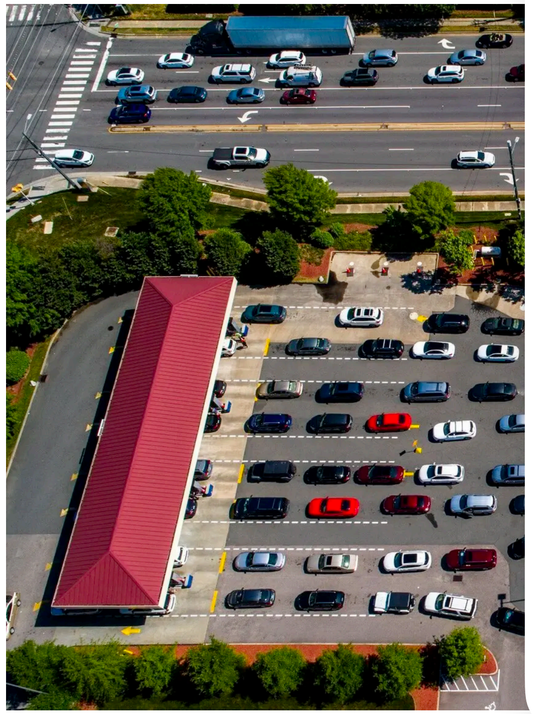
CMSC818I: Advanced Topics in Computer Systems; Large Language Models, Security, and Privacy

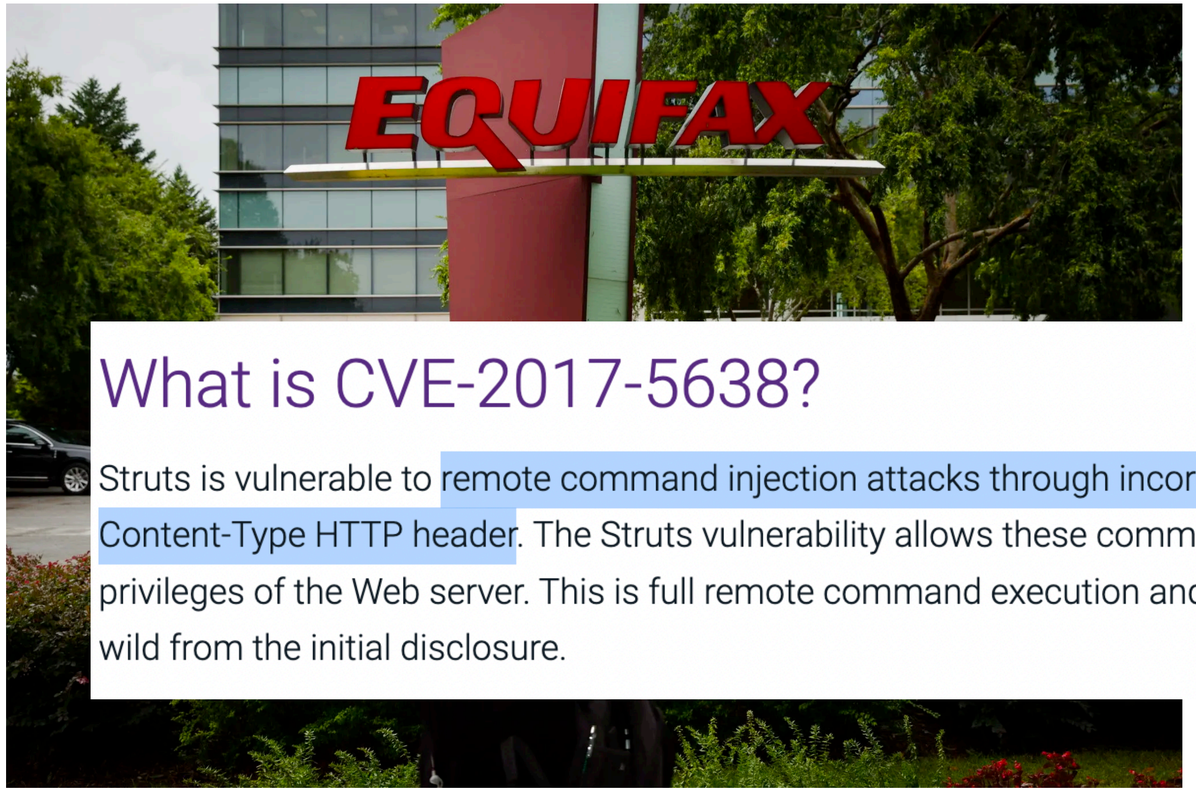
Large Language Models for Vulnerability Detection
8/31/2023

Cybersecurity Issues Affect Our Daily Lives

Colonial Pipeline in Ransom to Hac



Equifax Breach Affected 147 Million



What is CVE-2017-5638?

Struts is vulnerable to remote command injection attacks through incorrectly parsing an attacker's invalid Content-Type HTTP header. The Struts vulnerability allows these commands to be executed under the privileges of the Web server. This is full remote command execution and has been actively exploited in the wild from the initial disclosure.

Vulnerable Source Code Detection

- Security vulnerabilities cost companies billions of $ every year.
- https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html

Rank	ID	Name
1	CWE-787	Out-of-bounds Write
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
4	CWE-416	Use After Free
5	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
6	CWE-20	Improper Input Validation
7	CWE-125	Out-of-bounds Read

How?

- Static Analysis
  - e.g., CodeQL https://codeql.github.com/
  - e.g., Infer, Flawfinder
- Dynamic Analysis
  - e.g., Fuzzing, Taint Analysis
- Deep Learning

Deep Learning for Vulnerability Detection

- Why Deep Learning?
  - Also static
  - **Supposedly** better than rule-based static analyzer
    - https://github.com/github/codeql/tree/main/cpp/ql/src/Security/CWE
    - Tedious rule writing, heuristics, etc.
- **Does deep learning really work?**

SoTA Datasets

- Juliet, synthetic
  - e.g., https://samate.nist.gov/SARD/test-cases/231845/versions/2.0.0
- ReVeal
  - Chromium, Debian issues
  - e.g., https://security-tracker.debian.org/tracker/CVE-2023-4572
- Datasets that collect from National Vulnerability Database: **BigVul, CrossVul, CVEFixes**
  - e.g., https://nvd.nist.gov/vuln/detail/CVE-2014-3647

SoTA: ReVeal Classifier

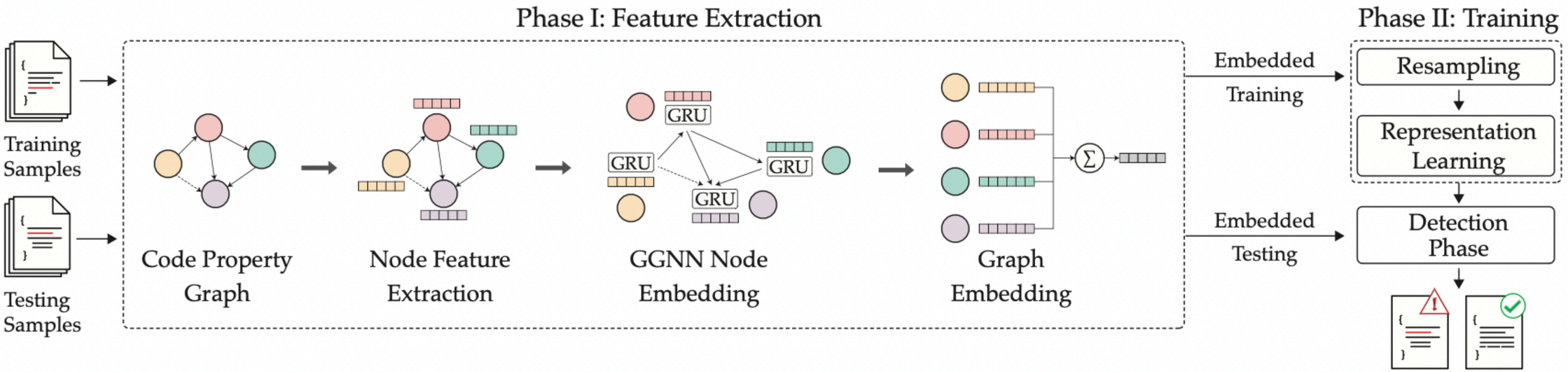
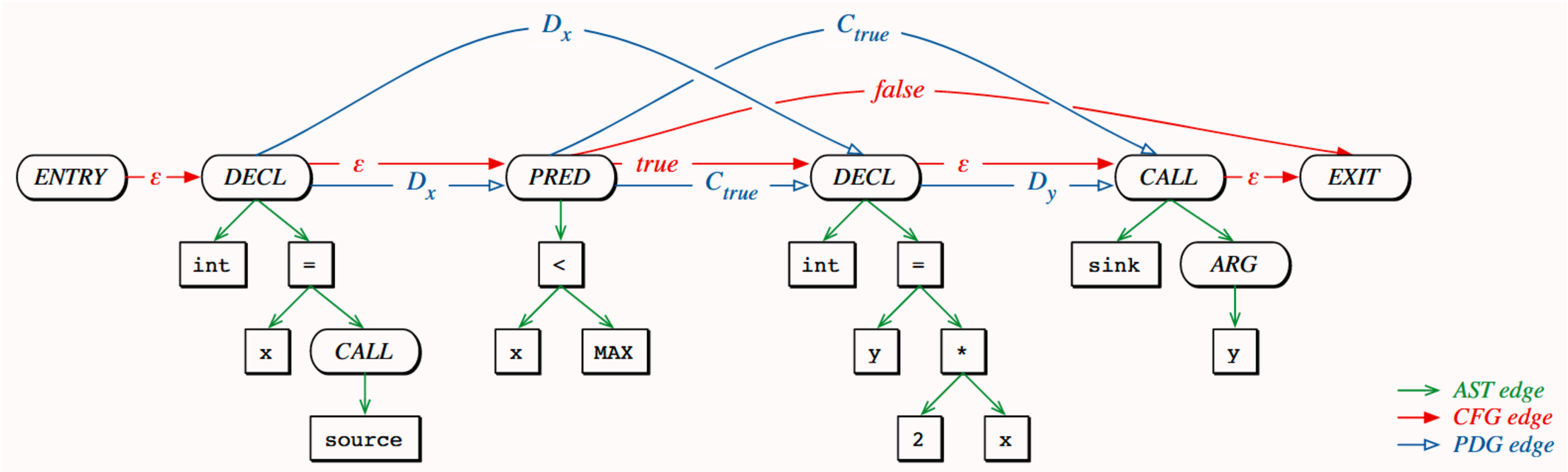


Figure 6: Overview of the REVEAL vulnerability prediction framework.

Code Property Graph

- https://coderpad.io/blog/development/code-property-graph-oriented-databases-source-code-analysis/
- **CPG = AST+CFG+PDG**

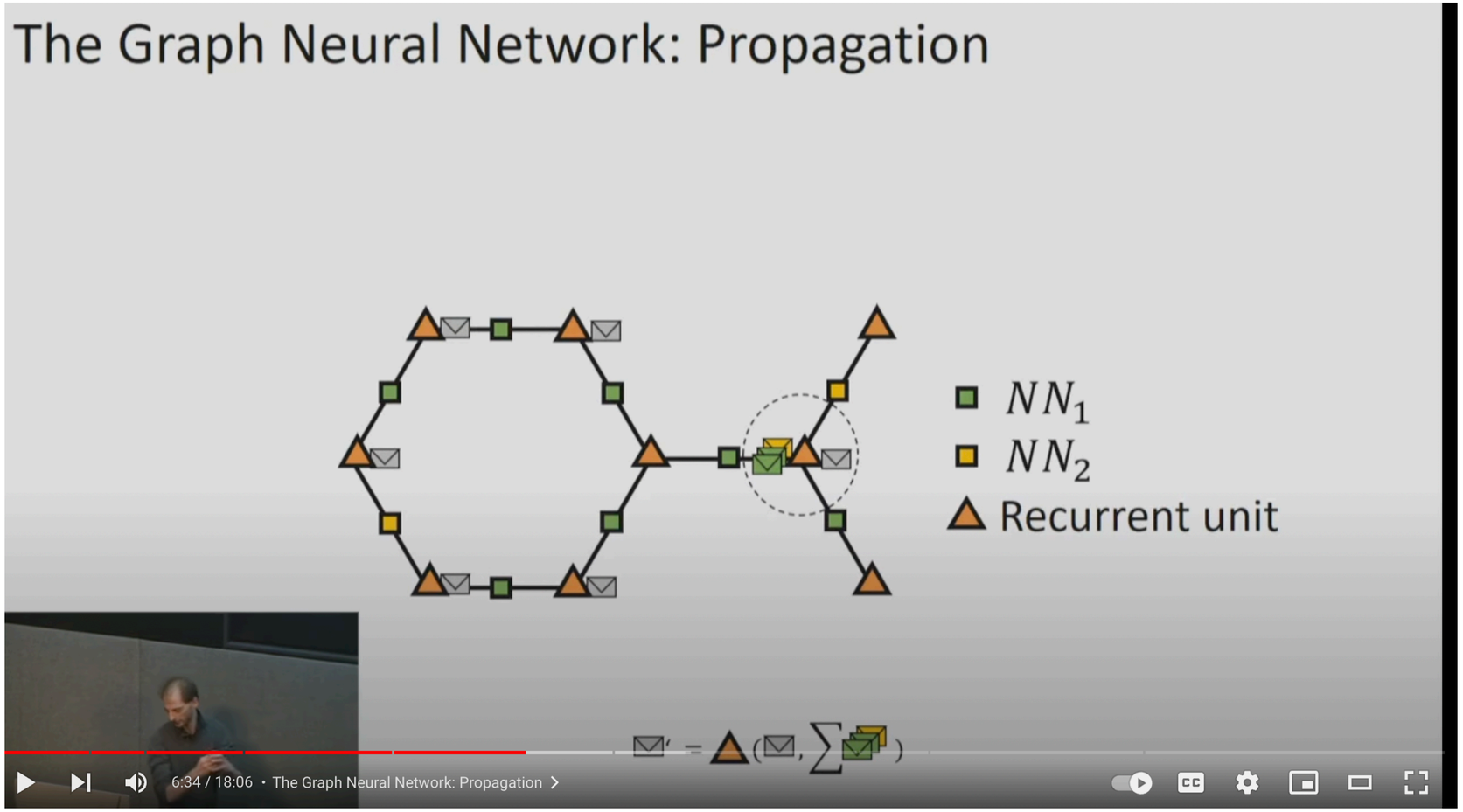
```
void foo(){    int x=source();    if (x<MAX)    {   int y = 2 * x;   sink(y);    }}
```



GNN

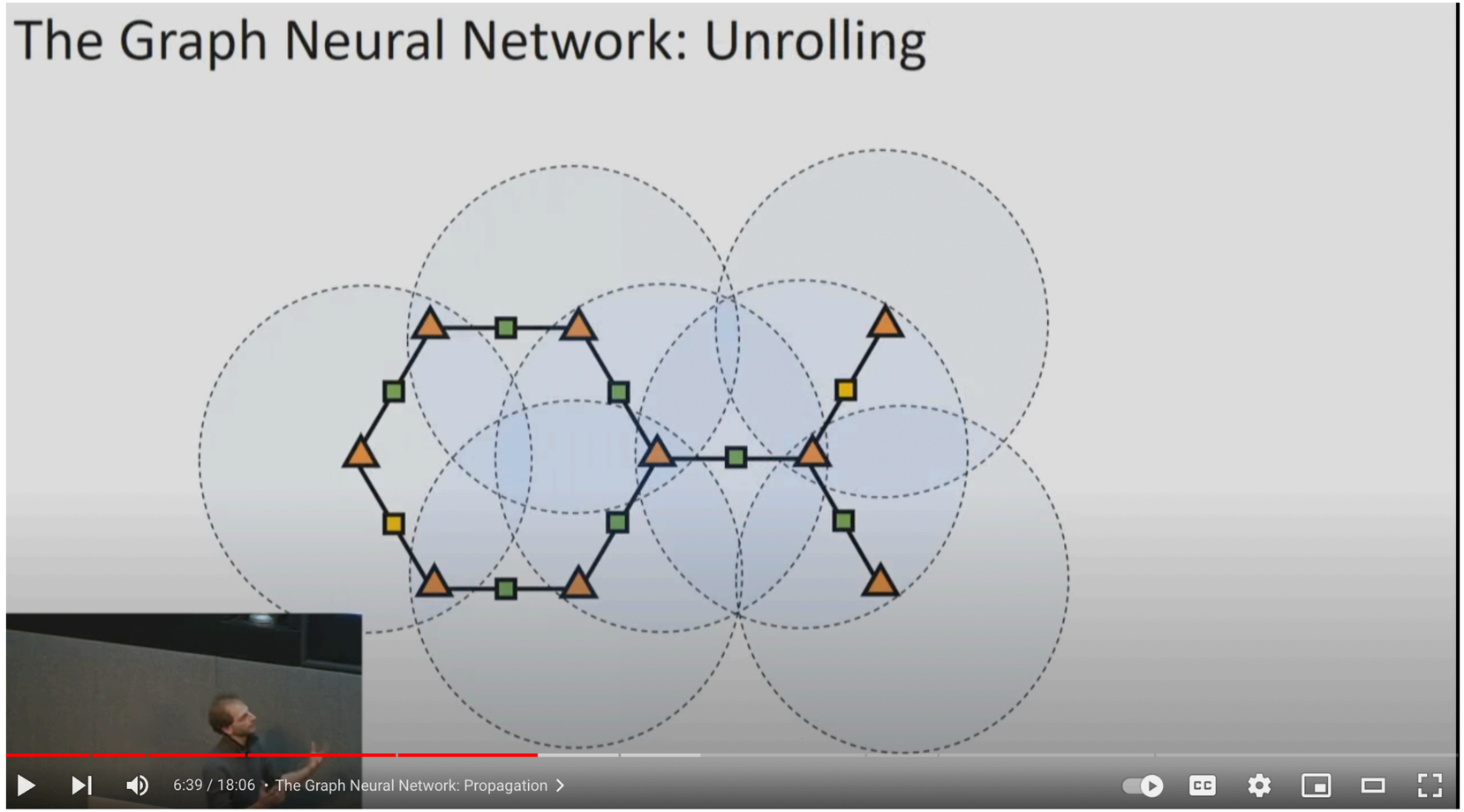
- Learn embeddings for each node, edge, graph
- Message passing

GNN



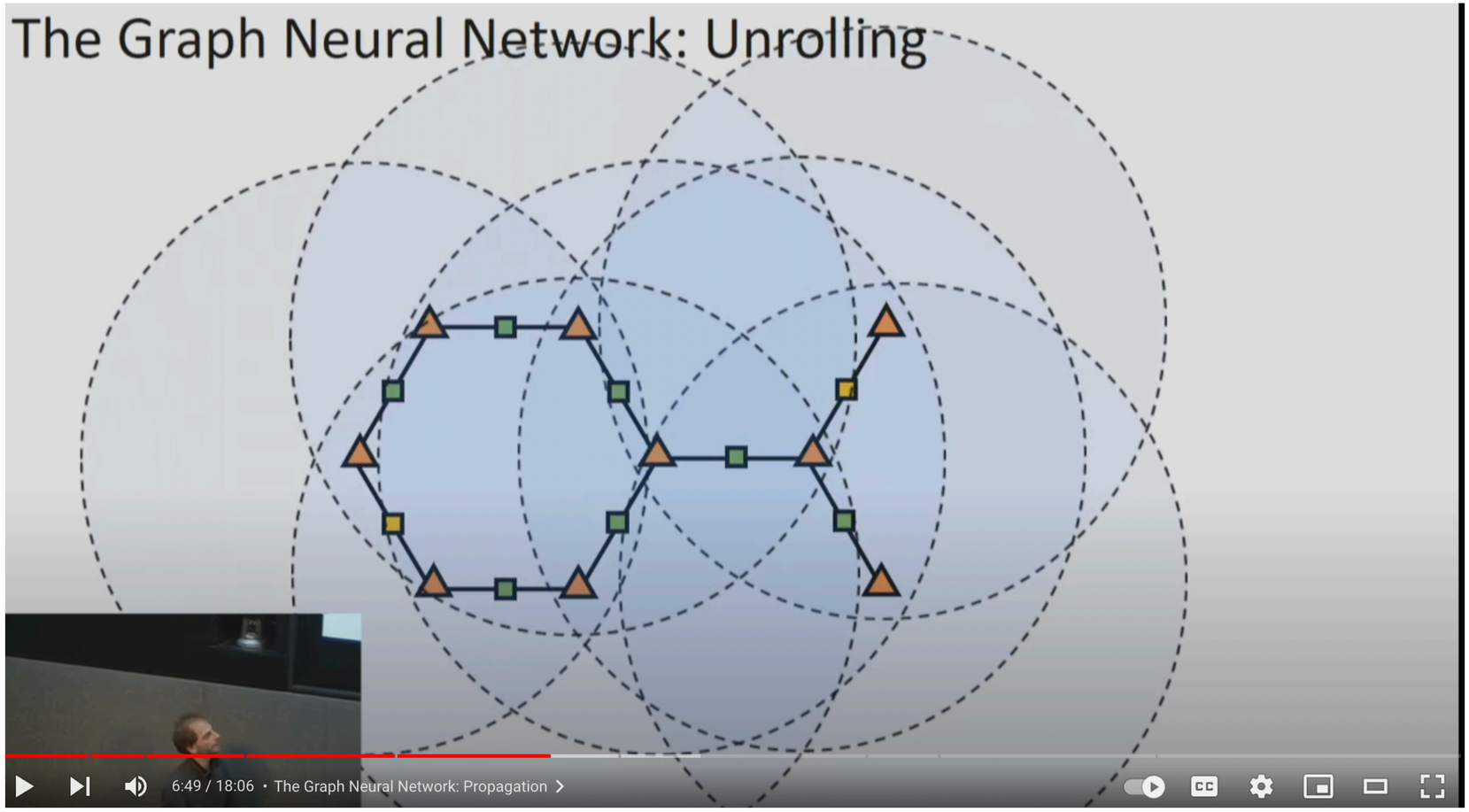
Source: “Microsoft Research: Graph neural networks: Variations and applications”

GNN



Source: “Microsoft Research: Graph neural networks: Variations and applications”

GNN



Source: “Microsoft Research: Graph neural networks: Variations and applications”

SoTA: ReVeal Classifier

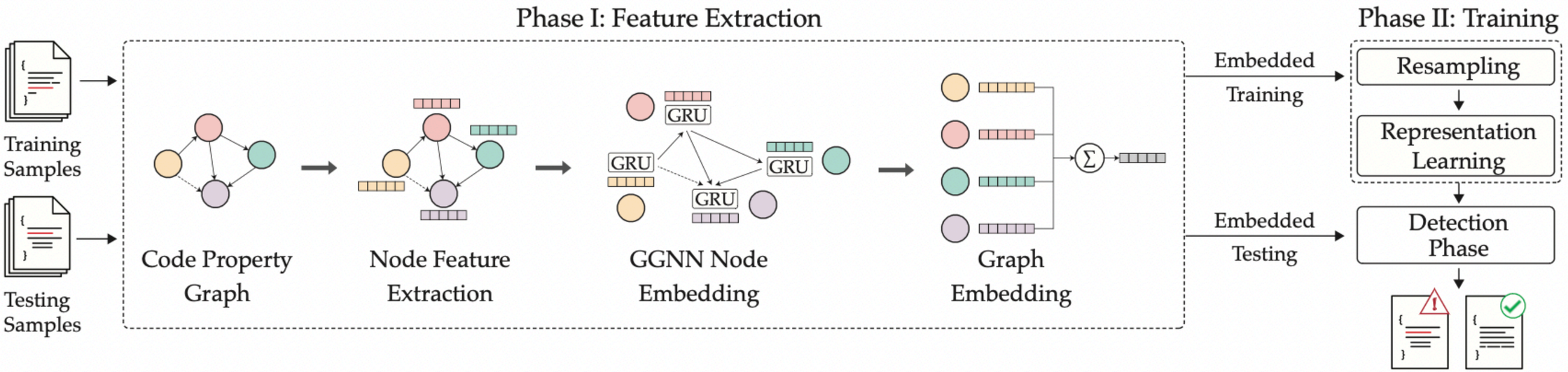


Figure 6: Overview of the REVEAL vulnerability prediction framework.

LLMs Trained on Code

- Summarize Code
- Generate Code from Description
- Detect Bugs
- Translate Code between Programming Languages
- …

Three Families of Models

GPT-2 Family

- GPT-2 Base
- CodeGPT
- PolyCoder

RoBERTa Family

- RoBERTa
- CodeBERT
- GraphCodeBERT

T5 Family

- T5 Base
- CodeT5 Small
- CodeT5 Base
- NatGen

GPT-2 Family

- GPT-2 Base, CodeGPT, PolyCoder
- 12 layers of Transformer **decoders**
- 117M to 160M
- Causal Language Modeling, i.e., next token prediction

RoBERTa Family

- RoBERTa, CodeBERT, and GraphCodeBERT
- 12 layers of Transformer **encoders**
- 125M model parameters
- Masked Language Modeling

T5 Family

- T5 Base, CodeT5 Small, CodeT5 Base, NatGen
- CodeT5 Small: 6 encoder layers and 6 decoder layers, 60M parameters
- Others: 12 encoder layers and 12 decoder layers, 220M parameters
- T5: masks spans of tokens
- Others: …

CodeT5

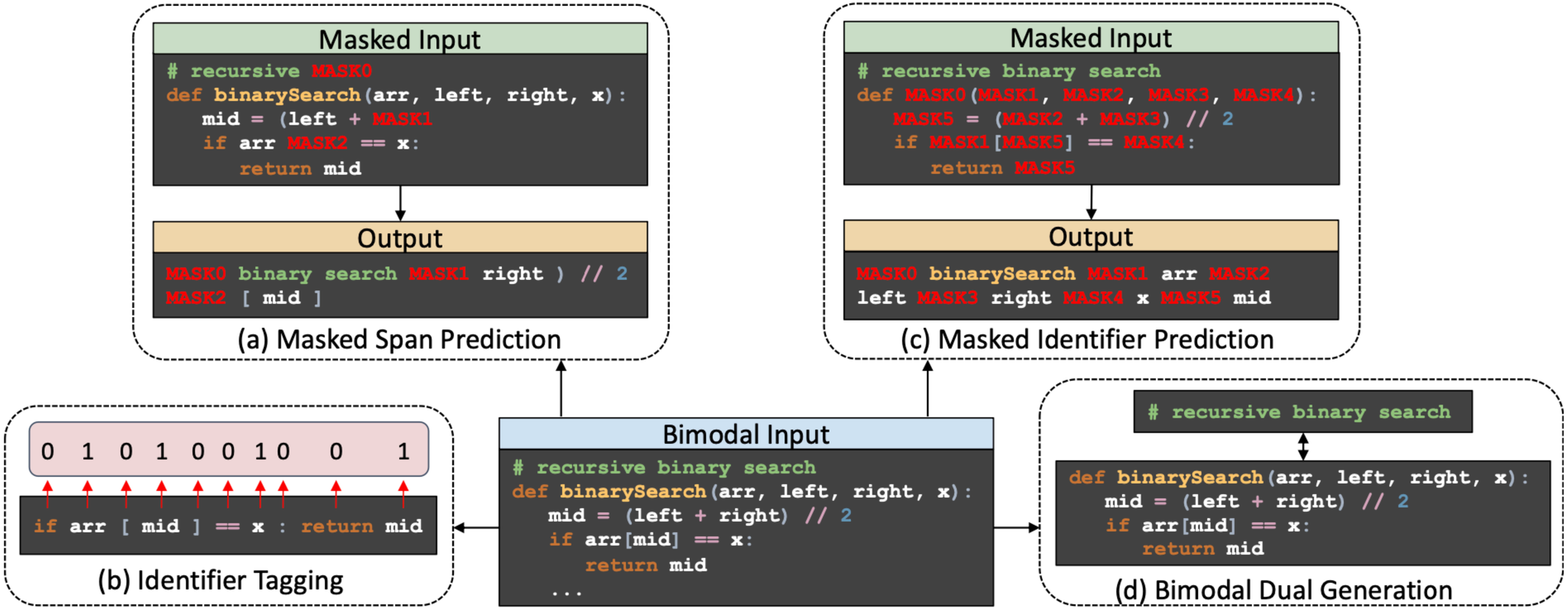


Figure 2: Pre-training tasks of CodeT5. We first alternately train span prediction, identifier prediction, and identifier tagging on both unimodal and bimodal data, and then leverage the bimodal data for dual generation training.

NatGen

```
int search(int[] arr, int key, int low, int high){
    while (low <= high) {
        int mid = low  + ((high - low) / 2);
        if(arr[mid] == key)  { return mid; }
        else { high = mid + 1; }
    }
    return -1;
}
```

(a) Original Code

```
int search(int[] arr, int key, int low, int high){
    for ( ; low <= high ; ) {
        int mid = low  + ((high - low) / 2);
        if(arr[mid] == key) { return mid; }
        else { high = mid + 1; }
    }
    return -1;
}
```

(b) Loop Transformation

```
int search(int[] arr, int key, int low, int high){
    while (low <= high) {
        int mid = low  + ((high - low) / 2);
        while ( i < i ) {
            high = mid + 1;
        }
        // ... Rest of the Code
    }
    return -1;
}
```

(c) DeadCode Insertion

```
int search(int[] arr, int key, int low, int high){
    while ( high >= low ) {
        int mid = low  + ((high - low) / 2);
        if( arr[mid] != key )  {
            high  =  mid + 1;
        }
        else { return mid; }
    }
    return -1;
}
```

(d) Block and Operand Swap

```
int search(int[] arr, int key, int low, int high){
    while (low <= high) {
        int mid = low  + ((high - low) / 2);
        if(arr[mid] == key) { return mid; }
        else {
            high = mid+ + ;
        }
    }
    return -1;
}
```

(e) Inserting confusing code element

```
int search(int[] var_1 , int key, int low, int var_2 ){
    while (low <= var_2 ) {
        int mid = low  + (( var_2 - low) / 2);
        if( var_1 [mid] == key) { return mid; }
        else { var_2 = mid + 1; }
    }
    return -1;
}
```

(f) Variable Renaming

Figure 2: Semantic preserving transformation used to prepare the pre-training data for NatGen.

Code LLM Evaluation

- Code Translation
- Code Clone Detection
- Code Summarization
- Defect Detection (some vulnerable source code)
- Text to Code Generation
- …

DiverseVul

- A New Dataset: DiverseVul
- A Measurement Study
  - Compare SoTA GNN and 10 LLMs
- Label Noise Analysis

Dataset Collection

- Identify security issue website
- Crawl them
- Extract git commit URLs, issue text, comments
- Get project, commit ID from git commit URLs
- Clone repos
- Process the commits
  - Vulnerable: changed functions before the commit
  - Nonvulnerable: fixed functions, all unchanged functions

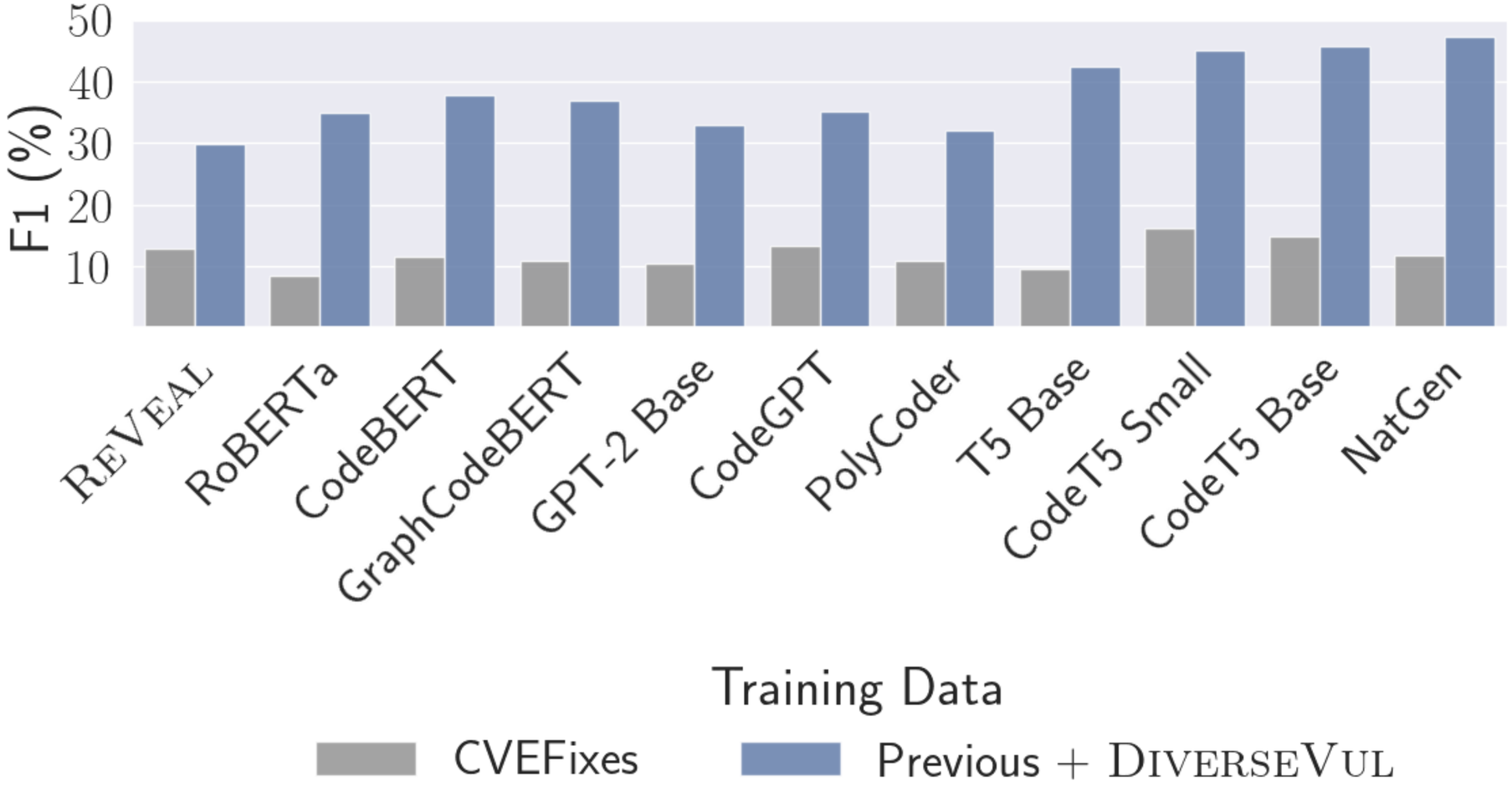
Merge Datasets

Dataset	# Projects	# CWEs	# Functions	# Vul Func	# Vul Func with CWE Info	# Commits
Devign	2▽	N/A	26,037	11,888	N/A	N/A
ReVeal	2◇	N/A	18,169	1,664	N/A	N/A
BigVul	348	91	264,919	11,823	8,783	3,754
CrossVul*	498	107	134,126	6,884	6,833	3,009
CVEFixes*	564	127	168,089	8,932	8,343	3,614
DiverseVul	**797**	**150**	**330,492**	**18,945**	**16,109**	**7,514**
Previous†	638	140	343,400	30,532	14,159	17,956
Previous + DiverseVul	**933**	**155**	**523,956**	**41,377**	**22,382**	**21,949**

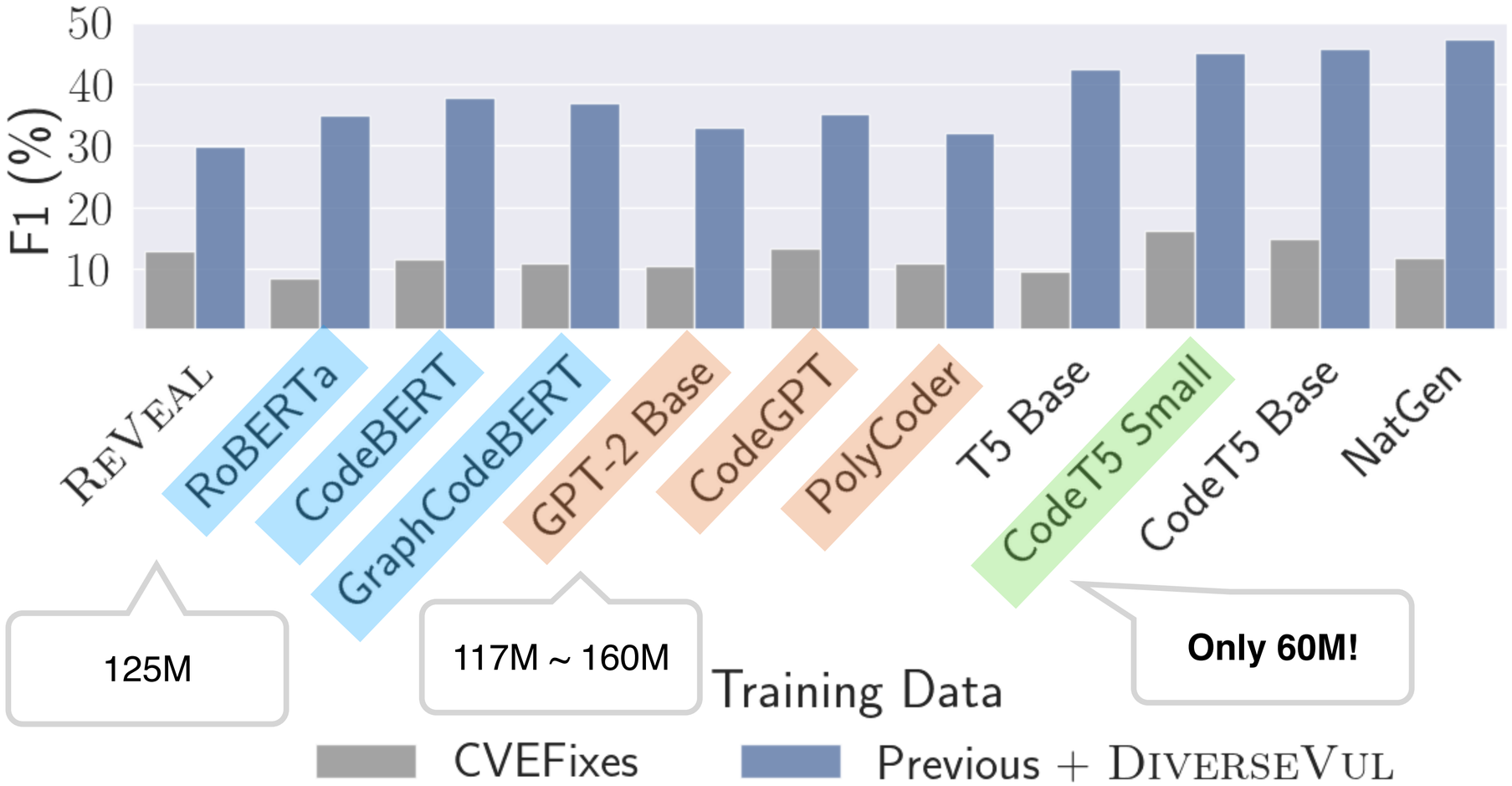
†: We aggregate previous five datasets by combining and deduplicating samples from Devign, ReVeal, BigVul, CrossVul, and CVEfixes.
*: CVEfixes and CrossVul are multi-language datasets. We report numbers for C/C++ code.
▽: Devign authors released data from two repositories: `FFMPeg+Qemu`. ◇: Chromium and Debian packages.

Table 3: Statistics about previous five datasets, DiverseVul, merged Previous dataset, and Previous + DiverseVul.

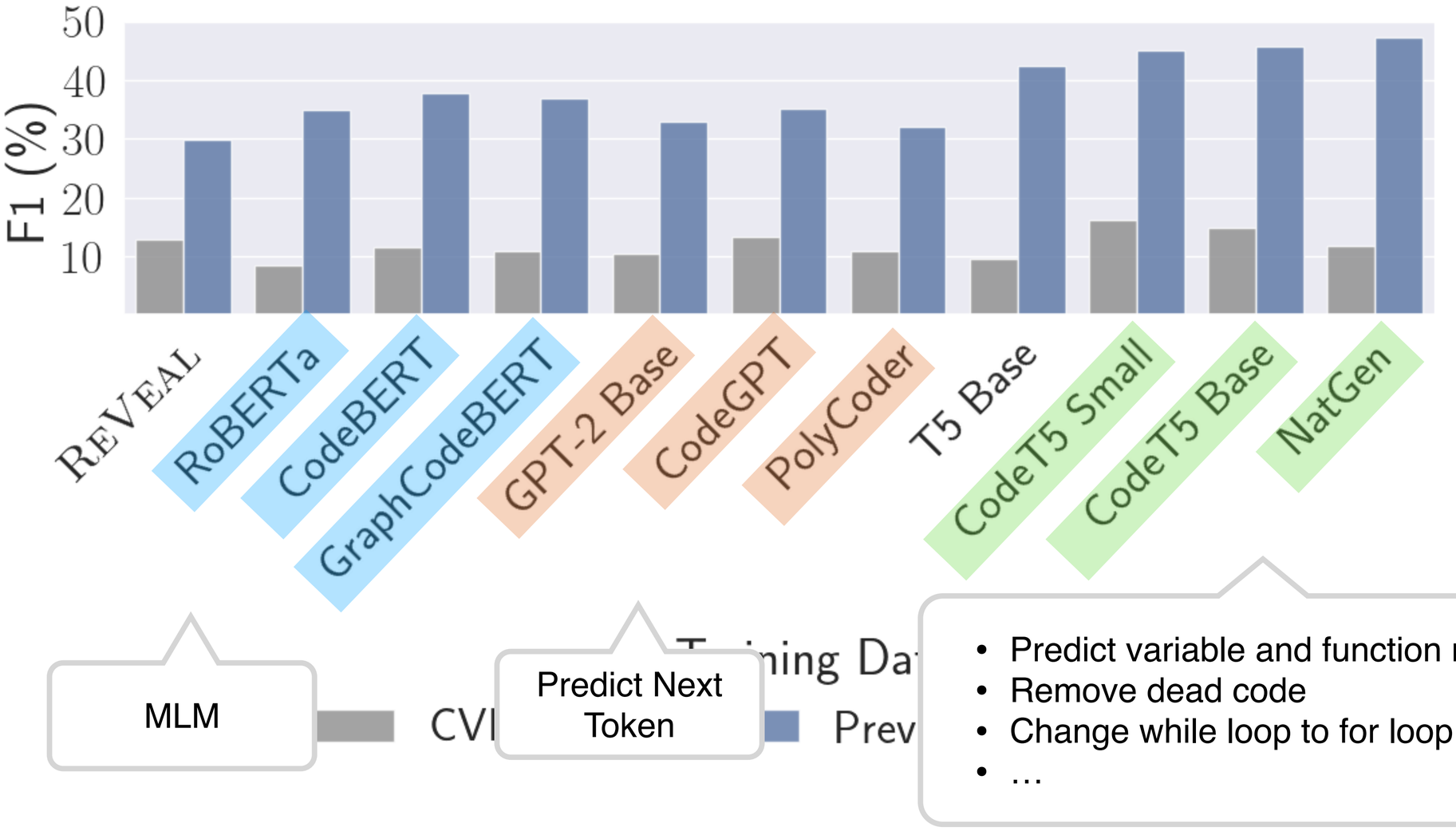
Smaller vs Larger Training Set



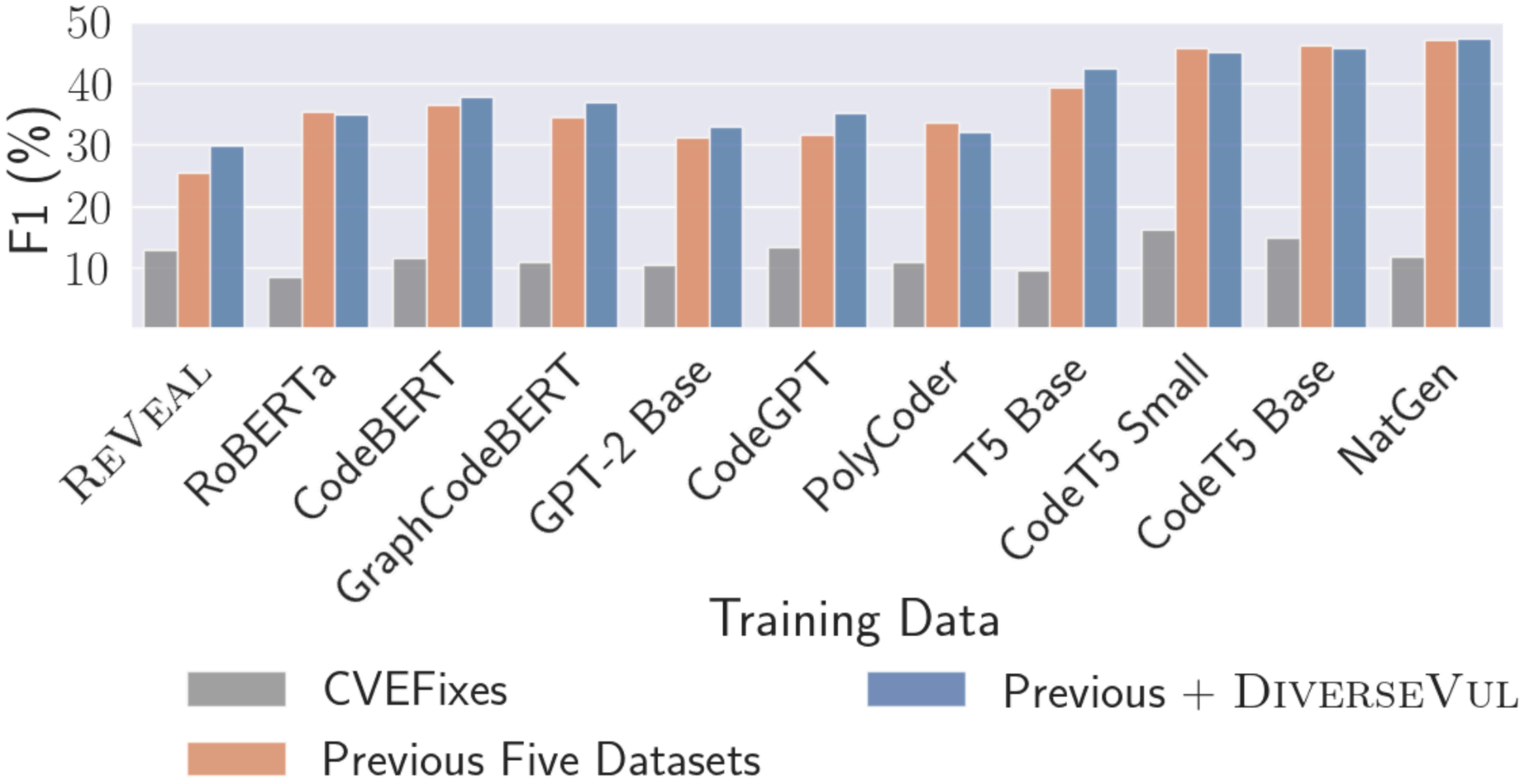
Is a larger model always better? No!



Code-Specific Pretraining Tasks

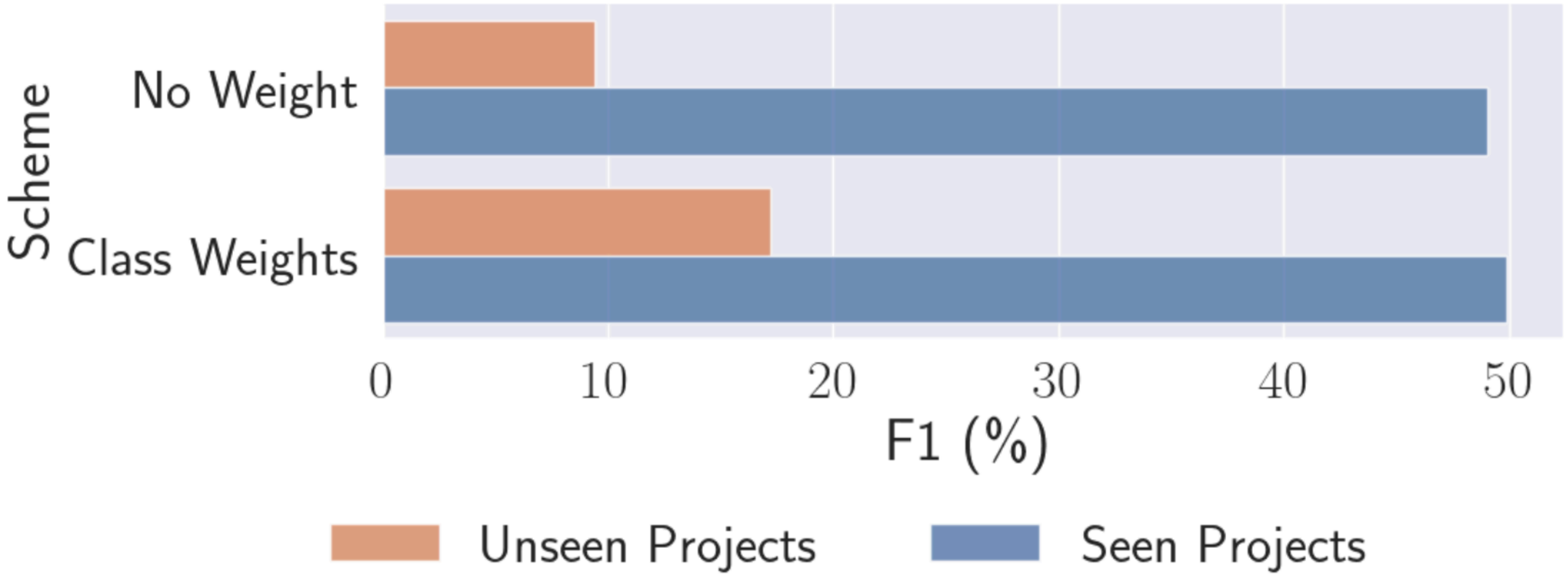


Does More Data Help?



Does More Data Help?

Generalization Issue



Label Accuracy Issue

Dataset	Correct Label	Wrong Label		
		Vulnerability Spread Across Multiple Functions	Relevant Consistency	Irrelevant
DiverseVul	60%	10%	12%	18%
CVEFixes ∪ BigVul ∪ CrossVul	36%	12%	12%	40%
CVEFixes	51.7%	10.3%	17.3%	20.7%
BigVul	25%	15.6%	9.4%	50%
CrossVul	47.8%	13%	21.8%	17.4%

Deep Learning Does Not Work

- High FPR
- Generalization Issue
- …

Thank you!

Paper: https://arxiv.org/abs/2304.00409