Static Bug Detection for ML Programs

Yiran Wang, Ph.D. Student
Department of Computer Science, Linköping University
yiran.wang@liu.se
WASP, Software Center Project 61

Background

Machine learning (ML) techniques have become widely popular and proven highly effective across various domains. However, developing ML programs can be challenging, often leading to bugs. To enhance the **quality of ML programs**, static analysis serves as a powerful tool for detecting bugs without the need to execute the code. Despite its potential, existing static analyzers struggle to manage statically unknown information related to data and ML libraries.
Jupyter notebooks, a widely used platform for ML prototyping and data analysis, provide a unique opportunity by offering valuable runtime information through their kernel after executing specific cells. To address the limitations of static analyzers, we propose a novel approach — **semi-static analysis** — which leverages runtime information from ML programs to improve the effectiveness of static analysis.

Bugs in ML notebooks

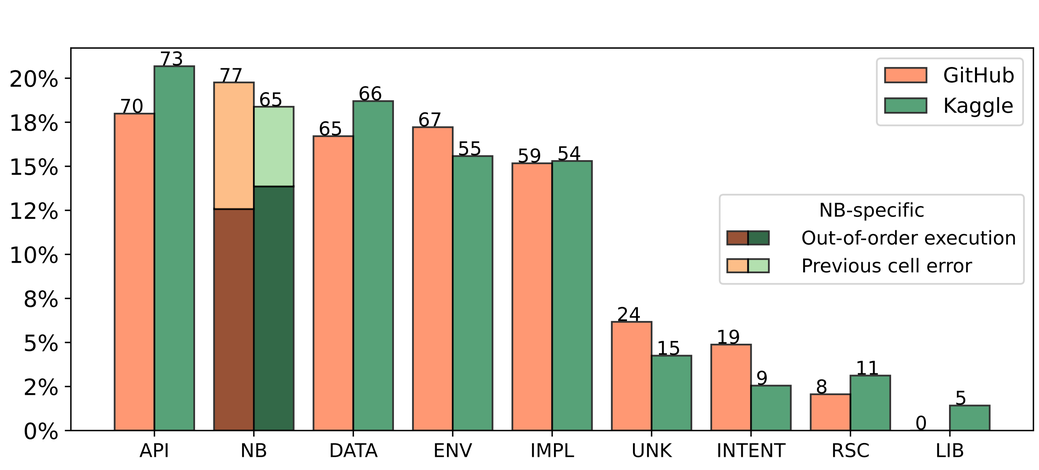
We conduct a comprehensive empirical study [1] to gain deeper insights into bugs in ML programs written in Jupyter Notebooks, focusing on crashes.

Data mining and analysis:

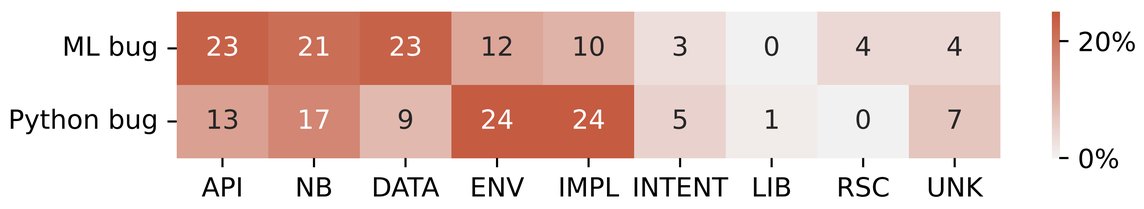
- **Data collection:** 64,031 ML notebooks from GitHub and Kaggle, containing 92,542 crashes/error cell outputs.
- **Manual analysis:** a representative sample of 746 crashes.

Analysis results:

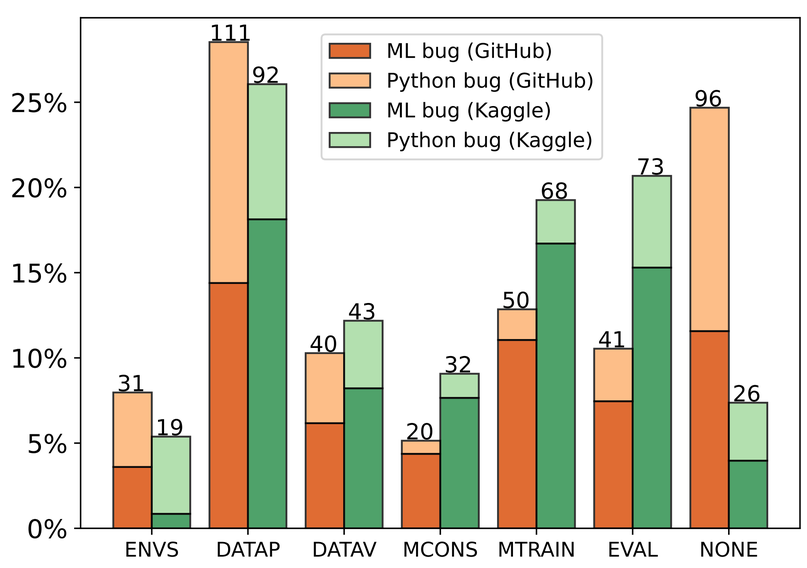
- **Exception types:** `NameError`, `ValueError`, and `TypeError` together account for 53% of all crashes.
- **Root causes:** Top causes include API misuse (19%), notebook-specific issues (19%, most common on Kaggle), and data confusion (18%).



- **ML vs. general Python bugs:** ML-specific bugs dominate, comprising 64% of the total crashes (59% on GitHub and 71% on Kaggle). Comparison of ML and Python bug root causes is shown below:



- **ML pipeline stages:** Crashes are most frequent during the data preparation stage (27%), followed by model training (16%) and evaluation/prediction (15%).



Static bug detection

We investigate our semi-static approach by examining how runtime information can enhance static analysis for bug detection in ML notebooks [2].

1. Load dataset

```
[1]: x = np.loadtxt("data.csv")
     x = tf.constant(x)
```

Run-time info after executing cell 1

x: <tf.Tensor: shape=(2, 3), dtype=int32, numpy= array([[43, 52, 73], [41, 18, 94]])>

2. Data processing

```
[ ]: x.set_shape([3,None])
     ...
```

Bug in cell 2

Error: Dimension 0 in both shapes must be equal, but are 2 and 3.

3. Build ML model
...

Experiment:

- **Targeted ML bug:** Tensor shape mismatch (TensorFlow).
- **Selected tools:** Classic static analyzer PYTHIA [4] and GPT-4 [3].
- **Dataset:** The Unaligned Tensor (UT) dataset [5], containing 14 buggy scripts and 14 fixed scripts *(adjusted to 15 buggy and 13 fixed upon uncovering a hidden bug in one "fixed" script with runtime information)*.

Result:

Analyzers	PYTHIA		GPT-4 (UNION)		GPT-4 (MAJORITY)	
RunInfo	✗	✓	✗	✓	✗	✓
Precision	84.62%	81.25%	75.00%	80.00%	66.67%	72.72%
Recall	73.33%	86.67%	60.00%	80.00%	40.00%	53.33%
F_1 score	78.57%	83.87%	66.67%	80.00%	50.00%	61.54%
FN	4	2	6	3	9	7
FP	2	3	3	3	3	3

Ongoing work

- **Benchmark:** Build a dataset of ML notebook crashes, categorized by bug type, root cause, and ML library, along with a reproducible setup.
- **Tooling:** Develop a bug detector for notebook environments that identifies coding errors before execution by integrating static analysis with LLMs.

Publication

[1] Yiran Wang, José Antonio Hernández López, Ulf Nilsson, Dániel Varró. Using Run-Time Information to Enhance Static Analysis of Machine Learning Code in Notebooks. FSE 2024, Brazil, doi: 10.1145/3663529.3663785

[2] Yiran Wang, Willem Meijer, José Antonio Hernández López, Ulf Nilsson, and Dániel Varró. Why do Machine Learning Notebooks Crash? 2024, arXiv:2411.16795

References

[3] John Schulman et al., ChatGPT: Optimizing Language Models for Dialogue. 2022.
[4] S. Lagouvardos et al., Static analysis of shape in TensorFlow programs. ECOOP 2020.
[5] Yuhao Zhang et al., An empirical study on tensorflow program bugs. ISSTA 2018.