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

Sunwoo Ahn
Department of Electrical and Computer Engineering
Seoul National University
Seoul, Korea

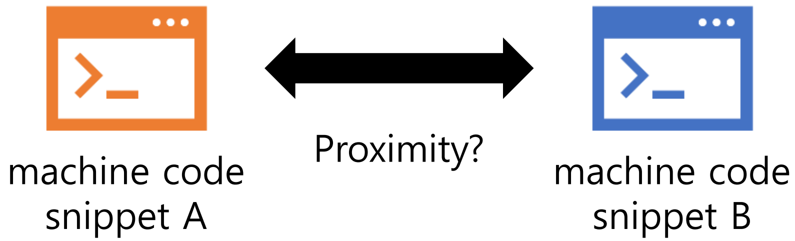
Seonggwan Ahn
Department of Electrical and Computer Engineering
Seoul National University
Seoul, Korea

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)

- BCSD Problem



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

Challenges

- Useful information is unavailable in a binary
  - e.g., variable name, structure, type, class hierarchy, etc.
- Binaries that have identical semantics can vary
  - compiler configuration, architecture, obfuscation, etc.
- Halting problem
  - Undecidable to prove
    the functional equivalency of two arbitrary programs

Existing Works

- Recent advances employ
  neural network with Siamese architecture

Model	Architecture
Gemini	GNN, Siamese NN
InnerEye	word2vec, LSTM, Siamese NN
Asm2Vec	PV-DM
PalmTree	BERT, GNN, Siamese NN
DeepSemantic	BERT, Softmax classifier

Existing Works

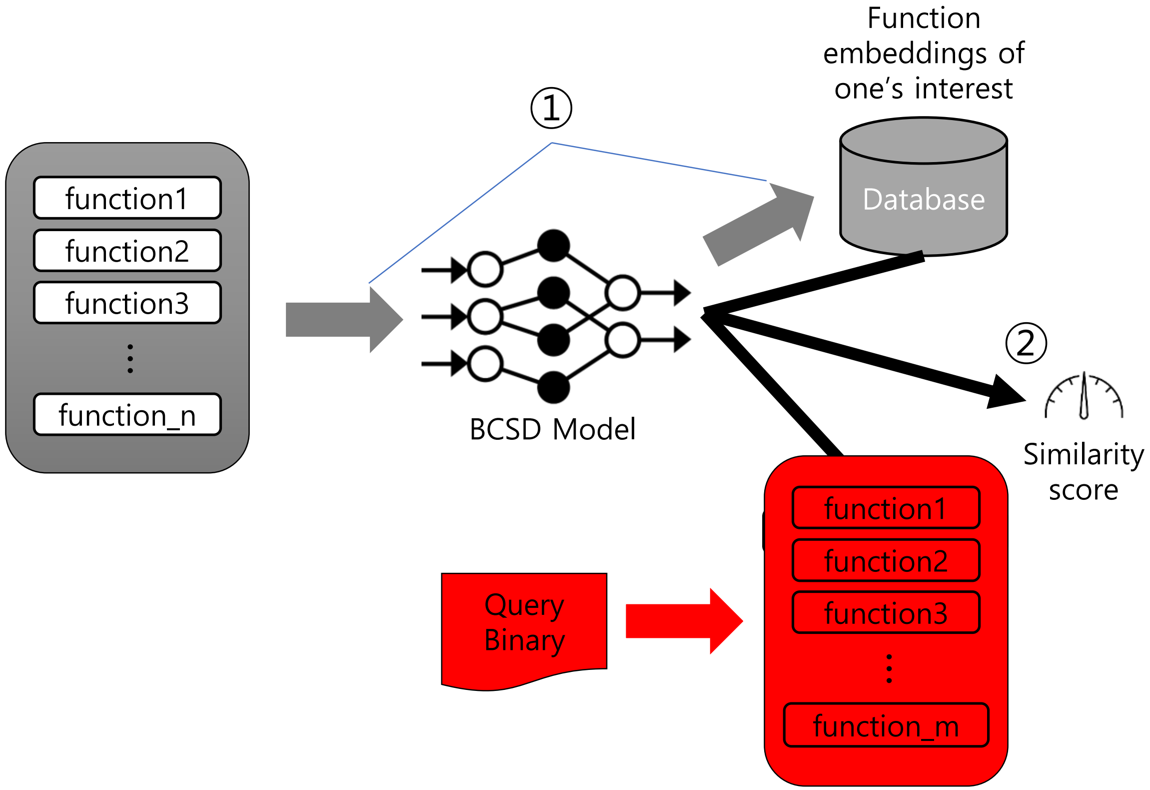
- Distance/loss function affects Siamese network (Marcelli et al., USENIX '22)

Model	Distance function	Loss function
Gemini	Cosine distance	Contrastive loss
InnerEye	Cosine distance	Contrastive loss
Asm2Vec	Cosine distance	Log probability
PalmTree	Cosine distance	Contrastive loss
DeepSemantic	None	Cross entropy

  - Scalar value → oversimplification

Problem

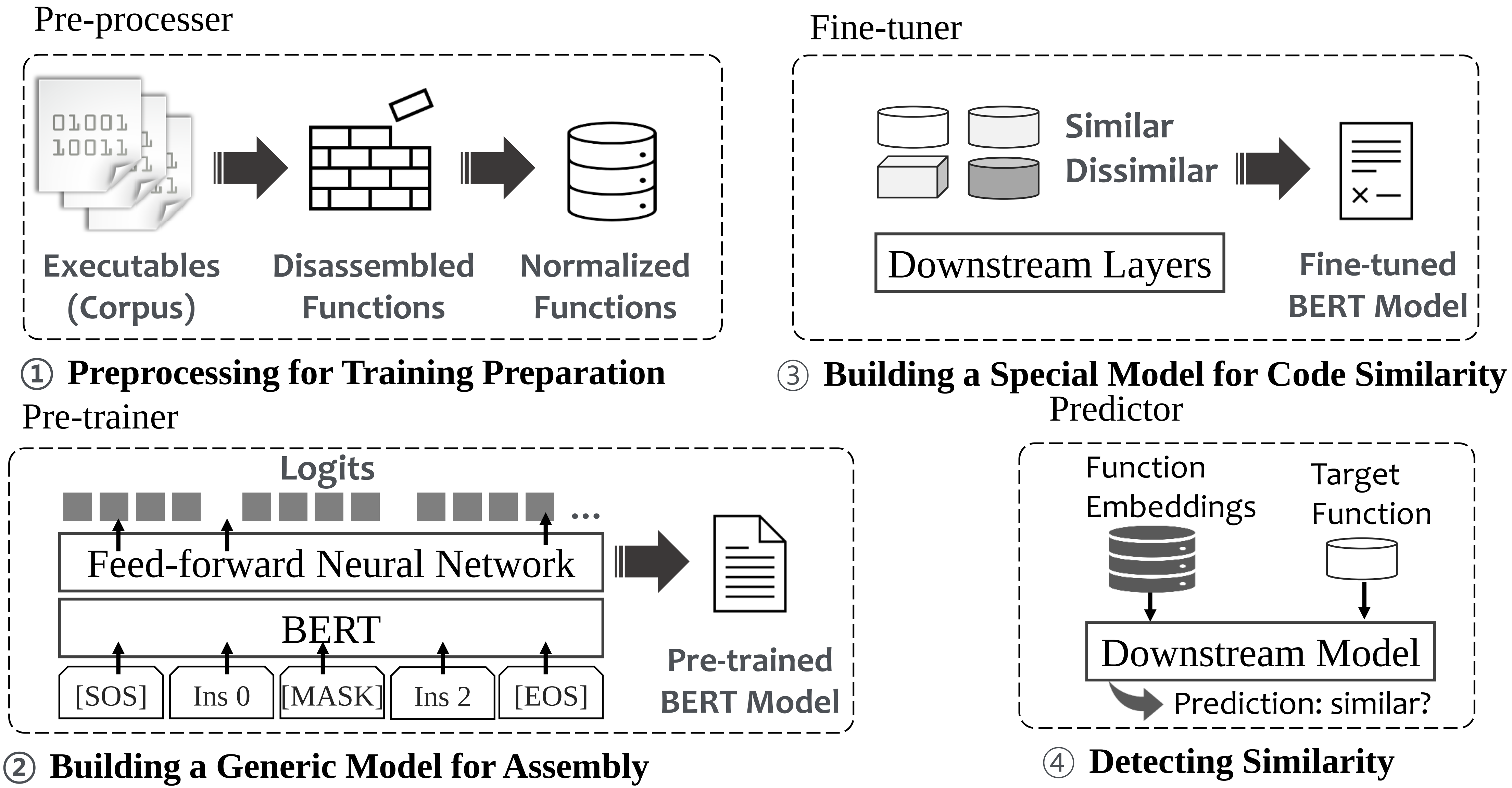
- We question existing work in a realistic scenario



Our Main Approach

- Goal: improve performance for unseen dataset
- Transferable similarity learning (BERT-based)
  - Learning a relationship btw instructions with pre-training
  - Repeatedly showing good performance on an assembly language
- Better similarity detection: learning a weighted distance vector with a binary cross entropy
  - Weighted distance → relationships are represented in a vector

BinShot



Pre-processer
01001
10011
Executables (Corpus)
Disassembled Functions
Normalized Functions
① Preprocessing for Training Preparation
Fine-tuner
Similar
Dissimilar
Downstream Layers
Fine-tuned BERT Model
③ Building a Special Model for Code Similarity
Pre-trainer
Logits
...
Feed-forward Neural Network
BERT
[SOS]
Ins 0
[MASK]
Ins 2
[EOS]
Pre-trained BERT Model
② Building a Generic Model for Assembly
Predictor
Function Embeddings
Target Function
Downstream Model
Prediction: similar?
④ Detecting Similarity

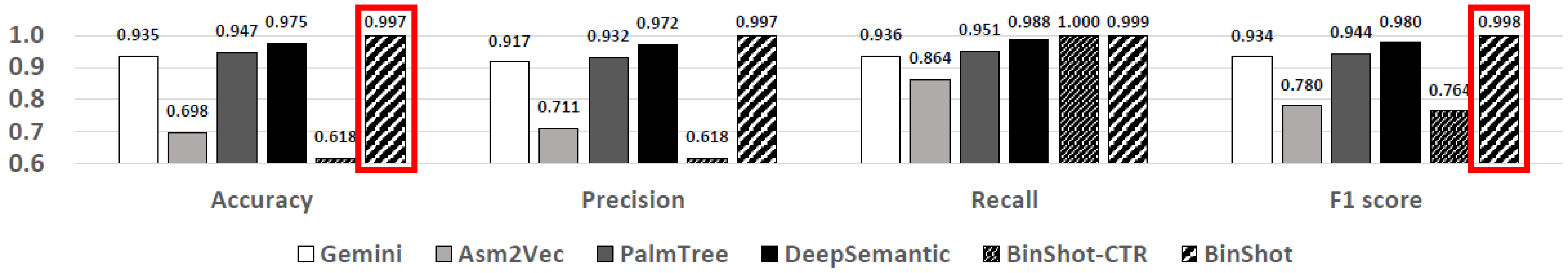
Experimental Setup

- Dataset
  - Compiled with 2 compiler (gcc, clang) & 4 optimization (O0-O3)
  - 1,400 binaries in total
    - GNU utilities – binutils, coreutils, diffutils, findutils
    - SPEC2006, SPEC2017
    - 11 Real-world programs (BusyBox, Libgmp, …)

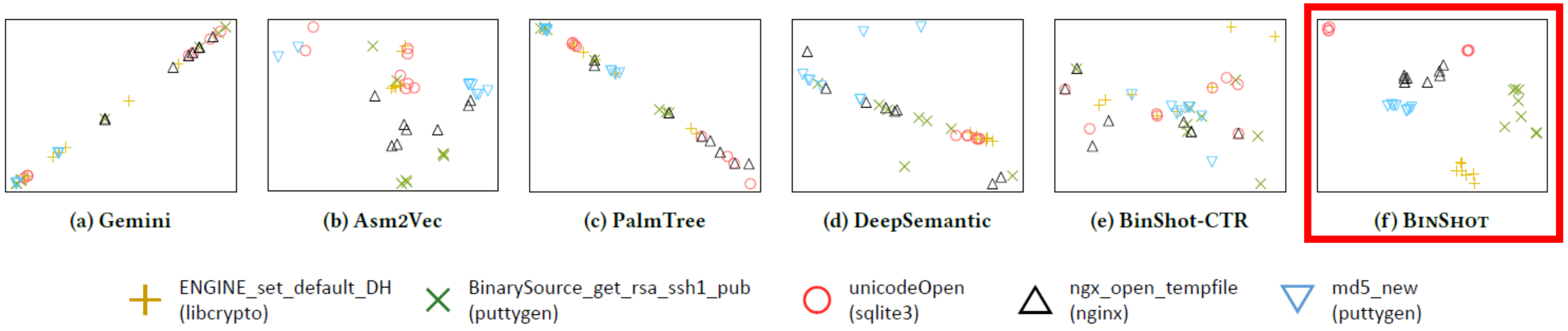
- Baseline models:
  - Gemini, Asm2vec, PalmTree, DeepSemantic
  - BinShot-CTR, BinShot

Evaluation - Effectiveness

- Evaluate whole dataset

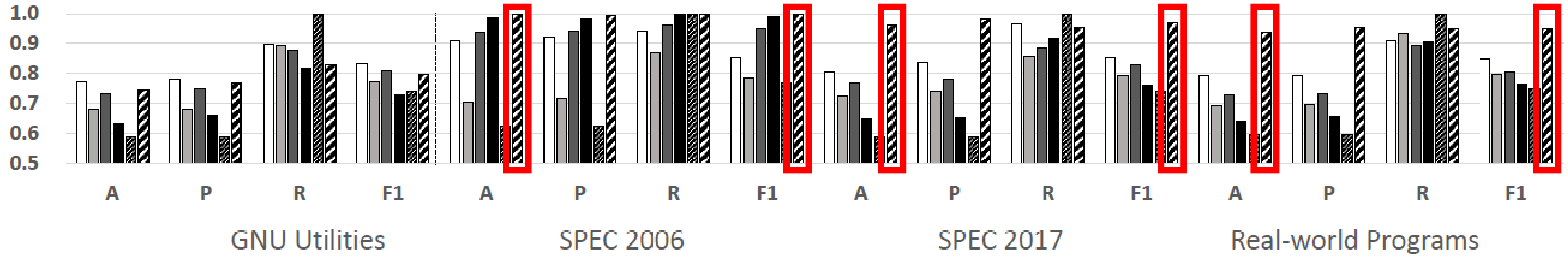


- t-SNE visualization



Evaluation - Transferability

- Trained with SPEC 2006



Evaluation – Vulnerable Function Detection

- Realistic scenario setup
  - Database contains vulnerable function embeddings
  - Query binary is stripped
  - Goal: find a vulnerable function from a 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
- Function inlining
- Code obfuscation and other code constructs
- Rarely appeared instructions

See our paper!

Wrap-up

- Learning a weighted distance with a binary cross entropy improves robustness against unseen function pairs

- Superiority of BinShot
  - effectiveness, practicality (transferability & runtime)

- The other models but ours shows poor performance in a realistic scenario

- Open source project: https://github.com/asw0316/binshot

Thanks!