How Machine Learning Is Solving the Binary Function Similarity Problem

USENIX ‘22

Binary Similarity

- Taking a pair of binaries and producing an output that represents their similarity
- Similar : Originate from the same source code
- Several practical applications (reverse engineering, vulnerability detection, malware clustering)
    - Researchers have published a lot of papers tackling this problem

Problem!

- Field is extremely fragmented
- Unable or difficult to reproduce
  - Incomplete artifacts
- Evaluation results are opaque
  - Different dataset, metric, implementation

Overview

- Binary Similarity Problem
  - Wide Range of Applications
  - Tackled by several research communities
  - Solved?

- Measurement Study on BCSD SOTA models
  - Dataset!

How is similarity measured?

- Direct vs Indirect
- Indirect
  - Compare an abstracted representation of binary instead
- Fuzzy Hash, Code Embedding, Graph Embedding

How is a Function represented?

CODE /DATA

- Raw bytes
- Assembly
- Normalized assembly
- Intermediate Representations

higher abstraction

STRUCTURE

- Control Flow Graph
- Annotated Control Flow Graph

OTHERS

- Data flow analysis
- Dynamic analysis
- Symbolic execution and analysis

Selected Models

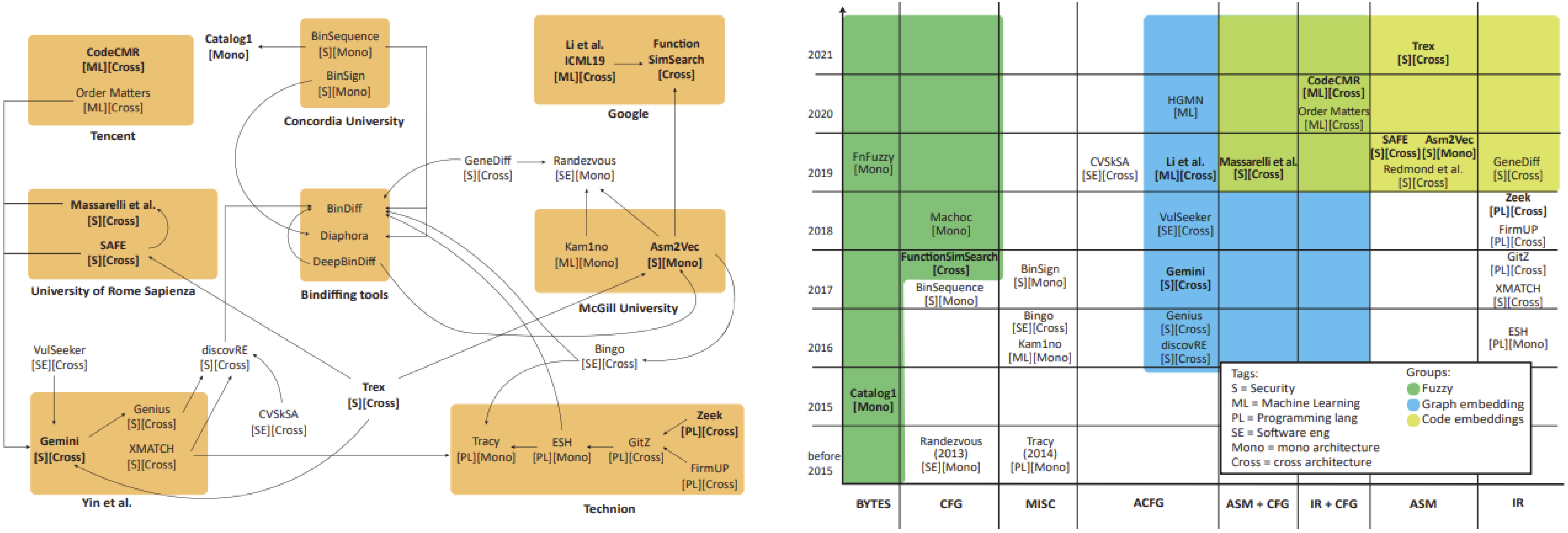

Figure 1: Function Similarity Systematization

Candidate Models (Fuzzy Hash)

- Catalog1
- FunctionSimSearch

Dataset

- Dataset -1
  - ClamAV, Curl, Nmap, OpenSSL, Unrar, Z3, and Zlib
  - Gcc, Clang (major release between 2015~2021)
  - O0, O1, O2, O3, and Os
  - 32/64
  - x86-64, MIPS, and ARM
- Dataset – 2 (Trex)
  - Binutils, Coreutils, Diffutils, Findutils, GMP, ImageMagick, Libmicrohttpd, LibTomCrypt, PuTTy, and SQLite
  - Gcc – 7.5
  - O0, O1, O2, O3
  - x86, x64, ARM 32, and MIPS 32

Experiment Method

X: diff O: same	Opt	Compiler	Bitness	Arch	Comp-version
XO	X	O	O	O	O
XC	X	X	O	O	X
XC+XB	X	X	X	O	X
XA	O	O	X	X	O
XA+XO	X	O	X	X	O
XM	X	X	X	X	X

Experiment Method

- Area Under Curve (AUC)
- Mean Reciprocal Rank (MRR)
- Recall@K

Simple Methods (Fuzzy Hash)

Table 2: Comparison of Catalog1 and FunctionSimSearch (FSS) on Dataset-1.

						XM					XC+XB	
	Description	XC	XC+XB	XA	XM	small	medium	large	MRR10	Recall@1	MRR10	Recall@1
Catalog1	B + size 16	0.66	0.60	0.48	0.54	0.54	0.53	0.54	0.08	0.07	0.25	0.23
Catalog1	B + size 128	**0.73**	0.66	0.43	0.55	0.54	0.55	0.58	0.12	0.09	**0.31**	**0.27**
FSS	G	0.72	**0.72**	**0.69**	**0.70**	**0.70**	**0.71**	**0.77**	**0.26**	**0.20**	0.29	0.23
FSS	G + M	**0.73**	0.71	0.58	0.65	0.64	0.66	0.70	0.17	0.13	0.29	0.24
FSS	G + M + I	**0.73**	0.70	0.58	0.65	0.64	0.66	0.71	0.15	0.09	0.28	0.23
FSS	*w* (G + M + I)	0.69	0.69	0.65	0.67	0.66	0.68	0.72	0.21	0.16	0.23	0.17

More Complex Methods

Table 3: Comparison of machine-learning models on Dataset-1.

	Description	XC	XC+XB	XA	XM	XM small	XM medium	XM large	XM MRR10	XM Recall@1
[67] Zeek (direct comparison)	Strands	0.84	0.85	0.84	0.84	0.85	0.83	**0.87**	0.28	0.13
[40] GMN (direct comparison)	CFG + BoW opc 200	0.85	0.86	**0.86**	0.86	**0.89**	0.82	0.79	**0.53**	**0.45**
[40] GMN (direct comparison)	CFG + No features	**0.86**	**0.87**	**0.86**	**0.87**	0.88	**0.85**	0.84	0.43	0.33
[40] GNN	CFG + BoW opc 200	**0.86**	**0.87**	**0.86**	**0.87**	**0.89**	0.84	0.76	0.52	0.44
[40] GNN	CFG + No features	0.82	0.83	0.82	0.82	0.85	0.80	0.76	0.37	0.29
[76] GNN (s2v)	CFG + BoW opc 200	0.81	0.82	0.78	0.81	0.82	0.78	0.74	0.36	0.26
[76] GNN (s2v)	CFG + manual	0.81	0.82	0.80	0.81	0.84	0.77	0.79	0.36	0.28
[76] GNN (s2v)	CFG + No features	0.69	0.70	0.69	0.70	0.70	0.69	0.75	0.12	0.07
[45] w2v + AVG + GNN (s2v)	CFG + N. asm 150	0.79	0.79	0.74	0.77	0.78	0.75	0.73	0.24	0.16
[45] w2v + wAVG + GNN (s2v)	CFG + N. asm 150	0.79	0.79	0.76	0.77	0.78	0.76	0.76	0.29	0.20
[45] w2v + RNN + GNN (s2v)	CFG + N. asm 150	0.79	0.80	0.79	0.80	0.82	0.77	0.80	0.27	0.17
[49] w2v + SAFE	N. asm 150	0.80	0.81	0.80	0.81	0.83	0.77	0.77	0.17	0.07
[49] w2v + SAFE	N. asm 250	0.82	0.83	0.82	0.83	0.84	0.81	0.82	0.22	0.09
[49] w2v + SAFE + trainable	N. asm 150	0.80	0.81	0.80	0.81	0.83	0.76	0.74	0.29	0.16
[49] rand + SAFE + trainable	N. asm 150	0.79	0.80	0.79	0.80	0.83	0.75	0.74	0.28	0.17
[14] Asm2Vec	10 CFG random walks	0.77	0.69	0.60	0.65	0.63	0.70	0.78	0.12	0.07
[38] PV-DM	10 CFG random walks	0.77	0.70	0.50	0.62	0.63	0.62	0.61	0.11	0.08
[38] PV-DBOW	10 CFG random walks	0.78	0.70	0.50	0.63	0.63	0.62	0.61	0.11	0.09

Table 4: Comparison of fuzzy hashing and machine-learning models on Dataset-2

Model name	Description	AUC			MRR10			Recall@1			Testing time (s)		
		XO	XA	XA+XO	XO	XA	XA+XO	XO	XA	XA+XO	Feat	Inf	Tot 100
[67] Zeek (direct comparison)	Strands	0.92	0.94	0.91	0.42	0.45	0.36	0.28	0.31	0.21	7225.41	67.00	9.92
[40] GMN (direct comparison)	CFG + BoW opc 200	**0.97**	**0.98**	**0.96**	**0.75**	**0.84**	**0.71**	**0.66**	**0.77**	**0.61**	1093.68	1005.00	1.83
[40] GMN (direct comparison)	CFG + No features	0.93	0.97	0.95	0.61	0.76	0.67	0.51	0.68	0.59	978.15	876.00	1.63
[40] GNN	CFG + BoW opc 200	0.95	0.97	0.95	0.67	0.79	0.67	0.57	0.73	0.57	1093.68	116.52	1.66
[40] GNN	CFG + No features	0.91	0.96	0.93	0.54	0.71	0.59	0.44	0.62	0.49	978.15	100.34	1.48
[76] GNN (s2v)	CFG + BoW opc 200	0.94	0.95	0.93	0.58	0.57	0.58	0.48	0.42	0.47	1093.68	118.59	1.66
[76] GNN (s2v)	CFG + Gemini	0.93	0.96	0.93	0.57	0.74	0.57	0.47	0.64	0.49	5139.91	98.40	7.18
[76] GNN (s2v)	CFG + No features	0.75	0.79	0.77	0.18	0.20	0.23	0.12	0.13	0.16	978.15	40.87	1.40
[45] w2v + AVG + GNN (s2v)	CFG + N. asm 150	0.90	0.88	0.87	0.46	0.31	0.42	0.38	0.18	0.33	1070.01	258.95	1.82
[45] w2v + wAVG + GNN (s2v)	CFG + N. asm 150	0.87	0.87	0.85	0.37	0.29	0.36	0.29	0.17	0.27	1070.01	253.72	1.81
[45] w2v + RNN + GNN (s2v)	CFG + N. asm 150	0.88	0.90	0.88	0.32	0.35	0.35	0.19	0.18	0.23	1070.01	685.50	2.41
[49] w2v + SAFE	N. asm 150	0.88	0.90	0.88	0.27	0.30	0.31	0.14	0.18	0.20	1031.23	33.33	1.46
[49] w2v + SAFE	N. asm 250	0.86	0.88	0.87	0.28	0.32	0.28	0.16	0.19	0.19	1031.23	33.33	1.46
[49] w2v + SAFE + trainable	N. asm 150	0.91	0.93	0.91	0.40	0.43	0.37	0.26	0.25	0.23	1031.23	33.57	1.46
[49] rand + SAFE + trainable	N. asm 150	0.90	0.91	0.90	0.28	0.33	0.31	0.14	0.17	0.21	1031.23	33.81	1.46
[14] Asm2Vec	Rand walks asm	0.94	0.69	0.75	0.60	0.07	0.22	0.49	0.02	0.18	978.15	5235.00	8.51
[38] PV-DM	Rand walks asm	0.94	0.66	0.72	0.64	0.08	0.23	0.51	0.05	0.19	978.15	5239.00	8.52
[38] PV-DBOW	Rand walks asm	0.94	0.66	0.72	0.63	0.07	0.23	0.50	0.03	0.20	978.15	3004.00	5.46
[60] Trex	512 Tokens	0.94	0.94	0.94	0.61	0.50	0.53	0.50	0.38	0.46	1493.58	1365.89	3.92
[74] Catalog_1	size 16	0.72	0.50	0.55	0.43	0.06	0.14	0.38	0.06	0.14	654.70	0.00	0.90
[74] Catalog_1	size 128	0.86	0.48	0.57	0.50	0.07	0.17	0.42	0.06	0.14	823.47	0.00	1.13
[18] FSS	G	0.77	0.81	0.77	0.26	0.35	0.32	0.18	0.26	0.26	1903.46	466.07	3.25
[18] FSS	G + M	0.79	0.68	0.69	0.29	0.15	0.21	0.23	0.09	0.15	1903.46	466.07	3.25
[18] FSS	G + M + I	0.80	0.68	0.69	0.30	0.16	0.20	0.23	0.10	0.14	1903.46	466.07	3.25
[18] FSS	w(G + M + I)	0.83	0.80	0.78	0.43	0.30	0.36	0.36	0.23	0.29	1903.46	466.07	3.25

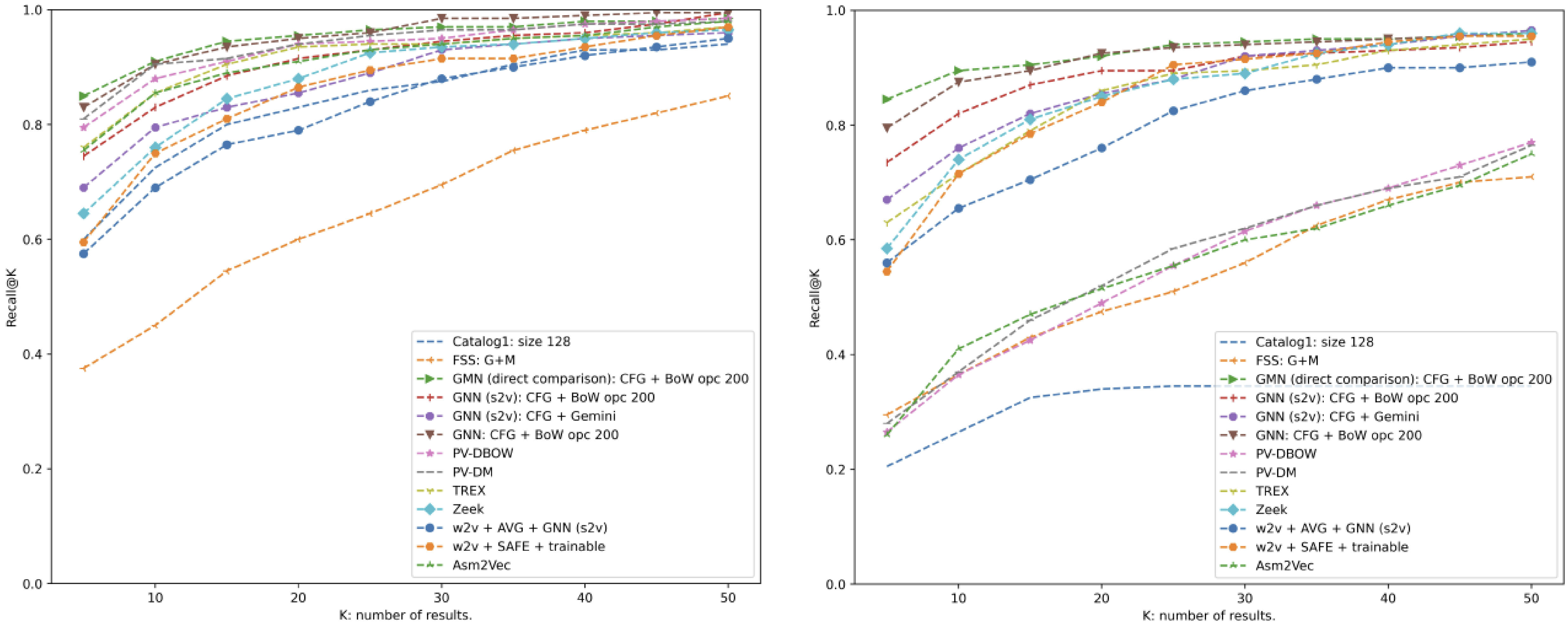

Figure 2: Comparison of the recall at different K values for XO (left) and XM (right) tasks.

CodeCMR (Tencent)

- Didn't Reimplement
- Authors of CodeCMR assisted in evaluation

Table 5: Comparison of CodeCMR/BinaryAI with GNN and bag of words (BoW) of opcodes (opc) or IDA microcode (IR).

						XM				
	Description	XC	XC+XB	XA	XM	small	medium	large	MRR10	Recall@1
[40] GNN	CFG + BoW opc 200	0.86	0.87	0.87	0.87	0.90	0.84	0.78	0.58	0.52
[40] GNN	CFG + BoW IR 80	0.87	0.87	0.87	0.88	0.89	0.86	0.81	0.62	0.56
[79] CodeCMR/BinaryAI	CFG + IR + Int + Strings	**0.98**	**0.98**	**0.98**	**0.98**	**0.99**	**0.97**	**0.93**	**0.86**	**0.83**

Takeaways – Contribution of novel ML solutions

- Deep-learning models provide an effective way of learning a function representation

- Siamese architecture in combination with a margin based loss introduced significant improvements

- GNNs are effective encoders and can be used in combination with others

Takeaways – Role of different set of features

- Using basic block features (e.g. ACFG) provides better results
- Minimal difference between manually engineered features and simpler ones (e.g. BoW opcodes)
- Dataflow information can boost the results especially for large functions

Takeaways – Cross Architecture vs Single Architecture

- Most ML models perform similarly on all the evaluated tasks

- Training on generic task data achieves performance close to the best of each task

- Asm2Vec and Catalog1 are limited to comparisons to a single architecture

Takeaways – Future Directions

- GNN models provide best results, but there are tens of different variants that need to be tested

- Combining GNNs with intermediate representations and dataflow information must be studied

- Training strategy and loss functions have been barely discussed in the past and only recently explored