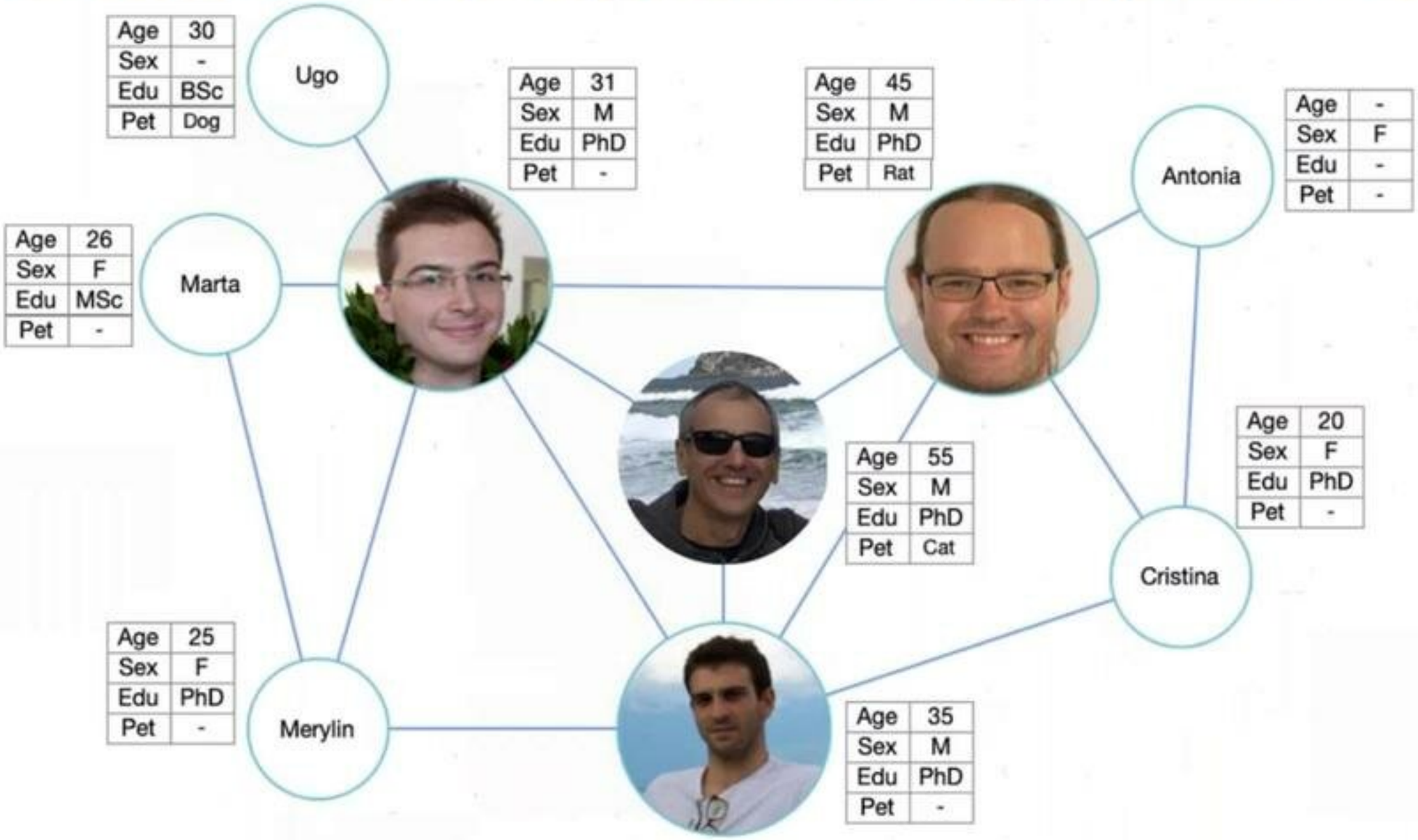


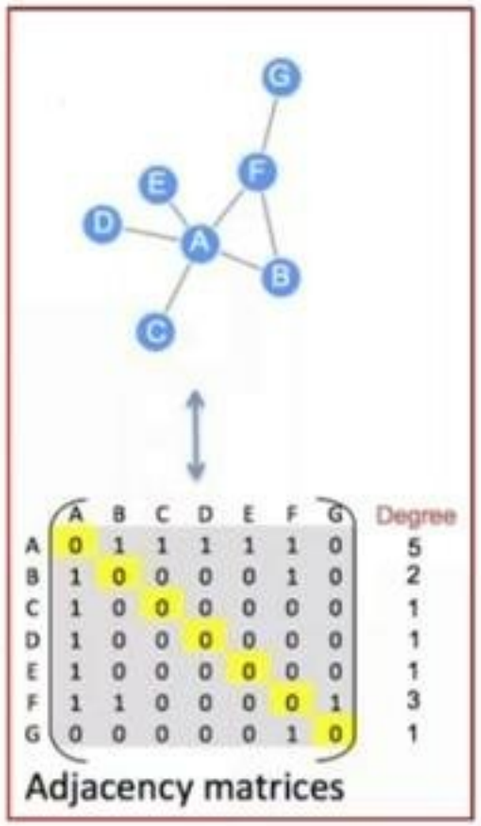
- In this talk, I presented different ways to merge:
  - Kernel methods for graphs
  - Neural Networks for Graphs
- Future work:
  - Other types of (pre) training
  - Recurrent GNNs
  - Work on relational graphs and graphs modelling more complex tasks

# Linear Graph Convolutional Networks

# Classification/Regression on graph nodes






- Dataset: a single huge graph:
  - $n$ vertices
  - $d$ **attributes** associated to nodes: $X \in \mathbb{R}^{n \times d}$
  - **target** associated to a small subset of nodes
- Given an unseen node, the task is to predict the correct target $y$