BinAdapter: Leveraging Continual Learning for Inferring Function Symbol Names in a Binary

Nozima Murodova Hyungjoon Koo*

SecAI Lab | SUNG KYUN KWAN UNIVERSITY 1398

acm ACM ASIACCS 2024, Singapore

Binary Reverse Engineering (Reversing)

- Binary reversing is the key to
  - Understand the underlying semantics of binary code when source code is unavailable

- Some utilities
  - Vulnerability discovery; Malware detection; Copyright infringement; etc.

- Main challenges
  - No high-level information; Complex transformation; Time-consuming; etc.

AI-assisted Binary Reversing

- Can handle a large volume of data
- Can be faster than manual analysis
- Can detect overlooked patterns and anomalies
- Can provide supportive insights and recommendations

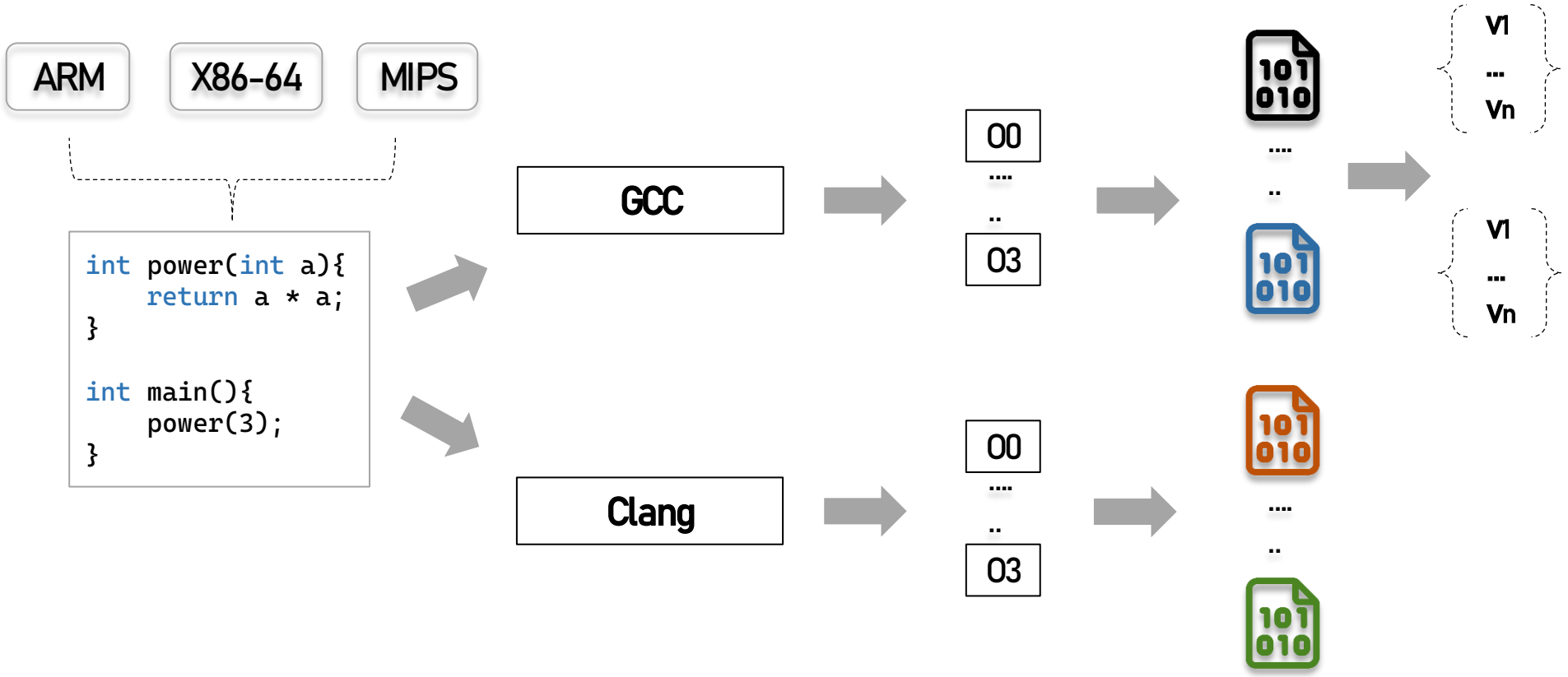
Classification

- Malware Detection
- Binary Code Similarity
- Function Boundary Detection

Generation

- Function Name Prediction (Our focus)
- Variable Name Prediction

Generating Numerous Binaries is Possible



Handling Incremental Data?

Traditional Training:
- Combining all datasets (old & new)
- Training a separate model per dataset

- Expensive ✖
- Time consuming ✖
- Old datasets may not be available ✖

Fine-tuning:
- Preparing a new dataset
- Fine-tuning the existing model

- Efficient training
- Better performance for new dataset
- Catastrophic forgetting for old dataset ✖

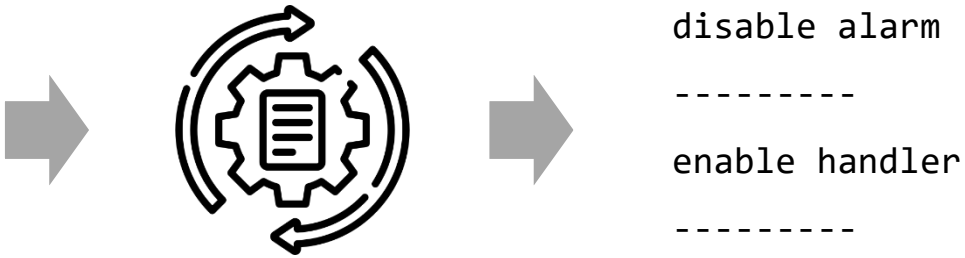
Continual (Lifelong) Learning (CL):
- Fine-tuning an existing model with a new dataset
- Preventing the performance degradation

- Mitigating catastrophic forgetting
- Learning new evolving features
- Efficient training

Methodology

- **AsmDepictor** – Transformer-based code-to-text translation model (ASIACCS'23)

```
mov    eax,DWORD PTR [rip+0x108f32e]
and    ah,0xdf
or     ah,0x40
mov    DWORD PTR [rip+0x108f322],eax
mov    eax,0x0
ret
---------
sub    rsp,0x8
call   QWORD PTR [rip+0x104c454]
mov    eax,0x0
add    rsp,0x8
ret
---------
```



```
disable alarm
---------
enable handler
---------
```

- **Adapter** – the first parameter efficient fine-tuning technique (ICML'19)

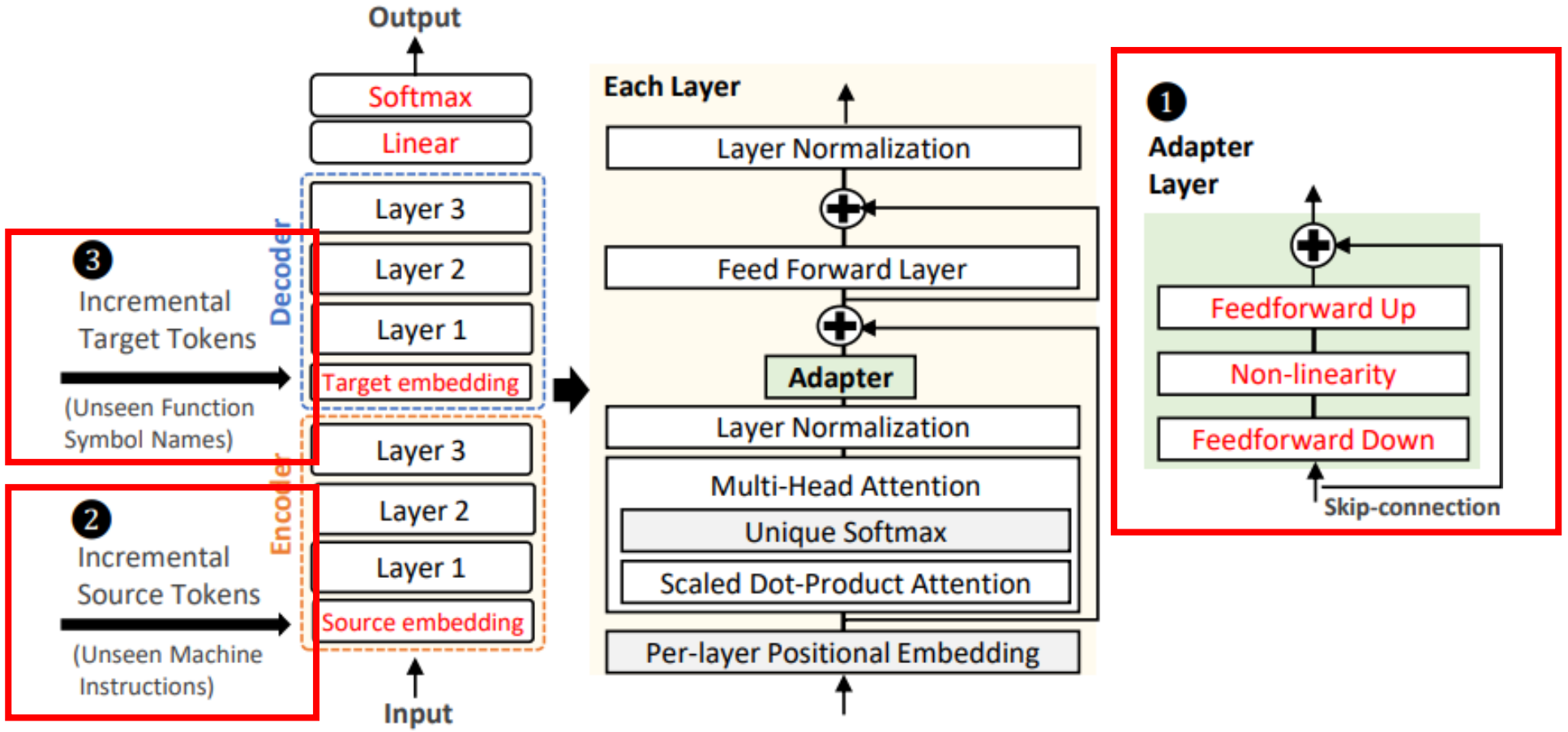
Possible Learning Scenarios for Reversing

1. **No new vocab** from (assembly) code and function symbol names

2. **New vocab** comes only from code

3. **New vocab** comes from both (code and function symbol names)

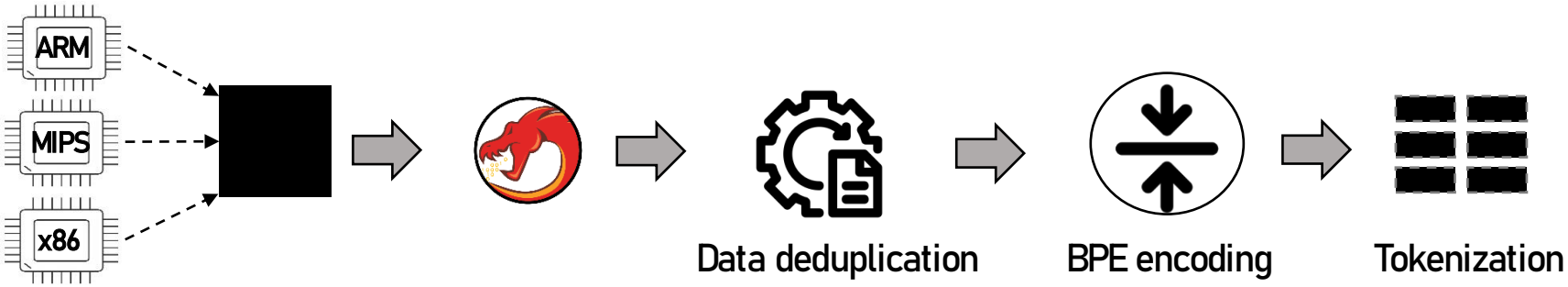
BinAdapter – CL Schema for Name Prediction



Experimental Setup

- Dataset
  - Scenario 1 – AsmDepictor Dataset (x86; **A0-A3**)
  - Scenario 2 – BinKit (ARM, MIPS, x86; **B1-B3**)
  - Scenario 3 – SPEC2006 (x86; **S1-S2**)

- Preprocessing



Baselines for Evaluation

Scenario 1:

- Fine-Tuning
- EWC (Elastic Weight Consolidation)
- SI (Synaptic Intelligence)

Scenario 2-3:

- Retraining
- Naïve M-NMT (CL for Multilingual Neural Machine Translation)

Research Questions

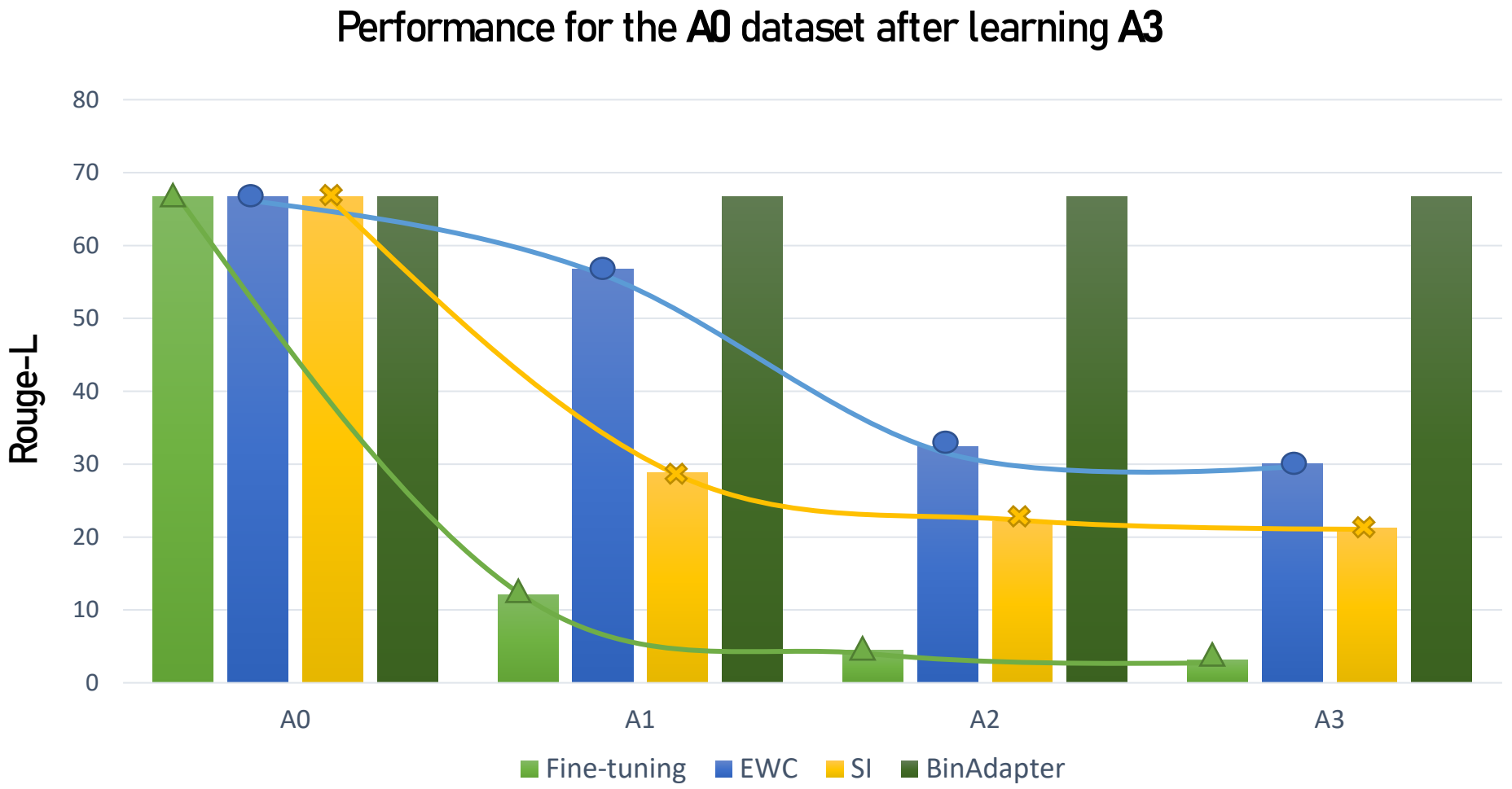
1. How effective is BinAdapter in different scenarios compared to other CL baseline techniques for a function name prediction task?

2. How efficient is BinAdapter in terms of training parameters, storage overheads, and inference time?

3. How appropriate is the current design of BinAdapter with ablation studies (e.g., number of adapters, fine-tuning layers)?

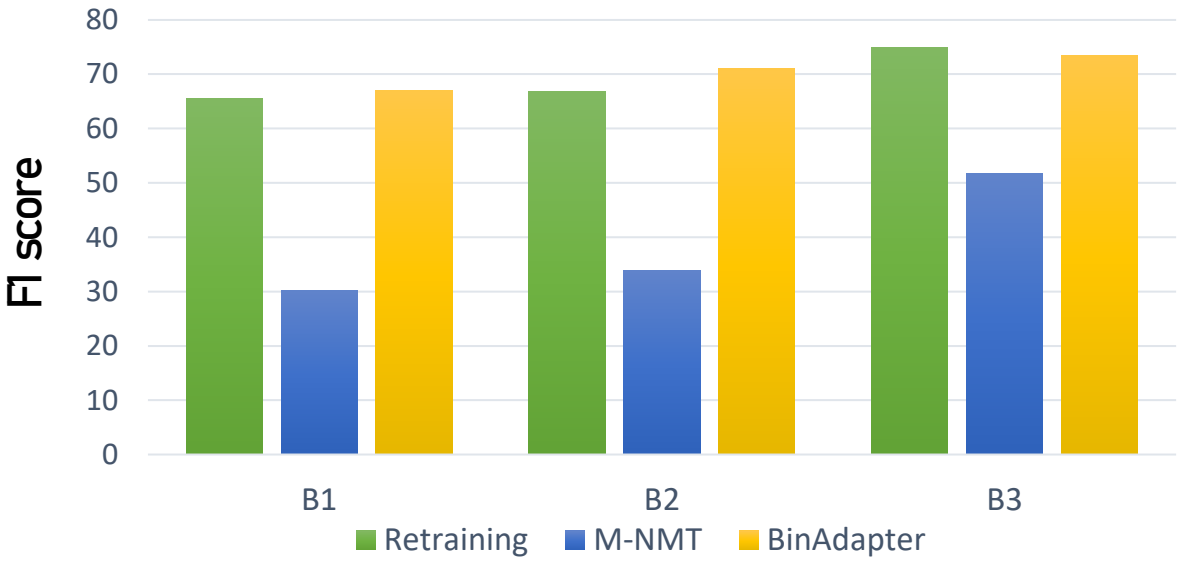
4. How does BinAdapter learn transformed code semantics like optimization levels?

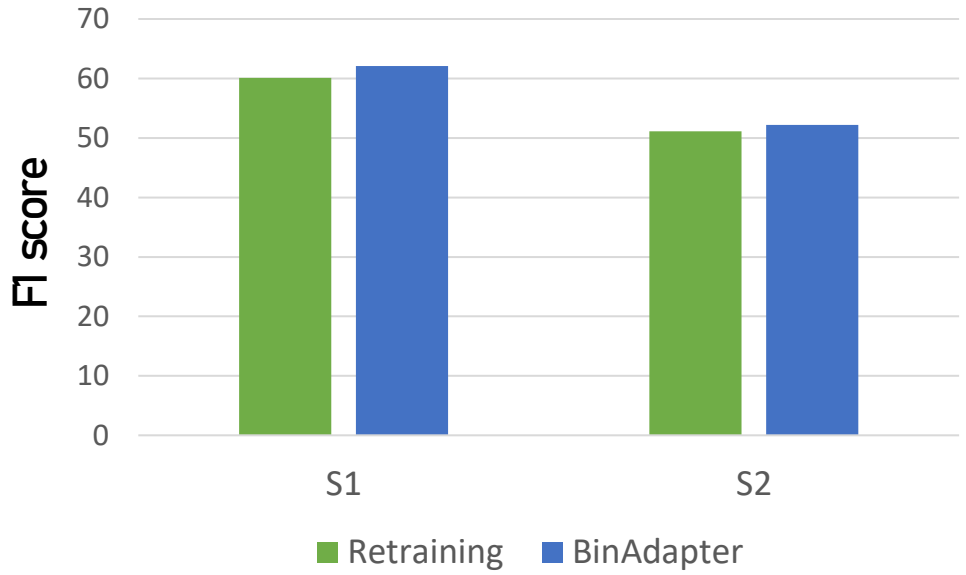
(RQ1) Effectiveness (Scenario 1)



A – AsmDepictor original dataset
A[0-3] – random splits of functions

(RQ1) Effectiveness (Scenario 1 and 2)





B1 – BinKit of ARM
B2 – BinKit of MIPS
B3 – BinKit of C++ programs

S1 – SPEC2006 split 1
S2 – SPEC2006 split 2

(RQ3) Ablation Study

Number of adapters for each layer

Fine-tuned layers	F1 score	Training Parameters
Src embedding	64.42	42.9%
Src embedding + 1 Encoder	62.89	51.1%
Src embedding + 2 Encoders	64.67	59.4%
Src embedding + 3 (all) Encoders	63.79	67.7%

Number of embeddings for training

# of Adapters Per Layer	F1 score	Training Parameters
1	61.17	3.1%
2	61.31	6.1%

Discussions & Limitations

- Robustness against code transformations
- Generalizability of CL to other models
- Storage overheads and inference time

Conclusion

- Static AI models struggle to handle incremental binary data
- **BinAdapter** – the first CL system for the Function Name Inference problem

- https://github.com/SecAI-Lab/BinAdapter

Thank you! (Q & A)