EasyChair Preprint

№ 12739

Open Source Machine Learning Model Safety Assurance for Embedded Edge Systems

Ching Hu and S.R. Prakash

March 27, 2024

Open Source Machine Learning Model Safety Assurance for Embedded Edge Systems

Ching Hu
Platform System Architecture, Safety and Security
ching.hu@sima.ai

S.R. Prakash
Platform System Architecture, Signal Processing
sr.prakash@sima.ai

***Abstract*—Machine learning (ML) model advancement is moving forward at a rapid pace compared to the safety and security critical system development life cycle. The behavior of the ML model depends on the data and software implementation of the algorithm. To leverage these advancements while keeping the system compliant for safety critical applications, this paper reviews the challenges of using open source ML models for safety critical applications and proposes metrics and workflow to improve model assurance for deployment in edge systems.**

***Index Terms*—safety, security, machine learning, safety assurance, open source, model validation, data bias, data variance, explainability, embedded edge**

I. Introduction

A. Challenges

Rapid advancements in the area of artificial intelligence or machine learning (AI/ML) are pushing many industries to evaluate and deploy the latest techniques into products for competitive advantage. The ML advancement is typically published as a technical paper describing a ML model, the background theory, the training methodology and datasets used in developing the model, and the procedure, datasets and results of performance evaluation on the model. Many of these significant innovations are open-source, permitting deployment under specified licensing conditions, and the papers reporting them include links to source code and datasets that aid in reproducing and verifying the reported results. However, transforming the findings in this paper into a system that can operate correctly and safely in real-world environments is a complicated process. Finding the right balance between leveraging the advantages offered by open-source innovation and assuring product safety and security remains a significant challenge. [1], a great reference, outlined auditing tools for open source ML models with a fairly comprehensive set of tools, but does not frame them into a best practice ML development life cycle for edge systems. This paper will aggregate practical methods and flows into a ML specific life cycle development process as illustrated in Figures 1, 2.

B. In Practice

As ML product matures, best practice and patterns of ML product development life cycle will emerge. SiMa.ai proposed CRISP-EML (Cross-Industry Standard Process for Embedded Machine Learning) development life cycle is illustrated in Figures 1, 2 as an example of interdependencies between different stages, personas, and goals for each stage. This framework is based on CRISP-ML(Q) as defined by MLOps (ml-ops.org). The emphasis on data and model validation throughout the life cycle is necessarily a part of safety assurance. The rest of this paper will discuss safety critical considerations that influence the safety assurance of ML Model in a production edge system in the context of the development process outlined in Figures 1, 2.

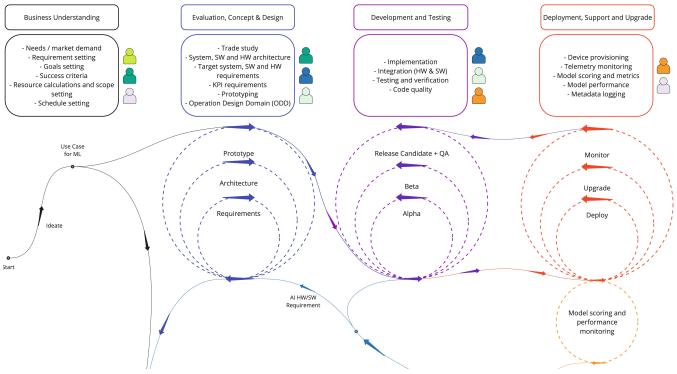

Fig. 1. ML Application Development and Deployment block; Credit: SiMa.ai

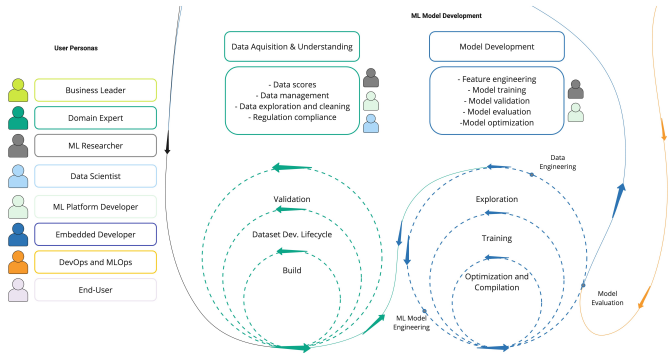

Fig. 2. ML model development; Credit: SiMa.ai

II. Safety Critical Considerations

A. Deterministic vs. Probabilistic

An inference system consists of a software-based ML model and a hardware platform that executes the ML model. The ML

Sponsored by SiMa.ai

model is a graph that encodes the sequence of mathematical transformations performed on the input to the system. Each of these two components of a system can contribute to the non-determinism of the system in different ways.

A deterministic system yields the same and absolutely predictable behavior or same outputs when fed the same set of input at different points in time, i.e. time invariant or independent. Non-deterministic system does not guarantee the same input-to-output absolute predictability relationship from trial to trial, i.e. input and time dependent. If a ML system can self adapt to an operating environment over time, then it's an adaptive ML system or time varying system. The goal of a safety assurance of such an adaptive ML system is to safety-bound the system's behavior.

Non-deterministic systems have two flavors, probabilistic and time-varying models. Probabilistic model means the outcome likely resides in a distribution of answers centered around a mean output. Time-varying mode's parameter and structure may change over time and the resulting outputs to the same stimulus can be different. Importantly, there is no bound on the magnitude of the difference from trial to trial. An inference system may exhibit non-determinism with different statistical characteristics that may be derived from a number of causes:

1) Platforms: Digital platforms are generally expected to be deterministic. However, some platform components such as CUDA can produce different results based on the execution order of threads across processing nodes [2]. Though this non-determinism may be mitigated with software control, as systems become more distributed, enforcing such mitigation control will become more challenging. These mitigation controls will also erode the computational efficiencies gained by distributed processing.
2) Data sources: Systems operating in real-world environments encounter a wide range of inputs often containing stochastic noise components. Practically, producing exactly identical inputs as the training data set is not possible . While the ML models are expected to be resistant to noise, the outputs are influenced by these noise components and the system would effectively exhibit a probabilistic non-determinism. Furthermore, the model's behavior will be different depending on in what sequence the new training data is inserted [3] [4] .
3) Models: ML models in the inference system can be completely static or dynamic. Static ML models means once deployed, the model's parameters will not be changed. Dynamic models, or time-varying models, would include systems that are periodically updated or replaced, or models that are continuously updated via reinforcement learning, or models that are trained and updated through a shared collective.

B. Data Bias vs. Variance

The behavior of a ML model depends on how it is trained and what it is trained with. Data bias and variance are two major considerations in training data as illustrated in Figure 3, as extracted from [5], [6]. Data bias is overweighting or overrepresented features or patterns in a dataset leading to biased predictions. Data variance is associated with variability of a model's prediction based on the data set.

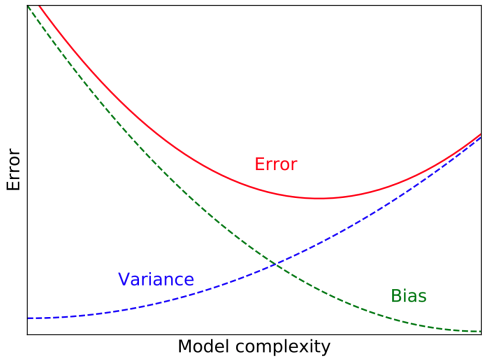

Fig. 3. Model complexity and error relationships with data bias, data variance, and error; Credit: EASA and Daedalean [5]

Even for large-scale datasets, bias can still be captured by modern neural networks resulting in model behavior uncertainty. [7] For open source dataset and pretrained models, the hidden or undiscovered bias can present significant challenges in safety assurance, as supported by [8].

It's also important to consider the application environment when constructing the training and validation data set. NIST (National Institute of Standards and Technology) presented a very comprehensive study and guidance to manage data bias in AI [9].

C. Black-box Model Explainability

There have been many research efforts covering the topic of explainable AI (XAI) due to the black box nature of ML models. [10] XAI is an exercise during design time to gain rationality to how a model arrived at its conclusion. This is particularly important in the context of safety for compliance, trust, and accountability. In the context of a production pipeline for safety critical applications, scaling the explainability output to be automated is a critical step. There are multiple options for model explanations in [11], e.g. general approach options are distillation / approximations, visualization, or intrinsic models. Couple of example approaches are LIME (Local Interpretable Model-agnostic Explanation) [12] and RISE (Randomized Input Sampling for Explanation) [13]

D. Model Optimization

Quantization and pruning both introduce modifications in the model's behavior, and these optimization is necessary for deployment where the host computing platform is resource limited, power limited, and timing constrained. The challenge is to ensure, as close as possible, the original behavior of the model. This is a stage where learning is transitioning to inference only processing. This transition from learning

environment to inference environment is where a formal safety compliant process is needed. These optimizations all occur after training is completed, thus changes here will impact model behavior, thus introducing unknown risks. Worth considering are technique such as quantization-enabled training [14].

a) Quantization: as part of the optimization options, is a mapping of continuous space to discrete space. The mapping can be a uniform quantization, i.e. equally spaced values, or non-uniform quantization. Uniform quantization is what's being used in most deployed systems due to its simplicity and practicality in hardware implementation. Quantization is inherently a lossy transform from input to output space. As the research in quantization advances, more methods will be available [15], and the accuracy of the quantized model will increase. It is important to verify this specific stage, and define an application relevant metric that ensures original intended behavior of the model. Quantization is also an unavoidable stage in the ML deployment process, as such QAT (quantization-aware training) [16] is often used to minimize the impact of this lossy stage.

b) Pruning: operates on the model's computational graph network by removing the components such as a neuron or weights toward a neuron. The model behavior will be modified at the cost of increasing model runtime performance.

E. Model Validation

The goals for model validation are to identify and add missing requirements and test cases. Two major categories for validation are data and model. For data, it consists of training data and test data. For the model, it can be validated qualitatively, but quantitative validation is very challenging. With the rising popularity of transformers, identifying which part of the input, i.e. attention, contributes the most to the output decision provides significant value in validating model behavior.

F. Security Risks

Edge systems live in the "wild" where Murphy's law dominates. Developing models vs. deployed models each have their own security risks. In the context of a safety critical system, a system is not safe if it is not secure. According to OWASP (Open Web Application Security Project) [17], the top ML security risks are:

1) Input manipulation attack *
2) Data poisoning attack *
3) Model inversion attack *
4) Membership inference attack *
5) Model theft
6) AI supply chain attack *
7) Transfer learning attack
8) Model skewing *
9) Output integrity attack
10) Model poisoning *

The risks tagged with "*" are particularly relevant to using open source models as reinforced with study in [18]. For example, an AI supply chain attack is when a malicious actor modifies or replaces a library, data set, or model used by the system. A recent example is that of an open source AI model found in Hugging Face that installs backdoors and other types of malware on end user machines [19]. According to JFrog's analysis, about 100 instances of malicious ML models on Hugging Face contain vulnerabilities such as object hijacking, reverse shells, and arbitrary code execution. Furthermore, about 95% of these malicious models were built with PyTorch [20]. When a system is jointly developed across multiple organizations internal or external, this risk increases as more entities and various development flows are introduced.

A MLSecOps(Machine Learning Security Operation) equivalent to OWASP vulnerability is found in [21]. Beyond vulnerabilities and attacks on ML based systems, in the context of safety, the primary concern is with the failure modes as identified in [22]. In particular, the unintended failure modes that directly affect the systems safety are reward hacking, side effects, distributional shifts, natural adversarial examples, common corruptions, and incomplete testing in realistic conditions. More unintended failure modes may be identified as ML technology advances. Other failure modes may lead to one or more of the following compromises: ML system integrity, availability, and confidentiality.

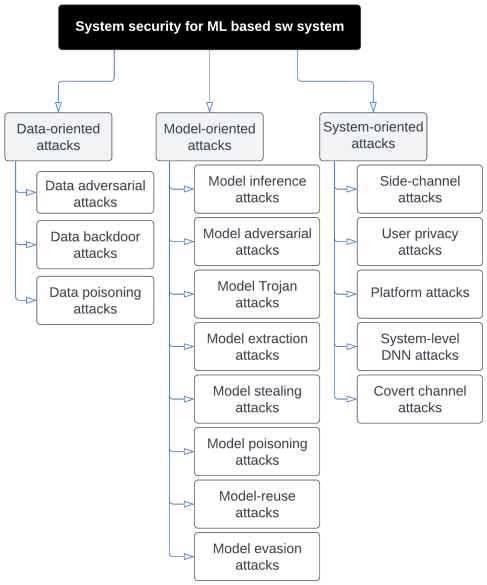

Fig. 4. Taxonomy of attacks on ML based system [23]

III. Model Assurance Framework for Safety

A model assurance framework is a necessary part of the ML life cycle transition from training to inference environment. This framework is a set of flows that requires both a development process that enables safety assurance as well as optimizing a model targeted for the deployment hardware. The following subsection would be part of the model assurance framework.

A. Model Tuning

Model tuning step is intended to find the optimal implementation, in terms of throughput, resource utilization, and accuracy, for a specific target host hardware. There will need to be a trade off between these different metrics and resulting in a balanced deployed system. The safety relevant tuning may include a latency model from the target host hardware and boundary conditions for bandwidth limited data throughput. The post tuning activity is to validate the consistency of model behavior from before and after tuning. The safety assurance subsection will propose an automated model validation flow to support assurance level assessment with supporting evidence.

B. Model Architecture

Model architecture aggregates data generation, data collection, feature engineering, training, evaluation, task orchestration, prediction, infrastructure, authentication, interaction, and monitoring. Safety relevant activities would be analyzing model architecture's fault tolerance, fault isolation, and fault recovery.

C. Model Algorithm

Model algorithm includes linear regression, logistic regression, decision tree, support vector machine (SVM), naive Bayes network, KNN (K-Nearest Neighbor), K-means, and random forest. Safety relevant activities would include analysis of plausibility of algorithm behavior, e.g. physical rules and legal rules within the ODD (operational design domain).

D. Safety Assurance

Safety assurance of model centers around process assurance and design assurance of the model with supporting artifacts or evidence. Safety compliance requires supporting evidence to support safety claims. Model explainability with comprehensive evidence would be a critical step in the model assurance pipeline. In the avionics domain, a ML development work flow for low criticality airborne applications is proposed in [24], where explainability is a key challenge, which impacts traceability through a ML model, a key safety requirement. This limitation constrains the ML application only to lower criticality systems.

This paper proposes a flow based on generated contextually realistic scenery data sets to automate model validation with explainability. For example, a video camera would require the synthesis of a photo-realistic image, whereas a LiDAR would require the synthesis of a point cloud data. The innovation in this flow is the usage of realistic scene and a set of rendered region(s) of interest (ROI) from the original image/video as shown in Figure 5. ROI can be a single object or a combination of different ROI. The amount of permutation can be exhaustive to provide all permutations of ROI. Model under test can operate on two paths: (1) normal complex realistic scene; (2) permutation of ROI combination of rendered photo-realistic images and even other environmental variables such as lighting condition, shadow regions, weather, and road surface conditions. First path would produce the baseline predictions as the model normally does. The second path would produce a set of high confidence predictions due to the noise-free conditions of the masked ROI. The results from these two paths can be compared, and false predictions can be quickly identified and the specific conditions on which the false prediction can be identified based on the permutation of the rendered images. This flow is model-agnostic, and can be used for diverse scenarios and use cases, where the scenes can be generated based on the ODD. Furthermore, for automotive use cases, OpenScenario and OpenODD may be used as a way to support generation of scenes, where the ROI is not limited to vehicles or pedestrians, but also may include weather, road condition, lighting level, or other environmental parameters. This would also enable full traceability from requirement to validation.

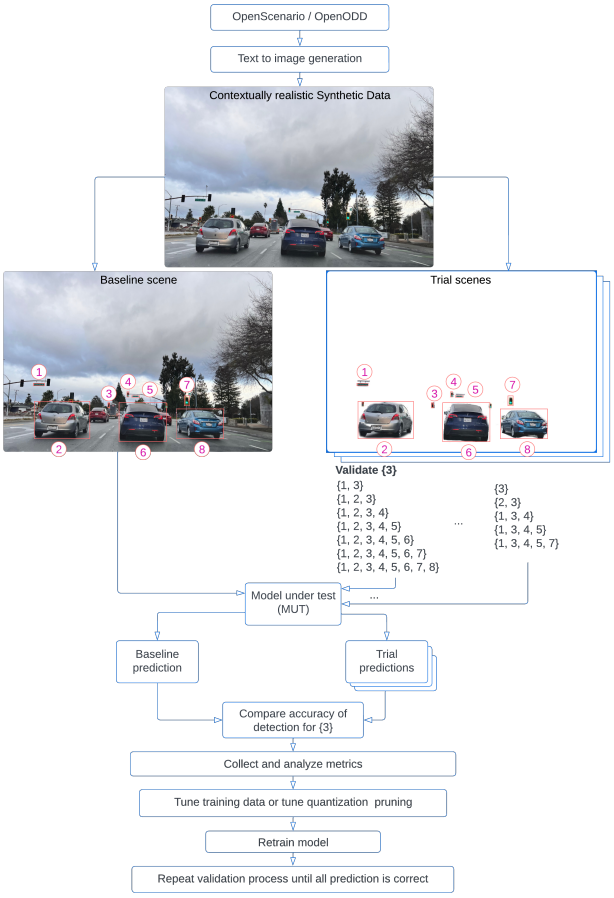

Fig. 5. Automated model validation with explainability using generated photo-realistic image sets

The commonly used mAP (mean Average Precision) metric, which uses the confusion matrix (TP, TN, FP, FN), IoU, Precision, and Recall, can be used for the normal prediction precision. But, in addition, for each permutation of the ROI image set, a context accuracy metric can be measured with mAP as well. The context accuracy would add assurance on understanding model behavior.

The context accuracy metric will be a significant measure for assurance of model behavior, and is especially important when large images are processed in tiles or at significantly

reduced resolution, which have reduced context, reduced field of view (i.e. missing context), or reduced details of context information.

The key benefits of this proposed flow are:

1) Automates model-agnostic explainability with traceable supporting evidence
2) Provides comprehensive coverage of scenarios
3) Identify which ROI influence the prediction for debugging overall model behavior
4) Provides metrics to have higher assurance in the model
5) Provide metrics to support targeted model tuning
6) Provides capability to compare pre- and post-quantized model for behavior consistency verification
7) Can be used for different domain or applications, not restricted to just automotive

E. Development process

The development process specifically for ML model development as illustrated in Figures 1, 2. For safety specific ML development process, the following should be considered:

1) Data source integrity and configuration management
2) Data cleaning flow
3) ML model safety metric definition
4) Model source integrity and authentication
5) Pre- and post- model optimization validation flow
6) Model explainability assurance metric definition

For safety compliance, explainability provides a comprehensive set of evidence to support safety arguments. [25] provides a great example of ML safety argument using GSN (Goal Structured Notation) for healthcare ML systems.

F. Security Assurance

The goal of security assurance is to provide a trustworthy and accountable development flow that protects against security risks identified in previous discussion. MLSecOps (Machine Learning Security Operations), which is a process to minimize ML model's vulnerability, is gaining traction in various fields. An example of which is proposed in [26] for the biotech industry that utilizes machine learning systems. While threats to each industries may have unique threat vectors and attack surfaces, the underlying MLSecOps and regulatory compliance philosophies are well aligned and can be cross leveraged.

IV. Conclusion

Outline of ML system development life cycle and the considerations needed for using opens source ML models in context of safety critical applications are presented. A framework of model assurance is outlined with emphasis on data and model validation methods. Furthermore, an automated model validation using synthesized dataset and respective metrics was proposed. With the growing usage of open source ML models and data sets, topics discussed and flow proposed in this paper will mature faster as more safety critical and mission critical applications leverage advancements made in the open source community.

Acknowledgment

We thank Carlos Davila of SiMa.ai for providing Figures 1, 2.

References

[1] C. M. Poland. (2022, Jun.) The right tool for the job: Open-source auditing tools in machine learning. [Online]. Available: https://arxiv.org/pdf/2206.10613.pdf

[2] D. Riach. (2019, Mar.) Determinism in deep learning (s9911). [Online]. Available: https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9911-determinism-in-deep-learning.pdf

[3] Z. Hammoudeh and D. Lowd. (2023, Jun.) Training data influence analysis and estimation: A survey. [Online]. Available: https://arxiv.org/pdf/2212.04612.pdf

[4] J. Mange, "Effect of training data order for machine learning," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, pp. 406–407.

[5] EASA and Daedalean. (2020, Mar.) Concepts of design assurance for neural networks (codann). [Online]. Available: https://www.easa.europa.eu/sites/default/files/dfu/EASA-DDLN-Concepts-of-Design-Assurance-for-Neural-Networks-CoDANN.pdf

[6] ——. (2021, May) Concepts of design assurance for neural networks (codann) ii. [Online]. Available: https://skybrary.aero/sites/default/files/bookshelf/6053.pdf

[7] Z. Liu and K. He. (2024, Mar.) A decade's battle on dataset bias: Are we there yet? [Online]. Available: https://arxiv.org/pdf/2403.08632.pdf

[8] M. Bernhardt, C. Jones, and B. Glocker. (2021, Dec.) Potential sources of dataset bias complicate investigation of underdiagnosis by machine learning algorithms. [Online]. Available: https://arxiv.org/pdf/2201.07856.pdf

[9] R. Schwartz, A. Vassilev, K. Greene, L. Perine, A. Burt, and P. Hall. (2022, Mar.) Nist special publication 1270: Towards a standard for identifying and managing bias in artificial intelligence. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1270.pdf

[10] A. Carrillo, L. F. Cantú, and A. Noriega. (2021, Apr.) Individual explanations in machine learning models: A survey for practitioners. [Online]. Available: https://arxiv.org/pdf/2104.04144.pdf

[11] A. S. Ravindran and J. Contreras-Vidal. (2023, Oct.) An empirical comparison of deep learning explainability approaches for eeg using simulated ground truth. [Online]. Available: https://www.nature.com/articles/s41598-023-43871-8.pdf

[12] M. T. Ribeiro, S. Singh, and C. Guestrin. (2016, Aug.) 'why should i trust you?' explaining the predictions of any classifier. [Online]. Available: https://arxiv.org/pdf/1602.04938.pdf

[13] V. Petsiuk, A. Das, and K. Saenko. (2018, Sep.) Rise: Randomized input sampling for explanation of black-box models. [Online]. Available: https://arxiv.org/pdf/1806.07421.pdf

[14] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. (2017, Dec.) Quantization and training of neural networks for efficient integer-arithmetic-only inference. [Online]. Available: https://arxiv.org/pdf/1712.05877.pdf

[15] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. (2021, Jun.) A survey of quantization methods for efficient neural network inference. [Online]. Available: https://arxiv.org/pdf/2103.13630.pdf

[16] X. Huang, Z. Liu, S.-Y. Liu, and K.-T. Cheng. (2023) Efficient quantization-aware training with adaptive coreset selection. [Online]. Available: https://arxiv.org/pdf/2306.07215.pdf

[17] S. Bhure, S. Singh, R. van der Veer, and A. Kang. (2023, Jan.) Owasp machine learning security top ten v0.3. [Online]. Available: https://owasp.org/www-project-machine-learning-security-top-10/

[18] D. Hintersdorf, L. Struppek, and K. Kersting. (2023, Aug.) Balancing transparency and risk: The security and privacy risks of open-source machine learning models. [Online]. Available: https://arxiv.org/pdf/2308.09490.pdf

[19] OECD.AI. (2024, Mar.) Hugging face, the github of ai, hosted code that backdoored user devices. [Online]. Available: https://oecd.ai/en/incidents/72360

[20] J. Burt. (2024, Feb.) Security vulnerabilities popping up on hugging face's ai platform. [Online]. Available: https://securityboulevard.com/2024/02/security-vulnerabilities-popping-up-on-hugging-faces-ai-platform/

[21] T. I. for Ethical AI & Machine Learning. (2024, Mar.) The mlsecops top 10 vulnerabilities. [Online]. Available: https://ethical.institute/security.html#four

[22] R. S. S. Kumar, D. O'Brien, K. Albert, S. Viljoen, and J. Snover. (2019, Nov.) Failure modes in machine learning. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1911/1911.11034.pdf

[23] H. Chen and M. A. Babar. (2023, Dec.) Security for machine learning-based software systems: a survey of threats, practices and challenges. [Online]. Available: https://arxiv.org/pdf/2201.04736v2.pdf

[24] K. Dmitriev, J. Schumann, and F. Holzapfel. (2022, Sep.) Toward certification of machine-learning systems for low criticality airborne applications. [Online]. Available: https://arxiv.org/pdf/2209.13975.pdf

[25] Y. Jia, J. McDermid, T. Lawton, and I. Habli. (2022, May) The role of explainability in assuring safety of machine learning in healthcare. [Online]. Available: https://arxiv.org/pdf/2109.00520.pdf

[26] N. Pervez and A. J. Titus. (2024, Feb.) Integrating mlsecops in the biotechnology industry 5.0. [Online]. Available: https://arxiv.org/pdf/2402.07967.pdf