Binary Similarity Using ML

Noam Shalev Nimrod Partush

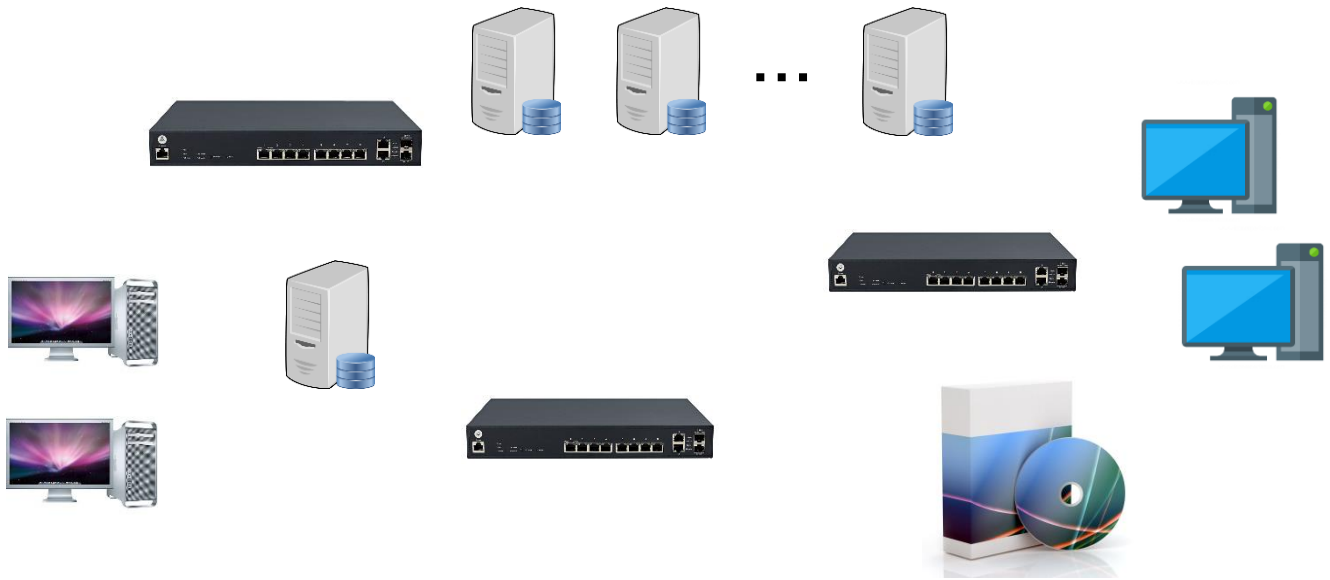
Technion – Israel Institute of Technology

PLAS 2018, October 19th , Toronto, Canada.

TECHNION
Israel Institute of Technology

Motivation

- Vulnerability Search



CEO

Security Officer

Motivation

- Copyright infringement

We do not use licensed code in our products !

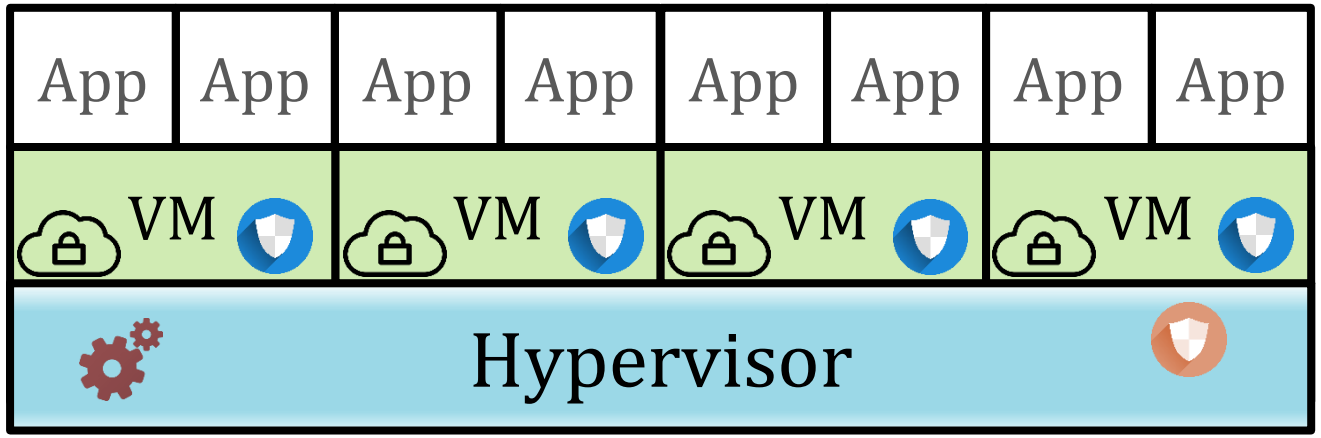
Startup CEO

But how can I be sure?

CEO

Motivation

- Guest protection in cloud setting
  - Scan memory of guest machines
  - Without inquiring guest data



Motivation

- Vulnerability Search
- Copyright infringement
- VM protection

Problem Definition

- Find the same code section after it has been compiled differently
  - And stripped

```
shr       eax, 8
lea       r14d,     [r12+13h]
mov       r13, rbx
lea       rcx, [r13+3]
mov       [r13+1], al
mov       [r13+2], r12b
mov       rdi, rcx
```

Heartbleed(), gcc v.4.9 -O3

?

≈

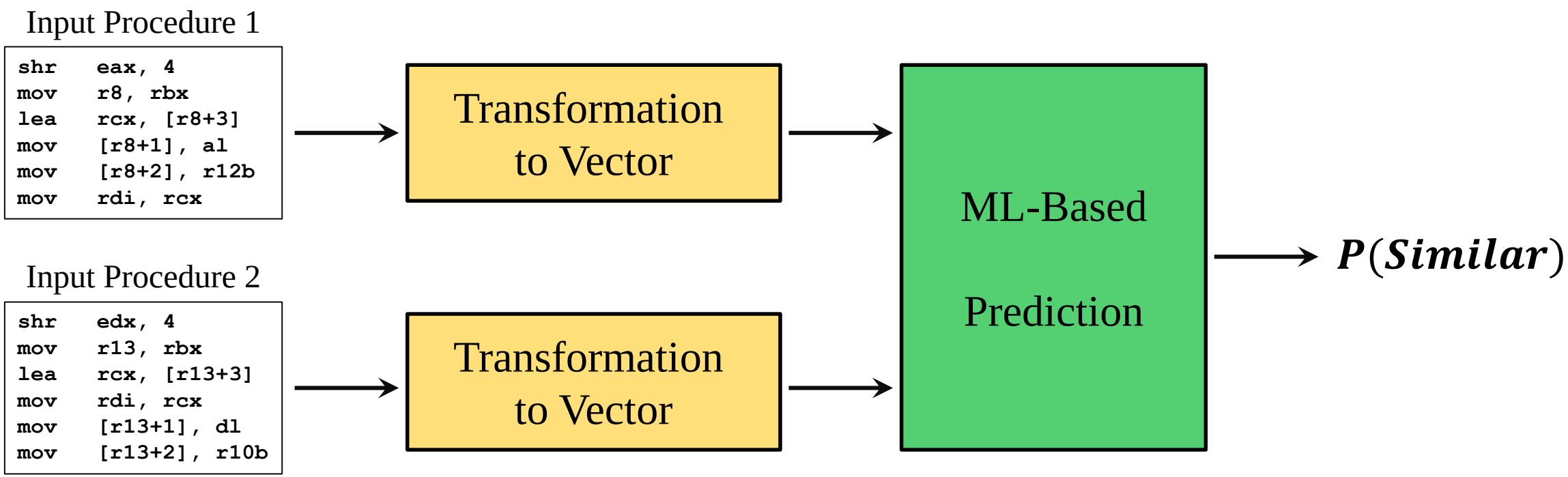
```
mov       r9, 13h
mov       r12, rbx
add       rbp, 3
mov       rsi, rbp
lea       rdi, [r12+3]
mov       [r12+2], bl
lea       r13d, [rcx+r9]
shr       eax, 8
```

Heartbleed(), clang v.3.5 -O3

Similarity Wish-List

- Precise
  - Avoid false positives
- Flexible – find similarities across
  - Compiler versions — `gcc-4.7` vs. `gcc-5.0`
  - Compiler vendors — `gcc-4.8` vs. `icc-15`
  - Optimization levels — `gcc-4.8-O0` vs. `icc-15-O3`
  - Different versions of the same code
- Work on stripped binaries (no debug info)

Algorithm Outline



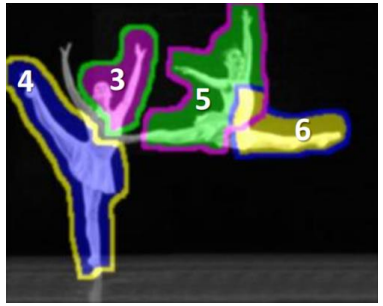
Similarity by Composition

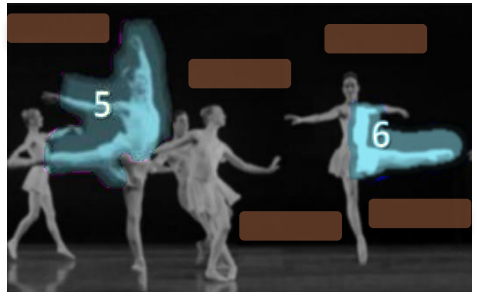
- Similarity principle [Irani et al. 2006]
  - Two signals are similar if one can "compose one signal from large contiguous chunks of the second signal"

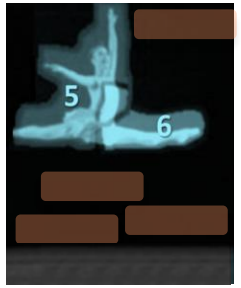
A

B

C

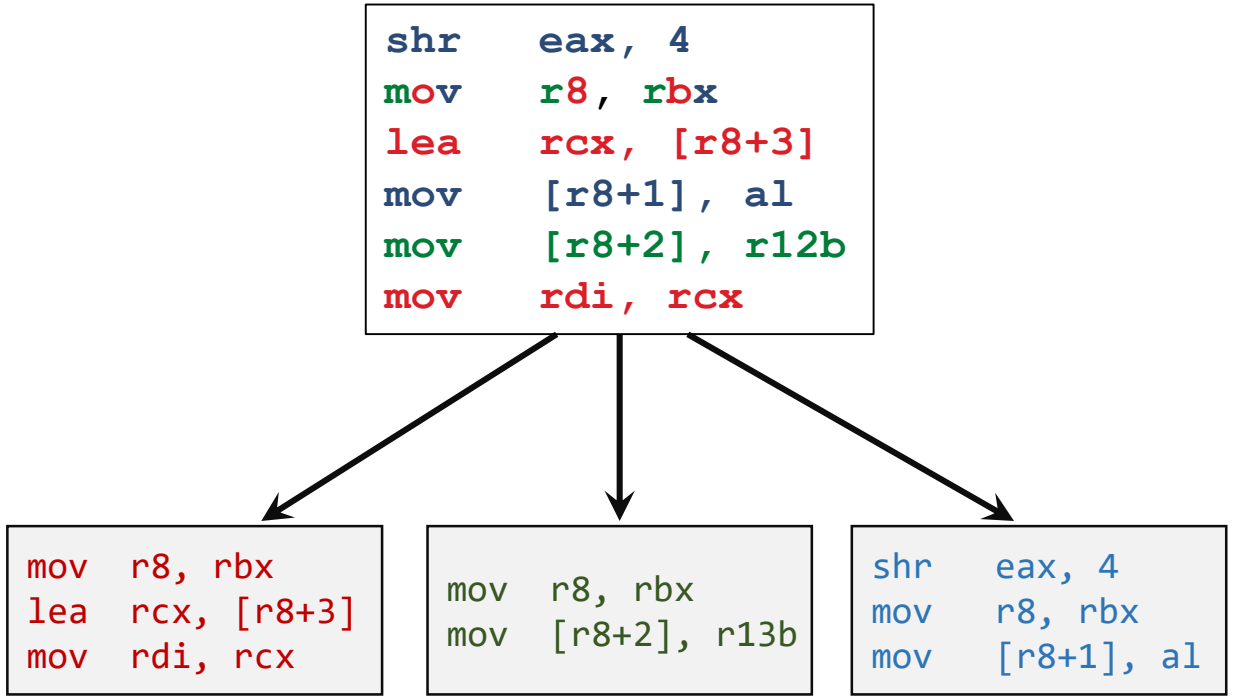






Code Decomposition

- Strand
  - Set of instructions required to compute the value of a certain variable



Code Decomposition

- Strand
  - Set of instructions required to compute the value of a certain variable

```
shr   eax, 4
mov   r8, rbx
lea   rcx, [r8+3]
mov   [r8+1], al
mov   [r8+2], r12b
mov   rdi, rcx
```

  - Not necessarily contiguous
  - Syntactically different strands can be equivalent

Similarity of Binaries

- Statistical similarity of binaries [David et al. 2016]
  1. Decomposition
  2. Pairwise semantic similarity
  3. Statistical similarity evidence

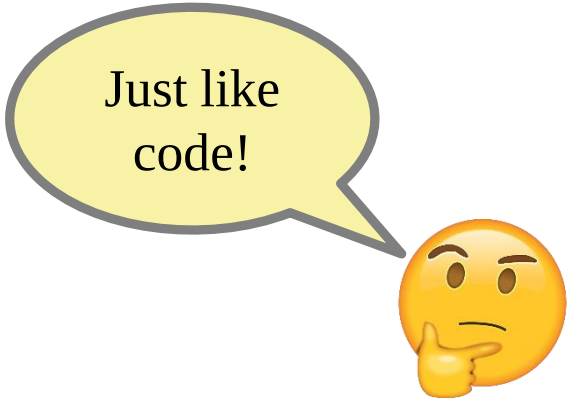
- Slow
  - ~ 20 seconds / comparison

We need a faster & scalable solution

Strands as Features

- Previous work used human-crafted features:
  - # of instructions / branches / calls
  - # of appearances of each assembly mnemonic
  - Specific numerical values

- We propose strands for representing code sections
  - Harder to see in human eyes



Strands as Features

- Challenge: determining equivalence requires a solver
  - Equivalent strands might look differently

- Idea: bring strands to a normalized form
  - In terms of textual representation
  - Requires some preprocessing

Introducing Proc2Vec

- Proc2vec – procedure to vector transformation algorithm
- Given a procedure, split to basic blocks
  - For each basic block

```
shr   eax, 4
mov   r8, rbx
lea   rcx, [r8+3]
mov   [r8+1], al
mov   [r8+2], r12b
mov   rdi, rcx
```

  - Decompose to strands

```
mov  r8, rbx
lea  rcx, [r8+3]
mov  rdi, rcx
```

```
mov  r8, rbx
mov  [r8+2], r13b
```

```
shr   eax, 4
mov   r8, rbx
mov   [r8+1], al
```

Introducing Proc2Vec

- Lift to intermediate representation

```
v1  := rbx
r8  := v1
v2  := r8 + 3
rcx := v2
v3  := rcx
rdi := v3
```

```
v1  := rbx
r8  := v1
v2  := r13
v3  := 64to8(v2)
v4  := r8
v5  := v4 + 2
M[v5] := v3
```

```
v1  := rax
v2  := 64to32(v1)
v3  := v2 / 16
rax := v3
v4  := rbx
r8  := v4
v5  := rax
v6  := 64to8(v5)
v7  := r13
v8  := v7 + 1
M[v8] := v6
```

- Optimize

```
rdi := rbx+3
```

```
M[rbx+2] := 64to8(r13)
```

```
M[r13+1] := 64to8(64to32(rax)/16)
```

- Normalize

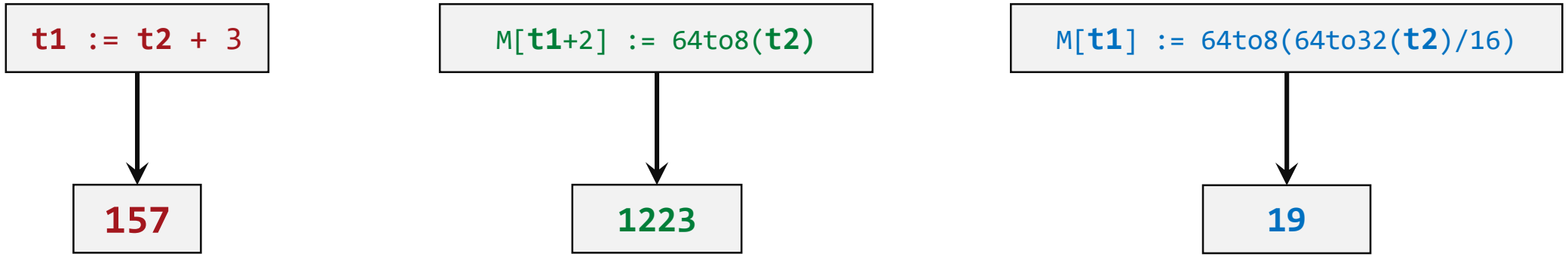
```
t1 := t2 + 3
```

```
M[t1+2] := 64to8(t2)
```

```
M[t1] := 64to8(64to32(t2)/16)
```

Introducing Proc2Vec

- Transform text to numbers by applying b-bit Md5 hash

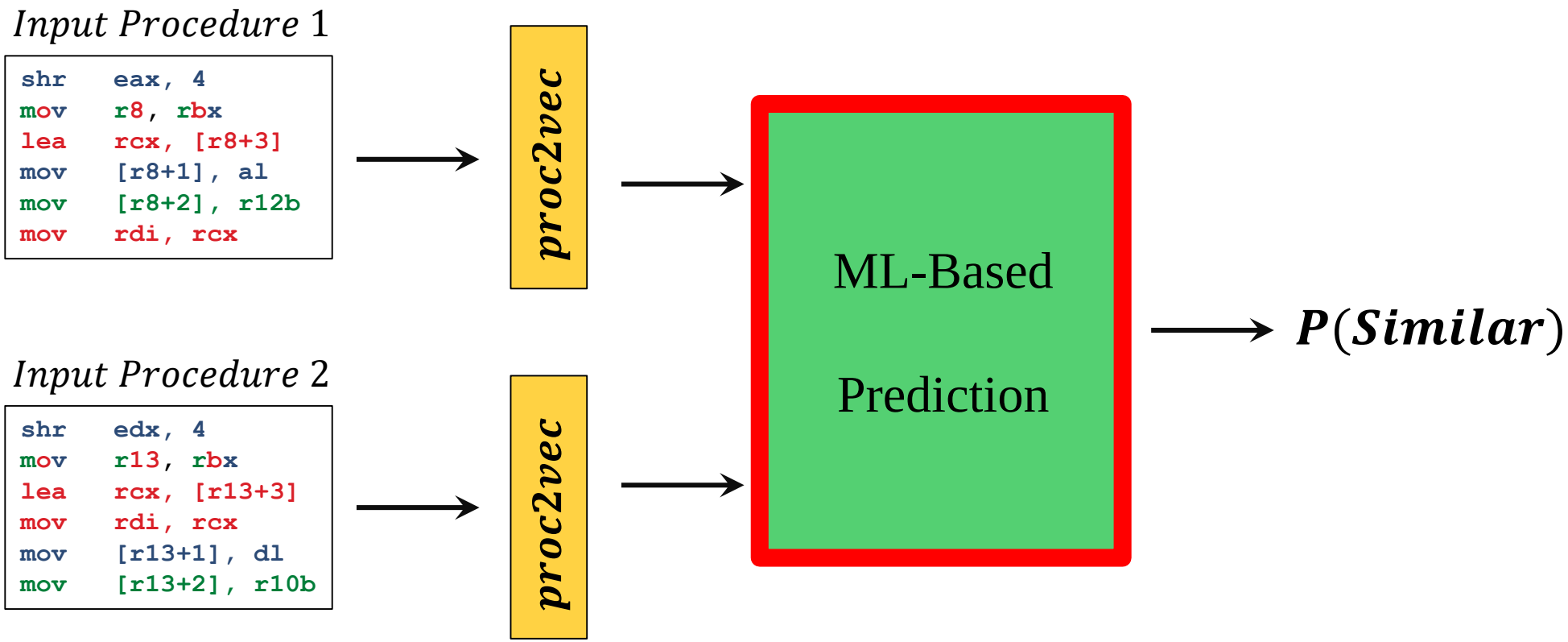


- Assemble the numbers into a vector

$$V[0: 2^b - 1] = (0, \dots, 0, 1, 0, \dots, 0, 1, 0, \dots, 0, 1, 0, \dots, 0)$$

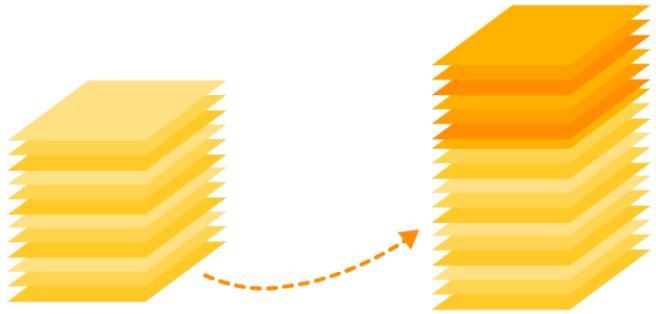
Index 19 Index 157 Index 1223

Algorithm Outline



Designing NN Predictor

- Dataset:
  - Open source projects from various fields:
    - `OpenSSL, binutils, bash, httpd, ntp, cURL, Snort, Git, ...`
  - Compiled using various
    - Compiler vendors (`gcc, icc, Clang`)
    - Compiler versions
    - Optimization levels (`-O{0,1,2,3,s}`)
    - Target architectures (`x86_64, AArch64`)
  - Overall ~1 Million procedures



Designing NN Predictor

- Challenges:
  - Similarity is symmetric
    - Generate symmetric data

$$(x, y) \rightarrow (x, x), (x, y), (y, x), (y, y)$$

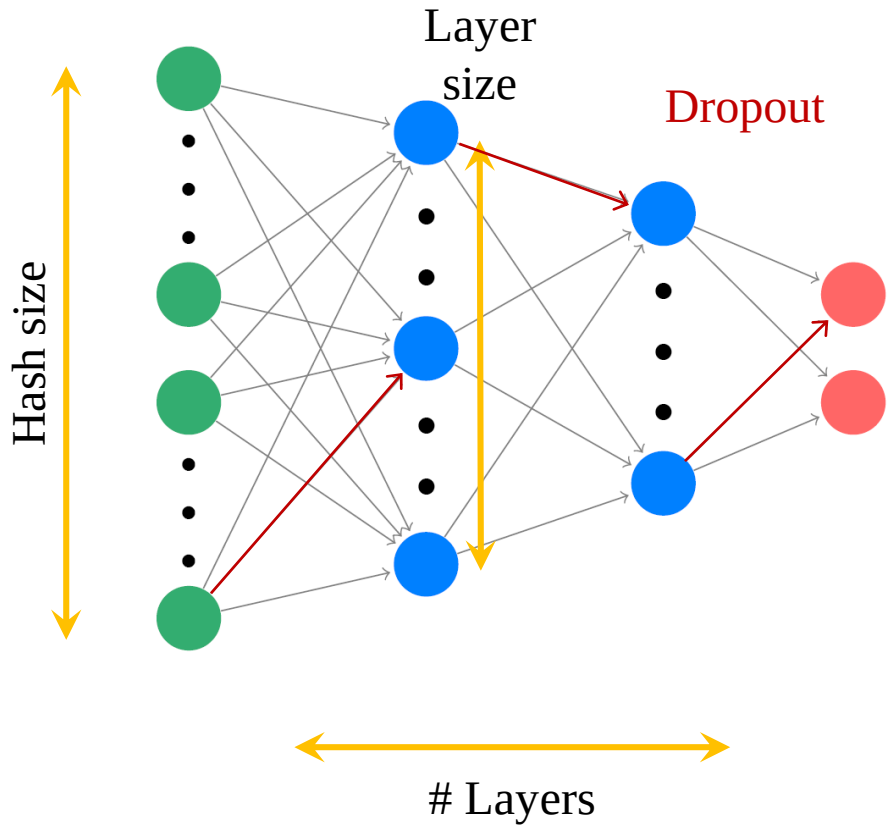
  - Procedures mostly don't match
    - Generate unbalanced dataset

$$match \quad \mathbf{1 : 6} \quad nonmatch$$

  - A predictor that always predict nonmatch gets high accuracy
    - Use CROC for measuring accuracy

Designing NN Predictor

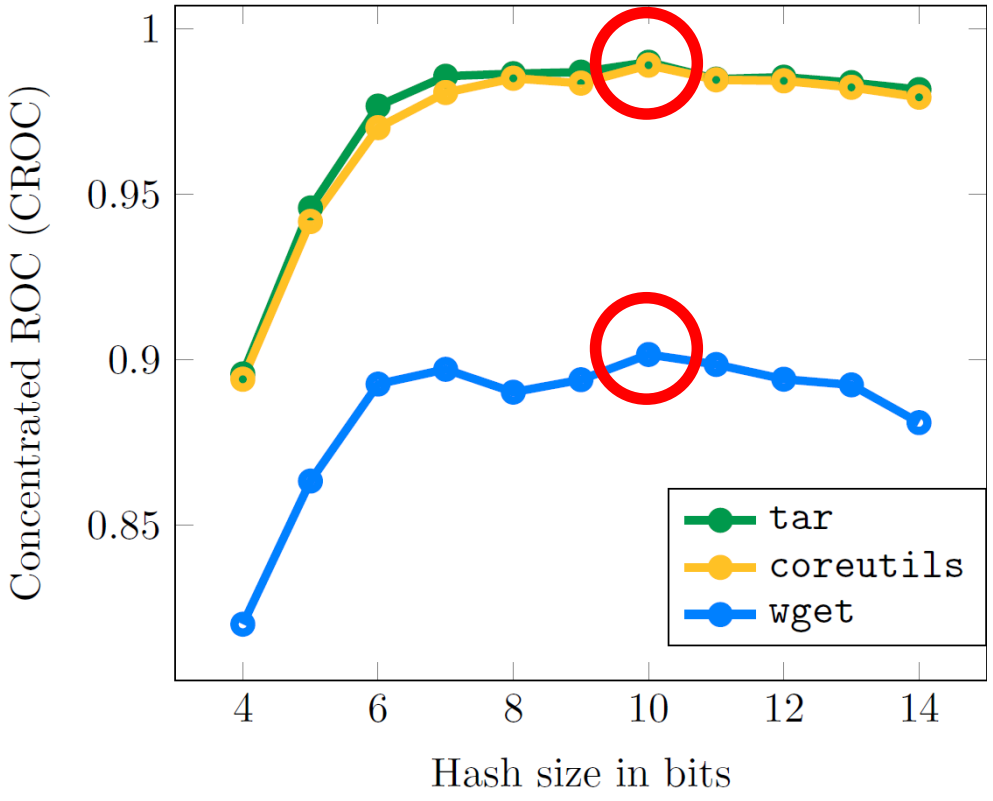
- Parameter tuning:
  - Train over 500K examples
    - Variable hash size
    - Variable number of hidden layers
    - Variable layer sizes
    - Variable regularization values



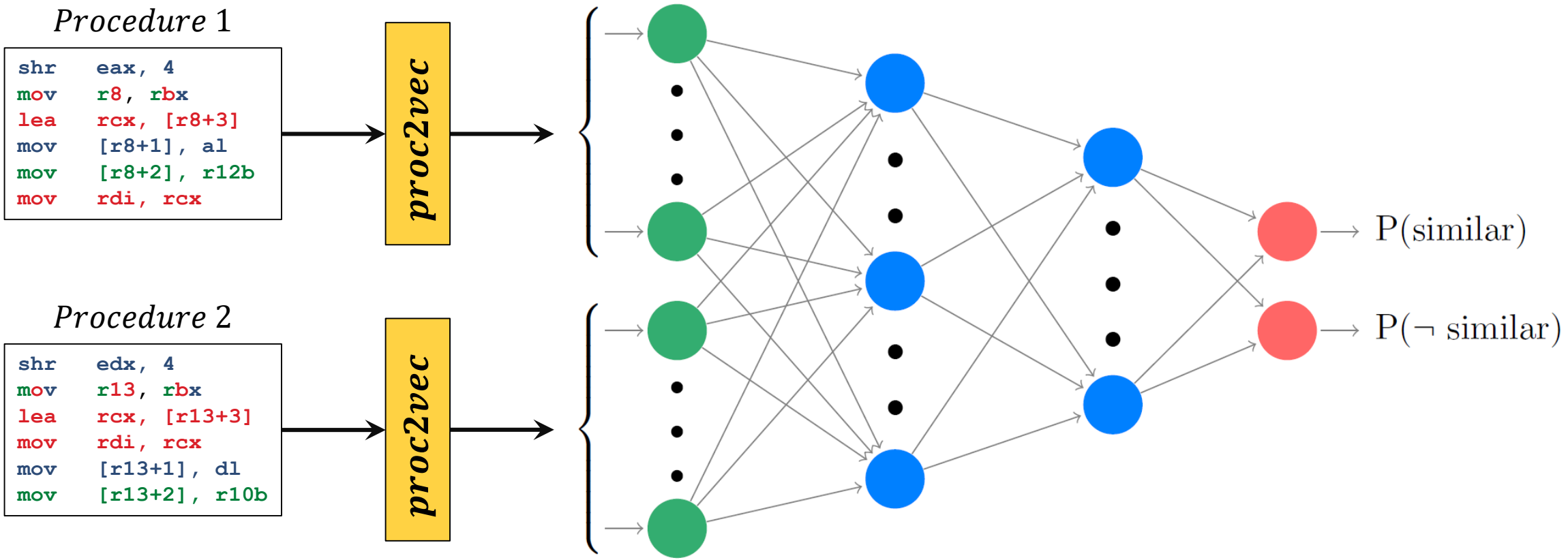
Designing NN Predictor

- Parameter tuning:
  - Train over 500K examples
    - Variable hash size

➢ Choose **10**



Algorithm Outline



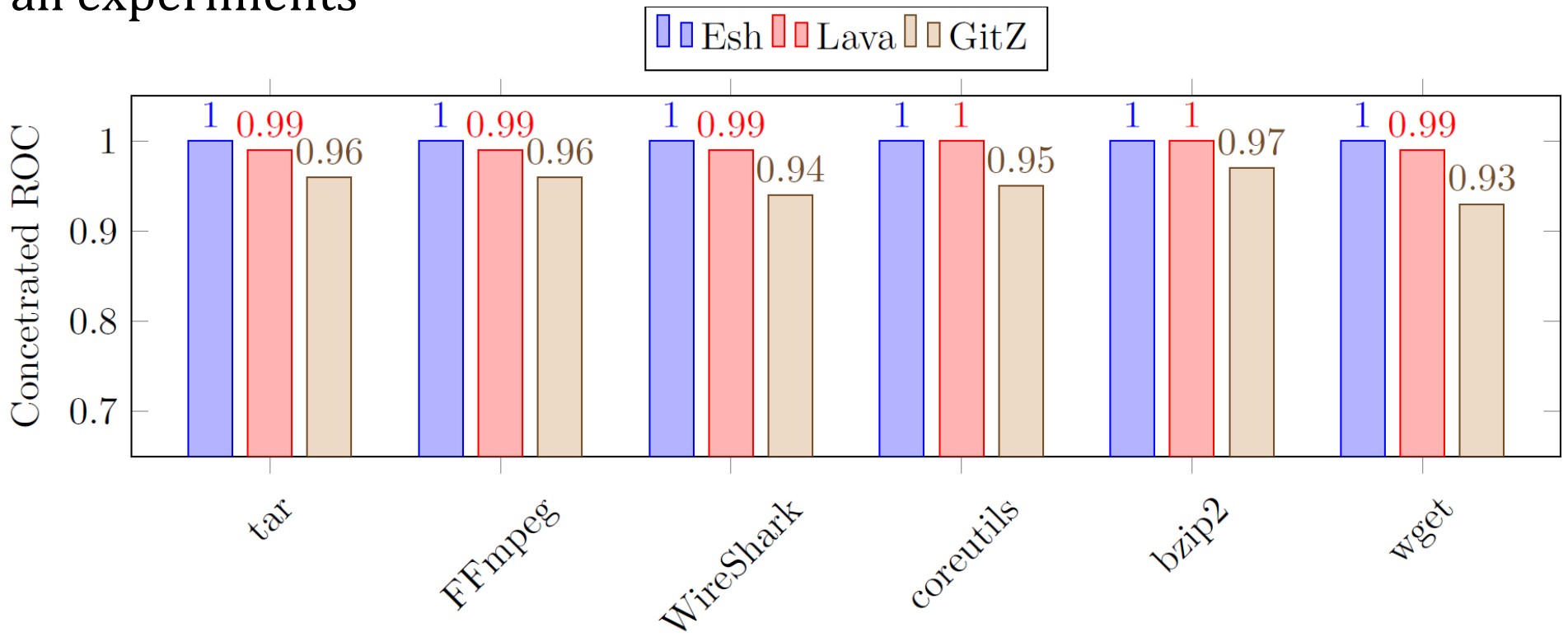
Evaluation

- Comparing to GitZ [David et. al 2017]
  - State-of-the-art fast tool

- All vs. all experiments
  1. Compile differently
  2. Predict similarities (n^2 predictions)
  3. Grade using CROC

Evaluation

- Accuracy results
  - Improved accuracy
  - In all experiments



Evaluation

- Throughput

  ~7000 predictions per second

  - On a single core
    - Intel Xeon E5-2640

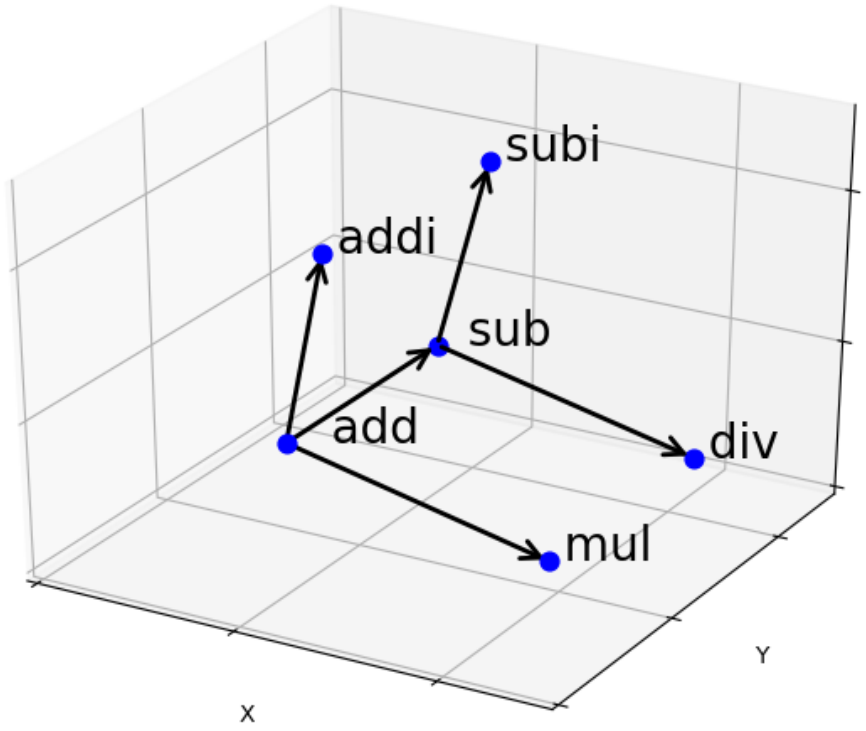
- GitZ throughput is ~30 predictions per second (200X improvement)

Evaluation

- Does it scale?
  - Yes!
    - With the number of cores
  - Procedure vectors are independent
  - Model can be replicated

Future Work

- Vision: inst2vec
  - Find meaningful representations for instructions
    - Capture latent factors
  - Employ NLP techniques for similarity detection



Binary Similarity Using ML

Noam Shalev Nimrod Partush

Thank You!

noams@technion.ac.il

TECHNION
Israel Institute of Technology