Graph Representation Learning applied to Transaction Networks

Midterm Assessment

Claudio Moroni
University of Turin
CENTAI Institute S.p.A.

Outline

1. Motivation
2. GNN Overview
3. Implementations and Contributions
4. Next Goals
5. Extras

UNIVERSITÀ DI TORINO

CENTAI

Motivation

1. Motivations

UNIVERSITÀ DI TORINO

CENTAI

Motivation

1. Motivations:
    1.1 Temporal Graph Benchmark
    1.2 Anti Money Laundering
    1.3 Embedding Directed Graphs

UNIVERSITÀ DI TORINO
CENTAI

Temporal Graph Benchmark

Modern TGNNs performance on **Temporal Node Classification** is lackluster:

Leaderboard for tgbn-trade

The MRR score on the test and validation sets. The higher, the better.

Package: >=0.8.0

Rank	Method	Test NDCG@10	Validation NDCG@10	Contact	References	Date
1	**Persistent Forecast**	0.855	0.860	TGB team	Paper, Code	August 6th, 2023
2	**Moving Average**	0.823	0.841	TGB team	Paper, Code	August 6th, 2023
3	**DyGFormer**	0.388 ± 0.006	0.408 ± 0.006	Le Yu (Beihang University)	Paper, Code	August 22nd, 2023
4	**TGN**	0.374 ± 0.001	0.395 ± 0.002	TGB team	Paper, Code	August 6th, 2023
5	**DyRep**	0.374 ± 0.001	0.394 ± 0.001	TGB team	Paper, Code	August 6th, 2023

PROPOSAL: Use **static representations**[1].

UNIVERSITÀ DI TORINO

CENTAI

1. Oettershagen et al. (2020)

Anti Money Laundering

Current approaches:

- Do not leverage network information[1];
- Do not successfully leverage temporal information[2];
- Do not leverage subgraph information[3];

PROPOSAL: Apply neural subgraph matching[4] to Static Repesentations.

1. Jullum et al. (2020), Lorenz et al. (2021), Jensen and Iosifidis (2023)
2. Weber et al. (2019), Johannessen and Jullum (2023)
3. Altman et al. (2023)
4. Rex et al. (2020), Fey et al. (2020)

UNIVERSITÀ DI TORINO

CENTAI

Embedding Directed Graphs

- LP is **crucial** in TNs for:
  - Preprocessing[1]
  - Graph generation[2]

Directed LP task is divided in three sub-tasks:

- General
- Biased
- Bidirectional

TBOK, current models perform well on **either** of the three, not all;

PROPOSAL: Come up with a training pipeline that does well on all three sub-tasks.

1. Li and He (2023), Lin et al. (2022)
2. Kipf and Welling (2016)

GNN Overview

1. Theoretical Background

2. Temporal Graph Neural Networks

UNIVERSITÀ DI TORINO

CENTAI

GNN Overview

1. Theoretical Background:

   1.1 Early Models;

   1.2 Message Passing Neural Networks.

2. Temporal Graph Neural Networks

UNIVERSITÀ DI TORINO

CENTAI

Early Models #1

- Before Deep Learning, features were **human-engineered** (Machine Learning);
- Early works can be organized in node/edge/graph-level **features**;

Node-level:

- **Features**: Degree, Eigenvector, Betweeness, Closeness **centralities** & Clustering Coefficient (**CC**)/Graphlet Degree Vector (**GDV**)
- **RW-based**: PageRank, **DeepWalk**, **Node2Vec** (not inductive!);

Edge-level: Local/Gobal Neighborhood overlap (**Katz Index**)

Graph-level: (Causal) Anonymous Walks, Graphlet Kernel, **Weisfeiler-Leman Kernel**.

Early Models #2

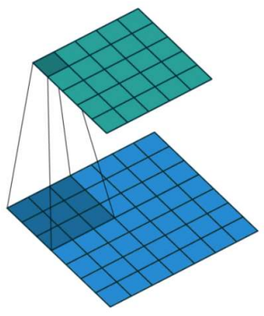
Limitations:

- Inherently **transductive**;
- Do not **incorporate** node/edge/graph features
- Computed node/edge/graph features are **task-independent**;

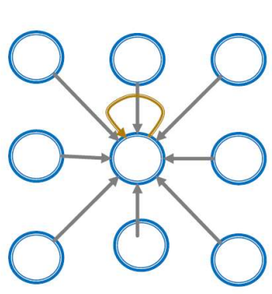
$\Longrightarrow$ Need for Graph Deep Learning i.e. **Graph Neural Networks**!

Message Passing Neural Networks #1

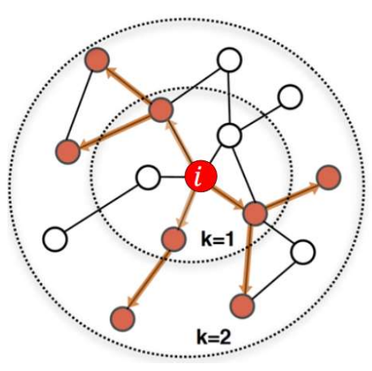
INTUITION: We'd like to produce **useful representations** for each node also taking into account their neighbors (ref. CNNs).

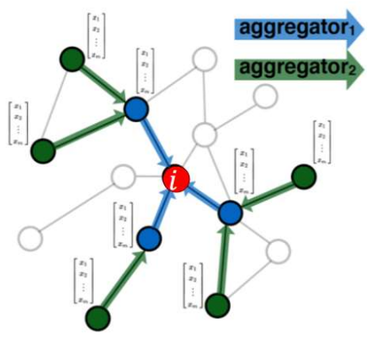


Image



Graph



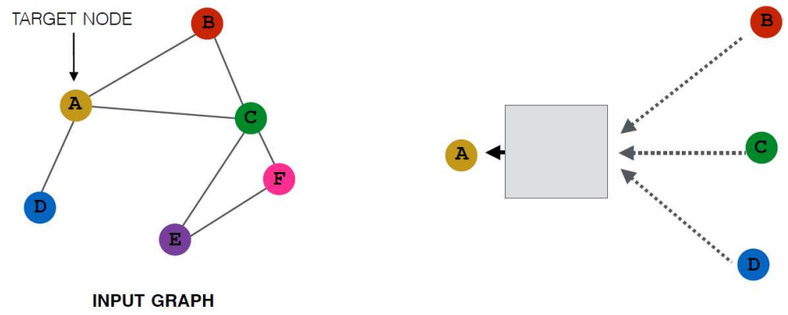


UNIVERSITÀ DI TORINO

CENTAI

Message Passing Neural Networks #2

$$\vec{h}_A^{(k+1)} = \sigma\left(\sum_{u \in N(A)} \frac{W^{(k)}\vec{h}_u^{(k)}}{|N(A)|} + B^{(k)}\vec{h}_A^{(k)}\right)$$



$$\vec{h}_v^{(k+1)} = \overrightarrow{\mathrm{COMB}}_k\left(\vec{h}_v^{(k)}, \overrightarrow{\mathrm{AGGR}}_k\left(\left\{\overrightarrow{\mathrm{MSG}}_k\left(\vec{h}_v^{(k)}, \vec{h}_u^{(k)}, e_{uv}\right) | u \in N(v)\right\}\right)\right)$$

1

1. Gilmer et al. (2017)

UNIVERSITÀ DI TORINO

CENTAI

UNIVERSITÀ
DI TORINO

CENTAI

GNN Overview

1. Theoretical Background

2. Temporal Graph Neural Networks:

   2.1 Framework

   2.2 ROLAND

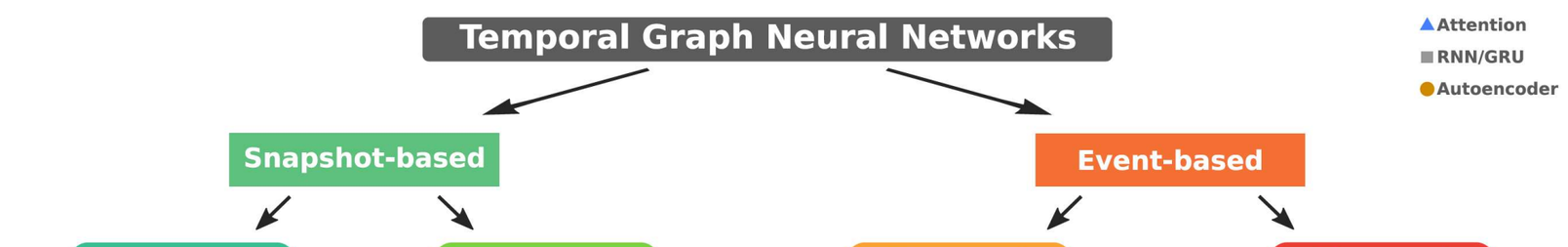
   2.3 TGAT

UNIVERSITÀ DI TORINO

CENTAI

Framework

Reviews: Longa et al. (2023), Gravina and Bacciu (2023), Gupta and Bedathur (2022)



	EvolveGCN	VGRNN	DySAT	DynGESN	Roland	SSGNN	TGAT	NAT	TGL	APAN	DGNN	TGN
f_{NC}	FT	✗	✗	✗	✗	✗	FT,FI	✗	FT	FT	PT	FT
f_{EC}	FT	✗	✗	✗	✗	✗	✗	✗	✗	FT	✗	✗
f_{GC}	✗	✗	✗	PT	✗	✗	✗	✗	✗	✗	✗	✗
f_{LP}	FT	FI,FT,PT	PT,FT	✗	FT	✗	FT,FI	PI,FT	FT	FT	FT	FI,FT
f_{EP}	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Clustering	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
LDE	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Regression (node)	✗	✗	✗	FT	✗	FT	✗	✗	✗	✗	✗	✗

UNIVERSITÀ DI TORINO

CENTAI

ROLAND

- A meta-model $\mathrm{GNN}^{(meta)}$, responsible for parameter initialization before **fine tuning** on each snapshot, is trained via **Meta-Learning** up to snapshot t;

- The GNN_{t+1} at snapshot $t+1$ is initialized $GNN \leftarrow \mathrm{GNN}^{(meta)}$ and **fine-tuned** on snapshot $t+1$ via:

$$H_{(k+1)}^{(t+1)} = GRU(H_{(k+1)}^{(t)}, \tilde{H}_{(k)}^{(t+1)})$$

So only **current snaphsot** and **previous final embeddings** are needed on GPU (**live-update**)

- The meta-model is updated:

$$GNN^{(meta)} \leftarrow (1-\alpha)GNN^{(meta)} + \alpha \mathrm{GNN}_{t+1}$$

And the cycle repeats.

TGAT

Time differences are represented via **Bochner**'s theorem (translation invariance can be proved):

$$\Phi(\Delta t) = \sqrt{\frac{1}{d}}[cos(\omega_1 \Delta t), sin(\omega_1 \Delta t), \dots, cos(\omega_d t), sin(\omega_d \Delta t)]$$

Given node v, we define:

$$Z^{(l-1)}(t) := [\vec{h}_v^{(l-1)}(t)||\vec{x}_{(v,v)}||\Phi(0), \vec{h}_{u_1}^{(l-1)}(t_1)||\vec{x}_{(u_1,v)}||\Phi(t-t_1), \dots$$
$$, \vec{h}_{u_1}^{(l-1)}(t_N)||\vec{x}_{(u_N,v)}||\Phi(t-t_N)]^t$$
$$\vec{q}_v^{(l-1)}(t)^t := \vec{Z}_v^{(l-1)}(t) W_Q := Z^{(l-1)}(t)[0,:]W_Q$$
$$K^{(l-1)}(t) := Z^{(l-1)}(t)[1:N,:]W_K$$
$$V^{(l-1)}(t) := Z^{(l-1)}(t)[1:N,:]W_V$$

UNIVERSITÀ DI TORINO

CENTAI

- A **convolutional self-attention**[1] layer is defined:

$$\tilde{\vec{h}}_v^{(l)}(t) = \sum_{u \in N(v;t)} \underbrace{softmax\left(\frac{\vec{q}_v^{(l-1)}(t)^t \vec{K}_u^{(l-1)}(t)}{\sqrt{d}}; N(v;t)\right) \vec{V}_u^{(l-1)}(t)}_{\tilde{\vec{h}}_{uv}^{(l)}(t)}$$

- Final embedding for v is:

$$\vec{h}_v^{(l)}(t) = \mathbf{ReLU}([\tilde{\vec{h}}_v^{(l)}(t)||\vec{h}_v^{(0)}(t)]W_0^{(l)} + \vec{b}_0^{(l)})W_1^{(l)} + \vec{b}_1^{(l)}$$

1. Vaswani et al. (2017)

Implementations and Contributions

1. Node Classification
2. Link Prediction

UNIVERSITÀ DI TORINO

CENTAI

Implementations and Contributions

1. Node Classification:

   3.1 Models

   3.2 Reddit NC Task

   3.3 OGBN-Arxiv NC Task

2. Link Prediction

Models

GraphSAGE GCN:

$$\vec{h}_v^{(k)} = \text{L2-NORMALIZE}\left(\sigma\left(W^{(k)}\text{MEAN}(\{\vec{h}_v^{(k-1)}\} \cup \{\vec{h}_u^{(k-1)}|u \in N(v)\})\right)\right)$$

GraphSAGE MEAN:

$$\vec{h}_v^{(k)} = \text{L2-NORMALIZE}\left(\sigma\left(W_N^{(k)}\text{MEAN}(\{\vec{h}_u^{(k-1)}|u \in N(v)\}) + W_S^{(k)}\vec{h}_v^{(k-1)}\right)\right)$$

ModGIN:

$$\vec{h}_v^{(k)} = \text{MLP}_C\left(\text{CONCAT}\left(\sum_{u \in N(v)}\text{MLP}_N(\vec{h}_u^{(k-1)}), \vec{h}_v^{(k-1)}\right)\right)$$

ModGIN_Att_Norm:

UNIVERSITÀ DI TORINO

CENTAI

$$\vec{\tilde{h}}_v^{(k)} = \mathrm{MLP}_C \left(\left(\sum_{u \in N(v)} \mathrm{MLP}_N (\vec{h}_u^{(k-1)} || \vec{h}_v^{(k-1)}) \right) || \vec{h}_v^{(k-1)} \right)$$

Reddit NC Task #1

41-way node classification on Reddit Dataset.

Reddit Dataset:

- Nodes are **posts** (232,965 total, 492 average degree)
- Two posts are linked if the **same user** comments below both (114615892 total edges)
- Label to predict is the **subreddit** a post belongs to
- Other node features: 300-dimensional **word embedding** of title || comments || score and number of comments
- **Metric**: micro-averaged F1 score
- **Training Set** size: 153431, **Validation Set** size: 23831, **Test Set** size: 55703

UNIVERSITÀ DI TORINO CENTAI

Reddit NC Task #2

- **Minibatch** + **Neighbor Sampling**;
- 10 epochs, 512 batch size
- $K = 2$
- $S_1 = 25$, $S_2 = 10$
- Embedding dimension: 256 (big), 128 (small)
- $\sigma = \mathrm{ReLU}$
- Adam Optimizer, LR Hyperparameter Search: $\{0.01, 0.001, 0.0001\}$
- Loss ℓ: LogSoftmax + Negative Log Likelihood;

UNIVERSITÀ DI TORINO CENTAI

Reddit NC Task #3

Model	NSI (F1)	FNI (F1)	epochs	secs/epoc
GraphSAGE_GCN	0.941	0.947	5	16.44
GraphSAGE_GCN_Sparse	0.941	0.948	10	**14.68**
GraphSAGE_MEAN	0.957	0.962	3	19.15
GraphSAGE_MEAN_Sparse	0.960	0.963	**2**	15
ModGIN	**0.961**	0.7	25	25.22
ModGIN_Att_Norm	0.959	**0.964**	10	46.27

UNIVERSITÀ DI TORINO

CENTAI

OGBN-Arxiv NC Task #1

TASK: 40-way node classification on OGBN-Arxiv.

OGBN-Arxiv:

- Nodes are **articles** (169343 total)
- Two articles are linked if one cites the other (directed graph, 1166243 total edges)
- Label to predict is the **subject area** an article belongs to (e.g. cs.AI, cs.OS, etc)
- Other node features: 128-dimensional feature vector obtained by **averaging** the embeddings of words in its title and abstract.
- **Metric**: micro-averaged F1 score (accuracy)
- **Training Set**: until 2017 (included), **Validation Set**: 2018, **Test Set**: 2019
- Evaluation same as Reddit

UNIVERSITÀ DI TORINO

CENTAI

OGBN-Arxiv NC Task #2

Model	Accuracy	epochs	secs/epoch
GraphSAGE_MEAN_OTS	**0.7164**	154	**0.12**
GraphSAGE_MEAN_MB_NS	0.704	**10**	6.64

Implementations and Contributions

1. Node Classification

2. Link Prediction:

   4.1 Overview;

   4.2 OGBL-DDI (Repr.);

   4.3 (V)GAE (Contrib.);

   4.4 (V)GNAE (Repr.);

   4.5 Gravity-(V)GAE (Contrib.);

   4.6 ST-(V)GAE (Contrib.);

   4.7 IT-AML (Contrib.);

   4.8 Conclusions;

Overview

- **Link Prediction** consists in predicting the probability of an unseen edge between two nodes;

- The models we'll be using are **(Variational) Graph Autoencoders**, they're composed of:
  - An **Encoder** that computes the embeddings of all nodes $H = Enc(A, X)$;
  - A **Decoder** that emits the probability of an edge $p_{ij} = Dec(\vec{h}_i, \vec{h}_j)$.

- The Variational models are also capable of **graph generation** through the added **stochasticity**;

- To perform LP, edges are split between **message**, **supervision**, **validation** and **test** sets.

OGBL-DDI (Repr.)

- **Preprocessing**: Learnable Features
- **Encoder**: 2-layer GraphSAGE
- **Decoder**: MLP
- **Setup**: BCEWithLogitsLoss, Early Stopping

We fully reproduce **performance** of OGB Team.

(V)GAE (Contrib.)

- **Dataset**: Cora (Undirected)
- **Encoder**: 2-layer GCN $ReLU(\tilde{A}\,\mathrm{ReLU}(\tilde{A}XW_0)W_1)$
- **Decoder**: $p_{ij} = \sigma(\vec{h}_i \cdot \vec{h}_j)$
- **Training Strategy**: Random Split + “General Setting”

UNIVERSITÀ DI TORINO

CENTAI

Ours:

	GAE	VGAE
AUC	0.92 +- 0.01	0.91 +- 0.006
F1	0.81 +- 0.01	0.799 +- 0.007
hitsk	0.66 +- 0.07	0.6 +- 0.05
AP	0.92 +- 0.02	0.905 +- 0.008
MRR	0.48 +- 0.06	0.42 +- 0.02

Theirs:

Method	Cora	
	AUC	AP
SC [5]	84.6 ± 0.01	88.5 ± 0.00
DW [6]	83.1 ± 0.01	85.0 ± 0.00
GAE*	84.3 ± 0.02	88.1 ± 0.01
VGAE*	84.0 ± 0.02	87.7 ± 0.01
GAE	91.0 ± 0.02	92.0 ± 0.03
VGAE	$\mathbf{91.4} \pm 0.01$	$\mathbf{92.6} \pm 0.01$

- **CONTRIBUTION**: Pytorch Geometric Implementation

UNIVERSITÀ DI TORINO

CENTAI

(V)GNAE (Repr.) #1

- GAE and VGAE embed **isolated** nodes near $\vec{0}$ $\implies$ all isolated nodes are **similar**;
- The authors propose to **normalize** the embeddings before **propagating** (**GNCN**):

$$\vec{h}_v^l = \frac{1}{d_i + 1} s \frac{\vec{h}_v^{l-1}}{||\vec{h}_v^{l-1}||} + \sum_{u \in N(v)} \frac{1}{\sqrt{d_v + 1}\sqrt{d_u + i}} s \frac{\vec{h}_u^{l-1}}{||\vec{h}_u^{l-1}||}$$

Encoder:

1. One-layer GNCN for the **GNAE**;
2. A GNCN layer (μ) and a GCN layer ($log\sigma$) for the **VGNAE**.

UNIVERSITÀ DI TORINO CENTAI

(V)GNAE (Repr.) #2

Task: Link prediction on undirected Cora

Results

Ours:

	GNAE	**VNGAE**
AUC	0.930 +- 0.004	~0.9
F1	0.793 +- 0.003	~0.76
AP	0.940 +- 0.003	~0.92

Theirs:

Data	Metric	Train	GAE	LGAE	ARGA	GIC	sGraph	GNAE	VGNAE
Cora	AUC	20%	0.782	0.866	0.795	0.880	0.845	0.887	**0.890**
		40%	0.856	0.908	0.844	0.914	0.840	0.926	**0.929**
		80%	0.922	0.938	0.919	0.933	0.885	**0.956**	0.954
	AP	20%	0.793	0.878	0.806	0.881	0.829	**0.901**	**0.901**
		40%	0.861	0.915	0.856	0.911	0.828	**0.936**	0.933
		80%	0.930	0.945	0.927	0.929	0.867	0.957	**0.958**

Gravity-(V)GAE (Contrib.) #1

- **Dataset**: Cora (Directed)
- **Encoder**: 2-layer GraphSAGE
- **Decoder**: $\hat{A}_{ij} = \sigma(m_j - \lambda log(||\vec{z}_i - \vec{z}_j||_2^2))$,
  $m_i = Enc(\vec{x}_i)[-1]$, $\vec{z}_i = Enc(\vec{x}_i)[:-1]$
- Directed LP is **divided** in three tasks: **General**, **Biased** and **Bidirectional**
- **CONTRIBUTIONS**:
  - **Early Stopping**;
  - Trainable λ;
  - **More Representative** training set for Biased and Bidirectional;
  - Pytorch Geometric Implementation (**to be made official**).

UNIVERSITÀ DI TORINO CENTAI

Gravity-(V)GAE (Contrib.) #2

Ours:

General Link Prediction:

	Gravity-GAE	**Gravity-VGAE**
AUC	0.857 +- 0.002	0.9041 +- 0.0005
F1	0.74 +- 0.01	0.741 +- 0.007
hitsk	0.72 +- 0.01	0.754 +- 0.004
AP	0.902 +- 0.002	0.9279 +- 0.0007

Biased Negative Samples Link Prediction:

	Gravity-GAE	**Gravity-VGAE**	**Gravity-GAE_altered**
AUC	0.83 +- 0.01	0.822 +- 0.003	0.848 +- 0.001
F1	0.722 +- 0.003	0.742 +- 0.002	0.751 +- 0.006
hitsk	0.445 +- 0.005	0.44 +- 0.01	0.37 +- 0.02
AP	0.832 +- 0.005	0.82 +- 0.002	0.833 +- 0.003

Bidirectionality Prediction:

UNIVERSITÀ DI TORINO

CENTAI

	Gravity-GAE	Gravity-VGAE	Gravity-GAE_altered	Gravity-VGAE_altered
AUC	0.775 +- 0.003	0.753 +- 0.01	0.794 +- 0.002	0.774 +- 0.003
F1	0.755 +- 0.006	0.74 +- 0.01	0.32 +- 0.01	0.4 +- 0.01
hitsk	0.55 +- 0.01	0.52 +- 0.04	0.57 +- 0.007	0.57 +- 0.02
AP	0.744 +- 0.003	0.7 +- 0.01	0.778 +- 0.004	0.742 +- 0.005

Theirs:

Dataset	Model	Task 1: General Link Prediction		Task 2: B.N.S. Link Prediction		Task 3: Bidirectionality Prediction	
		AUC (in %)	AP (in %)	AUC (in %)	AP (in %)	AUC (in %)	AP (in %)
Cora	*Gravity Graph VAE (ours)*	91.92 ± 0.75	92.46 ± 0.64	83.33 ± 1.11	84.50 ± 1.24	75.00 ± 2.10	73.87 ± 2.82
	Gravity Graph AE (ours)	87.79 ± 1.07	90.78 ± 0.82	83.18 ± 1.12	84.09 ± 1.16	75.57 ± 1.90	73.40 ± 2.53

UNIVERSITÀ DI TORINO

CENTAI

ST-(V)GAE (Contrib.)

- Everything the same as Gravity-(V)GAE (same paper);
- **Decoder**:

$$\hat{A}_{ij} = \sigma(\vec{s}_i \cdot \vec{t}_j)$$

where

$$\vec{s}_i = Enc(\vec{x}_i)[: emb\_size/2]$$
$$\vec{t}_i = Enc(\vec{x}_i)[emb\_size/2 :]$$

UNIVERSITÀ DI TORINO

CENTAI

- **CONTRIBUTIONS**:
  - **Early Stopping**;
  - Trainable λ;
  - **More Representative** training set for Biased and Bidirectional;
  - Pytorch Geometric Implementation (**to be made official**).
- Alternative (Kollias et al. 2022).

ST-(V)GAE (Contrib.)

Ours:

General Link Prediction:

	SourceTarget-GAE
AUC	0.86 +- 0.02
F1	0.5 +- 0.08
hitsk	0.66 +- 0.03
AP	0.894 +- 0.008

Biased Negative Samples Link Prediction:

	SourceTarget-GAE	**SourceTarget-GAE_altered**
AUC	0.61 +- 0.01	0.878 +- 0.006
F1	0.39 +- 0.09	0.8 +- 0.008
hitsk	0.17 +- 0.02	0.56 +- 0.02
AP	0.63 +- 0.01	0.869 +- 0.009

Bidirectionality Prediction:

	SourceTarget-GAE
AUC	0.74 +- 0.02
F1	0.69 +- 0.01
hitsk	0.52 +- 0.07

UNIVERSITÀ DI TORINO

CENTAI

	SourceTarget-GAE
AP	0.72 +- 0.03

Theirs:

Source/Target Graph AE	82.67 ± 1.42	83.25 ± 1.51	57.81 ± 2.64	57.66 ± 3.35	65.83 ± 3.87	63.15 ± 4.58

IT-AML (Contrib.) #1

- **Paper**: Altman et al. (2023)
- ABM model (*AMLworld*) that simulates transactions in a complex financial evironment (individuals, banks, companies, etc.)
- Model's output is a temporal sequence of transactions with labelled laundering patterns:

	Timestamp	From Bank	Account	To Bank	Account.1	Amount Received	Receiving Currency	Amount Paid	Payment Currency	Payment Format	Is Laundering
0	2022/09/01 00:08	11	8000ECA90	11	8000ECA90	3.195403e+06	US Dollar	3.195403e+06	US Dollar	Reinvestment	0
1	2022/09/01 00:21	3402	80021DAD0	3402	80021DAD0	1.858960e+03	US Dollar	1.858960e+03	US Dollar	Reinvestment	0
2	2022/09/01 00:00	11	8000ECA90	1120	8006AA910	5.925710e+05	US Dollar	5.925710e+05	US Dollar	Cheque	0
3	2022/09/01 00:16	3814	8006AD080	3814	8006AD080	1.232000e+01	US Dollar	1.232000e+01	US Dollar	Reinvestment	0
4	2022/09/01 00:00	20	8006AD530	20	8006AD530	2.941560e+03	US Dollar	2.941560e+03	US Dollar	Reinvestment	0
...	...	...	...	...	...	...	...	...	...	...	...
6924044	2022/09/10 23:39	71696	81B2518F1	71528	81C0482E1	3.346900e-02	Bitcoin	3.346900e-02	Bitcoin	Bitcoin	0
6924045	2022/09/10 23:48	271241	81B567481	173457	81C0DA751	1.313000e-03	Bitcoin	1.313000e-03	Bitcoin	Bitcoin	0
6924046	2022/09/10 23:50	271241	81B567481	173457	81C0DA751	1.305800e-02	Bitcoin	1.305800e-02	Bitcoin	Bitcoin	0
6924047	2022/09/10 23:57	170558	81A2206B1	275798	81C1D5CA1	4.145370e-01	Bitcoin	4.145370e-01	Bitcoin	Bitcoin	0
6924048	2022/09/10 23:31	170558	81A2206B1	275798	81C1D5CA1	3.427700e-02	Bitcoin	3.427700e-02	Bitcoin	Bitcoin	0

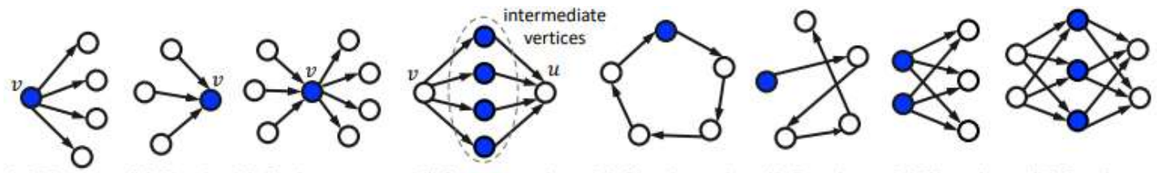
[6924049 rows x 11 columns]

IT-AML (Contrib.) #2

Laundering Data:

```
BEGIN LAUNDERING ATTEMPT - FAN-IN:  Max 3-degree Fan-In
2022/09/01 02:38,001812,80279F810,0110,8000A94C0,10154.74,Australian Dollar,10154.74,Australian Dollar,ACH,1
2022/09/02 14:36,022595,80279F8B0,0110,8000A94C0,5326.79,Australian Dollar,5326.79,Australian Dollar,ACH,1
2022/09/03 14:09,001120,800E36A50,0110,8000A94C0,4634.81,Australian Dollar,4634.81,Australian Dollar,ACH,1
END LAUNDERING ATTEMPT - FAN-IN

BEGIN LAUNDERING ATTEMPT - FAN-IN:  Max 8-degree Fan-In
2022/09/01 03:17,003671,801BF8E70,002557,8016B3750,8099.96,Euro,8099.96,Euro,ACH,1
2022/09/01 06:27,015,80074C7E0,002557,8016B3750,10468.56,Euro,10468.56,Euro,ACH,1
2022/09/01 10:04,002557,80107C9A0,002557,8016B3750,10270.07,Euro,10270.07,Euro,ACH,1
2022/09/02 06:35,012,800A9B180,002557,8016B3750,15645.21,Euro,15645.21,Euro,ACH,1
2022/09/03 09:12,021393,801271170,002557,8016B3750,14139.75,Euro,14139.75,Euro,ACH,1
2022/09/03 13:45,002175,801E25F20,002557,8016B3750,6276.26,Euro,6276.26,Euro,ACH,1
2022/09/03 16:17,020,800043BE0,002557,8016B3750,1042.63,Euro,1042.63,Euro,ACH,1
2022/09/03 23:09,022124,8011D0180,002557,8016B3750,12795.57,Euro,12795.57,Euro,ACH,1
END LAUNDERING ATTEMPT - FAN-IN
```



(a) Fan-out (b) Fan-in (c) Gather-scatter (d) Scatter-gather (e) Simple cycle (f) Random (g) Bipartite (h) Stack

Figure 2: Laundering Patterns Modelled

UNIVERSITÀ DI TORINO

CENTAI

IT-AML (Contrib.) #3

- The ABM is initialized via distributions, not real-world data;
- Modeling is detailed: transactions stratified by reason (e.g. salaries are monthly), companies own other companies, etc…
- 6 **synthetic datasets** are available online

Table 4: Public Synthetic Data Statistics. **HI** = Higher Illicit (more laundering). **LI** = Lower Illicit.

	Small		Medium		Large	
Statistic	**HI**	**LI**	**HI**	**LI**	**HI**	**LI**
# of Days Spanned	10	10	16	16	97	97
# of Bank Accounts	515K	705K	2077K	2028K	2116K	2064K
# of Transactions	5M	7M	32M	31M	180M	176M
# of Laundering Transactions	3.6K	4.0K	35K	16K	223K	100K
Laundering Rate (1 per N Trans)	981	1942	905	1948	807	1750

UNIVERSITÀ DI TORINO

CENTAI

IT-AML (Contrib.) #4

CONTRIBUTION N°1: Link Prediction Pass

Data Processing:

- Dataset size required vectorized methods
- Duplicate edges removed and temporal information discarded

Model. Gravity-GAE (best so far)

Split. Directed `torch.geometric.transforms.RandomLinkSplit` with `neg_sampling` = 1

UNIVERSITÀ DI TORINO

CENTAI

General Setting

	Gravity-GAE
AUC	0.894 +- 0.003
F1	0.8399 +- 0.0009
hitsk	0.00048 +- 3e-05
AP	0.86 +- 0.01
AUC_rev	0.617 +- 0.003
F1_rev	0.707 +- 0.002
hitsk_rev	5e-05 +- 1e-05
AP_rev	0.545 +- 0.003

Biased Setting

	Gravity-GAE
AUC	0.7827 +- 0.0002
F1	0.7416 +- 0.0005
hitsk	2e-05 +- 0.0
AP	0.7252 +- 0.0002

UNIVERSITÀ DI TORINO

CENTAI

IT-AML (Contrib.) #5

CONTRIBUTION N°2: Temporal Link Prediction Pass

UNIVERSITÀ DI TORINO

CENTAI

Conclusions

- For the ST models, we also implemented the **ST-VGAE** (not reported);

- Would be nice to try (Kollias et al. 2022), Edge Convolution/Line Graphs (Cai et al. 2020), Topology Adaptive GNN (Du et al. 2018) and experimenting with bidirectional heterogeneity.

Next Goals

1. Directed Link Prediction
2. Static Representations
3. AML via Neural Subgraph Matching

Next Goals

1. Directed Link Prediction

2. Static Representations

3. AML via Neural Subgraph Matching

UNIVERSITÀ DI TORINO

CENTAI

Directed Link Prediction

- Transaction networks are often **incomplete** (interbank transactions, Crypto APIs);
- LP performed with GNNs can make up for it and **improve** downstream tasks[1]
- **Graph Generation** useful to counteract data scarcity;
- (Salha et al. 2019) **splits** directed LP into three tasks, but comes short of finding a model that does well on **all** three;
- We began exploring this avenue initiating a collaboration with the author. I'd try:
  - **Weighting** reciprocal vs non-reciprocal vs random neagetive edges in the loss
  - Finding the **optimal amount** of random negatives;
  - **Ensembling** the three models.
- We'll also apply directed LP to a **graph generation** task in the near future.

1. Lin et al. (2022), Li and He (2023)

Next Goals

1. Directed Link Prediction

2. Static Representations:

   2.1 Motivation

   2.2 Reduced Graph Representation

   2.3 Directed Line Graph Expansion

   2.4 Static Kernel Expansion

   2.5 Conclusions

3. AML via Neural Subgraph Matching

UNIVERSITÀ DI TORINO

CENTAI

Motivation

- Current TGNNs suffer from low performances, and are very **complex** and difficult to **train**;
- Plethora of **static** metrics & kernels, very few ad hoc for temporal;
- **Lifting** metrics & kernels unfeasible $\implies$ need for **static representations**.
- (Oettershagen et al. 2020) proposes three ways to **statically represent** a graph s.t. the WL kernel retains most temporal information;
- In the following, assume $G = (V, E)$ where

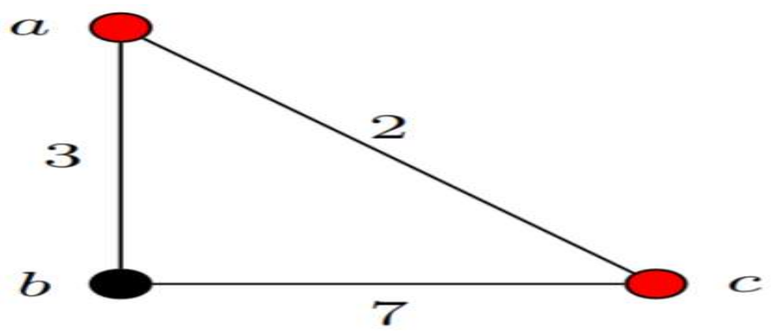
$$E = \{(\{u, v\}, t) | u, v \in V\}$$

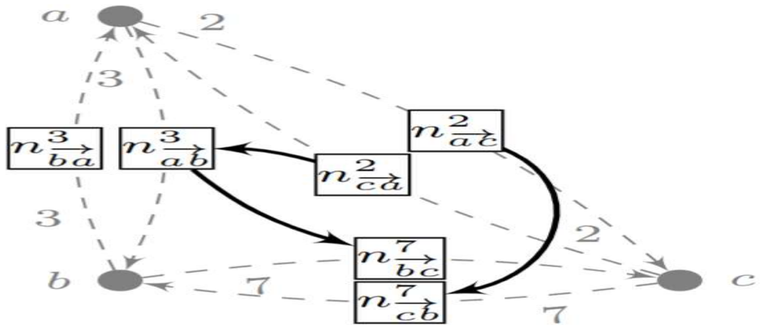
is a temporal undirected graph.

Directed Line Graph Expansion

Defined by $DL(G) = (V', E')$ where:

- $(\{u, v\}, t) \in E \implies n^t_{\overrightarrow{uv}}, n^t_{\overrightarrow{vu}} \in V'$;
- $(n^t_{\overrightarrow{uv}}, n^s_{\overrightarrow{xy}}) \in E' \iff v = x \wedge s > t$





UNIVERSITÀ DI TORINO

CENTAI

Static Expansion Kernel

Defined by $SE(G) = (V', E')$ where:

- $(\{u, v\}, t) \in E \implies (u, t), (v, t), (u, t+1), (v, t+1) \in V'$
- $E' = E_N \cup E_{W_1} \cup E_{W_2}$ where:
  - $E_N := \{((u, t), (v, t+1)), ((u, t+1), (v, t)) | (\{u, v\}, t) \in E\}$
  - $E_{W_1} := \{((w, i+1), (w, j)) | (w, i+1), (w, j) \in U \wedge i, j \in T(w) \wedge \tau_w(i) + 1 = \tau_w(j) \wedge i + i < j\}$
  - $E_{W_2} := \{((w, i), (w, j)) | (w, i), (w, j) \in U \wedge i, j \in T(w) \wedge \tau_w(i) + 1 = \tau_w(j) \wedge i + i < j\}$

Conclusions

- These static representations allow **non-parametric kernels** to produce informative features for **temporal graph classification**

- Might be worth **extending** the framework to GNNs and other tasks;

- We have initiated a(nother) **collaboration** with researchers from **UniTrento** to implement these ideas.

UNIVERSITÀ DI TORINO

CENTAI

Next Goals

1. Directed Link Prediction
2. Static Representations
3. AML via Neural Subgraph Matching

AML via Neural Subgraph Matching #1

- Current Machine/Deep Learning/(T)GNNs approach to AML make use of **node/edge classification**[1];

- Many of the above attempts act on **static** versions of temporal transaction network
  $\Longrightarrow$ **information loss**;

- **Time-aware** attempts (TGNNs) sometimes **outperformed** by simpler models and do not act at subgraph level[2] $\Longrightarrow$ **information loss**;

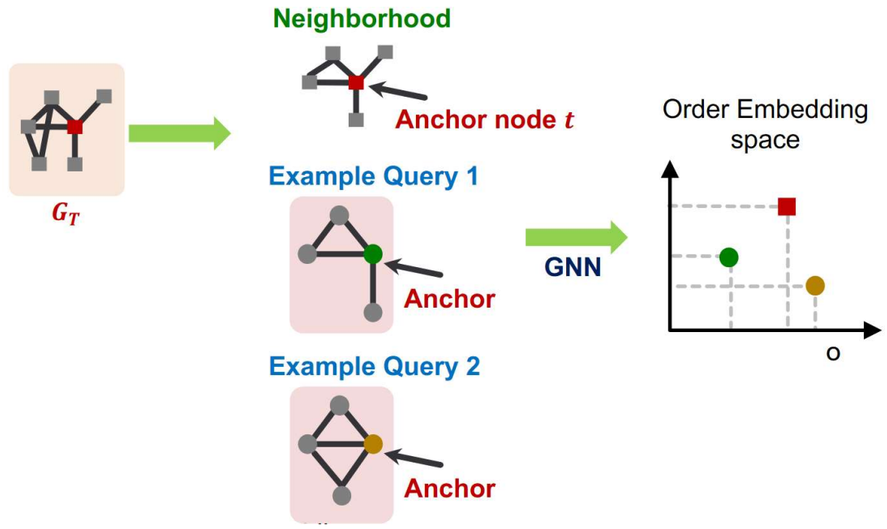
1. Jullum et al. (2020), Lorenz et al. (2021), Jensen and Iosifidis (2023), Weber et al. (2019), Liu et al. (2023), Johannessen Jullum (2023), Alarab and Prakoonwit (2022)
2. Johannessen and Jullum (2023), Weber et al. (2019)

AML via Neural Subgraph Matching #2

- **Statistical/Algorithmic** approaches to Subgraph Matching have been tried[1] but they're not **inductive** and may **scale** suboptimally
- **Neural** (approximate) Subgraph Matching/Mining/Counting algorithms Nguyes2023, Lan2022, Xu2022, (Roy et al. 2022), (Fey et al. 2020), (Rex et al. 2020)
- These methods may be applied to **static representations** of temporal networks.
- Alternatively, we may read the (very early) literature on **temporal neural subgraph matching**

1. Starnini et al. (2021)

AML via Neural Subgraph Matching #3



Loss:

$$l(G_q, G_t) = \sum_{i=1}^{D} (\max(0, \vec{z}_q[i] - \vec{z}_t[i]))^2$$

UNIVERSITÀ DI TORINO

CENTAI

Extras

1. Weisfeiler-Leman Kernel
2. Models
3. Graph Neural Networks
4. Expressivity of GNNs

Extras

1. Weisfeiler-Leman Kernel

2. Models

3. Graph Neural Networks

4. Expressivity of GNNs

Weisfeiler-Leman Kernel #1

Def.(Graph Isomorphism) : Two graphs (V_1, E_1) and (V_2, E_2) are **isomorphic** $\iff$

$$\exists \text{ bijection } f : V_1 \to V_2 | (u, v) \in E_1 \implies (f(u), f(v)) \in E_2$$

Obs. : Determining f is extremely **expensive** $\implies$ need for **approximate** algorithms $\implies$ **WL-kernel**

UNIVERSITÀ DI TORINO

CENTAI

Weisfeiler-Leman Kernel #2

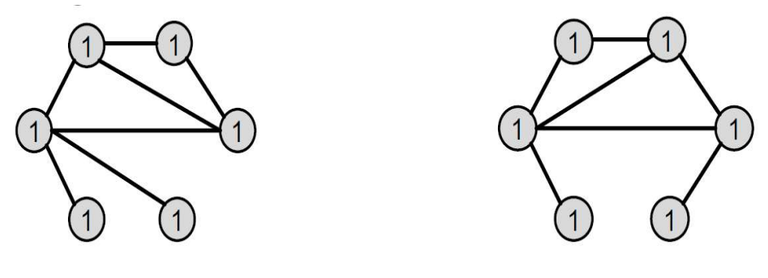
WL-Kernel:

1. Assign initial color $c_v^{(0)}$ to each node v;
2. Iteratively **refine** colors via:

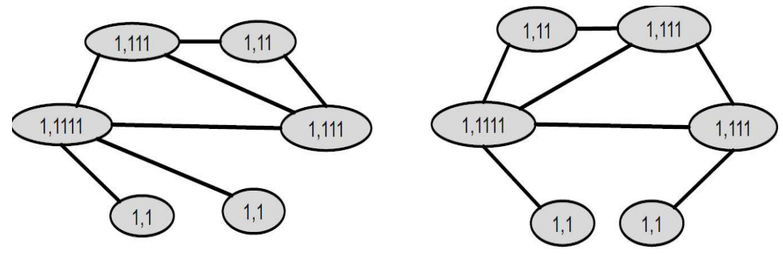
$$c_v^{(k+1)} = HASH\left(c_v^{(k)}, \{c^{(k)}{}_u | u \in N(v)\}\right)$$

3. Terminate when colors **stabilize** (#nodes iterations maximum);
4. Assign to each graph the empirical color distribution (histogram)

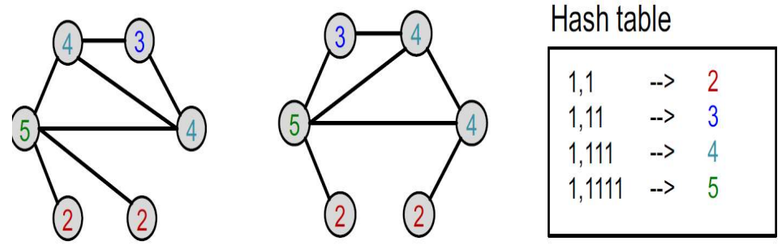
1.



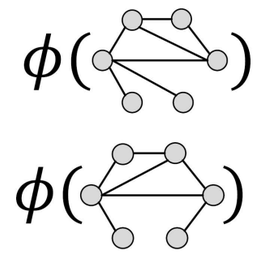
2.



3.



4



Colors
1,2,3,4,5,6,7,8,9,10,11,12,13
= [6,2,1,2,1,0,2,1,0, 0, 0, 2, 1]
Counts

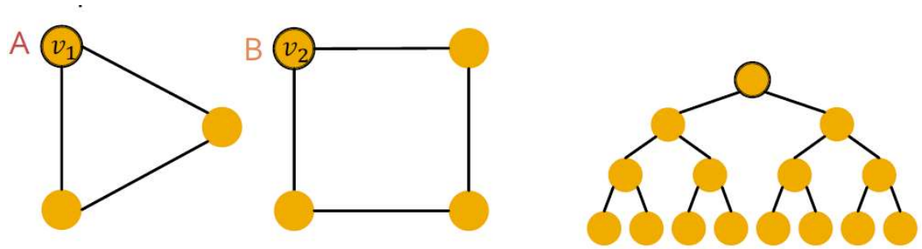
1,2,3,4,5,6,7,8,9,10,11,12,13
= [6,2,1,2,1,1,1,0,1, 1, 1, 0, 1]

UNIVERSITÀ DI TORINO

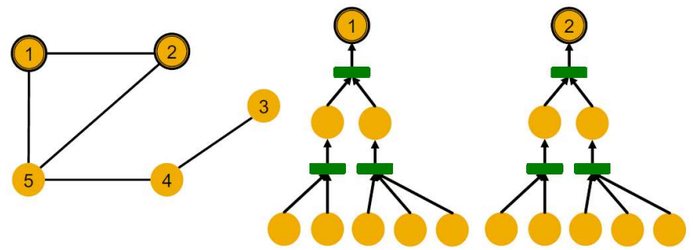
CENTAI

Weisfeiler-Leman Kernel #3

Obs. WL-Kernel **cannot distinguish** all non-isomorphic graphs e.g.:



And computes **same color** for different nodes:



Obs. : GNNs will be **differentiable** versions of $HASH \implies$ less expressive.

UNIVERSITÀ DI TORINO

CENTAI

Extras

1. Weisfeiler-Leman Kernel

2. Models

3. Graph Neural Networks

4. Expressivity of GNNs

Models #1

GraphSAGE GCN:

$$\vec{h}_v^{(k)} = \text{L2-NORMALIZE}\left(\sigma\left(W^{(k)}\text{MEAN}(\{\vec{h}_v^{(k-1)}\} \cup \{\vec{h}_u^{(k-1)}|u \in N(v)\})\right)\right)$$

```
class GraphSAGEConv_GCN(pyg.nn.MessagePassing):
    def __init__(self, in_channels, out_channels):
        super().__init__(aggr = "mean")
        self.W = torch.nn.Linear(in_channels, out_channels, bias = False)
        self.activation = torch.nn.ReLU()

    def message_and_aggregate(self, adj_t, x):

        return spmm(adj_t, x, reduce=self.aggr)

    def message(self, x_j):
        return x_j

    def update(self, inputs):

        unnormalized_embeddings = self.activation( self.W(inputs) ) # assumes add_self_loops was run before i

        normalized_embeddings = normalize(unnormalized_embeddings, p =2 , dim = 1)

        return  normalized_embeddings

    def forward(self, x, edge_index): #x, edge_index

```

UNIVERSITÀ DI TORINO

CENTAI

```
25          propagated  = self.propagate( edge_index, x=x)
```

Models #2

GraphSAGE MEAN:

$$\vec{h}_v^{(k)} = \text{L2-NORMALIZE}\left(\sigma\left(W_N^{(k)}\text{MEAN}(\{\vec{h}_u^{(k-1)}|u \in N(v)\}) + W_S^{(k)}\vec{h}_v^{(k-1)}\right)\right)$$

```python
class GraphSAGEConv_MEAN(pyg.nn.MessagePassing):
    def __init__(self, in_channels, out_channels):
        super().__init__(aggr = "mean")
        self.W_neigh = torch.nn.Linear(in_channels, out_channels, bias = False)
        self.W_self = torch.nn.Linear(in_channels, out_channels, bias = False)
        self.activation = torch.nn.ReLU()

    def message_and_aggregate(self, adj_t, x):

        return spmm(adj_t, x, reduce=self.aggr)

    def message(self, x_i, x_j):
        return x_j


    def forward(self, x, edge_index): #x, edge_index


        neigh_means  = self.propagate( edge_index, x=x)

        neigh_means_transformed = self.W_neigh(neigh_means)

        self_vecs_transformed = self.W_self(x)

        unnormalized_embeddings = self.activation(self_vecs_transformed + neigh_means_transformed)
```

UNIVERSITÀ DI TORINO

CENTAI

UNIVERSITÀ DI TORINO

CENTAI

Models #3

ModGIN:

$$\vec{h}_v^{(k)} = \text{MLP}_C\left(\text{CONCAT}\left(\sum_{u \in N(v)} \text{MLP}_N(\vec{h}_u^{(k-1)}), \vec{h}_v^{(k-1)}\right)\right)$$

```python
class ModGINConv(MessagePassing):
    def __init__(self, in_channels, out_channels):
        super().__init__(aggr = "add")

        self.in_channels = in_channels
        self.out_channels = out_channels

        self.neigh_net = Sequential(
            Linear(self.in_channels, self.out_channels),
            LeakyReLU(),
            Linear(self.out_channels, self.out_channels),
            LeakyReLU(),
        )

        self.concat_net = Sequential(
            Linear(self.out_channels + self.in_channels, self.out_channels),
            LeakyReLU(),
            Linear(self.out_channels, self.out_channels),
            LeakyReLU(),
        )

    def message_and_aggregate(self, adj_t, x):
```

```
23          return pyg.utils.spmm(adj_t, self.neigh_net(x), self.aggr) # torch_sparse.spmm
24
25      def forward(self, x, edge_index):
```

Models #4

ModGIN_Att_Norm:

$$\tilde{\vec{h}}_v^{(k)} = \text{MLP}_C\left(\left(\sum_{u \in N(v)} \text{MLP}_N(\vec{h}_u^{(k-1)} || \vec{h}_v^{(k-1)})\right) || \vec{h}_v^{(k-1)}\right)$$

```python
class ModGINConv(MessagePassing):
    def __init__(self, in_channels, out_channels):
        super().__init__(aggr = "add")

        self.in_channels = in_channels
        self.out_channels = out_channels

        self.neigh_net = Sequential(
            Linear(2*self.in_channels, self.out_channels),
            LeakyReLU(),
            Linear(self.out_channels, self.out_channels),
            LeakyReLU(),
        )

        self.concat_net = Sequential(
            Linear(self.out_channels + self.in_channels, self.out_channels),
            LeakyReLU(),
            Linear(self.out_channels, self.out_channels),
            LeakyReLU(),
        )

    def message(self, x_i, x_j):
```

```
23          return self.neigh_net(torch.cat((x_j,x_i), dim = 1))
24
25      def forward(self, x, edge_index):
26
```

Extras

1. Weisfeiler-Leman Kernel
2. Models
3. Graph Neural Networks
4. Expressivity of GNNs

Graph Neural Networks (GNNs) #1

Spectral derivation of GNNs:

1. Simultaneous discretization of a Riemannian Manifold $\mathcal{M}$ & Laplace-Beltrami operator yields a graph G and its laplacian $L = D - A$[1]

2. $\hat{L} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - \hat{A}$'s eigenvectors U used to define the Graph Fourier Transform:

$$\mathcal{F}(\vec{x}) = U^T\vec{x}$$

3. $\mathcal{F}$ used to define Graph Convolution:

$$\vec{g}_{\vec{\theta}} * \vec{x} = U\vec{g}_{\vec{\theta}}U^T\vec{x}$$

4. Interpret $\vec{g}_{\vec{\theta}}$ as function of $\hat{L}$'s eigenvalues $\hat{\Lambda}$, and approximate using Chebishev's polynomials:

$$\vec{g}_{\vec{\theta}} \approx \sum_{k=0}^{K} \theta_k T_k(\hat{\Lambda})$$

5. Truncate to **first-order**, **group** parameters, **renormalize**, apply **non-linearity** and generalize to matrices:

$$Z \approx \mathrm{ReLU}\left(\tilde{A}X\Theta\right)$$

where $\tilde{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$[2]

1. Nahmias (2017), Hein, Audibert, and Luxburg (2005), Burago, Ivanov, and Kurylev (2014), Lu (2020), Aubry (2013), García Trillos et al. (2019)
2. Kipf and Welling (2017)

Graph Neural Networks (GNNs) #2

Kipf's convolution **recovered** in MPNN framework by:

$$\begin{cases} \overrightarrow{\mathrm{COMB}}_t & = \mathrm{ReLU}(\cdot)\ (\text{no self-information}) \\ \overrightarrow{\mathrm{AGGR}}_t & = \sum \\ \overrightarrow{\mathrm{MSG}}_t \left(\vec{h}_v^{(t)}, \vec{h}_u^{(t)}, e_{uv}\right) & = \frac{W_t}{\sqrt{deg(u)}\sqrt{deg(v)}} \vec{h}_u^{(t)} \end{cases}$$

Spectral framework:

- Mathematically **rigid**;
- **Not necessary** for extensions;
- Included in MPNN framework.

$\Longrightarrow$ refer to **MPNN** framework

Other architectures are **GAT**, **GraphSAGE**, **SGCN**, etc[1]

1. Hamilton, Ying, and Leskovec (2017), Veličković et al. (2018),Chanpuriya and Musco (2022)

UNIVERSITÀ
DI TORINO

CENTAI

Extras

1. Weisfeiler-Leman Kernel
2. Models
3. Graph Neural Networks
4. Expressivity of GNNs

Expressivity of GNNs #1

- Best GNN? No **free-lunch**;
- Look for **most expressive** GNN i.e. GNN that can **distinguish** most different nodes/graphs;

Refer to:

$$\vec{h}_v^{(k+1)} = \overrightarrow{\mathrm{COMB}}_k \left(\vec{h}_v^{(k)}, \overrightarrow{\mathrm{AGGR}}_k \left(\left\{ \overrightarrow{\mathrm{MSG}}_k \left(\vec{h}_v^{(k)}, \vec{h}_u^{(k)}, e_{uv} \right) | u \in N(v) \right\} \right) \right)$$

Desiderata:

- $\overrightarrow{\mathrm{MSG}}_k$ injective over $\mathbb{R}^{n_k}$
- $\overrightarrow{\mathrm{AGGR}}_k$ injective over $\mathcal{M}(\mathbb{R}^{n_k}) = \left\{ \{\{\vec{h}_1^{(k)}\}, \{\vec{h}_2^{(k)}\}, \dots\} | \vec{h}_i^{(k)} \in \mathbb{R}^{n_k} \right\}$

- $\overrightarrow{\mathrm{COMB}}_k$ injective over $\mathbb{R}^{n_k} \times \mathbb{R}^{n_k}$

UNIVERSITÀ DI TORINO

CENTAI

Expressivity of GNNs #2

- By **Kolmogorov-Arnold's theorem**[1], every symmetric function can be represented as:

$$f(x_1, \ldots, x_n) = \rho\left(\sum_{i=1}^{n} \phi(x_i)\right) \text{ where } \rho \text{ and } \phi \text{ are injective nonlinearities}$$

- Thus, by the **Universal Approximation Theorem**, the **GIN** architecture[2] has a chance to match WL expressivity:

$$\vec{h}_v^{(t+1)} = \text{MLP}\left((1+\epsilon)\vec{h}_v^{(k)} + \sum_{u \in N(v)} \vec{h}_u^{(k)}\right)$$

- Result: GIN is **at most** as expressive as WL-Test

- More expressive GNNs may be obtained by mimicking k-WL kernel, or by extending the MPNN framework[3].

1. Wagstaff et al. (2022), Zaheer et al. (2017), wikipedia
2. Xu et al. (2019)
3. Morris et al. (2021), Du et al. (2018), Michel et al. (2023)

Thanks 🙏

UNIVERSITÀ DI TORINO

CENTAI

References

Alarab, Ismail, and Simant Prakoonwit. 2022. “Graph-Based LSTM for Anti-Money Laundering: Experimenting Temporal Graph Convolutional Network with Bitcoin Data.” *Neural Processing Letters* 55 (1): 689–707. https://doi.org/10.1007/s11063-022-10904-8.

Altman, Erik, Jovan Blanuša, Luc von Niederhäusern, Béni Egressy, Andreea Anghel, and Kubilay Atasu. 2023. “Realistic Synthetic Financial Transactions for Anti-Money Laundering Models.” https://arxiv.org/abs/2306.16424.

Aubry, Erwann. 2013. “Approximation of the Spectrum of a Manifold by Discretization.” arXiv. https://doi.org/10.48550/ARXIV.1301.3663.

Burago, Dmitri, Sergei Ivanov, and Yaroslav Kurylev. 2014. “A Graph Discretization of the Laplace–Beltrami Operator.” *Journal of Spectral Theory* 4 (4): 675–714. https://doi.org/10.4171/jst/83.

Cai, Lei, Jundong Li, Jie Wang, and Shuiwang Ji. 2020. “Line Graph Neural Networks for Link Prediction.” https://arxiv.org/abs/2010.10046.

Chanpuriya, Sudhanshu, and Cameron N Musco. 2022. “Simplified Graph Convolu with Heterophily.” In *Advances in Neural Information Processing Systems*, edit

Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. https://openreview.net/forum?id=jRrpiqxtrWm.

Du, Jian, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, and Soummya Kar. 2018. "Topology Adaptive Graph Convolutional Networks." https://arxiv.org/abs/1710.10370.

Fey, Matthias, Jan E. Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. 2020. "Deep Graph Matching Consensus." In *International Conference on Learning Representations*. https://openreview.net/forum?id=HyeJf1HKvS.

García Trillos, Nicolás, Moritz Gerlach, Matthias Hein, and Dejan Slepčev. 2019. "Error Estimates for Spectral Convergence of the Graph Laplacian on Random Geometric Graphs Toward the Laplace–Beltrami Operator." *Foundations of Computational Mathematics* 20 (4): 827–87. https://doi.org/10.1007/s10208-019-09436-w.

Gilmer, Justin, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. "Neural Message Passing for Quantum Chemistry." In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1263–72. ICML'17. Sydney, NSW, Australia: JMLR.org.

Gravina, Alessio, and Davide Bacciu. 2023. "Deep Learning for Dynamic Graphs: Models and Benchmarks." https://arxiv.org/abs/2307.06104.

Gupta, Shubham, and Srikanta Bedathur. 2022. “A Survey on Temporal Graph Representation Learning and Generative Modeling.” https://arxiv.org/abs/2208.12126.

Hamilton, William L., Rex Ying, and Jure Leskovec. 2017. “Inductive Representation Learning on Large Graphs.” In, 1025–35. NIPS’17. Red Hook, NY, USA: Curran Associates Inc.

Hein, Matthias, Jean-Yves Audibert, and Ulrike von Luxburg. 2005. “From Graphs to Manifolds – Weak and Strong Pointwise Consistency of Graph Laplacians.” In *Lecture Notes in Computer Science*, 470–85. Springer Berlin Heidelberg. https://doi.org/10.1007/11503415_32.

Jensen, Rasmus Ingemann Tuffveson, and Alexandros Iosifidis. 2023. “Qualifying and Raising Anti-Money Laundering Alarms with Deep Learning.” *Expert Systems with Applications* 214: 119037. https://doi.org/https://doi.org/10.1016/j.eswa.2022.119037.

Johannessen, Fredrik, and Martin Jullum. 2023. “Finding Money Launderers Using Heterogeneous Graph Neural Networks.” https://arxiv.org/abs/2307.13499.

Jullum, Martin, Anders Løland, Ragnar Bang Huseby, Geir Ånonsen, and Johannes Lorentzen. 2020. “Detecting Money Laundering Transactions with Machine Learning.” *Journal of Money Laundering Control* 23 (1): 173–86. https://doi.org/10.1108/jmlc-07-2019-0055.

Kipf, Thomas N., and Max Welling. 2016. “Variational Graph Auto-Encoders.” https://arxiv.org/abs/1611.07308.

———. 2017. “Semi-Supervised Classification with Graph Convolutional Networks.” https://arxiv.org/abs/1609.02907.

Kollias, Georgios, Vasileios Kalantzis, Tsuyoshi Idé, Aurélie Lozano, and Naoki Abe. 2022. “Directed Graph Auto-Encoders.” https://arxiv.org/abs/2202.12449.

Li, Zhiyuan, and Enhan He. 2023. “Graph Neural Network-Based Bitcoin Transaction Tracking Model.” *IEEE Access* 11: 62109–20. https://doi.org/10.1109/ACCESS.2023.3288026.

Lin, Dan, Jiajing Wu, Qi Xuan, and Chi K. Tse. 2022. “Ethereum Transaction Tracking: Inferring Evolution of Transaction Networks via Link Prediction.” *Physica A: Statistical Mechanics and Its Applications* 600: 127504. https://doi.org/https://doi.org/10.1016/j.physa.2022.127504.

Liu, Jiayi, Changchun Yin, Hao Wang, Xiaofei Wu, Dongwan Lan, Lu Zhou, and Chunpeng Ge. 2023. “Graph Embedding-Based Money Laundering Detection for Ethereum.” *Electronics* 12 (14): 3180. https://doi.org/10.3390/electronics12143180.

Longa, Antonio, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. 2023. “Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities.” https://arxiv.org/abs/2302.01018.

Lorenz, Joana, Maria Inês Silva, David Aparício, João Tiago Ascensão, and Pedro Bizarro. 2021. “Machine Learning Methods to Detect Money Laundering in the Bitcoin Blockchain in the Presence of Label Scarcity.” https://arxiv.org/abs/2005.14635.

Lu, Jinpeng. 2020. “Graph Approximations to the Laplacian Spectra.” *Journal of Topology and Analysis* 14 (01): 111–45. https://doi.org/10.1142/s1793525320500442.

Michel, Gaspard, Giannis Nikolentzos, Johannes Lutzeyer, and Michalis Vazirgiannis. 2023. “Path Neural Networks: Expressive and Accurate Graph Neural Networks.” https://arxiv.org/abs/2306.05955.

Morris, Christopher, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2021. “Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks.” https://arxiv.org/abs/1810.02244.

Nahmias, David. 2017. “An Exposition of Spectral Graph Theory.” https://dnahmias.com/docs/harmonicfinalpaper.pdf.

Oettershagen, Lutz, Nils M. Kriege, Christopher Morris, and Petra Mutzel. 2020. “Temporal Graph Kernels for Classifying Dissemination Processes.” In *Proceedings of the 2020 SIAM International Conference on Data Mining (SDM)*, 496–504. https://doi.org/10.1137/1.9781611976236.56.

Rex, Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. 2020. “Neural Subgraph Matching.” https://arxiv.org/abs/2007.030[illegible]

UNIVERSITÀ DI TORINO

CENTAI

Roy, Indradyumna, Venkata Sai Baba Reddy Velugoti, Soumen Chakrabarti, and Abir De. 2022. “Interpretable Neural Subgraph Matching for Graph Retrieval.” *Proceedings of the AAAI Conference on Artificial Intelligence* 36 (7): 8115–23. https://doi.org/10.1609/aaai.v36i7.20784.

Salha, Guillaume, Stratis Limnios, Romain Hennequin, Viet-Anh Tran, and Michalis Vazirgiannis. 2019. “Gravity-Inspired Graph Autoencoders for Directed Link Prediction.” In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 589–98. CIKM ’19. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3357384.3358023.

Starnini, Michele, Charalampos E. Tsourakakis, Maryam Zamanipour, André Panisson, Walter Allasia, Marco Fornasiero, Laura Li Puma, et al. 2021. “Smurf-Based Anti-Money Laundering in Time-Evolving Transaction Networks.” In *Lecture Notes in Computer Science*, 171–86. Springer International Publishing. https://doi.org/10.1007/978-3-030-86514-6_11.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc.

UNIVERSITÀ
DI TORINO

CENTAI