Practical Binary Code Similarity Detection with BERT-based Transferable Similarity Learning

Sunwoo Ahn

Department of Electrical and Computer Engineering

Seoul National University

Seoul, Korea

Seonggwan Ahn

Department of Electrical and Computer Engineering

Seoul National University

Seoul, Korea

Hyungjoon Koo

Department of Computer Science and Engineering

Sungkyunkwan University

Suwon, Korea

Yunheung Paek

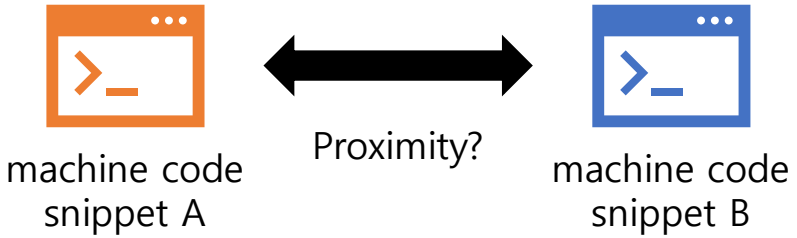
Department of Electrical and Computer Engineering

Seoul National University

Seoul, Korea

Binary Code Similarity Detection (BCSD)

- Problem Definition



- Papers → since 1999 [HAQ et al., 2021]

- Usage
  - Code clone detection
  - Malware detection
  - Malware family classification
  - Known vulnerability discovery
  - Code patching verification

Challenges

- Elimination/transformation of semantic information
  - e.g., variable name, structure, type, class hierarchy
- Binary variants
  - compiler configuration, architecture, obfuscation, etc.
- Halting problem
  - It is undecidable to prove
    that two arbitrary programs are functionally equivalent

Scope

- Comparison type
  - One-to-one
  - One-to-many
  - Many-to-many
- Comparison target(binary)
  - Compiler
  - Compiler options/versions/opt lv
  - Architecture
  - Obfuscation
- Comparison granularity
  - Basic block
  - Function
  - Binary
- Comparison techniques
  - static/dynamic/symbolic approaches
  - Hashing, indexing, embedding
  - Cosine/L1/L2 distance

Scope

- Comparison type
  - One-to-one
  - One-to-many
  - Many-to-many
- Comparison target(binary)
  - Compiler
  - Compiler options/versions/opt lv
  - Architecture – x86_64
  - Obfuscation
- Comparison granularity
  - Basic block
  - Function
  - Binary
- Comparison techniques
  - static/dynamic/symbolic approaches
  - Hashing, indexing, embedding
  - Cosine/L1/L2 distance
    - Weighted distance

Existing Works

- Recent works employ Siamese network

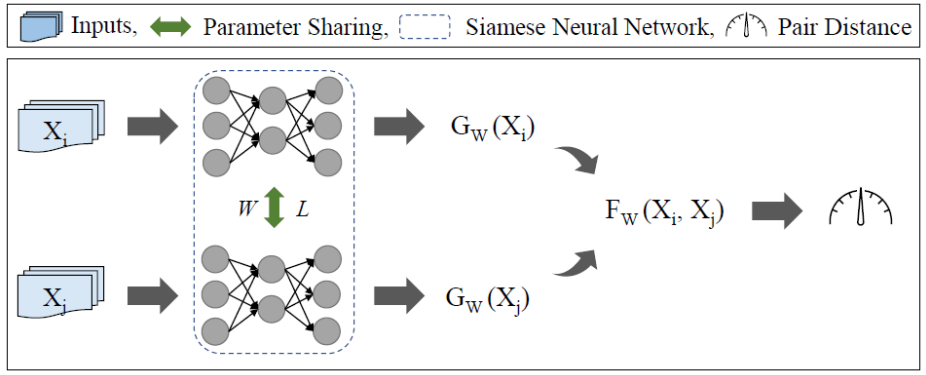
Model	Distance function	Loss function	Architecture
Gemini	Cosine distance	Contrastive loss	GNN, Siamese NN
InnerEye	Cosine distance	Contrastive loss	word2vec, LSTM
Asm2Vec	Cosine distance	Log probability	PV-DM
PalmTree	Cosine distance	Contrastive loss	BERT, GNN, Siamese NN
DeepSemantic	None	Cross entropy	BERT, Softmax classifier

Existing Works

- Recent works employ Siamese network

Model	Distance function	Loss function	Architecture
Gemini	Cosine distance	Contrastive loss	GNN, Siamese NN
InnerEye	Cosine distance	Contrastive loss	word2vec, LSTM
Asm2Vec	Cosine distance	Log probability	PV-DM
PalmTree	Cosine distance	Contrastive loss	BERT, GNN, Siamese NN
DeepSemantic	None	Cross entropy	BERT, Softmax classifier

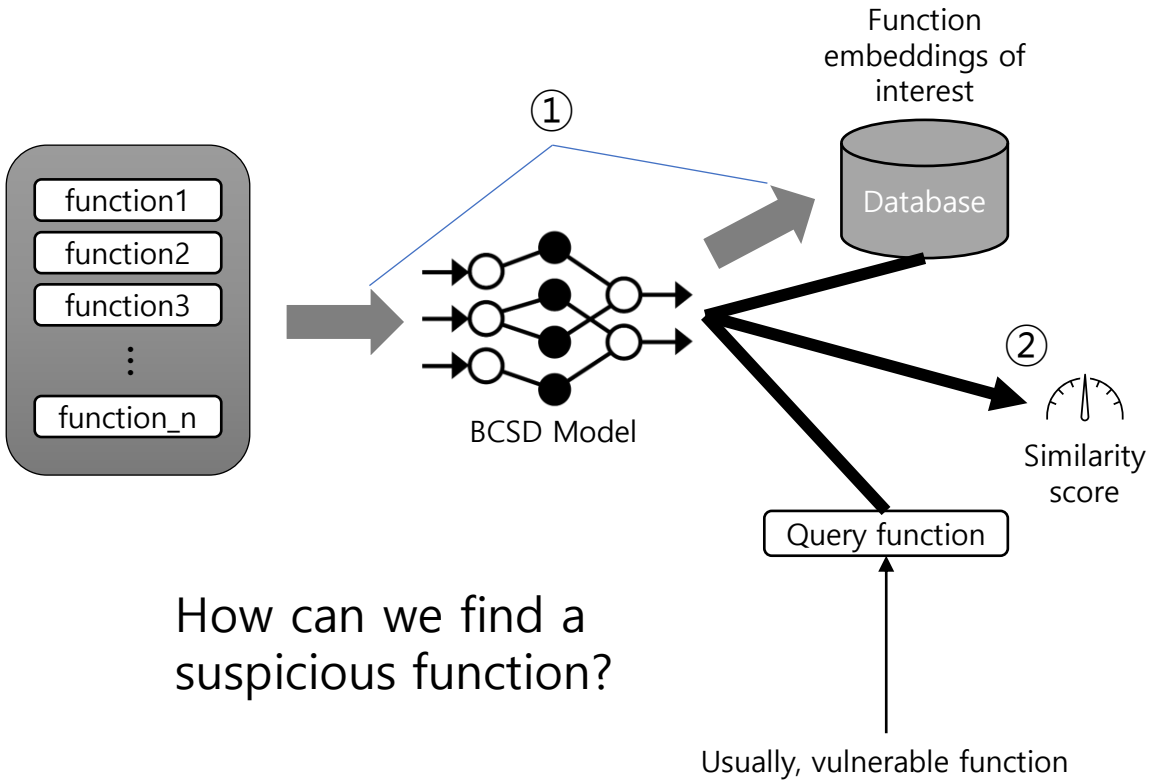
- Scalar value → oversimplification



$$L_c(W, Y, X_i, X_j) = Y\frac{1}{2}(F_W)^2 + (1-Y)\frac{1}{2}(max(0, 1-F_W))^2$$

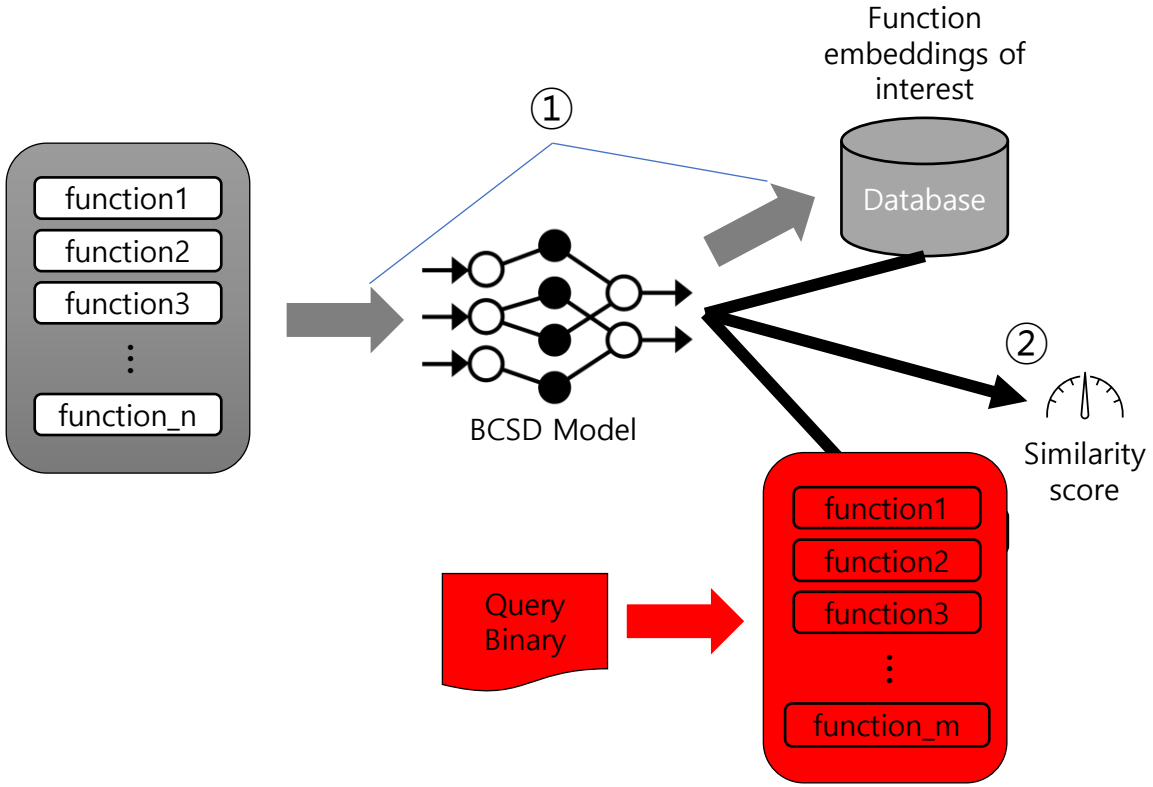
Problem

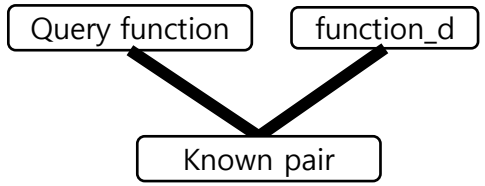
- Prediction scenario in existing works

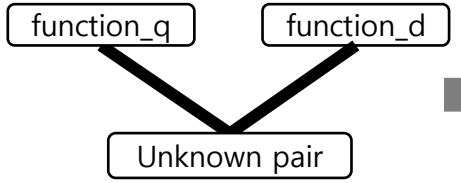


Problem

- Our newly proposed realistic scenario







Existing models → many false positives

Model need to be more robust against unknown functions!

Problem

- Recent works employ Siamese network

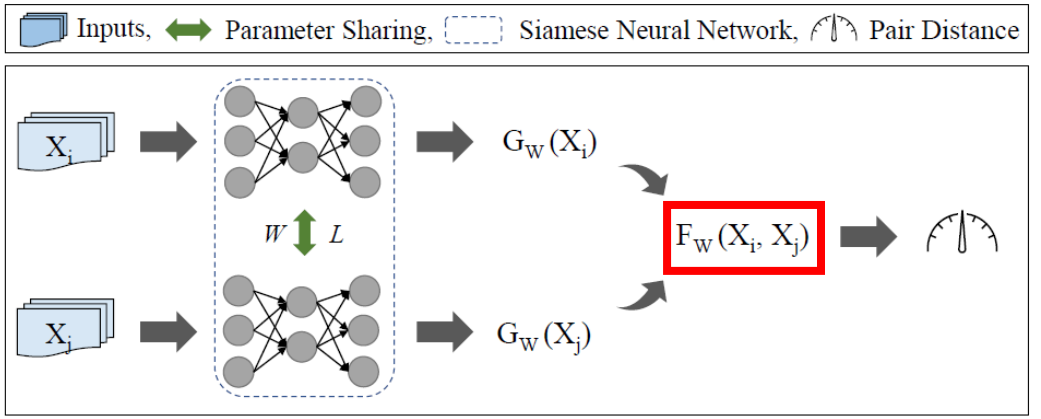
Model	Distance function	Loss function	Architecture
Gemini	Cosine distance	Contrastive loss	GNN, Siamese NN
InnerEye	Cosine distance	Contrastive loss	word2vec, LSTM
Asm2Vec	Cosine distance	Log probability	PV-DM
PalmTree	Cosine distance	Contrastive loss	BERT, GNN, Siamese NN
DeepSemantic	None	Cross entropy	BERT, Softmax classifier

- Two elements affects performance [Marcelli et al., 2022]
  → We explore some options

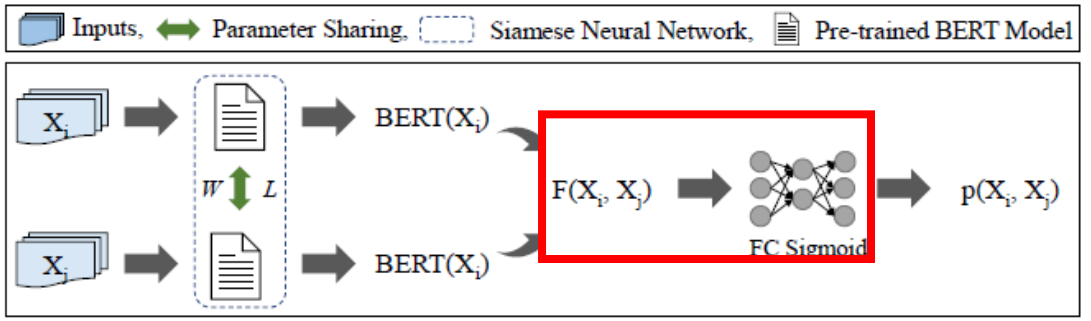
Solutions

- Pretraining BERT
  - learns relationship btw instructions
  - understands assembly language

- Siamese network learning
  weighted distance with binary cross entropy
  - Koch et al. proposed it
  - More robust against unseen data



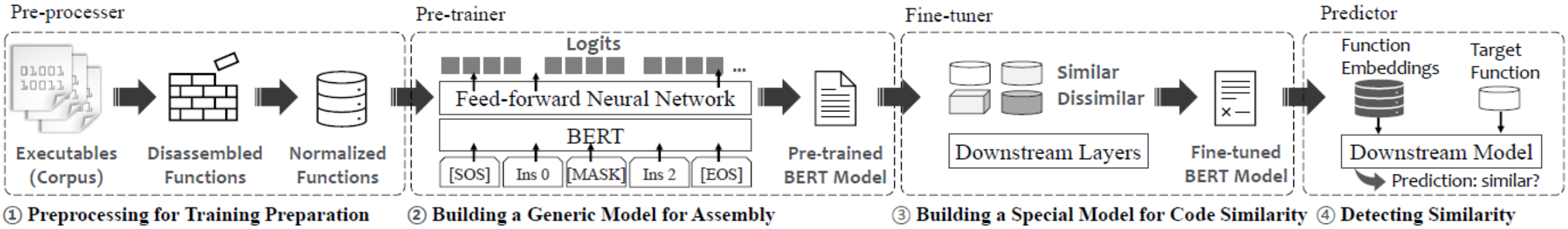
$$L_c(W, Y, X_i, X_j) = Y\frac{1}{2}(F_W)^2 + (1-Y)\frac{1}{2}(max(0, 1-F_W))^2$$



$$L = Y \log p(X_i, X_j) + (1-Y)\log(1-p(X_i, X_j))$$

$$p(X_i, X_j) = \sigma(FC(F(X_i, X_j)))$$

BinShot



- Preprocessor
  - Disassemble binaries
  - Instruction normalization
    - Prevents OOV problem
- Pretrainer
  - MLM task → same with original BERT
  - NSP task → exclude from original BERT
    - Function invocation rather than locations
- Finetuner
  - Downstream task = BCSD
  - Learn weighted distance with Siamese network
- Predictor
  - For efficient inference in our newly proposed realistic scenario

You can find more details in our paper!

Experimental Setup

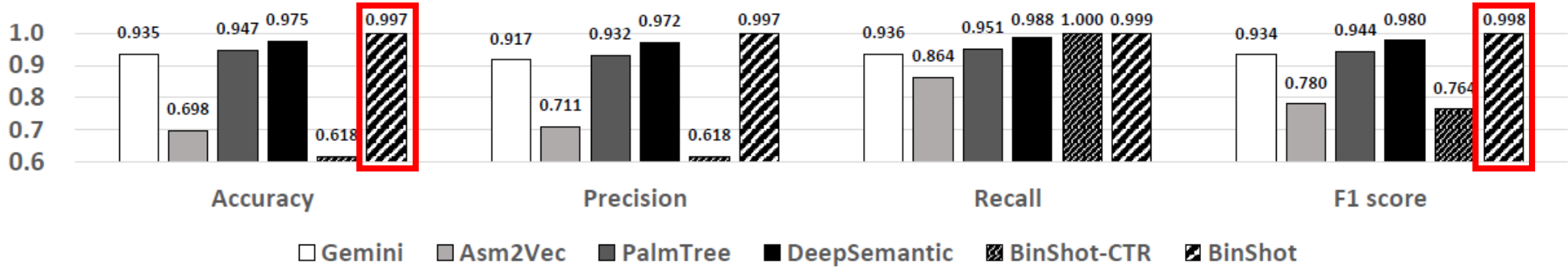
- Dataset
  - compiled with 2 compiler (gcc, clang) & 4 optimization (O0-O3)
  - Projects
    - GNU utilities – binutils, coreutils, diffutils, findutils
    - SPEC CPU – 2006, 2017
    - Real-world programs
      - BusyBox, Libgmp, ImageMagick, Libcurl, LibTomCrypt, OpenSSL, SQLite, zlib, PuTTYgen, Nginx, vsftpd

- Baseline models

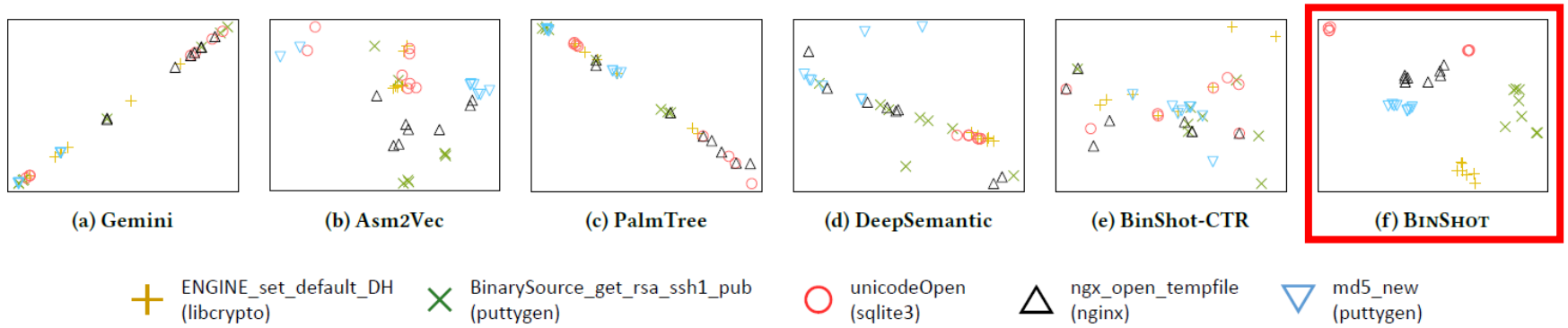
Model	Distance function	Loss function	Architecture
Gemini	Cosine distance	Contrastive loss	GNN, Siamese NN
Asm2Vec	Cosine distance	Log probability	PV-DM
PalmTree	Cosine distance	Contrastive loss	BERT, GNN, Siamese NN
DeepSemantic	None	Cross entropy	BERT, Softmax classifier
BinShot-CTR	L2 nrom	Contrastive loss	BERT, Siamese NN
BinShot	Weighted squared error	Binary cross entropy	BERT, Siamese NN

Evaluation - effectiveness

- Evaluate whole dataset
  - #positive : #negative = 1:1

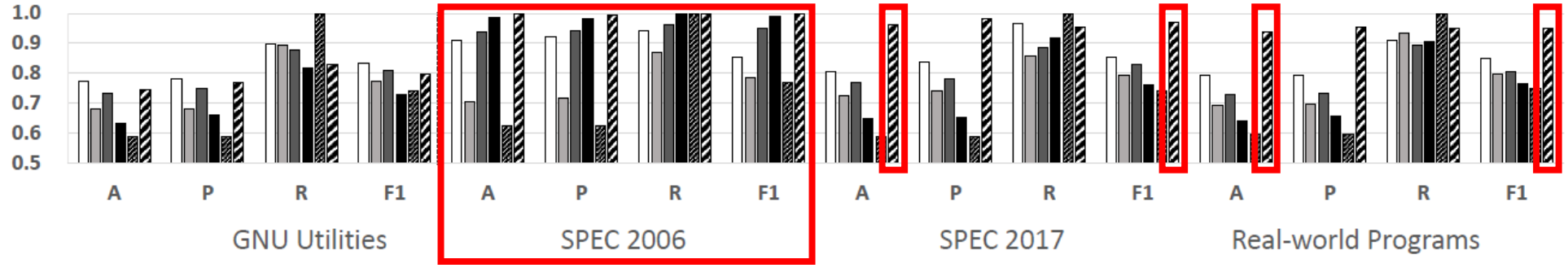


- t-SNE visualization

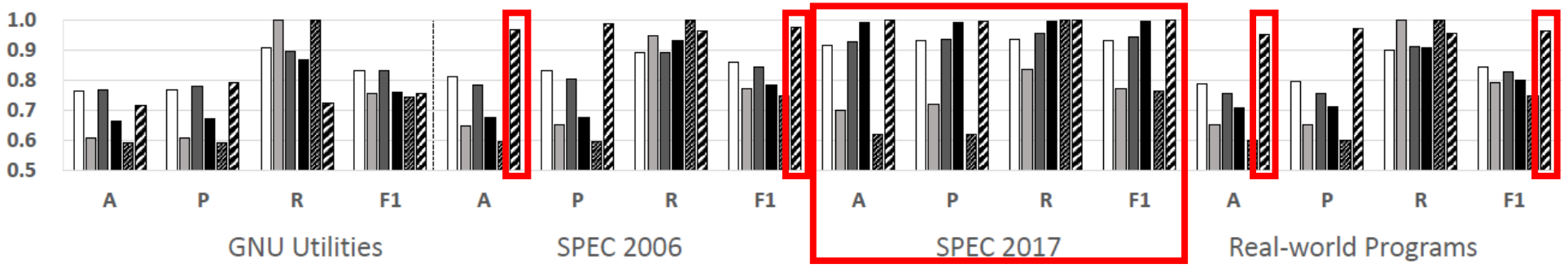


Evaluation - transferability

- Trained with SPEC 2006



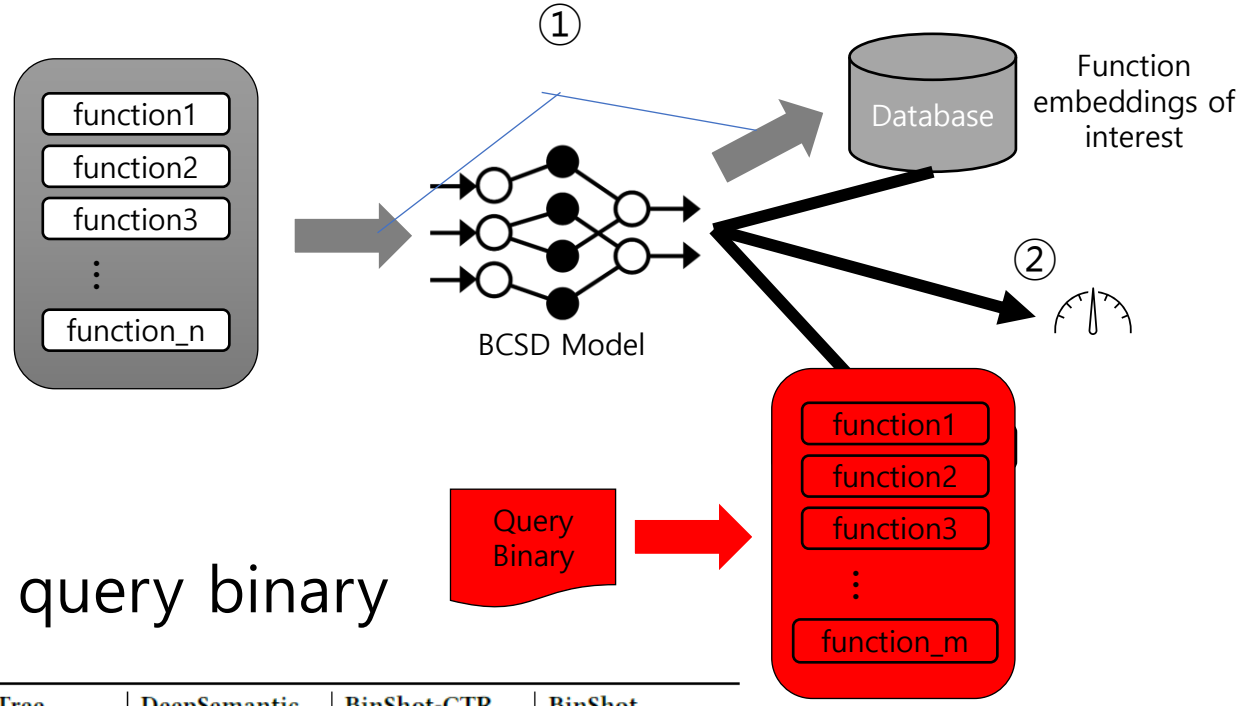
- Trained with SPEC 2017



- Note: GNU utilities → a few sharing functions w/ static library

Evaluation - vulnerable function detection

- Realistic scenario setup
  - Database contains vulnerable function embeddings
    - 4 binary variants (gcc O0-3)
  - Query binary is stripped
    - 4 binary varinats (clang O0-3)
  - Goal: find vulnerable function in query binary



Program	CVE	Vulnerable function	Gemini O0–O3	Gemini A/R	Asm2Vec O0–O3	Asm2Vec A/R	PalmTree O0–O3	PalmTree A/R	DeepSemantic O0–O3	DeepSemantic A/R	BinShot-CTR O0–O3	BinShot-CTR A/R	BinShot O0–O3	BinShot A/R
OpenSSL v1.0.1e*	2014-0160 [13]	tls1_process_heartbeat	✓✓✓✓	0.0033/ 1.0000	✓✓✓✓	0.1179/ 1.0000	✓✓✓✓	0.0140/ 1.0000	✓✗✓✗	0.3656/ 0.6000	✓✓✓✓	0.0033/ 1.0000	✓✓✓✓	0.9009/ 1.0000
		dtls1_process_heartbeat	✓✓✓✓		✓✓✓✓		✓✓✓✓		✓✗✓✗		✓✓✓✓		✓✓✓✓	
	2014-0221 [14]	dtls1_get_message_fragment	✓✓✓✓		✓✓✓✓		✓✓✓✓		✓✗✓✗		✓✓✓✓		✓✓✓✓	
	2014-3508 [15]	OBJ_obj2txt	✓✓✓✓		✓✓✓✓		✓✓✓✓		✓✓✓✗		✓✓✓✓		✓✓✓✓	
	2015-1791 [17]	ssl3_get_new_session_ticket	✓✓✓✓		✓✓✓✓		✓✓✓✓		✗✓✓✓		✓✓✓✓		✓✓✓✓	
NTP v4.2.7p10	2014-9295 [16]	crypto_recv	- - - -	0.0055/ 1.0000	- - - -	0.1588/ 1.0000	- - - -	0.0083/ 1.0000	- - - -	0.4505/ 1.0000	- - - -	0.0064/ 1.0000	- - - -	0.7940/ 1.0000
		ctl_putdata	✓✓✓ -		✓✓✓ -		✓✓✓ -		✓✓✓ -		✓✓✓ -		✓✓✓ -	
		configure	✓ -✓✓		✓ -✓✓		✓ -✓✓		✓ -✓✓		✓ -✓✓		✓ -✓✓	
libav v0.8.3	2012-2776 [12]	decode_cell_data	✓✓✓✓	0.0007/ 1.0000	✓✓✓✓	0.1215/ 1.0000	✓✓✓✓	0.0065/ 1.0000	✗✓✗✓	0.0003/ 0.5000	✓✓✓✓	0.0007/ 1.0000	✓✓✓✓	0.9497/ 1.0000

Evaluation – runtime efficiency

- Runtime efficiency
  - Exp1 - Each function pair
  - Exp2 - 82300 function pairs (100 in database, 823 in query binary) with our predictor

Model	Gemini	Asm2Vec	PalmTree	DeepSemantic	BinShot-CTR	BinShot
Exp1 (ms)	0.10	81.94	1.33	1.34	1.30	1.32
Exp2 (s)	1.16	6,734.66	29.03	1.51	1.45	1.54

Discussions & Limitations

- Mangled Names
- Code obfuscation and other code constructs
  - See scope slide (target binary)
- Function inlining
- Rarely appeared instructions

Summary

- Superiority of BinShot
  - effectiveness, practicality (transferability & runtime), visualization
- Learning weighted distance & pretraining
  improve robustness against unseen function pair
- The other models but ours shows poor performance
  in our newly proposed realistic scenario
- Open source project: https://github.com/asw0316/binshot

Thanks!