2019 SIAM International Conference on DATA MINING

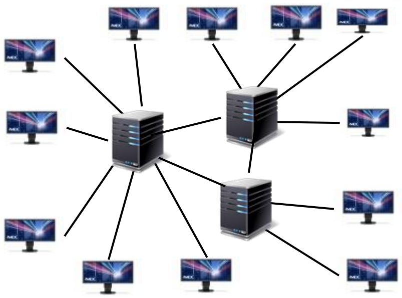
UIC

NEC Laboratories America
Relentless passion for innovation

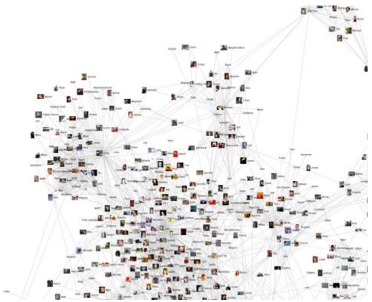
Attentional Heterogeneous Graph Neural Network: Application to Program Reidentification

Shen Wang[1], Zhengzhang Chen[2],
Ding Li[2], Zhichun Li[2], Lu-An Tang[2], Jingchao Ni[2],
Junghwan Rhee[2], Haifeng Chen[2] and Philip S. Yu[1]
[1]University of Illinois at Chicago
[2]NEC Laboratories America

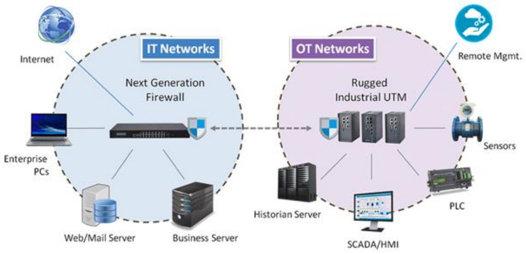
Background: Graphs/Networks



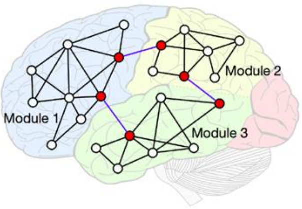
Enterprise networks



Social networks



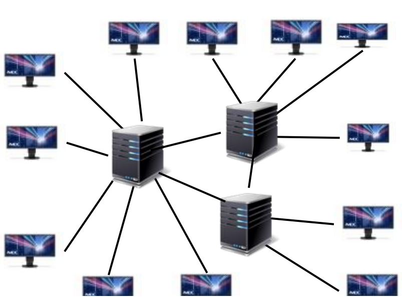
IT/OT networks

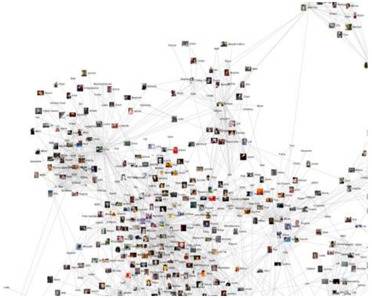


Brain networks

- **Ubiquitous** in real world
  - Graph is extensively employed within different fields
- A **flexible** and **general data structure**
  - Nature representation for linked data
- **Big Data**
  - Large Scale with Rich attributes

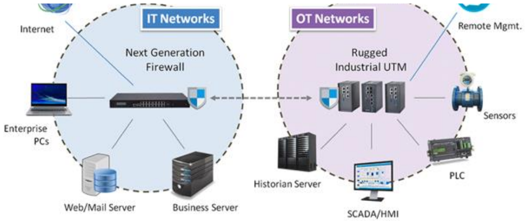
Background: Graphs/Networks



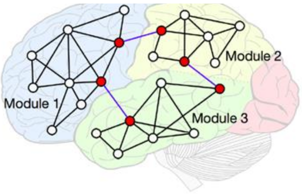


- **Ubiquitous** in real world
  - Graph is extensively employed within different fields

It requires a effective and efficient way to represent the Graphs.



IT/OT networks

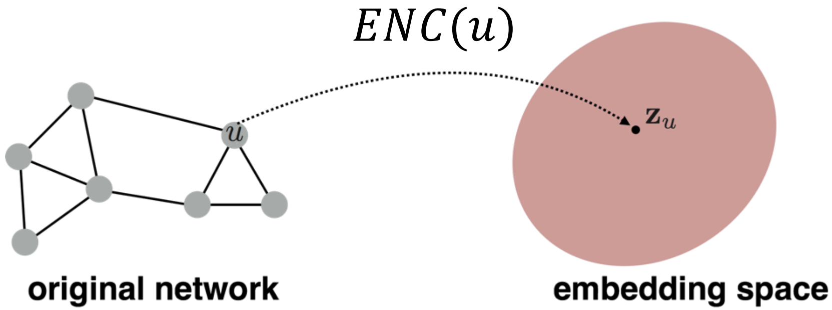


Brain networks

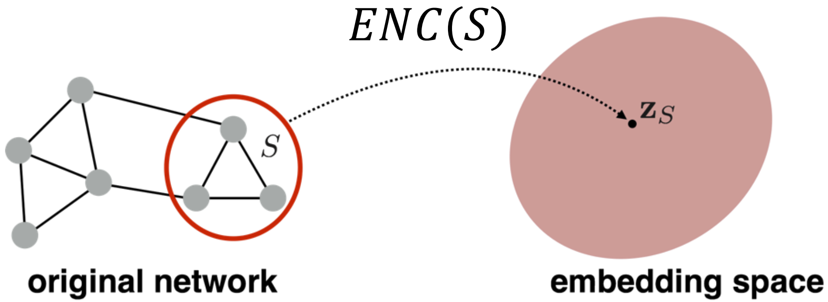
linked data

- **Big Data**
  - Large Scale with Rich attributes

Background: Network Embedding



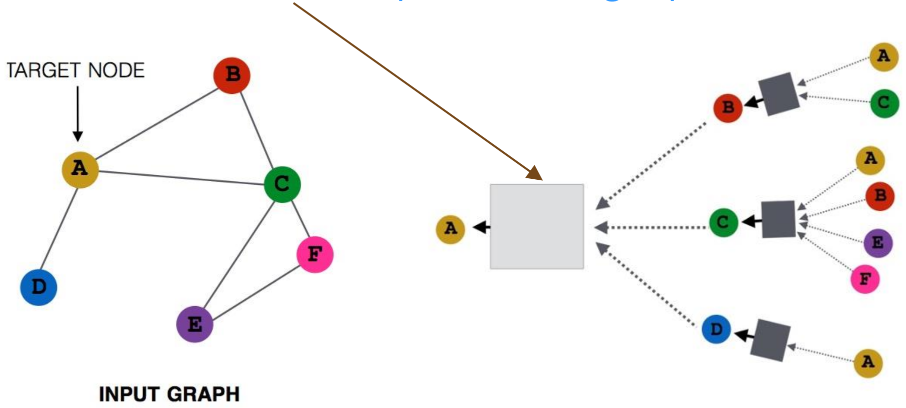
- Encode **nodes** so that the similarity in the embedding space approximates similarity in the original network
- $similarity(u_1, u_2) \approx z_{u_1}^T z_{u_2}$

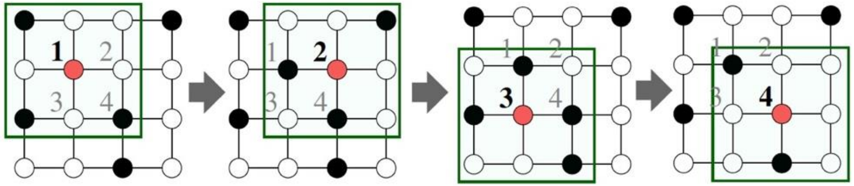


- Encode **subgraph/graph** so that the similarity in the embedding space approximates similarity in the original network
- $similarity(S_1, S_2) \approx z_{S_1}^T z_{S_2}$

Background: Graph Neural Network: an effective and efficient way of Network Embedding

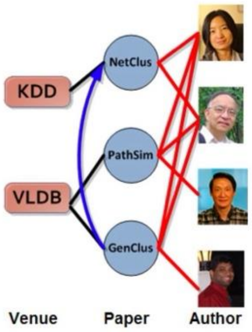
$ENC(V) =$ complex function that depends on graph structure.



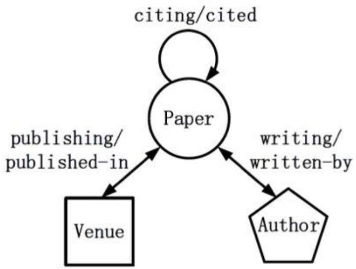


- GNN Methods:
  - Generate **node embeddings** based on **local neighborhoods**
  - **Nodes aggregate information** from their **neighbors** using **neural networks**
  - Leverage a **center-surround filter**
- Examples:
  - GCN, Diffusion Convolution Network, GraphSAGE, Gated Graph Neural Network
- **Limitations**
  - Only apply to **homogeneous** graph
  - Only focus on **node embedding**

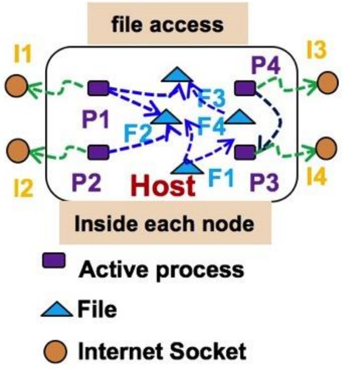
Background: Heterogeneous vs Homogeneous



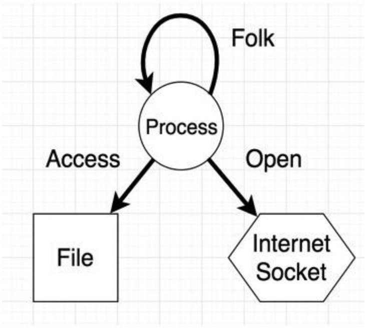
(a) Network instance



(b) Network schema



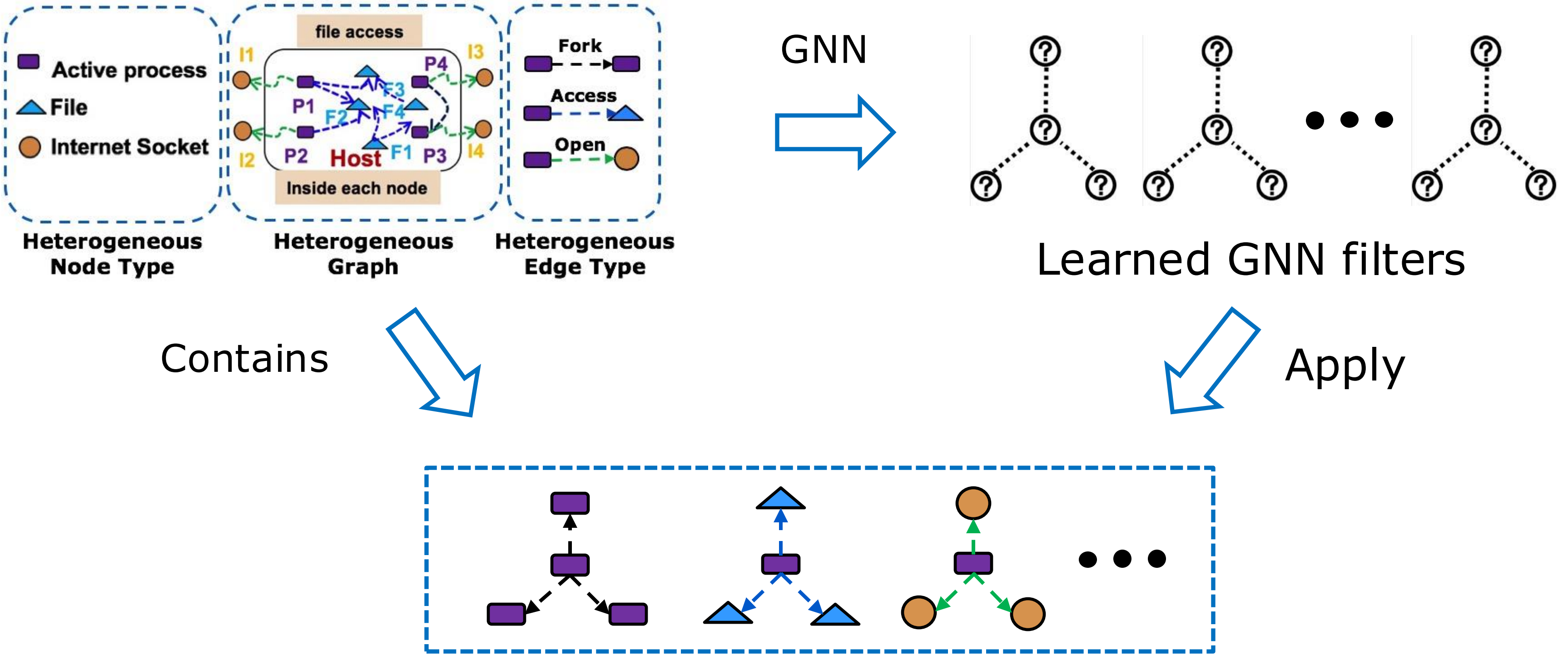
(a) Network instance



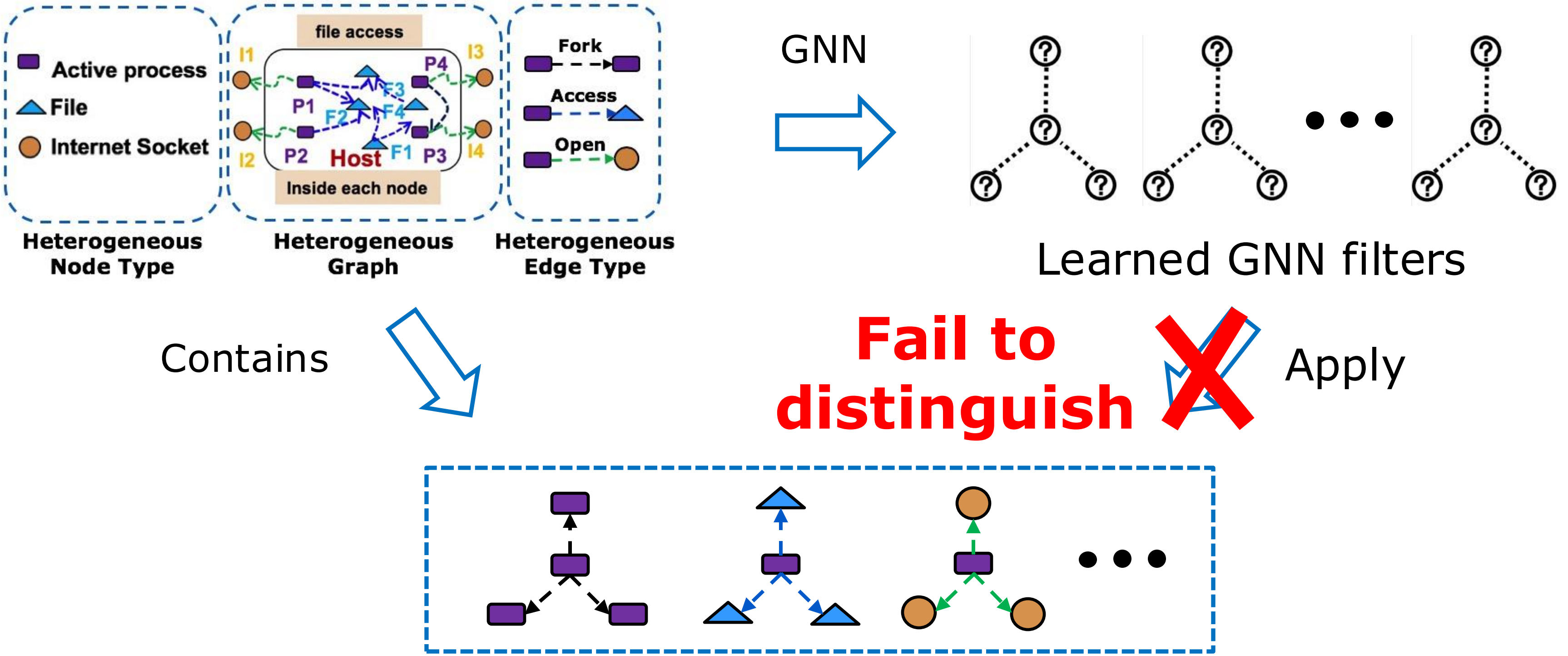
(b) Network schema

- Most real-world graph are **Heterogeneous Graph**
- **Heterogeneous Graph** vs **Homogeneous Graph**
  - **Entities/Nodes:**
    - **multiple types** vs **single type**
  - **Links/Edges:**
    - **multiple types** vs **single type**

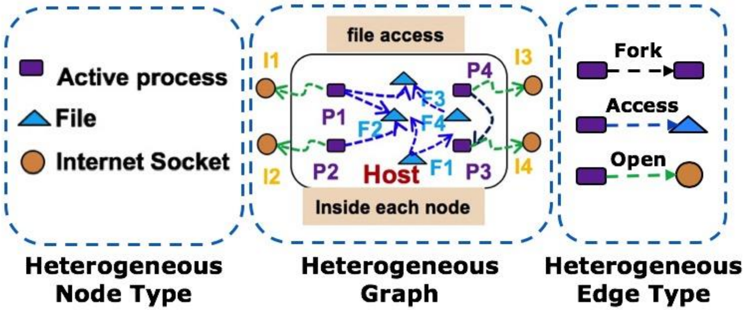
Tradition GNNs on Heterogeneous Graph



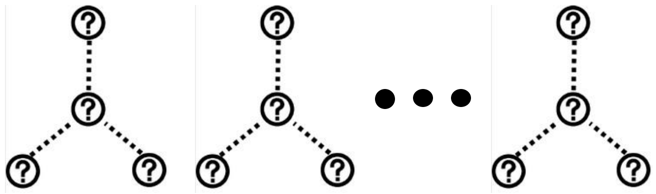
Tradition GNNs on Heterogeneous Graph



Tradition GNNs on Heterogeneous Graph



GNN

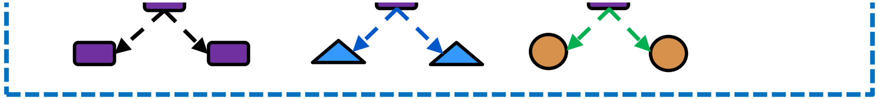


Learned GNN filters

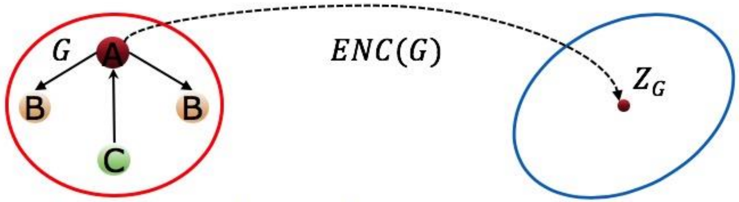
Contains

Apply

How to learn GNN filter to preserve the heterogeneous graph structure?



Problem Statement



- Assume we have a graph G with:
  - V_A, V_B, V_C are different sets of **vertices** belong to **different types**, and each vertex has high-dimensional **features** (categorical attributes, text, image data, node degrees, clustering coefficients, indicator vectors)
  - $E_{A\to B}, E_{C\to A}$ are the sets of **edges** belong to **different types**
- **Goal**
  - Find a **neural network based function** that **encodes** the **graph** G into a low-dimensional **vector**

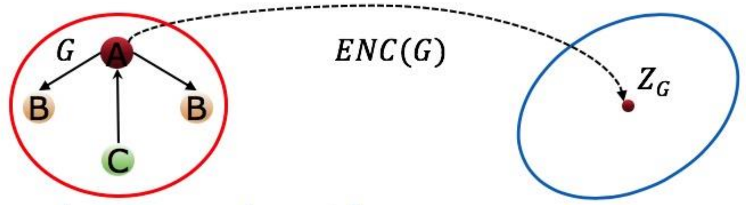
$$ENC(G) = Z_G$$

d-dimensional embedding

input graph

- **Applications**
  - Program Reidentification: given a target program with corresponding event data during a time window and a claimed name/ID, check whether it belongs to the claimed name/ID

Problem Statement



Challenge 1: How to preserve the heterogeneous graph structure?

- $E_{A\to B}, E_{C\to A}$ are the sets of **edges** belong to **different types**

- **Goal**
  - Find a **neural network based function** that **encodes** the **graph** G into a low-dimensional **vector**

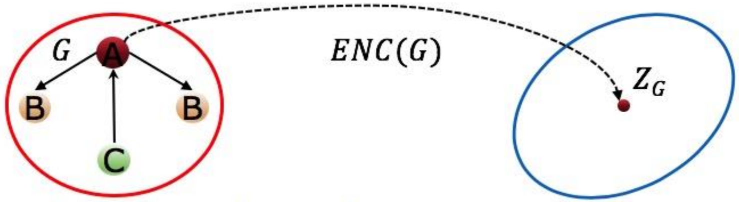
$$ENC(G) = Z_G$$

d-dimensional embedding

input graph

- **Applications**
  - Program Reidentification: given a target program with corresponding event data during a time window and a claimed name/ID, check whether it belongs to the claimed name/ID

Problem Statement



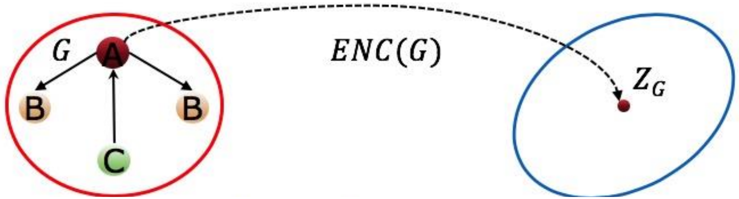
Challenge 1: How to preserve the heterogeneous graph structure?

Challenge 2: How to capture the hierarchy of different dependencies from simple to complex?

input graph

- **Applications**
  - Program Reidentification: given a target program with corresponding event data during a time window and a claimed name/ID, check whether it belongs to the claimed name/ID

Problem Statement



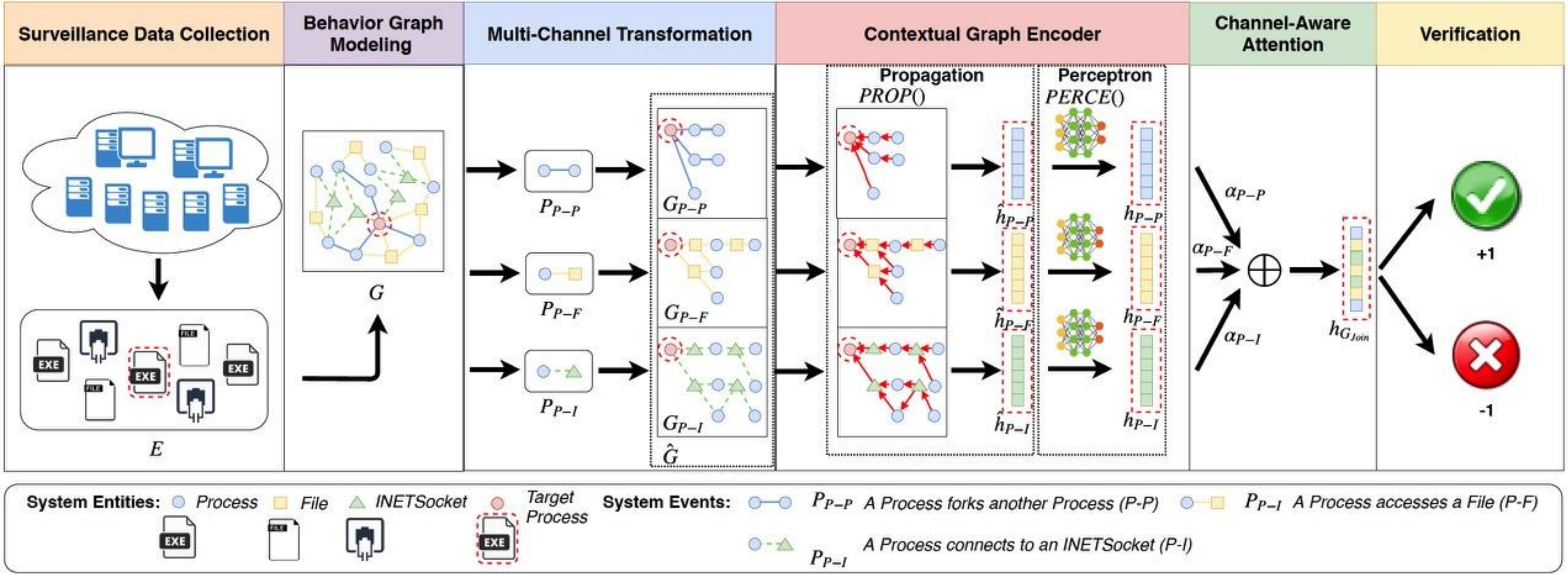
Challenge 1: How to preserve the heterogeneous graph structure?

Challenge 2: How to capture the hierarchy of different dependencies from simple to complex?

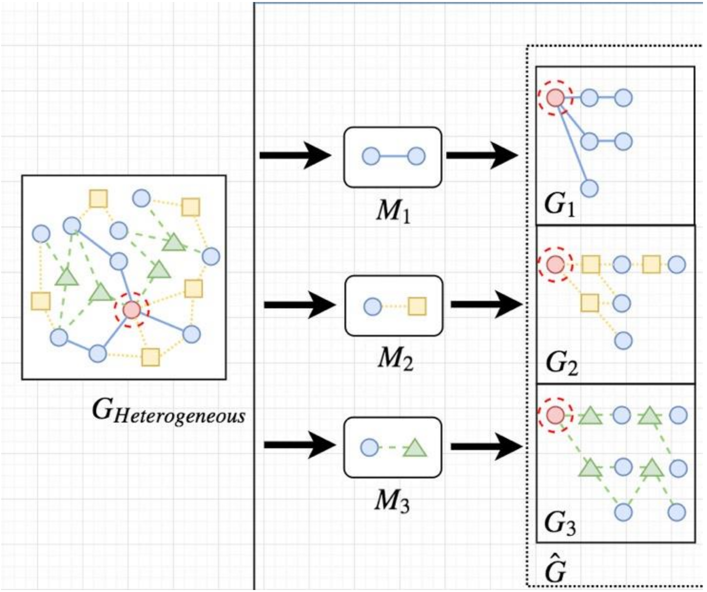
Challenge 3: How to deal with the different importances of different dependencies?

during a time window and a claimed name/ID, check whether it belongs to the claimed name/ID

An overview of the proposed DeepHGNN for program reidentification



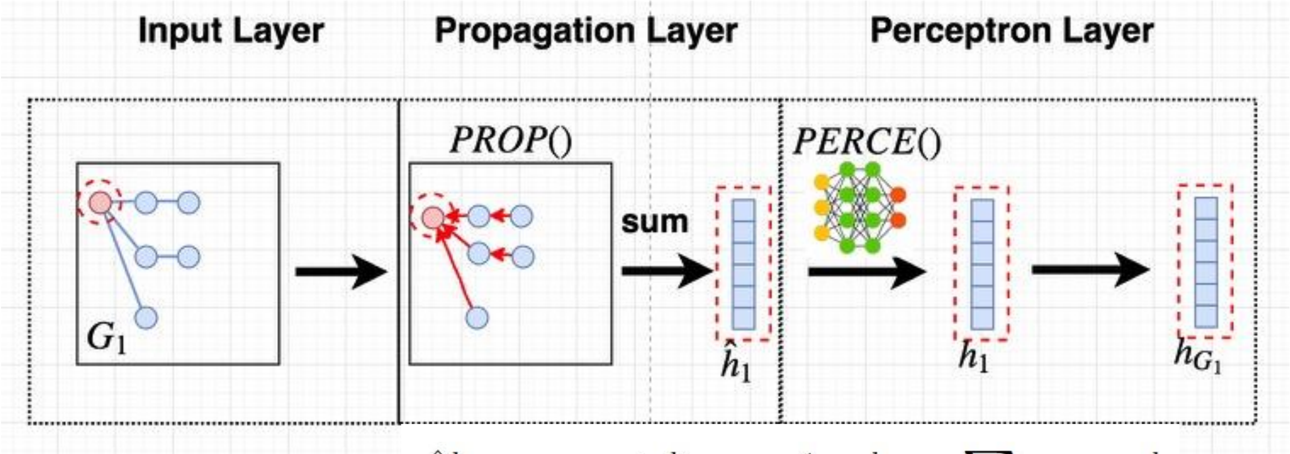
Multi-Channel Transformation



$$\hat{G} = \{G_i | G_i = (V_i, E_i, A_i), i = 1, 2, \ldots, |C|\}$$

- **Motivation:** GNN filter is required to capture the **heterogeneous network structure**
- **How to:** Transform the **heterogeneous graph** to **multi-channel graph** with the guide of **meta-paths**
  - **Meta-path:** a path that connects entity types via a sequence of relations over a heterogeneous network.
    - A process forks another process (P -> P)
    - A process accesses a file (P -> F)
    - A process opens an Internet socket (P -> I)
    - Two processes access the same file (P -> F <- P
    - Two processes open the same Internet socket (P -> I <- P)
- **Contribution: Heterogeneous-aware** filter can be learned
  - Diverse filter can capture heterogeneous structure

Contextual Graph Encoder (CGE)

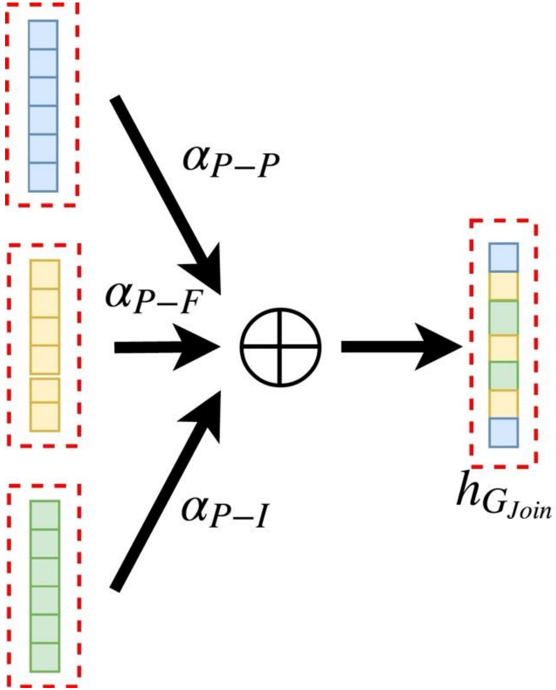


- Propagation function $\hat{h}^l = PROP(h^l) = D^{-1}Ah^l = \sum_{u \in N(v_t)} P_{uv_t} h^l$
  - Propagation matrix: P
  - Graph receptive field: $\mathcal{F} = \{N(v_t)\},$
- Intuition
  - Propagate contexts via **diffusion process** characterized by a **random walk** on the graph with a specific probability $q \in [0,1]$ and a state transition matrix $D^{-1}A$
  - Propagation layer computes **weighted sum** of the **full set** of **1-hop** contexts' current representation

Contextual Graph Encoder(CGE): General View

- **Key Idea:** Generate **graph embeddings** based on **local contexts**.
- **Intuition:** Nodes **aggregate information** from their **context** using **neural networks**
- **Architecture:**
  - **Input layer:** **extract** node features
  - **Propagation Layer:** **aggregate** contexts information
  - **Perceptron Layer:** **map** the aggregated contexts information to specific nonlinear space

Channel-Aware Attention



- **Motivation:** Leverage the **correlation of different channels** to assign each channel a **specific weight**
- **How to:**
  - Compute the **attention weight**

$$\alpha_i = \frac{\exp(\sigma(a[W_a h_{G_i} || W_a h_{G_k}]))}{\sum_{k' \in |C|} \exp(\sigma(a[W_a h_{G_i} || W_a h_{G_{k'}}]))}$$

  - Compute the **attentional joint embedding**

$$h_{G_{Join}} = \sum_{i=1}^{|C|} ATT(h_{G_i}) h_{G_i}$$

- **Contribution:** Help to learn the **joint embedding** with considering the **importance of different channels**

Experiment Setup

- Baselines:
  - LR and SVM
  - XGB
  - MLP
- Dataset:
  - Real-world system events monitoring data in Windows OS
- Evaluation Metrics:
  - ACC
  - F-1 score
  - AUC score
  - Precision
  - Recall

Synthetic Experiment Results

Meta-Path	Evaluation Criteria		
	ACC	F-1	AUC
DeepHGNN$_{pp}$	0.838	0.864	0.843
DeepHGNN$_{pf}$	0.821	0.855	0.838
DeepHGNN$_{pi}$	0.579	0.635	0.592
DeepHGNN$_{con}$	0.876	0.901	0.890
DeepHGNN$_{att}$	**0.905**	**0.929**	**0.908**

Table 1: Reidentification results of different meta-paths.

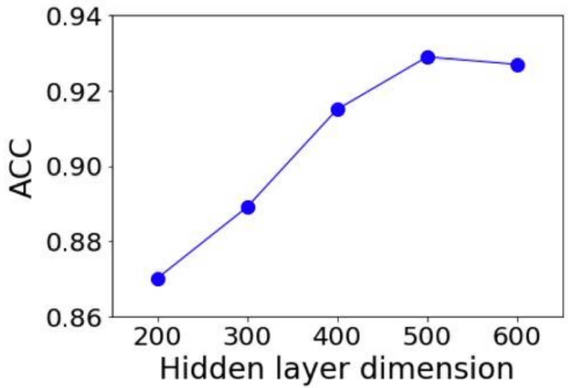
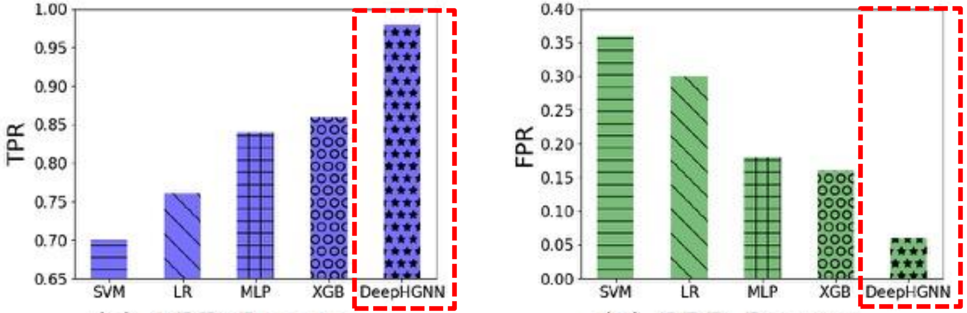


Figure 2: Parameter sensitivity analysis results.

Method	Settings	Evaluation Criteria				
		ACC	F-1	AUC	Precision	Recall
LR	*fea-1*	0.693	0.755	0.699	0.632	0.948
	fea-2	0.705	0.770	0.703	0.655	0.950
	fea-3	0.724	0.772	0.727	0.675	0.948
SVM	*fea-1*	0.502	0.662	0.502	0.505	0.970
	fea-2	0.795	0.778	0.725	0.701	0.935
	fea-3	0.504	0.652	0.504	0.505	**0.975**
XGB	*fea-1*	0.775	0.802	0.776	0.732	0.930
	fea-2	0.833	0.860	0.846	0.821	0.936
	fea-3	0.855	0.866	0.856	0.827	0.937
$MLP_{shallow}$	*fea-1*	0.633	0.745	0.643	0.626	0.938
	fea-2	0.775	0.808	0.779	0.724	0.932
	fea-3	0.778	0.808	0.780	0.726	0.932
MLP_{deep}	*fea-1*	0.633	0.743	0.653	0.625	0.945
	fea-2	0.801	0.830	0.805	0.769	0.921
	fea-3	0.815	0.831	0.816	0.778	0.923
DeepHGNN$_{shallow}$	/	0.905	0.929	0.908	0.905	0.933
DeepHGNN$_{deep}$	/	**0.929**	**0.961**	**0.935**	**0.932**	0.936

Table 2: Comparison on normal program reidentification.

Real-world Experiment Results



(a) TPR Results. (b) FPR Results.

Figure 3: Disguised program detection results.

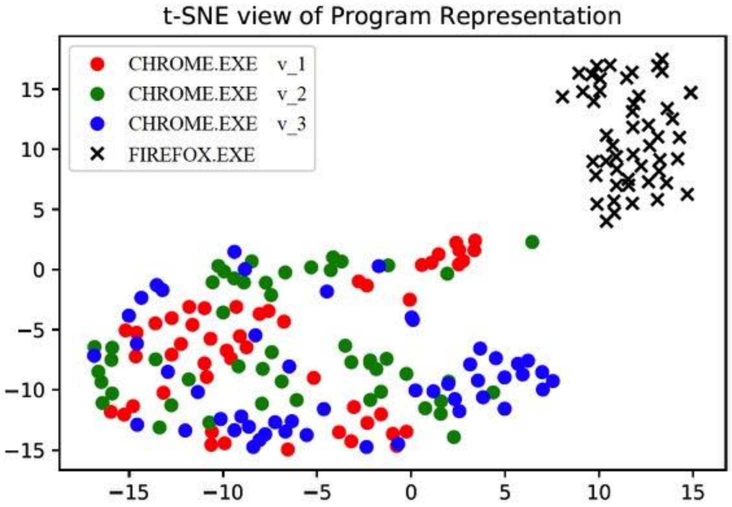
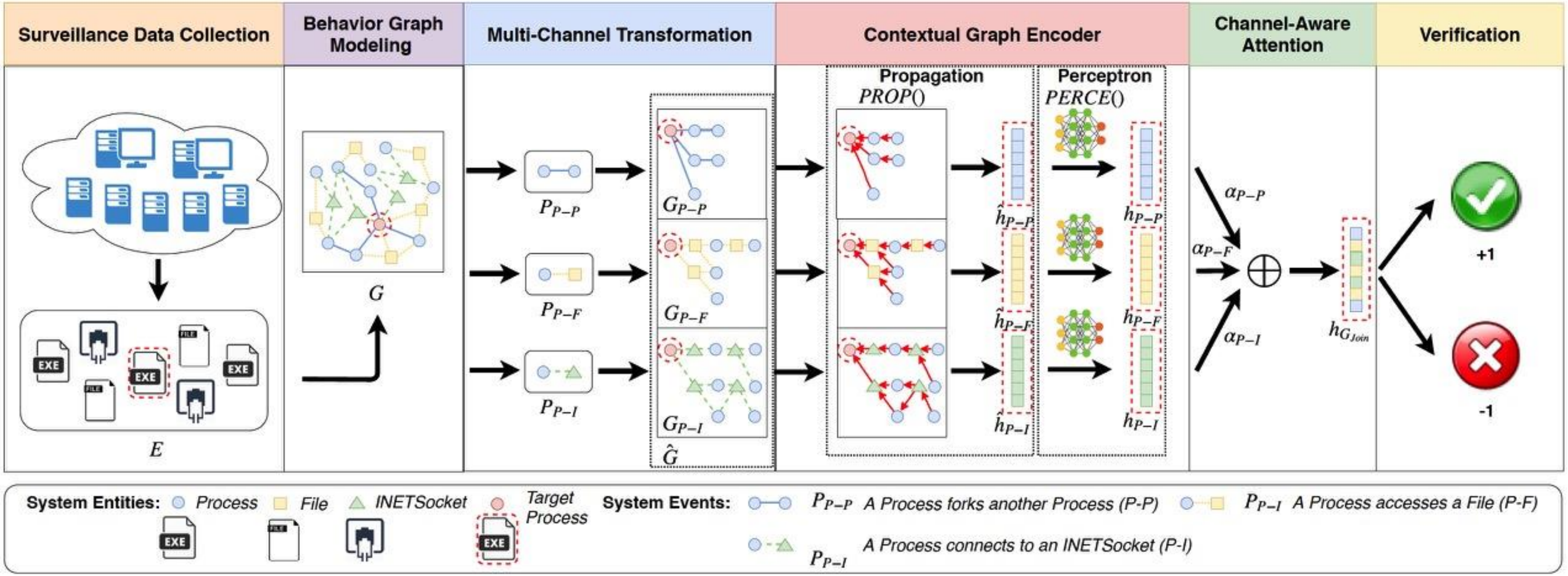


Figure 4: Scatter plot embedding of different versions of CHROME.EXE vs FIREFOX.EXE.

Summary

- **First attempt** to study the **Graph Neural Network** on **Heterogeneous Graph** in an **attentional mechanism**
  - Directly handle the heterogeneous graph and preserve the heterogeneous relationship
- We propose **Deep Heterogeneous Graph Neural Network**
  - General graph embedding framework based on graph neural network
- **Effectively** and **efficiently** applied in the real-world tasks of program reidentification

Thank you!


Surveillance Data Collection
Behavior Graph Modeling
Multi-Channel Transformation
Contextual Graph Encoder
Channel-Aware Attention
Verification
Propagation
PROP()
Perceptron
PERCE()
EXE
FILE
E
G
P_{P-P}
P_{P-F}
P_{P-I}
G_{P-P}
G_{P-F}
G_{P-I}
$\hat{G}$
$\hat{h}_{P-P}$
$\hat{h}_{P-F}$
$\hat{h}_{P-I}$
h_{P-P}
h_{P-F}
h_{P-I}
α_{P-P}
α_{P-F}
α_{P-I}
$h_{G_{Join}}$
+1
-1
System Entities: Process File INETSocket Target Process
System Events: P_{P-P} A Process forks another Process (P-P) P_{P-I} A Process accesses a File (P-F)
P_{P-I} A Process connects to an INETSocket (P-I)


Q&A