Research Statement | Sumon Biswas

Engineering AI-Enabled Software for Fairness and Safety

> *A wide array of attributes such as robustness, accuracy, fairness, explainability, and privacy [of AI systems], presents a complicated set of challenges... System designers, regulators, policymakers [...] will have to devise a set of tools to handle them.*
>
> *National Academies of Sciences, Engineering, and Medicine (NASEM) Workshop, Washington DC, 2021* [1]

The rapid increase of artificial intelligence (AI)-enabled software is introducing significant ethical and safety risks to human lives and society. My research at the intersection of **Software Engineering (SE) and AI** aims at modeling, verifying, and mitigating such risks in different stages of machine learning pipeline, e.g., preprocessing, deployment. I adopted both formal and empirical approaches to ensure algorithmic fairness and safety of AI-enabled systems, through rigorous analysis of the software abstractions and their real-world implementations.

Despite the transformational progress, ML software often violates critical fairness and safety properties. Unlike traditional software, most of the ML models are under-specified, i.e., the behaviors lack formal characterizations and are evaluated based on labeled examples. Furthermore, the decision logic of the ML models are learned from large datasets as opposed to explicit instructions, which leads to opaque and uncertain software systems. Consequently, software engineers need novel techniques and tools to understand the new challenges and mitigate them. My research focuses on identifying such issues through modeling and analysis of the ML components, and designing SE techniques for mitigation. In particular, I conceived **algorithmic fairness** as an SE problem and localized biases, i.e., discrimination of people based on protected attributes, e.g., race, ethnicity, sex, age, etc. Recognizing the salient problem of fairness-accuracy trade-off, I developed bias mitigation strategy that preserves performance. While these techniques made significant progress in engineering ML software for fairness, they can not provide guarantees. To address this, I built fairness verification technique that can certify properties in critical systems such as sentencing, lending, recruiting, etc.

AI-enabled systems also evolve in unanticipated ways and cause long-term fairness and safety violations. For instance, the decisions of an AI system (e.g., predictive policing) can bring certain changes to the environment, which, in turn, can push the system towards an unsafe state over time. I built a simulation-based framework for long-term analyses of AI-enabled systems, which facilitates developers to identify impactful design choices and build interventions proactively. Besides ML models and systems, I also focused on improving the ML process and programming paradigm, which contribute to the overall software quality. I built an infrastructure to mine AI-based open-source repositories (millions of lines of code and metadata), that enabled studying the data science software architecture *in-the-small* (e.g., Jupyter notebooks), and *in-the-large* (e.g., GitHub projects). Leveraging the infrastructure, I identified a new set of technical debts (TD) in ML programs, and how the widespread use of large language models (LLM) impact the TD maintenance, especially debt repayment and reproduction. In summary, my works focus on the following directions in the **SE for AI** area:

1. **Fairness verification and reasoning:** I developed the first SMT-based *individual fairness* verification technique for neural networks (ICSE'23 [1]). I also applied causal reasoning to build the first component-level fairness evaluation in ML pipeline (FSE'21 [2]), and demonstrated fairness composition in ensemble learning (ICSE'23 [3]).
2. **Designing fair and safe AI systems:** I conducted a large-scale study on fairness engineering (FSE'20 [4]) and designed state-of-the-art mitigation strategy to achieve fairness-accuracy trade-off (FSE'23 [5]). In addition, based on my envisioned systemic analysis framework (SE4SafeML@FSE'23 [6]), I proposed a Monte-Carlo simulation technique to analyze long-term impact of AI system configurations (submitted [7]).
3. **AI engineering and static analysis:** For large-scale mining and static program analysis of AI software, I built an infrastructure using *Boa* (MSR'19 [8, 9]), which I leveraged to study the AI pipelines (ICSE'22 [10]). I also identified various technical debts in open-source ML (FSE'22 [11]), and their impacts on LLM (ICSE'24 [12]).

I am currently working on the DARPA project, *Verified Security and Performance Enhancement of Large Legacy Software (V-SPELLS)*. Specifically, I am designing model-based engineering techniques to ensure safety of learning-enabled Unmanned Aerial Vehicles (UAV). My prior works also contributed to the NSF TRIPODS proposal that received institute-grant called *D4: Dependable Data-Driven Discovery*. I aim to achieve the overarching goal of removing societal discrimination and safety risks by improving AI software development. Many opportunities remain from eliciting correct requirements to ensuring the properties in production. I plan to build a *correct-by-construction* framework so that software engineers can provide guarantees. I would further extend my focus on responsible AI engineering and explore the synergy between the quality attributes. Finally, I intend to build sustainable design of AI systems as a crosscutting attribute so that the qualities endure over time (Section 4).

[1] NASEM workshop proceedings on Assessing and Improving AI Trustworthiness: `https://nap.nationalacademies.org/catalog/26208`

1 Ensuring Fairness through Formal Verification and Modular Reasoning

AI-enabled software are increasingly used in high-stake applications that impact human lives directly such as loan approval, criminal sentencing, hiring employees, etc. Several incidents of discrimination against certain race, sex, age, etc., have highlighted the need for ensuring fairness of ML models. However, the existing testing and mitigation techniques lack formal guarantees. The main barrier to providing such guarantees is the complexity and size of the ML models, e.g., neural networks. I utilized modular reasoning and analysis to certify the models using two key ideas: 1) Modularize the specification by input space partitioning, relaxation of fairness constraints, and weakest precondition computation. 2) Perform white-box analysis on the ML model. As shown in Figure 1, I adopted a modular approach to verify neural networks, and identify unfair components in ML pipelines and ensemble models [1, 2, 3]. In addition to the formal guarantee, this approach also brings practical benefits to the developers such as deploying verified partitions of the model and conduct targeted repair on the remaining ones.

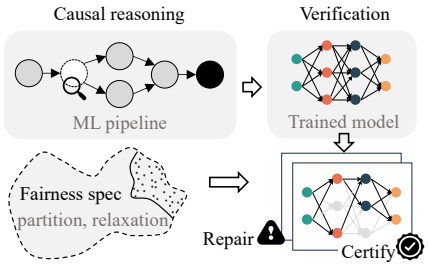

Figure 1: Modular reasoning and verification

Fairness Verification in Neural Networks. I built the first SMT-based individual fairness verification technique, *Fairify*, for neural networks [1]. The **individual fairness** property ensures that any two individuals with *similar* attributes except the protected attribute(s), should receive the same predictions. Formally, for any two individuals x and x', if $d(x, x') \leq \epsilon$, then $f(x) = f(x')$, where d is a distance metric, ϵ is a threshold, and f is the classifier. Verifying the property is harder than the adversarial robustness since it requires *global* safety, for any $\langle x, x' \rangle$. I extracted the pre- and post-conditions for the input features, transferred them to neurons using weakest precondition semantics, and then leveraged the SMT solver to verify. The verification has been made tractable in three steps: 1) Input partitioning, 2) Sound neural pruning, and 3) Heuristic-based pruning. The key idea behind the technique is that many neurons always remain inactive and hence do not impact the decision-making when we consider a relaxed input specification. Therefore, Fairify employs interval arithmetic and layer-wise verification on each neuron to prune the network. Fairify also profiles the activation heuristics to prune the network further. The pruned networks are then certified, and the developer may choose to deploy subset of pruned networks and leverage the counterexamples to repair the remaining partitions.

Fairness Composition in ML Pipeline through Causal Reasoning. ML pipelines consist of several data processing stages such as normalization, imputation, sampling, etc. Additionally, in ensemble learning, multiple dependent or independent classifiers are composed to make the final prediction using voting, stacking, etc. Each of these ML components has individual fairness impact as well as they compose in a non-trivial way, e.g., two unfair data transformers can be combined for fair predictions. However, the existing methods quantify bias wholistically based on the final outcomes, hence, developers can not identify and compare the biased components. I addressed this problem by localizing bias in ML pipelines through causal reasoning [2]. From pipeline $\mathcal{P}$, I constructed a reference pipeline $\mathcal{P}^*$ by removing or replacing a transformer and computed the causal changes between the predictions $\hat{Y}(\mathcal{P})$ and $\hat{Y}(\mathcal{P}^*)$. Based on four existing **group fairness** criteria such as demographic parity and equal opportunity, I proposed causal fairness measures of the components. I also conducted a case study to choose downstream data transformer that can mitigate unfairness of an upstream one. In addition, I explored the fairness composition in dominant ensemble ML algorithms, e.g., heterogeneous-vs-homogeneous learners, etc., instead of treating them as monolithic components [3], which enables modular mitigation and transparency.

Future work. My works show prospects for an automatic modularization and verification framework that would enable developers to conduct 1) interactive verification and repair, and 2) quantitative verification. The framework would analyze the UNSAT core and patterns in the counterexamples to guide provable repair, e.g., biased neuron removal and weight modifications. I also plan to leverage the white-box analysis to generate assume-guarantee rules for neural verification and further plan to collaborate with HCI researchers to document fairness assumptions and guarantees for accountability in critical decision-making.

2 Sustainable Design of Fair and Safe AI Systems

Sustainable software engineering entails creating and enduring value in the long term without hindering sociotechnical well-being and maintainability. Designing sustainable AI is challenging due to large and complex configurations and CACE principle, i.e., *changing anything changes everything*. The AI system, coupled with its dynamic operating environment, can generate unforeseen side effects that lead to hazardous states. My research aims at addressing these challenges by enabling fair and safe design for software engineers and data scientists [4, 5, 6, 7].

Bias Identification and Mitigation with Accuracy-Fairness Trade-off. I framed the ML fairness as an SE problem and identified several software engineering issues such as the lack of fairness-aware specification and API

design, etc. [4]. I investigated various kinds of societal biases and the impacts of existing mitigation techniques on open-source ML models. The empirical evaluation unveiled the software constructs that directly control the trade-off between different fairness and accuracy metrics. To that end, I built a bias mitigation technique that outperforms the existing ones by achieving generalization over different metrics, datasets, and classifiers [5]. The mitigation technique is built as an **AutoML** approach, that finds the optimal set of hyperparameters in two steps. *First*, I proposed a novel Bayesian multi-objective optimization that dynamically balances fairness and accuracy based on the inputs. *Second*, I ensured efficiency by mining a large amount of dataset-models from OpenAI (offline phase), and guiding the search using heuristics that *match* with the current inputs (online phase).

Long-Term Fairness and Safety. Many sociotechnical systems such as drug monitoring, predictive policing, etc., employ AI models that degrade with time. Designing sustainable safety of for them requires system-level understanding including the environment, agents, and policies. One emerging problem in these systems is unintended side effects from feedback loops. A **feedback loop** occurs when the AI system affects the environment, which, in turn, affects the system's future decisions. For example, a *predictive policing* system can allocate more police to certain neighborhoods, leading to more arrests. These inflated crime reports are fed into the AI software, which reinforces to increasingly over-police the area. I envisioned a safety design framework that models the system, environment and their interactions over a period of time [6]. Then I conducted simulation of the *model* under various configurations and policies, to detect and characterize possible feedbacks (e.g., reinforcing or balancing). Because of the uncertainties (e.g., crime incidents, distribution shifts), the simulation generates a large number of possible traces. Then I built a Monte-Carlo simulation technique called FAIRSENSE, to efficiently sample the traces and evaluate sensitivity of system parameters that cause long-term fairness violations [7]. The results unraveled many counter-intuitive design insights such as short-term fair configurations often result in long-term unfairness.

Future work. While most of the existing tools provide *myopic* design guidance, I will continue to focus on the sustainable design of AI systems. In particular, I will extend the safety framework to enable causal analysis, i.e., automatic extraction of causal loop diagram and stock-and-flow diagram, to identify leverage points. This will inform the design of feedback loop interventions, e.g., delaying an existing feedback or creating one to balance. I also plan to synthesize adaptive interventions (e.g., dynamic policies) to maintain the *equilibrium* of the properties.

3 AI Engineering through Mining and Static Program Analysis

While ML toolbox provides quicker development of many complex solutions, it has been characterized as the *high-interest credit card* — accumulating a new set of technical debts (TD) in the development process. Because of the emphasis on exploratory and iterative development, ML engineers often adopt ad-hoc processes leading to such issues. I built an analysis framework for ML-enabled software [8, 9], and then leveraged that to improve the quality and maintenance of ML programs and processes [10, 11, 12].

The Art and Practice of AI/ML Software. To understand novel issues and best practices, I built a large-scale mining and static analysis infrastructure for AI/ML software. I extended the Boa framework to mine Python source code and Jupyter notebooks and construct representations such as AST, CFG, etc. Then I mined a few large datasets from GitHub for the research on **data science (DS) pipelines** — the recent one contains more than 102K repositories and 290 million file snapshots. DS pipeline is a software architecture consisting of sequential stages such as preprocessing, modeling, training, etc., typically followed in AI/ML projects. To facilitate research and practice on DS pipelines, it is essential to understand their semantics, organization and characteristics. I conducted a comprehensive study of DS pipelines in theory (literature), in-the-small (computational notebooks), and in-the-large (GitHub projects) [10]. For the computational pipelines, I developed a static analyzer to automatically extract temporal order of APIs and call sequences. I further combined qualitative and quantitative methodologies to elicit representative views and (anti-)patterns of pipelines including implicit feedback loops, tightly coupled stages, etc.

Identifying ML Technical Debt and its Impact on LLM. Despite the frequent presence of TD in ML software, no comprehensive study has been conducted on their types and how they could be fixed. I conducted the first empirical study to build a taxonomy of machine learning TD [11]. I mined *self-admitted* TD, i.e., source code with comments that admit the debts, and identified 23 types of TD such as ML knowledge debt, interpretability debt, dependency debt, etc., under five broad types of traditional TD, e.g., requirement debt, design debt, etc. I also computed the effort needed to *repay* each type of TD by analyzing the code diff in debt-resolution commits. Furthermore, developers are extensively adopting **large language models** (LLM) such as GitHub Copilot, to write code. Since the LLMs are trained on large corpus of open-source data, they can potentially regenerate the existing TD and introduce several safety risks. I explored the impact of LLMs on the maintenance and evolution of TD [12]. The work revealed that LLM produces technically indebted code as much as 35.36% of the time. On the positive side, I proposed a prompt engineering technique to be applied on the existing TD to automatically repay the debts.

Future work. I plan to develop a technique that identifies unsafe TD and prioritizes them for resolution in critical systems. Furthermore, akin to the system-level long-term analysis (Section 2), I plan to study AI software evolution by mining the changes made through commits. Some initial investigations on deep learning models (at the code-level) show repetitive change patterns such as tensor manipulation, training scheduling, etc., many of which are breaking changes, e.g., vanishing or exploding gradients. I will develop a change clone detection method for ML programs and libraries, that would suggest or prevent breaking changes with respect to the upstream.

4 Research Agenda

I plan to continue my research in the area of *software engineering (SE) for AI*, and combine formal reasoning and empirical SE to bridge the gap between the fail-safe software designs and implementation. As outlined in the previous sections, my **short-term plan (1-5 years)** is to pursue the following directions — §1: Propose interactive fairness verification and provable repair, §2: Design for AI systems that prevent long-term discrimination and safety risks, §3: Build program analysis techniques to improve the evolution and maintenance of ML software.

My **long-term vision (5-10 years)** is to conduct research on ensuring dependability of intelligent software systems, broadly defined. I would empower *software engineers and data scientists* with techniques that ensure trustworthiness to the *end-users and society*. In addition to my prior focus on classical ML models, neural networks, ensembles, AutoML and LLM, I will investigate emerging AI such as Artificial General Intelligence (AGI). Specifically, I intend to build rigorous SE techniques and tools in the following directions:

- ***Responsible AI Engineering:*** I will focus on responsible AI engineering, through exploration of fairness and safety, and their synergy with robustness and interpretability. This will involve redefining the properties through domain-specific requirement engineering and specification inference. While most of the techniques take a "build, then fix" approach, I aim to build a "correct by construction" framework that ensures alignment of the properties in production. Specifically, the framework would generate contracts using input preconditions and neural properties, e.g., correctness of a ConvNet on certain subpopulation will be verified using local filtering and pooling operations. Furthermore, the counterexamples of the contracts and their negative counterparts will be used to provide reliable explanations. Thus, the framework will decide when to trust ML decisions and override the risks in presence of *known unknowns*, e.g., data shifts, model drifts. For the *unknown unknowns*, I will build monitoring tools to prevent accidents by combining software and human-in-the-loop interventions.
- ***Design Software Safety in Leaning-Enabled Systems:*** I plan to build modular design techniques for learning-enabled systems such as medical devices and autonomous vehicles. Through analysis of several ML and non-ML components and their intricate interactions in these hybrid systems, I will explore the benefits of modularity, i.e., separation of concerns, modular debugging, and reuse. Particularly, I am interested in developing a platform that would lift (from source code) or build (from specification) models of the independent software components and ensure safe feature interaction. For example, the developers would be able to design dynamic prioritization of features of the failsafe battery and ML-based object recognition modules of a delivery-drone, to ensure safe landing over faster delivery in a weather condition. This will require creation of domain-specific languages and model-driven analyses that capture unprecedented feature interactions of components.
- ***Sustainable Software Engineering:*** While significant progress has been made on hardware sustainability, software sustainability is still in an early stage, leading to the premature obsolescence of many AI systems. I plan to explore sustainable SE as a crosscutting attribute, both vertically along the AI pipeline and horizontally across responsible quality attributes. In particular, I will focus on collecting and creating sustainable design patterns and metrics through retrospective analysis of software evolution. This will lead to development of AI processes for modular upgrade and disposal. For example, developers would be able to upgrade pre-trained models by forgetting certain inputs, change of parameters (patching), or adding new functionality (e.g., multi-modality), etc. I will focus on building techniques such as version control of model, expert model creation, and reuse of predictive capabilities that systematically improve compatibility and life expectancy of the AI systems.

Funding and grant opportunities: The White House AI Bill of Rights 2022 highlighted that – *"designers, developers, and deployers of automated systems should take proactive and continuous measures to protect [...] from algorithmic discrimination"*. A recent executive order on Safe, Secure, and Trustworthy AI (October 2023) also underscored future research in the area. Currently, my works in the DARPA project advances their continuous effort on the safety of high-assurance software. My prior works also contributed to establishing the NSF institute at ISU called "Dependable Data Driven Discovery". The recent workshops by NASEM and NIST, and DCLs from NSF propels software engineering research for fairness, safety and sustainability. As a faculty member, I plan to apply for financial support through university, industry, and government programs, e.g., NSF CAREER, CCF-SHF, etc., and build collaborative efforts with researchers from academia and industry towards dependable next-generation software systems.

References

[1] **Sumon Biswas** and Hridesh Rajan. Fairify: Fairness verification of neural networks. In *ICSE'2023: The 45th International Conference on Software Engineering*, page 1546–1558, 2023.

[2] **Sumon Biswas** and Hridesh Rajan. Fair preprocessing: Towards understanding compositional fairness of data transformers in machine learning pipeline. In *ESEC/FSE'2021: The 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 981–993, 2021.

[3] Usman Gohar, **Sumon Biswas**, and Hridesh Rajan. Towards understanding fairness and its composition in ensemble machine learning. In *ICSE'2023: The 45th International Conference on Software Engineering*, page 1533–1545, 2023.

[4] **Sumon Biswas** and Hridesh Rajan. Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness. In *ESEC/FSE'2020: The 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 642–653, 2020.

[5] Giang Nguyen, **Sumon Biswas**, and Hridesh Rajan. Fix fairness, don't ruin accuracy: Performance aware fairness repair using AutoML. In *ESEC/FSE'2023: The 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1–13, 2023.

[6] **Sumon Biswas**, Yining She, and Eunsuk Kang. Towards safe ML-based systems in presence of feedback loops. In *ESEC/FSE'2023 Workshop (SE4SafeML): Dependability and Trustworthiness of Safety-Critical Systems with Machine Learned Components*, pages 1–4, 2023. (A position paper).

[7] Yining She*, **Sumon Biswas***, Christian Kästner, and Eunsuk Kang. Long-term fairness analysis of ML-enabled system. In *Under submission*, pages 1–22, 2024. (*: Equal contribution).

[8] Boa: Mining ultra large scale software repositories — Python and ML Datasets (February'22/Python, August'21/Python, Jan'21/ML-Verse, August'20/Python-DS). `http://boa.cs.iastate.edu/boa/index.php`.

[9] **Sumon Biswas**, Md Johirul Islam, Yijia Huang, and Hridesh Rajan. Boa meets Python: A Boa dataset of data science software in Python language. In *MSR'19: 16th International Conference on Mining Software Repositories*, pages 577–581, May 2019.

[10] **Sumon Biswas**, Mohammad Wardat, and Hridesh Rajan. The art and practice of data science pipelines: A comprehensive study of data science pipelines in theory, in-the-small, and in-the-large. In *ICSE'2022: The 44th International Conference on Software Engineering*, page 2091–2103, 2022.

[11] David OBrien, **Sumon Biswas**, Sayem Imtiaz, Rabe Abdalkareem, Emad Shihab, and Hridesh Rajan. 23 shades of self-admitted technical debt: An empirical study on machine learning software. In *ESEC/FSE'2022: The 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 734–746, 2022.

[12] David OBrien, **Sumon Biswas**, Sayem Imtiaz, Rabe Abdalkareem, Emad Shihab, and Hridesh Rajan. Are prompt engineering and TODO comments friends or foes? an evaluation on GitHub copilot. In *ICSE'2024: The 46th International Conference on Software Engineering*, pages 1–12, 2024.