Learning Graph-based Code Representations for Source-level Functional Similarity Detection

Jiahao Liu[*], *Jun Zeng*[*], *Xiang Wang, and Zhenkai Liang*

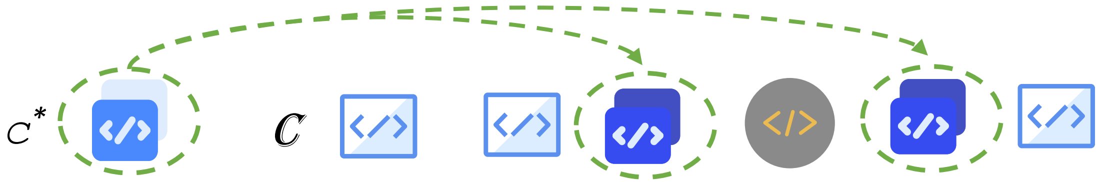
IEEE/ACM ICSE, May 2023

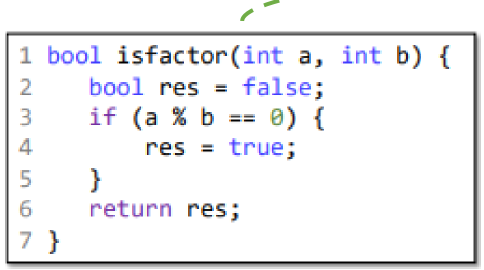
NUS National University of Singapore

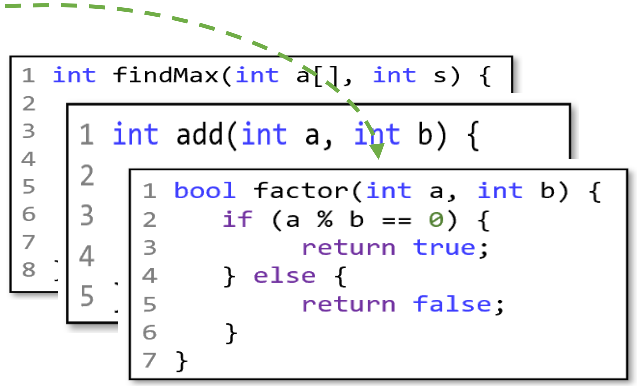
1958 University of Science and Technology of China

Code Functional Similarity Detection

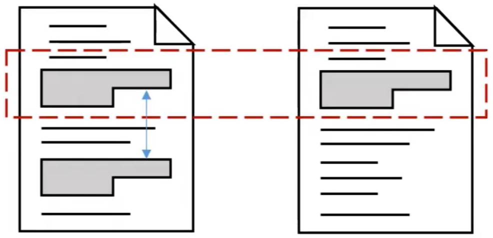
Given a code fragment c^*, and a corpus of code fragments $\mathbb{C} = \{c_1, c_2, \ldots\}$, how to *identify* candidates in $\mathbb{C}$ that are *functionally similar* with c^*?



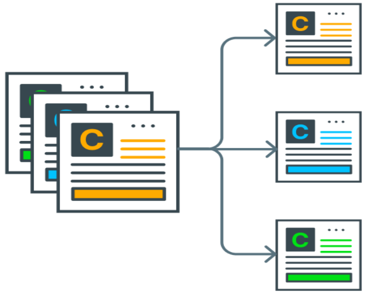




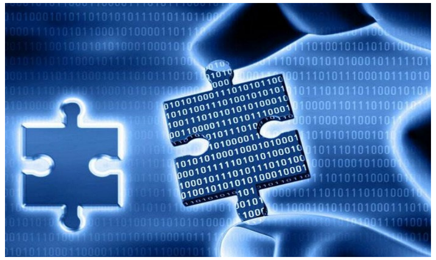
The cornerstone of various software engineering tasks:



Code clone detection



Code classification



Code search

Bug detection

Previous Solutions Detecting Code Similarity

Token-based methods [Sourcerercc @ICSE'16, NIL @FSE'21, …]

- Lack of program structures → textually different yet structurally similar codes

```
bool isfactor(int a, int b) {
    bool res = false;
    if (a % b == 0) {
        res = true;
    }
    return res;
}
```

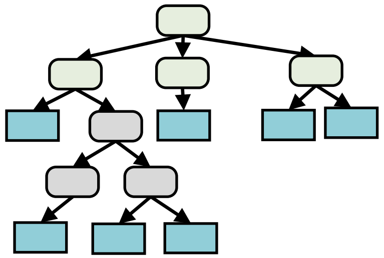
bool isfactor int a int b bool res = …

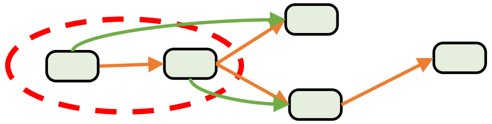
Tree-based methods [ASTNN @ICSE'19, InferCode @ICSE'21, …]

- Agnostic to program semantics → semantically similar programs with different syntax

Graph-based methods [Deepsim @FSE'18, …]

- Focus on local information → lack of overall graph structure





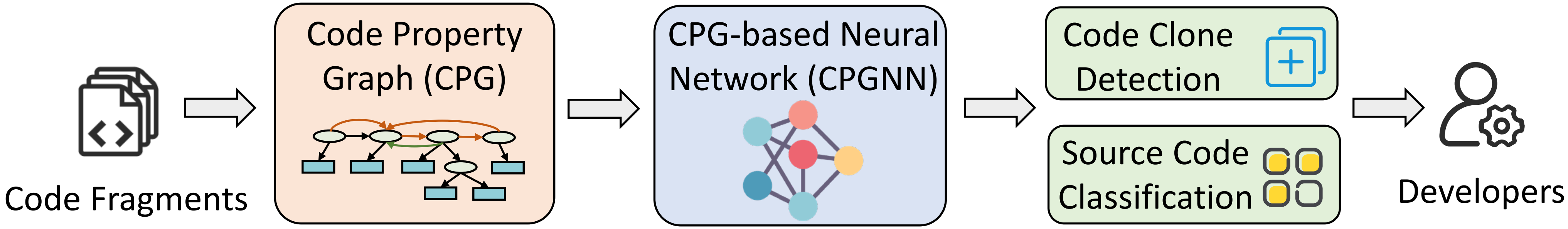
Our Insights

- Which code representations should be used to comprehend programs?
  - Abstract Syntax Tree (AST) well describes program syntax; Control Flow Graph (CFG) and Data Flow Graph (DFG) carry semantic information
  - Combining *AST, CFG,* and *DFG* together as *Code Property Graph* (CPG) benefits program understanding

- How to capture useful information from CPG for similarity detection?
  - Graph neural network (GNN) is powerful at capturing graph-structure features
  - GNN is not originally designed for program analysis
  - Tailor a GNN to learn graph-based code representation for similarity detection

Goal: Customize a **Graph Neural Network** on **Code Property Graph** to facilitate functional similarity detection

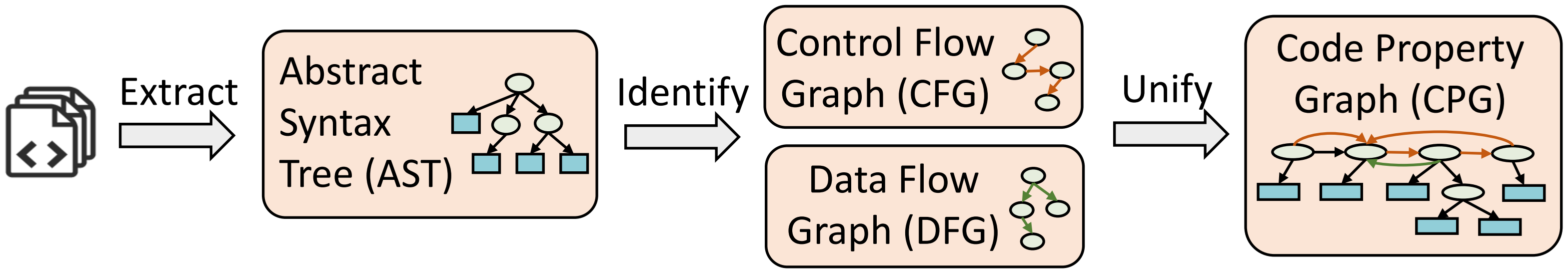
TAILOR: Overview



- Transform code fragments into *code property graph* (CPG)
- Model code representations with *GPG-based Neural Network* (CPGNN)
- Detect *code functional similarity* (code clone and classification)

Code Property Graph (CPG)

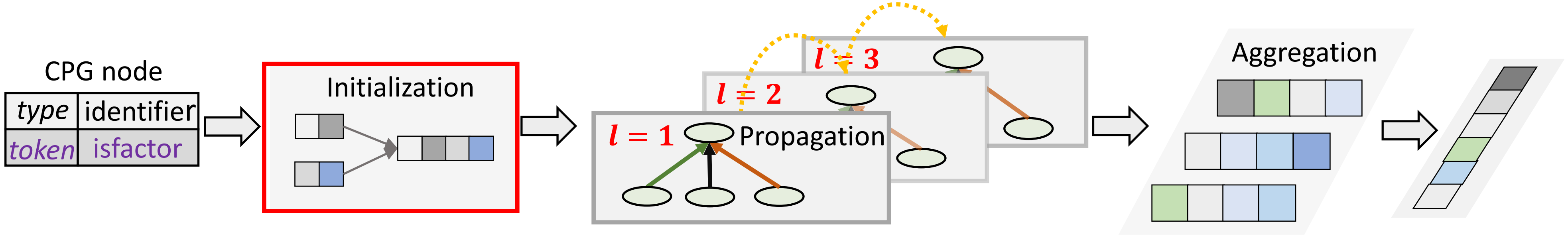
Code property graph (CPG) integrates *abstract syntax tree* (AST), *control flow graph* (CFG), and *data flow graph* (DFG)



- Extract AST as the basis for generating other code representations
- Identify CFG by analyzing control flows within and across functions
- Identify use-def variables in code statements to form data-flow edges
- Unify AST, CFG, and DFG into a joint CPG, whose nodes are the same with AST nodes, and edges include AST, CFG, DFG edges

CPG-based Graph Neural Network

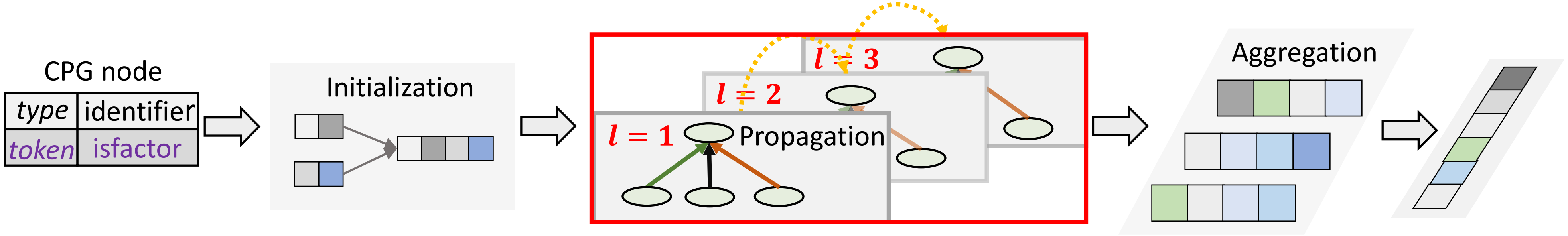
Key Idea: design a graph neural network (CPGNN) tailored to learn code representations for functional similarity detection



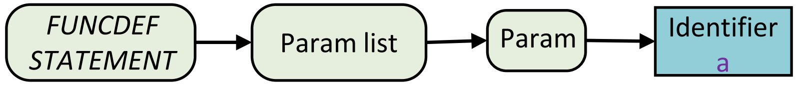
- ***Fuse*** type and token features to represent CPG nodes
  - Employ word2vec to embed type and token symbols into the vector space
  - Concatenate type and token embeddings to generate CPG node embedding

CPG-based Graph Neural Network

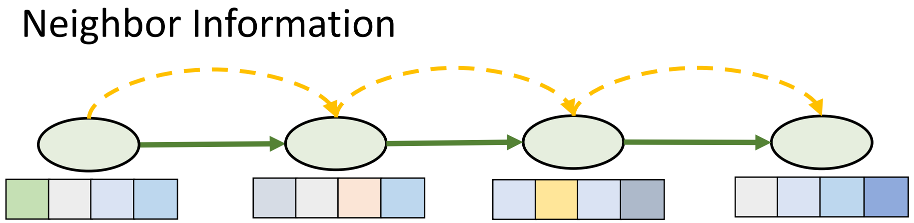
Key Idea: design a graph neural network (CPGNN) tailored to learn code representations for functional similarity detection



- ***Propagate*** node embeddings along CPG edges to learn graph structures
  - Enrich node semantics and contexts with graph structure, e.g., how variables are defined and used

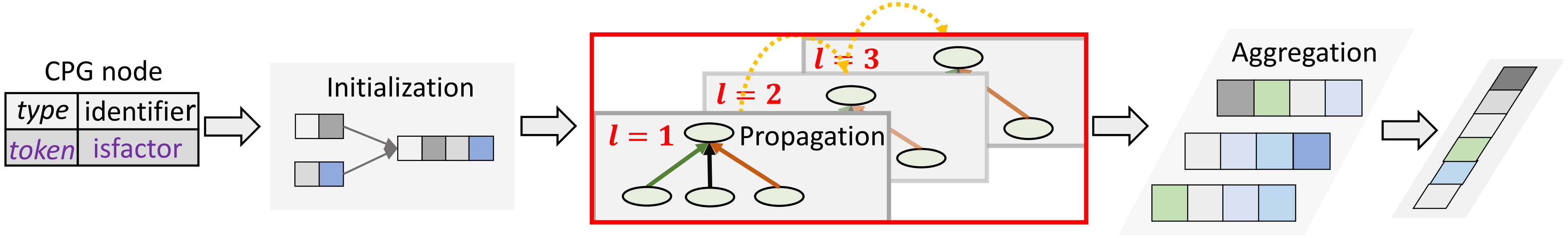


  - Propagate neighbor information along CPG paths

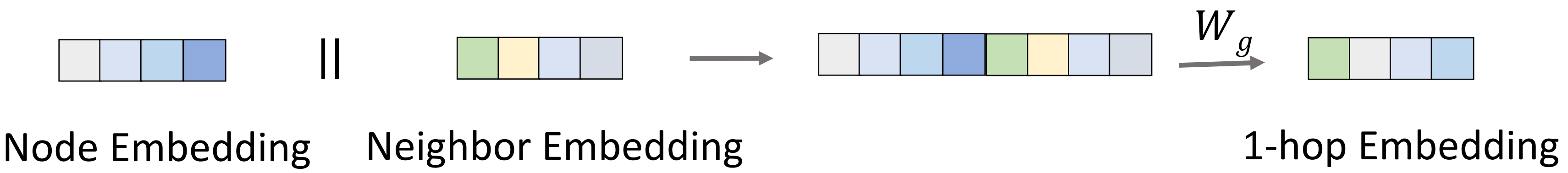


CPG-based Graph Neural Network

Key Idea: design a graph neural network (CPGNN) tailored to learn code representations for functional similarity detection

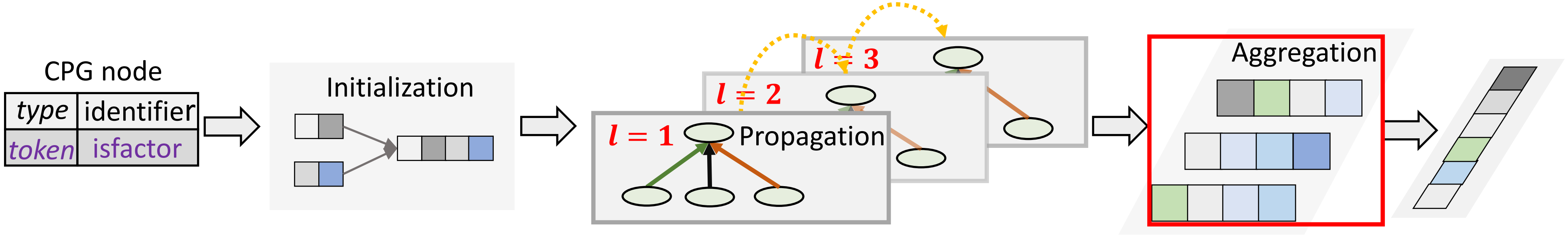


- ***Update*** node embeddings with neighbor information
  - Recent studies apply GRU from off-the-shelf GGNN [Devign @NeurIPS'19]
  - Limitation: Increase training difficulty and decrease GNN's effectiveness
  - Solution: Concatenation with a trainable matrix

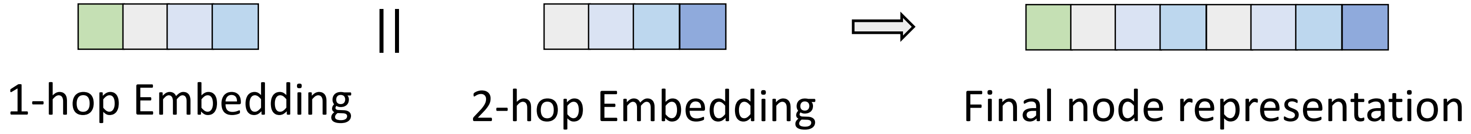


CPG-based Graph Neural Network

Key Idea: design a graph neural network (CPGNN) tailored to learn code representations for functional similarity detection



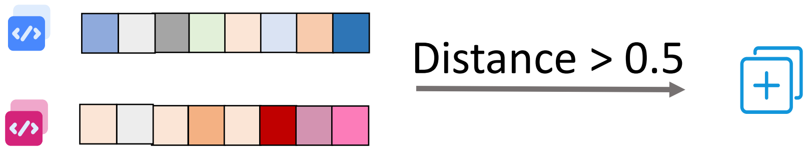
- ***Stack*** multi-hop neighbor information
  - Neighbor information at different hops describes global graph structure
  - Concatenate multi-hop neighbor embeddings to generate final node representation



Learning to Code Similarity Detection

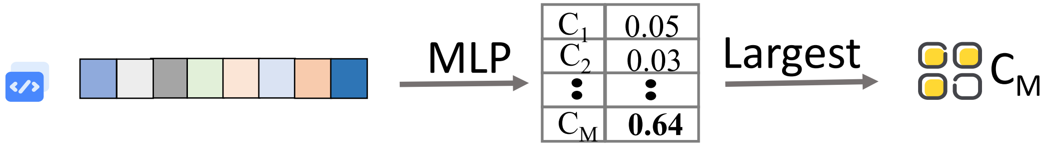
To produce neural representations for a code fragment, we combine its CPG nodes using the pooling operation

- Code clone detection (CCD): calculate Euclidean distance of code representations to infer clone pairs



Code clone detection (CCD)

- Source code classification (SCC): apply fully connected layers (MLP) to classify it into different categories



Source code classification (SCC)

Evaluation

Experiment Setup:

- **OJClone** constructed from 52,000 C programs belonging to 104 tasks
  - Code Clone Detection (CCD): 19,800 clone pairs, 300,000 non-clone pairs
  - Source Code Classification (SCC): all 52,000 programs
- **BigCloneBench**, constructed from 25,000 Java systems
  - Code Clone Detection (CCD): 71,677 clone pairs and 20,000 non-clone pairs

Evaluation aspects:

- How **effective** is TAILOR in code clone detection and classification?
- How does CPGNN **contribute** to TAILOR compared with off-the-shelf GNNs?
- To what extent do different design choices of CPGNN **affect** TAILOR's performance?

Effectiveness in Functional Similarity Detection

Compare TAILOR with state-of-the-art approaches on code clone detection (*F-score*) and source code classification (*Accuracy*)

Code Clone Detection (CCD)

	Token-based		Tree-based			Graph-based			
Dataset	SourcererCC	NIL	RtvNN	Code2Vec	ASTNN	FCDetector	FA-AST	Mocktail	**TAILOR**
OJClone	16.4	54.3	69.4	85.2	95.1	91.8	/	94.5	**99.9**
BCB	57.9	66.1	83.7	93.0	97.2	/	98.5	/	**99.8**

Source Code Classification (SCC)

	Tree-based			Graph-based	
Dataset	Code2Vec	InferCode	ASTNN	Mocktail	**TAILOR**
OJClone	64.2	93.0	97.9	85.5	**98.3**

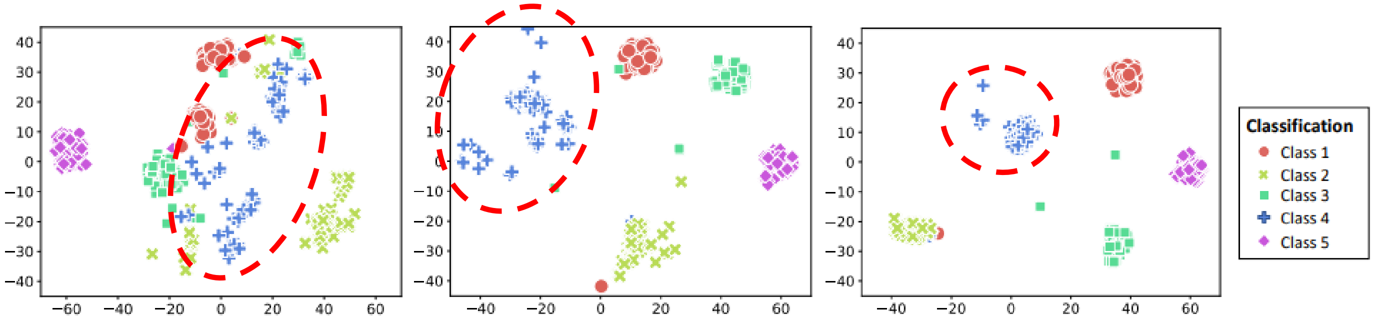
TAILOR **achieves** state-of-the-art performance in functional similarity detection

Comparison of Different GNNs

- Compare different off-the-shelf GNN variants on OJClone

Task	Metric	CPGNN-NP	LightGCN	GCN	GGNN	KGAT	CPGNN
CCD	**F-score**	0.0	13.1	98.7	98.2	96.0	**99.9**
SCC	**Accuracy**	72.0	79.6	97.6	96.7	95.2	**98.3**

- Visualize code representations produced by CPGNN-NP, GCN, and CPGNN



(a) CPGNN-NP (b) GCN (c) CPGNN

CPGNN shows **the strongest capability** to model code representations

Evaluating the Design of CPGNN

Investigate the effect of embedding **initialization**, CPG **representation**, and CPGNN **layer number** (A=AST, C=CFG, D=DFG)

			Embedding Initialization			Code Representation				CPGNN Layer		
Task	**Metric**	**Dataset**	**Type**	**Token**	**Comb**	**A**	**A+C**	**A+D**	**A+C+D**	**1**	**3**	**5**
CCD	**F-score**	**OJClone**	99.6	99.6	**99.9**	99.4	99.8	99.8	**99.9**	98.9	99.5	**99.9**
CCD	**F-score**	**BCB**	99.5	99.7	**99.8**	99.4	99.6	99.6	**99.8**	99.5	99.6	**99.8**
SCC	**Accuracy**	**OJClone**	97.7	97.8	**98.3**	97.9	98.1	98.0	**98.3**	96.4	97.6	**98.3**

TAILOR achieves the best performance (F-score in CCD, Accuracy in SCC):

- Node type and token both **contribute** to learn code representations

Evaluating the Design of CPGNN

Investigate the effect of embedding **initialization**, CPG **representation**, and CPGNN **layer number** (A=AST, C=CFG, D=DFG)

			Embedding Initialization			Code Representation				CPGNN Layer		
Task	**Metric**	**Dataset**	**Type**	**Token**	**Comb**	**A**	**A+C**	**A+D**	**A+C+D**	**1**	**3**	**5**
CCD	**F-score**	**OJClone**	99.6	99.6	**99.9**	99.4	99.8	99.8	**99.9**	98.9	99.5	**99.9**
CCD	**F-score**	**BCB**	99.5	99.7	**99.8**	99.4	99.6	99.6	**99.8**	99.5	99.6	**99.8**
SCC	**Accuracy**	**OJClone**	97.7	97.8	**98.3**	97.9	98.1	98.0	**98.3**	96.4	97.6	**98.3**

TAILOR achieves the best performance (F-score in CCD, Accuracy in SCC):

- Node type and token both **contribute** to learn code representations
- Code property graph (A+C+D) provides a **comprehensive** view of programs

Evaluating the Design of CPGNN

Investigate the effect of embedding **initialization**, CPG **representation**, and CPGNN **layer number** (A=AST, C=CFG, D=DFG)

			Embedding Initialization			Code Representation				CPGNN Layer		
Task	**Metric**	**Dataset**	**Type**	**Token**	**Comb**	**A**	**A+C**	**A+D**	**A+C+D**	**1**	**3**	**5**
CCD	**F-score**	**OJClone**	99.6	99.6	**99.9**	99.4	99.8	99.8	**99.9**	98.9	99.5	**99.9**
CCD	**F-score**	**BCB**	99.5	99.7	**99.8**	99.4	99.6	99.6	**99.8**	99.5	99.6	**99.8**
SCC	**Accuracy**	**OJClone**	97.7	97.8	**98.3**	97.9	98.1	98.0	**98.3**	96.4	97.6	**98.3**

TAILOR achieves the best performance (F-score in CCD, Accuracy in SCC):

- Node type and token both **contribute** to learn code representations
- Code property graph (A+C+D) provides a **comprehensive** view of programs
- Including multi-hop neighbors is **beneficial** to understand programs

Conclusion

- We propose TAILOR:
  - Learn high-quality graph-based code representations
  - Detect code functional similarity (code clone detection & source code classification)
- Insights:
  - CPG carries essential information to present program syntax and semantics
  - Customize a GNN to learn graph-base code representations by propagating node information along CPG structures

Artifact: https://github.com/jun-zeng/Tailor