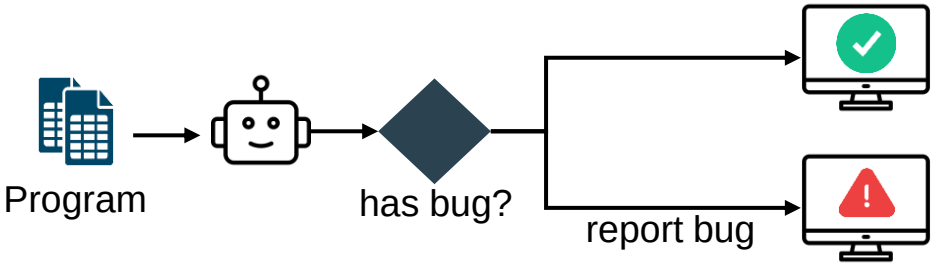
The Power of Types: Exploring the Impact of Type Checking on Neural Bug Detection in Dynamically Typed Languages

Boqi Chen[1], José Antonio Hernández López[2,3], Gunter Mussbacher[1], Dániel Varró[1,2]

[1]Electrical and Computer Engineering, McGill University, Montreal, Canada
[2]Department of Computer and Information Science (IDA), Linköping University, Linköping, Sweden
[3]Department of Computer Science and Systems, University of Murcia, Murcia, Spain

Neural bug detectors (NBDs)

- Neural networks designed for bugs ***beyond the scope*** of traditional detection tools, such as parsers, compilers and type checkers



Variable misuse bugs

- A category of bugs about **misuse of variables**

```
def count_even_odd(numbers):
    evens = 0
    odds = 0
    for num in numbers:
        if num % 2 == 0:
            evens += 1
        else:
            evens += 1
    return evens, odds
```

A logic-related variable misuse error

```
def take_last_assignment(source):
    first=True
    last=None
    for assn in source:
        if first:
            last=assn
            first=False
        if (assn[1]!=first[1]):
            (yield last)
        last=assn
    if (last is not None):
        (yield last)
```

A type-related variable misuse error

programs can be parsed / compiled without issues

Variable misuse bugs

- Logic-related variable misuse bugs

```
def count_even_odd(numbers):
    evens = 0
    odds = 0
    for num in numbers:
        if num % 2 == 0:
            evens += 1
        else:
            evens += 1
    return evens, odds
```

A logic-related variable misuse error

The variable **odd** should be used

Need to understand the function logic to identify the error

An NBD can help in this case

Variable misuse bugs

- The type-related variable misuse bugs

```
def take_last_assignment(source):
    first=True
    last=None
    for assn in source:
        if first:
            last=assn
            first=False
        if (assn[1]!=first[1]):
            (yield last)
        last=assn
    if (last is not None):
        (yield last)
```

A type-related variable misuse error

`first` is defined as a `Boolean` variable

Here is trying to use it as a `list`

It is a **type-related** bug and could been effectively caught by a **type checker**

Variable misuse bugs

- The type-related variable misuse bugs

```
de
    if (last is not None):
        (yield last)
```

A type-related variable misuse error

riable

nd
could been effectively caught by
a **type checker**

These bugs can be identified by type checkers, do we still need NBDs for them?

Research questions

The type-related bugs…

% **Prevalence** in datasets

Research questions

The type-related bugs…

% **Prevalence** in datasets

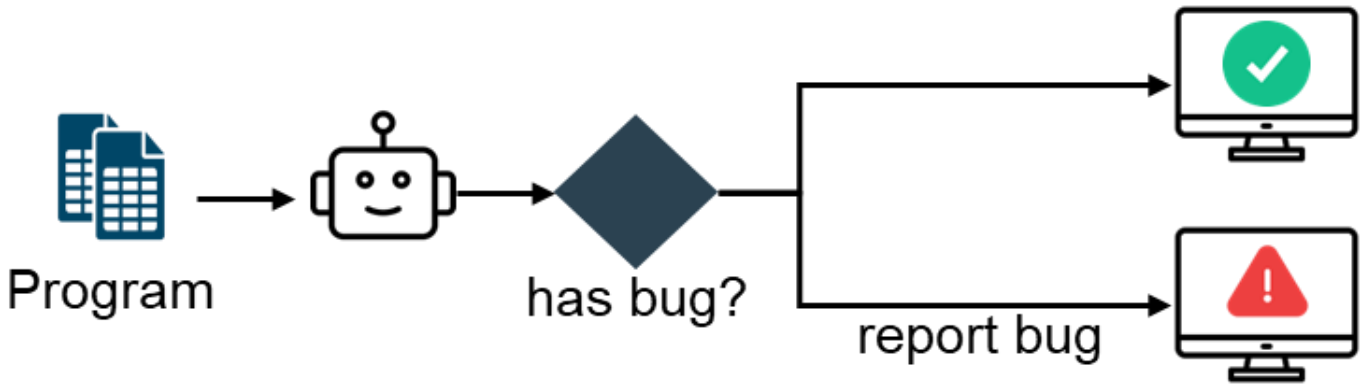
A significant portion of the dataset (5% - 19.56%)

pytype -

: mypy

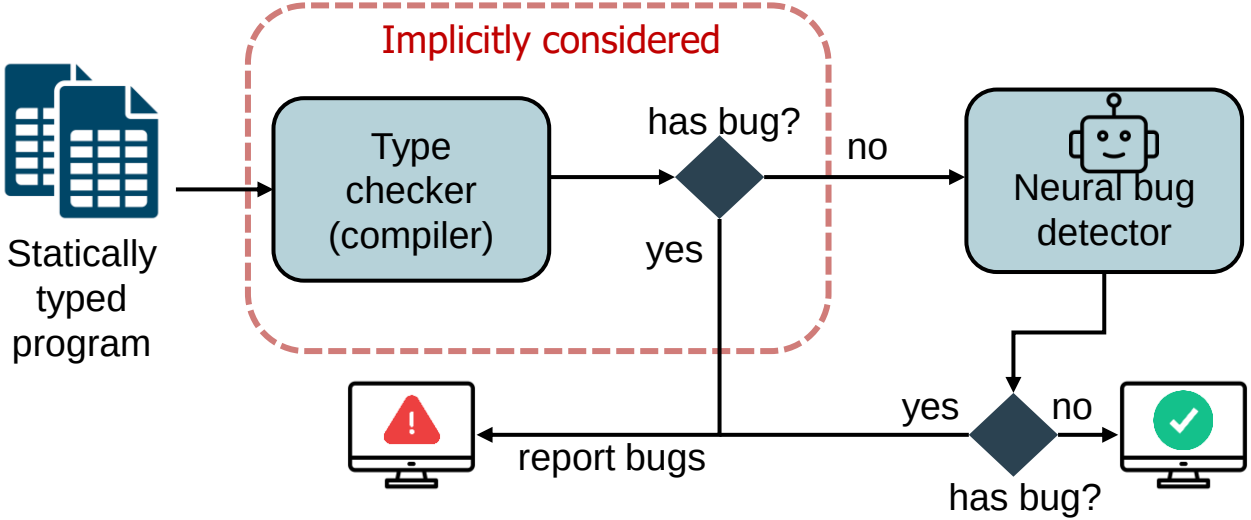
Why do these samples exist in the datasets?

Recall the typical workflow of neural bug detectors



Why do these samples exist in the dataset?

There is one more implicit step...



This is not the case for dynamically typed languages

Research questions

The type-related bugs…

% **Prevalence** in datasets

A significant portion of the dataset (5% - 19.56%)

Use a type checker to identify them?

☑ Impact of **type checkers**

Impact of type checkers

Neural bug detectors only → **`Pytype` + NBD** on Precision (P) and Recall (R)

NBDs	P	R
CodeBERT	↓	↑
GraphCodeBERT	↓	↑
UniXcoder	↓	↑
GGNN	↑	↑
GREAT	↑	↑

Unannotated real-world programs

With explicit type annotation →

NBDs	P	R
CodeBERT	↑	↑
GraphCodeBERT	↑	↑
UniXcoder	↑	↑
GGNN	↑	↑
GREAT	↑	↑

Annotated real-world programs

Integrating type checkers is useful when **recall of bugs** is more relevant, or **explicit type annotation** exists

Research questions

The type-related bugs…

Prevalence in datasets

A significant portion of the dataset (5% - 19.56%)

Impact of **type checkers**

Improving the overall performance

Is the evaluation over-estimated?

Impact on **evaluation**

Influence on evaluation

What if the type-related bugs are removed from the testing datasets?

NBDs	ΔP (%)	ΔR (%)
All NBDs	-5.67 to -10.35 ⬇	-3.55 to -6.16 ⬇

Unannotated real-world programs

With explicit type annotation

NBDs	ΔP (%)	ΔR (%)
All NBDs	-3.18 to -12.83 ⬇	-5.12 to -17.32 ⬇

Annotated real-world programs

Removing type related bugs causes **significant performance drops**

Research questions

The type-related bugs…

Prevalence in datasets

A significant portion of the dataset (5% - 19.56%)

Impact of **type checkers**

Improving the overall performance

Impact on **evaluation**

Filtering them notably decreases the performance

Impact on **training**

Filter these bugs during training?

Research questions

The type-related bugs…

Prevalence in datasets

A significant portion of the dataset (5% - 19.56%)

Impact of **type checkers**

Improving the overall performance

Impact on **evaluation**

Filtering them notably decreases the performance

Impact on **training**

Filtering training dataset slightly improves the performance

% **Prevalence** in datasets

A significant portion of the dataset (5% - 19.56%)

Impact of **type checkers**

Improving the overall performance

Impact on **evaluation**

Filtering them notably decreases the performance

Impact on **training**

Filtering training dataset slightly improves the performance

How to better integrate existing SE tools with NBDs so that we can take advantage of both?

Preprint: