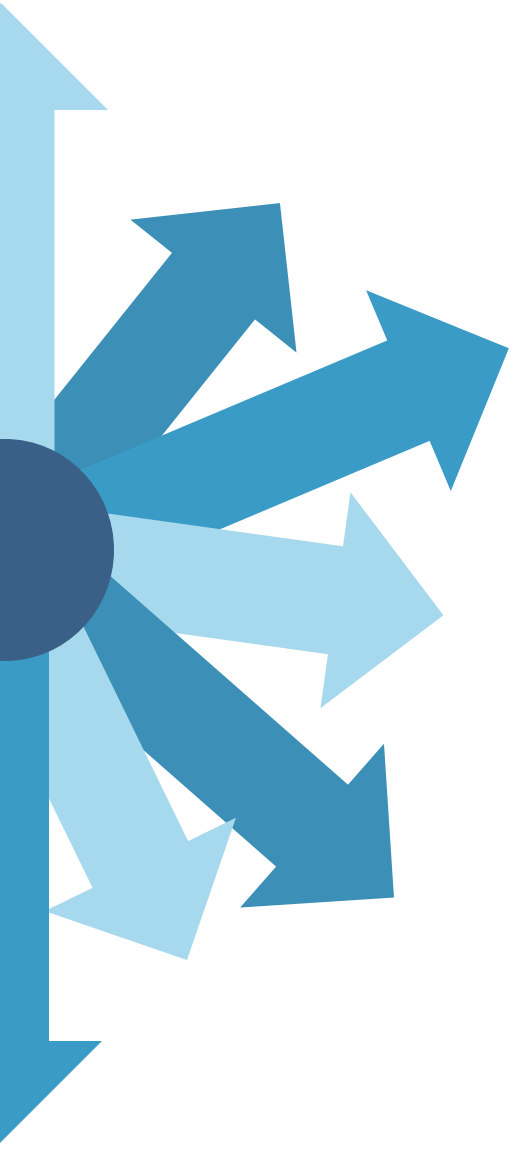


# FlakyXbert

## A Few-Shot Learning Framework for Detecting and Classifying Flaky Tests

**Riddhi More, Jeremy S. Bradbury**

Software Engineering & Education Research Lab
Ontario Tech University, Oshawa, Canada

http://www.seerlab.ca

SEER LAB | OntarioTech UNIVERSITY

# LLMs for Software

- Task automation
- Defect Prediction
- Code review
- Bug fixing
- Test Case generation
- Etc..

# LLMs for Software

- Task automation
- Defect Prediction
- Code review
- Bug fixing
- Test Case generation
- Etc..

- Computational resources?
- Sustainability?
- Carbon footprint?
- Reproducibility?
- Etc..

# Publications*

- Riddhi More, Jeremy S. Bradbury. **"An Analysis of LLM Fine-Tuning and Few-Shot Learning for Flaky Test Detection and Classification."** *Proc. of the International Conference on Software Testing, Verification and Validation (ICST 2025)*, Napoli, Italy, Mar./Apr. 2025. *[to appear]*
- Riddhi More, Jeremy S. Bradbury. **"Assessing Data Augmentation-Induced Bias in Training and Testing of Machine Learning Models."** *Proc. of the 1st International Workshop on Fairness in Software Systems (FAIRNESS 2025)*, Montreal, PQ, Canada, Mar. 2025. *[to appear]*

*preprints available for download at https://www.seerlab.ca

# What are Flaky Tests?

*Flaky tests exhibit **non-deterministic** behavior during execution, and they may **pass** or **fail** without any changes to the program under test.*

```
const assert = require('assert');

async function fetchData() {
    // Simulating a network request with a random delay
    const delay = Math.random() * 1000;
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve('data');
        }, delay);
    });
}

describe('Async Test', function() {
    it('should return data within 500ms', async function() {
        const startTime = Date.now();

        const data = await fetchData();

        const endTime = Date.now();
        const duration = endTime - startTime;

        // Flaky assertion: test may fail if fetchData() takes longer than 500ms
        assert(duration < 500, `Test failed due to timeout. Duration: ${duration}
        assert.strictEqual(data, 'data');
    });
});
```

***Random delay in fetchData()****: The fetchData() function simulates a network request with a random delay up to 1000 milliseconds. The test checks that the response is received within 500 milliseconds, but this is inconsistent since the delay could be longer than 500 ms, making the test pass or fail depending on the random timing.*

# A Survey of Flaky Tests

OWAIN PARRY, University of Sheffield, UK
GREGORY M. KAPFHAMMER, Allegheny College, USA
MICHAEL HILTON, Carnegie Mellon University, USA
PHIL MCMINN, University of Sheffield, UK

Tests that fail inconsistently, without changes to the code under test, are described as *flaky*. Flaky tests do not give a clear indication of the presence of software bugs and thus limit the reliability of the test suites that contain them. A recent survey of software developers found that 59% claimed to deal with flaky tests on a monthly, weekly, or daily basis. As well as being detrimental to developers, flaky tests have also been shown to limit the applicability of useful techniques in software testing research. In general, one can think of flaky tests as being a threat to the validity of any methodology that assumes the outcome of a test only depends on the source code it covers. In this article, we systematically survey the body of literature relevant to flaky test research, amounting to 76 papers. We split our analysis into four parts: addressing the causes of flaky tests, their costs and consequences, detection strategies, and approaches for their mitigation and repair. Our findings and their implications have consequences for how the software-testing community deals with test flakiness, pertinent to practitioners and of interest to those wanting to familiarize themselves with the research area.

CCS Concepts: • **Software and its engineering** → *Software testing and debugging*;

Additional Key Words and Phrases: Flaky tests, software testing

**ACM Reference format:**
Owain Parry, Gregory M. Kapfhammer, Michael Hilton, and Phil McMinn. 2021. A Survey of Flaky Tests. *ACM Trans. Softw. Eng. Methodol.* 31, 1, Article 17 (October 2021), 74 pages.
https://doi.org/10.1145/3476105

## A Survey of Flaky Tests

OWAIN PARRY, University of Sheffield, UK
GREGORY M. KAPFHAMMER, Allegheny College, USA
MICHAEL HILTON, Carnegie Mellon University, USA
PHIL MCMINN, University of Sheffield, UK

Tests that fail inconsistently, without changes to the code under test, are described as *flaky*. Flaky tests do not give a clear indication of the presence of software bugs and thus limit the reliability of the test suites that contain them. A recent survey of software developers found that 59% claimed to deal with flaky tests on a monthly, weekly, or daily basis. As well as being detrimental to developers, flaky tests have also been shown to limit the applicability of useful techniques in software testing research. In general, one can think of flaky tests as being a threat to the validity of any methodology that assumes the outcome of a test only depends on the source code it covers. In this article, we systematically survey the body of literature relevant to flaky test research, amounting to 76 papers. We split our analysis into four parts: addressing the causes of flaky tests, their costs and consequences, detection strategies, and approaches for their mitigation and repair. Our findings and their implications have consequences for how the software-testing community deals with test flakiness, pertinent to practitioners and of interest to those wanting to familiarize themselves with the research area.

CCS Concepts: • **Software and its engineering** → *Software testing and debugging*;

Additional Key Words and Phrases: Flaky tests, software testing

**ACM Reference format:**
Owain Parry, Gregory M. Kapfhammer, Michael Hilton, and Phil McMinn. 2021. A Survey of Flaky Tests. *ACM Trans. Softw. Eng. Methodol.* 31, 1, Article 17 (October 2021), 74 pages.
https://doi.org/10.1145/3476105

- By identifying and isolating flaky tests, developers can focus on genuine test failures, streamlining the debugging process and reducing false alarms.
- Categorizing flaky tests can help in understanding their nature and root causes.
- Ignoring flaky test failures can significantly impact software stability.

# Datasets

- International Dataset of Flaky Tests (IDoFT) [1]
  - Flaky (3195) & non-flaky tests (618)
  - Tests organized by open-source projects
- FlakyCat Dataset [2]
  - Only flaky tests (369)
  - Diverse data from multiple open source projects
  - Contains augmented data

TABLE I: IDoFT and FlakyCat datasets

| **IDoFT - Flaky vs. Non-Flaky** | **# Tests** |
|---|---|
| Flaky tests | 3195 |
| Non-Flaky tests | 618 |
| **Total** | **3813** |
| **IDoFT - Flaky Test Categories** | |
| Non-deterministic-order-dependent (NDOD) | 84 |
| Non-order-dependent (NOD) | 226 |
| Order-dependent ()OD) | 932 |
| Non-idempotent-outcome (NIO) | 196 |
| Implementation-dependent (ID) | 1617 |
| Unknown-dependency (UD) | 140 |
| **Total** | **3195** |
| **FlakyCat - Flaky Test Categories** | |
| Async wait (Asyn.) | 125 |
| Concurrency (Conc.) | 48 |
| Time | 42 |
| Test Order Dependency (OD) | 103 |
| Unordered Collections (UC) | 51 |
| **Total** | **369** |

**[1]** "International dataset of flaky tests (IDoFT)," https://github.com/TestingResearchIllinois/idoft.
**[2]** A. Akli, G. Haben, S. Habchi, M. Papadakis, and Y. Le Traon, "Flakycat: predicting flaky tests categories using few-shot learning," in 2023 IEEE/ACM Int. Conf. on Automation of Software Test (AST 2023), pp. 140–151.

Our research is motivated by the need for an improved understanding of the trade-offs between ***fine-tuning*** and ***few-shot learning (FSL)*** techniques in addressing flaky test challenges.

# Fine-tuning vs FSL

**Fine-Tuning:**

- Adapts pre-trained models (e.g., BERT, GPT) to task-specific datasets.
- Requires large labeled datasets for precision.
- Enhances model accuracy and robustness but is resource-intensive.

# Fine-tuning vs FSL

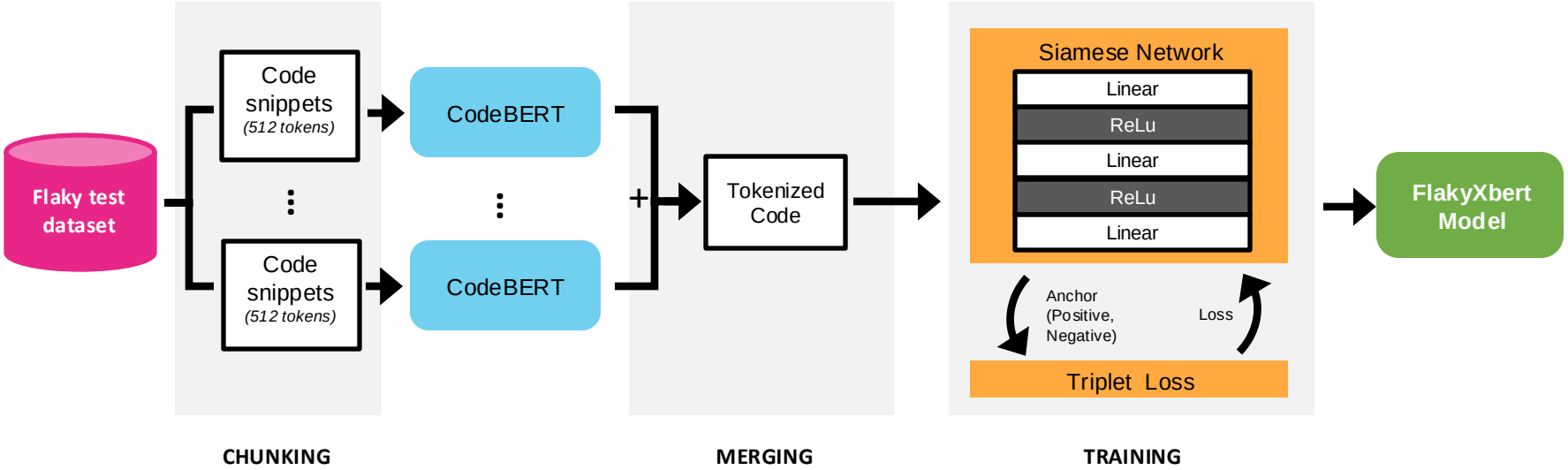
**Fine-Tuning:**

- Adapts pre-trained models (e.g., BERT, GPT) to task-specific datasets.
- Requires large labeled datasets for precision.
- Enhances model accuracy and robustness but is resource-intensive.

**Few-Shot Learning (FSL):**

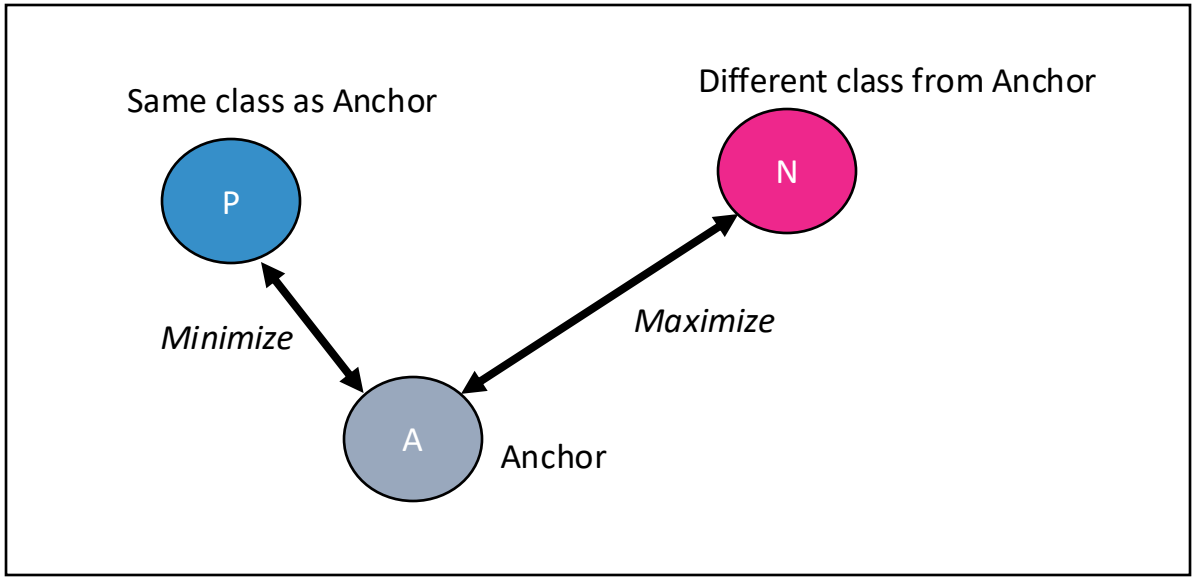
- Works with minimal data.
- Ideal for data-constrained environments or rapid deployment.
- Faster and more flexible, aligning with agile development practices.

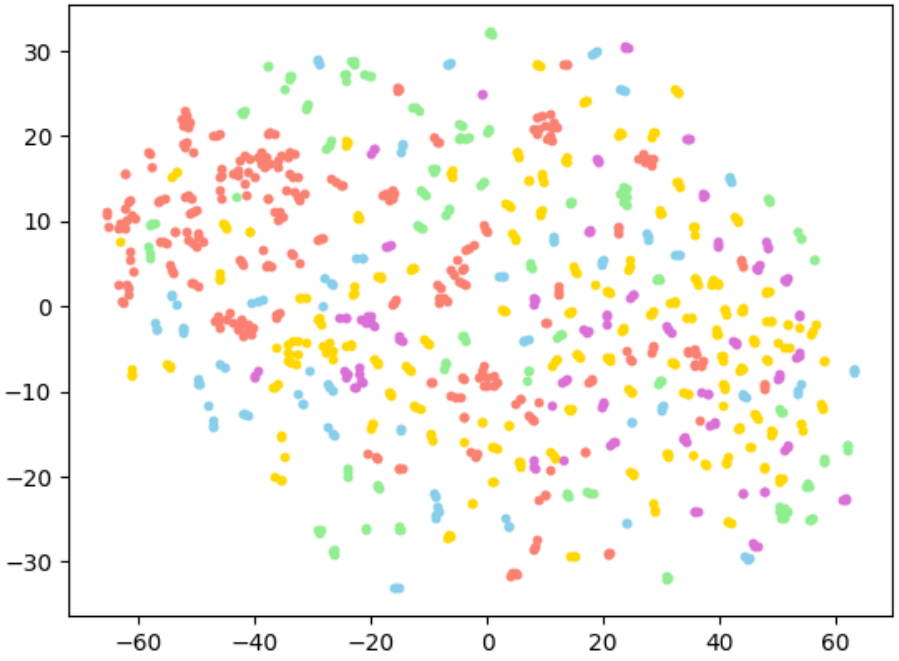
# FlakyXbert: Architecture



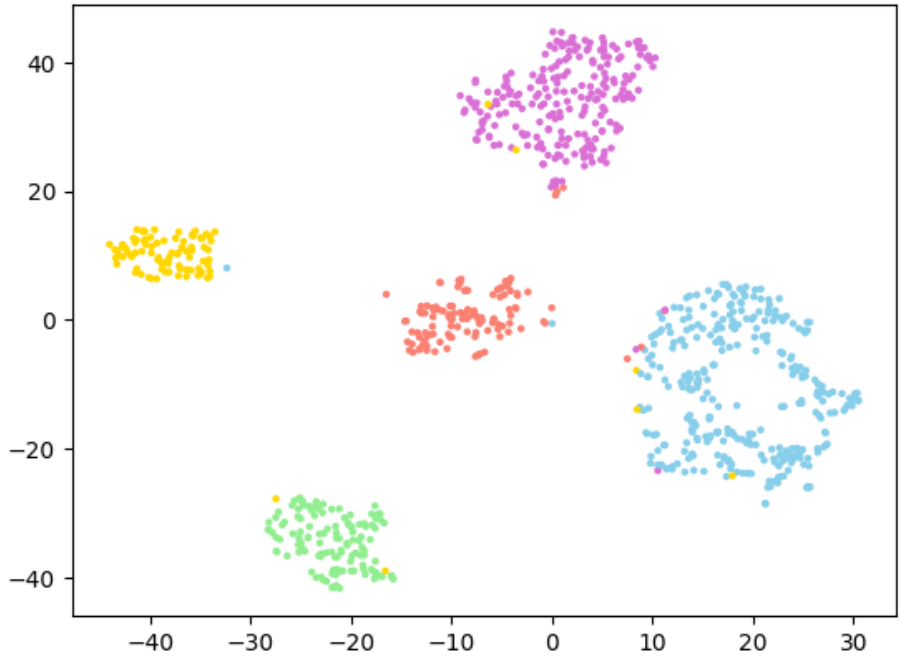
$$\mathrm{TL} = \max(\|f(a) - f(p)\|^2 - \|f(a) - f(n)\|^2 + \mathrm{margin}, 0)$$

Triplet Loss function






Before FlakyXbert




After FlakyXbert

# Research Questions

- **RQ1:** How does the performance of FSL and fine-tuning compare for flaky test detection and classification across different data scenarios?
  - **RQ1.1:** What is the performance of FSL compared to fine-tuning on small per-project data? (IDoFT)
  - **RQ1.2:** What is the performance of FSL compared to fine-tuning with a diverse data set? (FlakyCat)

- **RQ2:** What is the cost of FSL vs. fine-tuning?

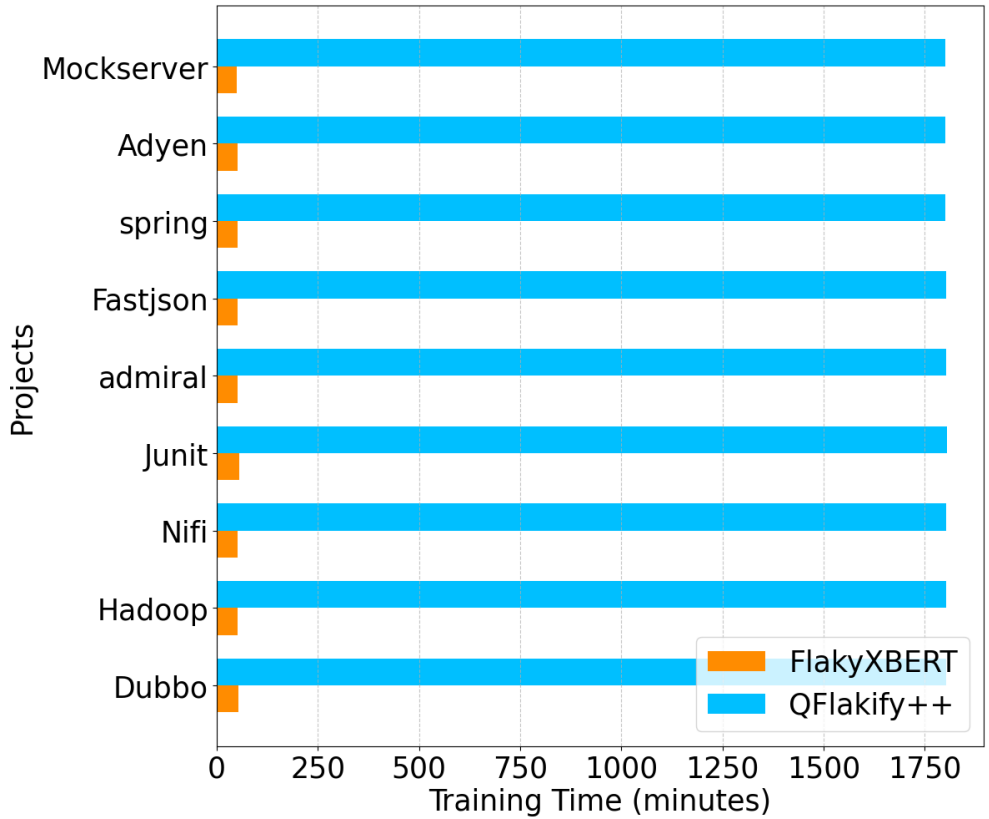
# Results: IDoFT Detection – F1-Score

| Project | Support | FlakyXbert | Flakify++ | Q-Flakify++ | FlakyQ_RF |
|---|---|---|---|---|---|
| Dubbo | 186 | 88.7 | 87.0 | 91.0 | 88.0 |
| Hadoop | 149 | 95.0 | 99.0 | 100.0 | 100.0 |
| Nifi | 146 | 91.5 | 99.0 | 100.0 | 100.0 |
| Junit | 250 | 94.0 | 99.0 | 99.0 | 99.0 |
| Admiral | 113 | 91.3 | 99.0 | 99.0 | 99.0 |
| Fastjson | 109 | 91.3 | 91.0 | 93.0 | 93.0 |
| spring | 68 | 100.0 | 100.0 | 100.0 | 100.0 |
| Adyen | 89 | 30.0 | 43.0 | 52.0 | 45.0 |
| Mockserver | 39 | 100.0 | 100.0 | 100.0 | 100.0 |
| Total/ Weighted Avg. | 2105 | 95.1 | 95.6 | 96.0 | 95.4 |

*Note: To see the full version, please refer to the original paper [2].*

**[2]** S. Rahman, A. Baz, S. Misailovic, and A. Shi, "Quantizing large- language models for predicting flaky tests," in *Proc. of the 17th IEEE Int. Conf. on Software Testing, Verification and Validation (ICST 2024)*.

# Results: IDoFT Detection – Training Time



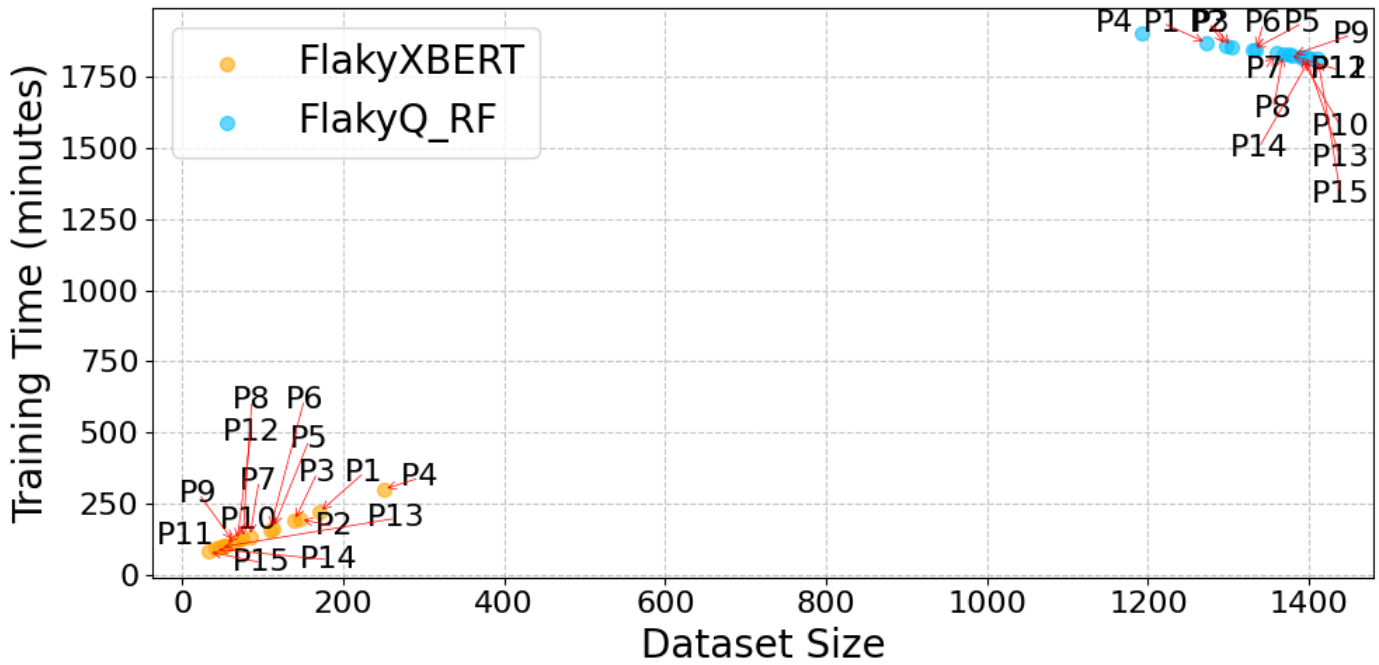
# Results: IDoFT Classification – F1-Score

| Project | Support | FlakyXbert | Flakify++ | Q-Flakify++ | FlakyQ_RF |
|---|---|---|---|---|---|
| Dubbo | 170 | 71.0 | 77.0 | 77.0 | 73.0 |
| Hadoop | 146 | 51.0 | 90.0 | 88.0 | 91.0 |
| Nifi | 139 | 91.0 | 100.0 | 100.0 | 100.0 |
| Junit | 250 | 94.0 | 98.0 | 98.0 | 98.0 |
| Ormlite | 113 | 96.0 | 99.0 | 97.0 | 97.0 |
| admiral | 109 | 63.0 | 85.0 | 77.0 | 88.0 |
| Wildfly | 84 | 74.0 | 97.0 | 98.0 | 98.0 |
| Mapper | 75 | 100.0 | 93.0 | 80.0 | 100.0 |
| Fastjson | 64 | 82.0 | 91.0 | 88.0 | 94.0 |
| Java | 54 | 85.0 | 87.0 | 87.0 | 87.0 |
| Biojava | 51 | 91.0 | 19.0 | 16.0 | 32.0 |
| spring | 68 | 90.0 | 100.0 | 100.0 | 100.0 |
| Hbase | 47 | 76.0 | 98.0 | 95.0 | 98.0 |
| hive | 41 | 100.0 | 98.0 | 96.0 | 98.0 |
| Nacos | 32 | 96.0 | 100.0 | 97.0 | 97.0 |
| Total/ Weighted Avg. | 1810 | 76.5 | 90.2 | 93.0 | 94.8 |

*Note: To see the full version, please refer to the original paper [2]*

**[2]** S. Rahman, A. Baz, S. Misailovic, and A. Shi, "Quantizing large- language models for predicting flaky tests," in *Proc. of the 17th IEEE Int. Conf. on Software Testing, Verification and Validation (ICST 2024).*

# Results: IDoFT Classification – Training Time vs. Dataset Size

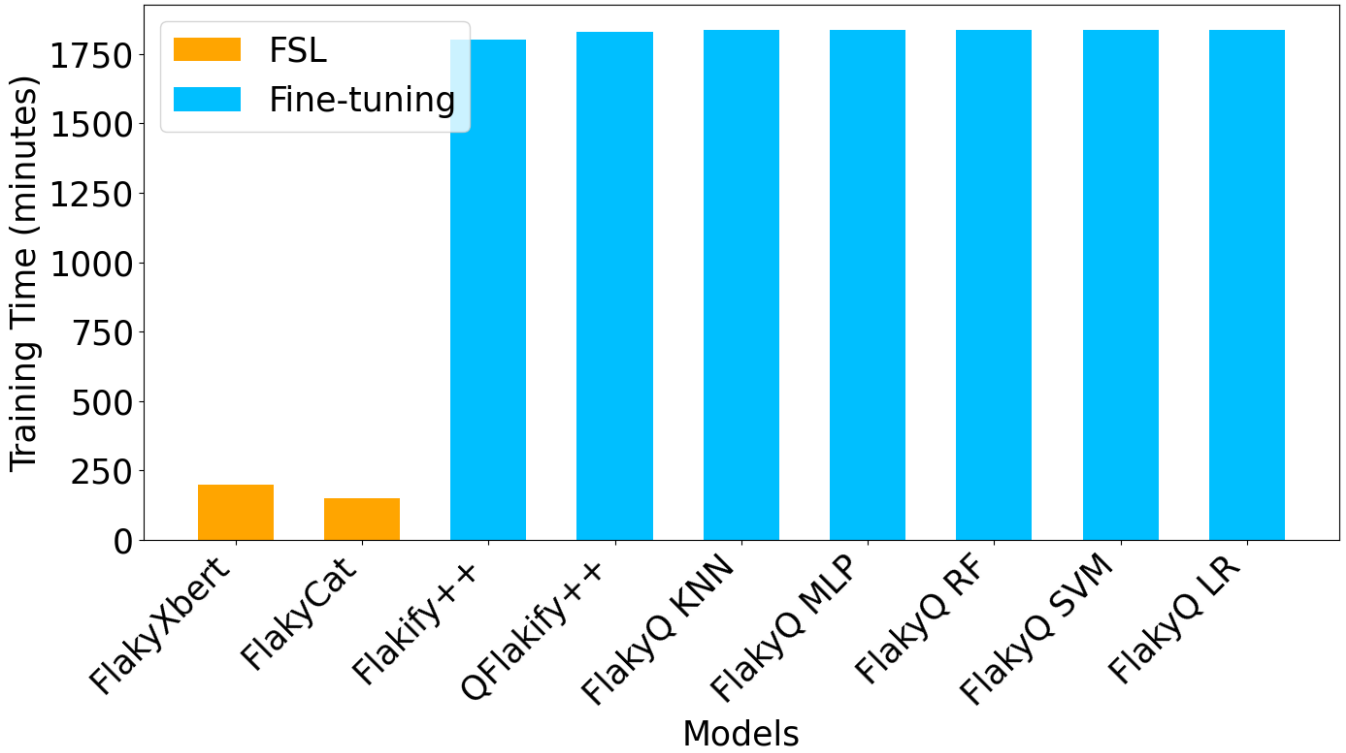


# Results: FlakyCat Classification – F1-Score

| Technique | Classifier | Asyn. | Conc. | Time | UC | OD | Weighted Avg. |
|---|---|---|---|---|---|---|---|
| Few-shot Learning (FSL) | FlakyXbert | 98.0 | 90.0 | 93.0 | 97.0 | 99.0 | 96.0 |
| | FlakyXbert (without augmentation) | 52.0 | 80.0 | 36.0 | 43.0 | 78.0 | 60.0 |
| | FlakyCat | 72.0 | 36.0 | 75.0 | 72.0 | 73.0 | 67.5 |
| Fine-tuning (FT) | Flakify++ | 94.8 | 93.3 | 96.9 | 96.1 | 97.1 | 95.6 |
| | Q-Flakify++ | 92.6 | 87.1 | 96.9 | 95.0 | 95.8 | 93.6 |
| | FlakyQ_KNN | 93.1 | 90.7 | 95.5 | 95.0 | 96.3 | 94.2 |
| | FlakyQ_MLP | 94.0 | 89.7 | 95.5 | 94.8 | 96.6 | 94.5 |
| | FlakyQ_RF | 94.3 | 91.5 | 95.5 | 94.8 | 96.6 | 94.8 |
| | FlakyQ_SVM | 93.8 | 89.0 | 95.5 | 93.2 | 96.1 | 93.9 |
| | FlakyQ_LR | 92.7 | 89.8 | 95.5 | 94.8 | 96.1 | 93.9 |
| Hybrid (FSL + FT) | FSL++ | 93.7 | 90.3 | 97.9 | 95.9 | 96.7 | 91.5 |

*Note: Asyn. = Async Wait, Conc. = Concurrency, Time = Test Order Dependency, UC = Unordered Collections, OD = Other Dependencies*

# Results: FlakyCat Classification – Training Time



# Conclusion

***FSL (Few-Shot Learning)*** in the FlakyXbert model is effective in environments with sparse data.

- It leverages fewer examples to generalize well across flakiness categories.

***Fine-tuning***, which requires more extensive data, excels by adapting to a broader range of features.

- It handles diversity better and typically offers more accurate predictions.
- It demands greater computational resources and longer training time.

Using ***FSL with augmentation*** provided better results than Fine-Tuning.

- However, these results might not reflect general trends.

# Conclusion

- The choice between ***FSL*** and ***Fine-tuning*** depends on balancing trade-offs between:
  - Data availability.
  - Computational efficiency.
  - Adaptability to diverse flaky test characteristics.

*Threats to Validity - we used publicly available datasets and consistent evaluation metrics across models, addressing class imbalances with augmentation. While further tuning could improve performance, variability in results across projects and uncertainty in generalizing our findings remain. More experimentation is needed to confirm broader applicability beyond flaky test detection.*

# Data Augmentation

*"...involves enhancing the sufficiency and diversity of training examples without explicitly collecting new data... The essence of data augmentation lies in generating new data by altering existing data points through various transformations"* [1].

**[1]** Y. Zhou, C. Guo, X. Wang, Y. Chang, and Y. Wu, "A survey on data augmentation in large model era," arXiv preprint:2401.15422, 2024.

# Data Augmentation in FlakyCat Dataset

- To avoid self-confirmatory bias in our case study, we rely on an existing augmentation technique that was applied to the **FlakyCat** data set by a third-party **[2]**.
- For each original tests case (**v0**) there are two augmented versions (**v1, v2**) created using an adapted Synthetic Minority Over-sampling Technique (SMOTE) that involved generating variants through controlled mutations of non-flakiness-related elements.

[2] A. Akli, G. Haben, S. Habchi, M. Papadakis, and Y. Le Traon, “FlakyCat: Predicting flaky tests categories using few-shot learning,” in Proc.of IEEE/ACM International Conference on Automation of Software Test (AST 2023), May 2023, pp. 140–151.

Our analysis of data augmentation is motivated by a desire to understand any potential **bias** considerations.

# Research Questions

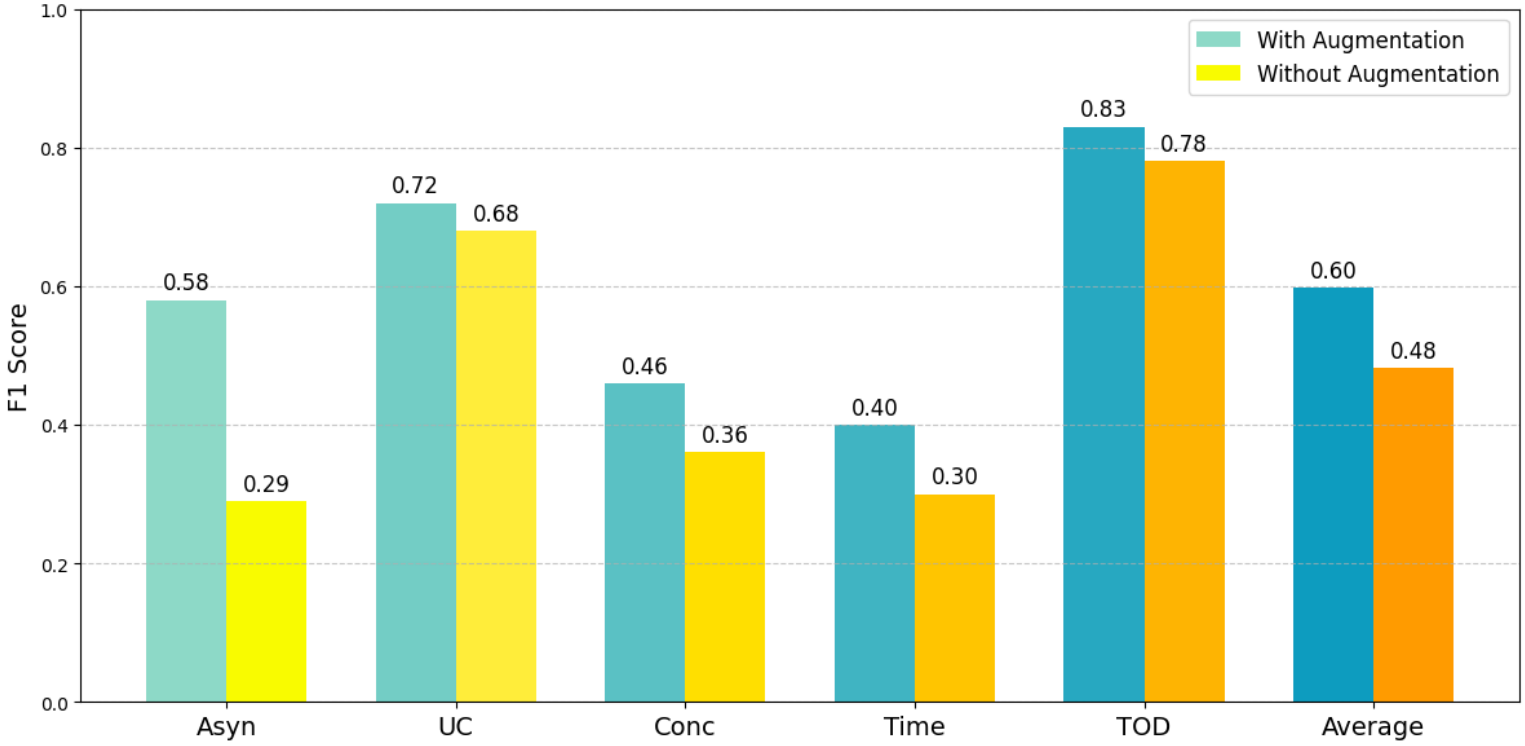
- **RQ1:** What is the impact of data augmentation in flaky test classification (FlakyXbert) with the FlakyCat dataset?

- **RQ2:** What is the bias of data augmentation in flaky test classification (FlakyXbert) with the FlakyCat dataset?

# Experiment 1: Design

- **RQ1: What is the impact of data augmentation in flaky test classification (FlakyXbert) with the FlakyCat dataset?**
- **Phase A (baseline):** Using only original test cases divide the test cases into training and testing sets.
- **Phase B (augmentation):** Start with the same original test cases divided into the training and testing sets. For each test in the training data also add corresponding augmented tests and for each test in the testing data add corresponding augmented tests as well.
- This ensures a more accurate evaluation of the model's ability to generalize to unseen cases and assess whether it learns meaningful patterns rather than artifacts of augmentation.

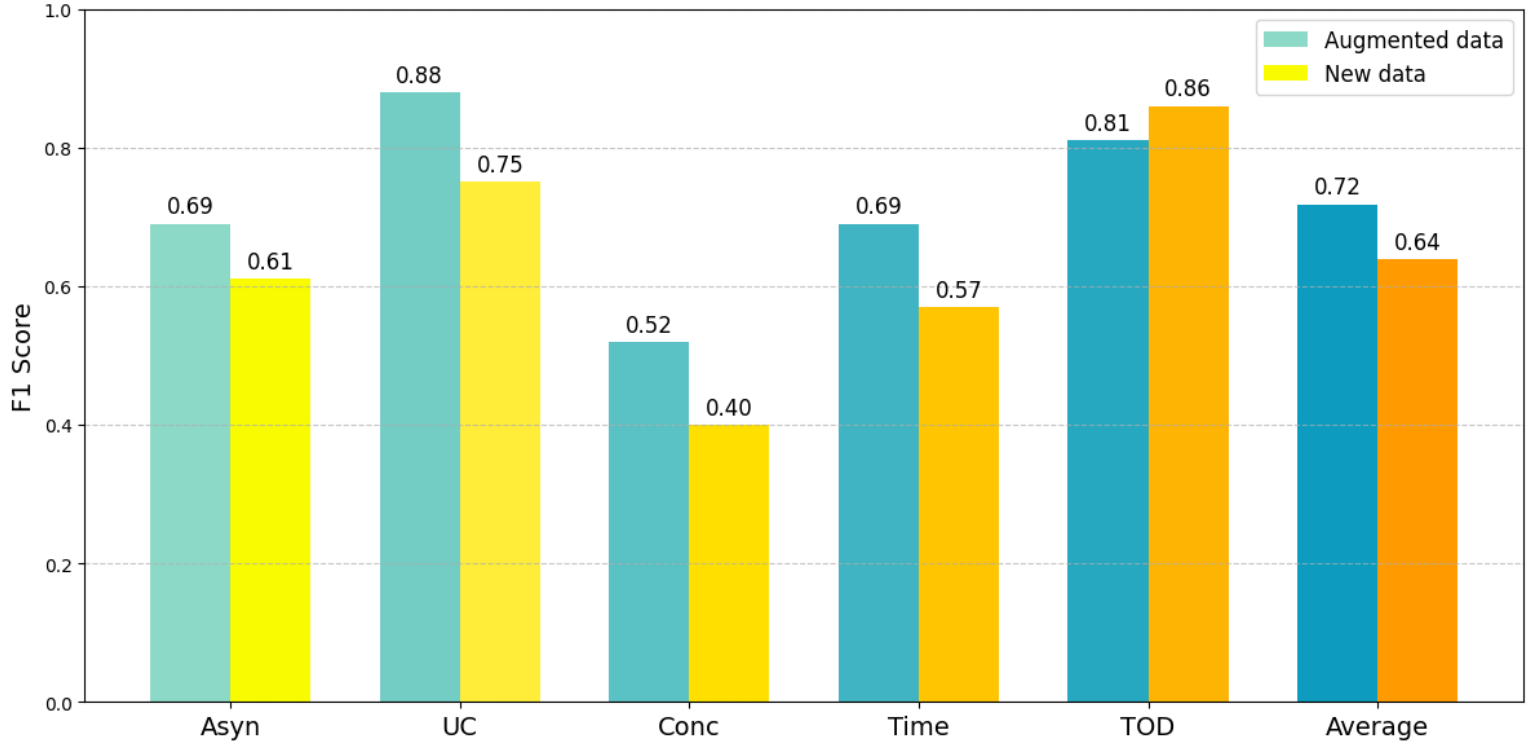
# Experiment 1: Results



# Experiment 2: Design

- **RQ2: What is the bias of data augmentation in flaky test classification (FlakyXbert) with the FlakyCat dataset?**
- **Training:** The model is trained only on original test cases.
- **Testing:** The model is evaluated in two contexts:
  - On a test set of unseen original test cases.
  - On a test set of test cases that are augmentations of the training test cases.
- This design examines whether the model performs differently on augmented variants of familiar cases versus new cases, providing insight into its ability to generalize flaky test characteristics or its tendency to overfit augmentation patterns.

# Experiment 2: Results

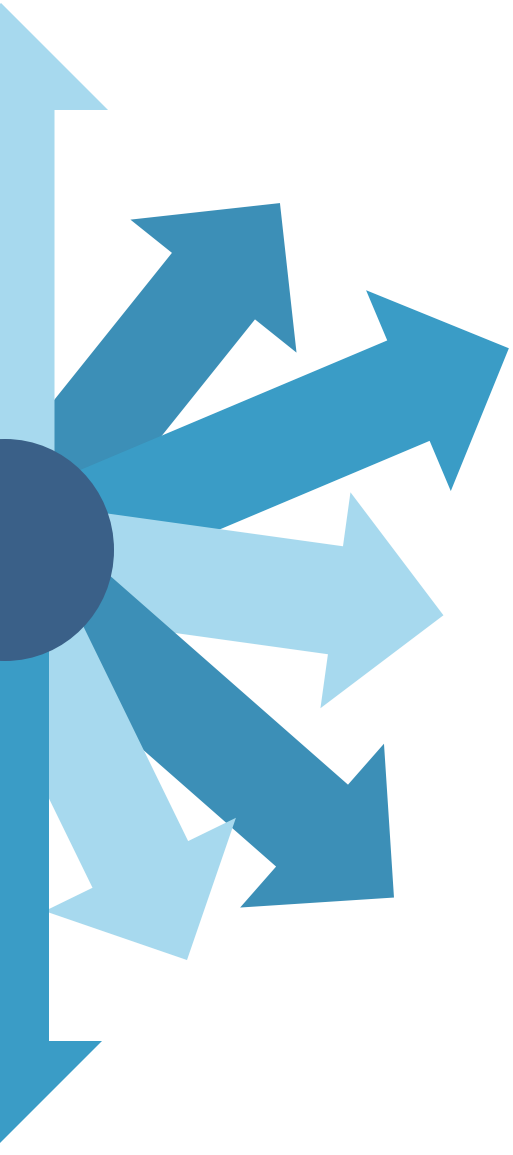


# Preliminary Recommendations

- The systematic performance gap between augmented and new test cases (8% average F1 score difference) suggests **there is a need to maintain a completely separate validation set of original, non-augmented data during model evaluation and testing.**
- The varying effectiveness of augmentation across different flaky test categories indicates **there is a potential benefit of category-specific augmentation strategies rather than a one-size-fits-all approach to data augmentation.**
- Bias in data augmentation likely extend beyond flaky test detection. **Researchers in other areas of SE may benefit from assessing bias in augmented data as well as strategies to reduce any identified bias.**

# Conclusions

- Our experiments reveal both **benefits** and **biases** from using the augmented FlakyCat dataset
- Our findings highlight the need for **careful use of augmented data** and **robust evaluations to detect biases** and ensure results reflect real-world model performance.



# FlakyXbert

## A Few-Shot Learning Framework for Detecting and Classifying Flaky Tests

**Riddhi More, Jeremy S. Bradbury**

Software Engineering & Education Research Lab
Ontario Tech University, Oshawa, Canada

http://www.seerlab.ca

SEER LAB | OntarioTech UNIVERSITY