Tool Report

Evaluation Report on RobustML

European Trustworthy AI Foundation
ISX-ETF-LIV-1775

contact@confiance.ai | www.confiance.ai/foundation

Document reference: 440C

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Martin Gonzalez	IRT SystemX	Research Scientist
Reviewer	Raphael Braud	IRT SystemX	Team Leader

Document Control

Revision	Date	Commentary	Author
v0.1	01-03-2025	Initiate Document	Martin Gonzalez
v1.0	31-03-2025	Ready for review	Martin Gonzalez

Contents

A. Introduction

A.1. General introduction to trustworthy AI challenges

Trustworthiness in AI within critical systems (systems that can directly or indirectly affect human life and moral entities) is essential for its widespread adoption (by the industry, the decision makers, the general public, etc.) and poses the following significant challenges.

- First, how to design AI models, so that, by construction, they satisfy trustworthy properties (accuracy, robustness. . .).
- Secondly, how to characterize these AI models, for example to understand and explain their behavior and their adequacy to the operational domain.
- Then, how to implement and embed those AI models on hardware, by making them fit for the target without losing their trustworthy properties.
- Another question is, what methods of data engineering to apply in order to, among other topics, manage important volumes of data and adapt to the evolution of the operational domain.
- At system level, what verification and certification processes to consider specifically for AI-based systems.
- Finally, a federation of all these matters is necessary to build an end-to-end methodological approach, supported by a consistent engineering environment compatible with industrial practices. These are the challenges, among others, that the Confiance.ai program addresses.

A.2. Rationale for this Evaluation Report

This document aims to provide a technical and methodological evaluation of the RobustML asset, developed within the Confiance.ai program, and in particular the developments and maturation that this library underwent in the EC4 Team by the end of the program.

A.3. Target audience and disclaimer

The target audience of this document consist of machine learning engineers, not necessarily expert but familiar with ML adversarial robustness techniques and best practices.

A.4. How to use this document

There are three different uses of this document:

1. The reader may use this document as a final evaluation report on RobustML and its positioning with respect to other adversarial evaluation libraries such as the Adversarial Robustness Toolbox (ART) and the RobustBench libraries.

2. The reader may also use this document to position the added values of RobustML on an industry-oriented focus and obtain information about promising new directions on which this asset can be further matured.

B. Evaluation Report

B.1. A straight dive-in to the insights of this report

In order to properly evaluate and characterize the RobustML tool developed within the Confiance.ai program with other libraries, much context needs to be addressed. However, for the sake of transparency, we expose upfront the key arguments we will develop and will constitute the main evaluative dimensions of this report.

- **RobustML is an *industry-oriented* Robustness Evaluation tool:** RobustML differentiates from RobustBench in that the latter was initially created as a proxy for evaluating *scientific* progress on the field of Adversarial Machine Learning. This proxy is constructed in RobustBench so that each leader-board ultimately matches a prespecified optimization objective. For fixing one, one needs to pre-specify, besides a dataset, a noise perturbation amplitude, a norm, and a choice on whether attacks are targeted or not. But the fact is that there are no universally best-performing methods across different optimization objectives. On the other hand, RobustML produces robustness degradation curves at increasing noise perturbation amplitudes which allow industries to better delimitate the robustness expectations of their models and be able to choose tailored solutions at different thresholds of perturbation amplitudes.

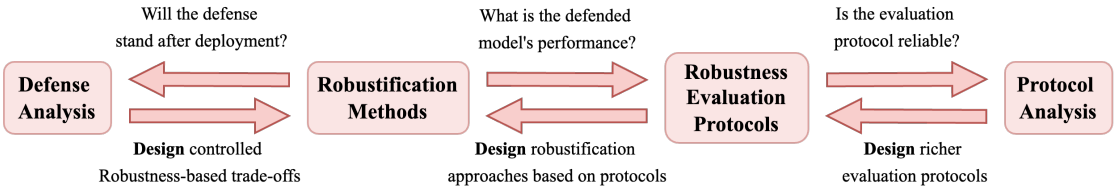
- **RobustML's Adversarial Robustness Evaluation service is aligned with the concept of *worst-case in-distribution robustness analysis*:** RobustML differentiates from ART in that the latter is meant to be a library that lists available adversarial attack methods for different purposes. Nevertheless, ART does not propose state of the art evaluation protocols for robustness, which is why ART is not utilized in RobustBench. The fact is that state of the art protocols do not consist on simply applying all methods listed in ART but rather crafting principled combinations of well-chosen methods that make more damage than taking those methods separately. RobustML aligns with the original RobustBench evaluation protocols for evaluating models. This is a point we will develop in detail concerning the distinctions on how the AutoAttack method was implemented in RobustBench in comparison to ART. This will have major implications as it shows that one cannot recreate such protocols within the perimeter of the ART library.

- **RobustML's associated methodological guideline:** RobustML's ongoing developments provide concrete templates for adversarial robustness evaluation protocols, adapted to specific ML Component's design architecture. Such templates can be consulted in Chapter C of the RUM Methodology, although we will revisit a good amount of it in Subsection B.3 and the overall next subsection for a high-level picture.

B.2. Overview on adversarial Robustness Evaluation

In this section, we lay the ground for the high-level methodology for evaluating adversarial robustness, which is the main topic of the "Evaluation service" of RobustML. This is meant to give context to the next section, where we give emphasis on the relation of RobustML with other libraries such as ART and AutoAttack, upon which RobustML is partially built upon, and articulate the specificity and differentiation points that create the added value of RobustML over such libraries.

B.2.1 Robustness Evaluation across different methodologies



Confiance.ai's robustness assessment consists on the workflow in the above figure, based on:

- **Robustness Evaluation Protocols:** These transform the theoretical evaluation concept into an algorithmic protocol. For instance, crafting adversarial attacks over the test sets realizes the idea of worst-case In-Distribution analysis while Monte-Carlo sampling is used to estimate a theoretically continuous region around a sample to be certified. It usually serves as a design guide to craft robustification approaches.

- **Robustification Methods:** The very structure of each evaluation protocol usually serves as a design guide to craft defense approaches. Such robustification methods are tightly bounded with the evaluation protocols from which they were crafted and their goal is to improve that protocol's metrics. For instance, adversarial training is crafted to improve a model's performance in the context of worst-case evaluation (i.e. adversarial attacks).

- **Protocol Analysis:** Beyond addressing different questions, protocols are themselves implementations of theoretical methods and need to be the subject of a validation/certification process. For instance, worst-case evaluation is thought to be conducted with the strongest available adversarial attacks within a predefined cyber-security breach context. As such, if the protocol only uses weak adversarial attacks against a well-defended model, it will generate an over-estimated robust accuracy due to a poor quality evaluation protocol rather than due to a poor defense mechanism.

- **Defense Analysis:** analyzing the quality of a robustification method goes beyond metric improvement and its process is tightly related to protocol analysis. For instance, it is a fact that too strong adversarial training overfits and poorly generalizes to adversarial threats which were not seen during training. As a consequence, the worst-case analysis protocol must be enriched to account for such strength-generalizability trade-off.

B.2.2 Worst-case In-Distribution Robustness Analysis

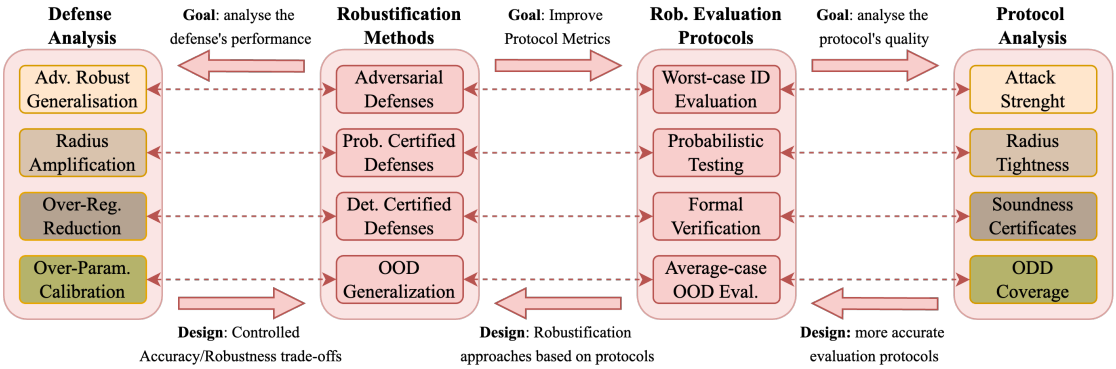
The topic commonly known as Adversarial Robustness Evaluation in the scientific literature can be reframed as algorithmically implementing tools for addressing the concept of worst-case In-

Distribution Robustness Analysis for Machine Learning models. This is to be distinguished to 3 other Robustness dimensions for ML Models :

- a part of these deal with certified defenses (both deterministic and probabilistic). For this conception of robustness, the objective is to characterize local stability properties of models in an attack-agnostic fashion. This means that the objective is to guarantee a property that will work universally for adversarial attacks under extremely constrained scenarios that are driven by scientific progress.

- another part deals with OOD Robustness, which addresses the concept of average-case out-of-distribution robustness analysis for ML Models. This subject is very industry-oriented, although, as evaluation is done in an average-case fashion, implementing its protocols is done in a completely different way than for adversarial robustness and has its own set of challenges, which need to be first addressed separately from the "adversarial robustness" evaluative dimension.

Overall, the system of evaluation protocols, protocol analysis, robustification methods and analysis of such methods for these different dimensions can be encompassed in the following picture.



In this document we will mainly concentrate on the first row of this picture, and very specifically in the two columns on the right which are the main service of RobustML : providing rigorous adversarial robustness evaluation protocols to better address worst-case in-distribution robustness analysis, and analysis existing evaluation protocols and in particular the way the strength of attacks implemented in such protocols has an impact on the reliability of provided robustness evaluation.

B.3. RobustML Evaluation Protocols and Positioning

RobustML is a python library developed within the Confiance.ai's program. This library is simultaneously built on top of modified versions of two well-known libraries, ART and RobustBench (which is the maintained version of the original AutoAttack library that is no longer maintained), that RobustML encompasses and handles in their specificity.

As said earlier, RobustML differentiates from ART in that the latter does not fully exploit state of the art adversarial robustness evaluation protocols. In this subsection we will develop the

concrete example of the AutoAttack method, its rationale, the differences in implementation from ART and RobustBench, and finally the position of RobustML around what methodology to implement.

AutoAttack is a robust adversarial attack framework designed to evaluate the adversarial robustness of deep neural networks. It was introduced in the paper "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks" (Croce & Hein, 2020). AutoAttack consists of a suite of attacks that are parameter-free and highly effective in breaking defenses against adversarial attacks.

The original version of AutoAttack's functioning can be resumed as follows:

- The first attack on the list of attacks is operated on an image, if the attack is not successful, then we pass to use the second attack on this list and so on, until an attack succeeded to fool a model on the image. In this case, we chose new image and proceed likewise. This way, by the end of the evaluation procedure, images that were not attacked are those for which none of the available attack methods succeeded to craft a viable adversarial noise within the pre-specified normed ball that is the search space of the attack

- During the above procedure, AutoAttack's algorithm also inspects the ML model's nature and in particular its defense mechanism and can readapt the list of attacks to be performed to better exploit known weaknesses for several defenses. In particular, it well check for randomization defenses, non-deterministic outputs, zero-gradients (gradient-masking).

- If the standard version was used, AutoAttack also produces protocolary checks to report on whether the use of its standard version is reliable or not, and, in case it is unreliable, it indicates examples of attacks that one may want to use using the custom version of AutoAttack. This is particularly helpful when considering test-time adaptive defenses.

- Finally, it also ensures that the number of classes for targeted attacks is consistent with the attacking objective.

Only the first item on the above list is made for the ART implementation of AutoAttack. Very concretely, instead of providing the entire evaluation protocol, the ART Implementation instantiates AutoAttack simply as a callable attack class. Next, the list of attacks used in ART's default AutoAttack are nowhere near the amount and diversity found in the original implementation of AutoAttack, which is the one that RobustML supports, as shown in Table B.1.

We needs to identify here a crucial methodological error : targeted attacks are ruled-out de-facto while they should be used if a worst-case evaluation is aimed. Moreover, their algorithm sets to standard as FALSE for the possibility to use targeted attacks even when explicitly providing a list of attacks one wishes AutoAttack to be run across. The very important Fast Adaptive Boundary attack (2019) is nowhere to be found in ART while it is actively used in the original version of AutoAttack and is used as a discriminative factor to determine if standard AutoAttack evaluation is or not reliable. As such, the original version of AutoAttack cannot be built within ART (even in its 2025 version), and ART cannot provide any significant indicator for the reliability of the adversarial robustness evaluation protocol of choice.

Next, the ART implementation of AutoAttack does not account for attacks that are known to be

AutoAttack's version	ART	Original & RobustML		
	-	standard	plus	rand
untargeted APGD-CE (no restarts)		•	○	
targeted APGD-DLR (9 target classes)		•	○	
targeted FAB (9 target classes)		•	○	
Square Attack (5000 queries)	•	•	•	
DeepFool (eps=10^{-3}, 10 class gradients)	•			
untargeted APGD-CE (5 restarts)	•		•	
untargeted APGD-DLR (5 restarts)	•		•	
untargeted FAB (5 restarts)			•	
targeted APGD-DLR (9 target classes)			•	
untargeted APGD-CE (no restarts, 20 iterations for EoT)				•
untargeted APGD-DLR (no restarts, 20 iterations for EoT)				•

Table B.1: AutoAttack in ART and in the original implementation as used in RobustBench and RobustML. The • sign means that the method is part of the list of attacks to be performed. The ○ sign means that the "plus" version of AutoAttack implicitly performs those attacks as part of the stronger signaled attacks.

able to adapt to well-known defenses. The "Rand" version of AutoAttack automatically explores if the model possesses randomization procedures such as classifiers with stochastic components. In this case, it is known that one can combine AutoAttack with Expectation over Transformation (EoT) as in (Athalye et al., 2018) when initializing AutoAttack.

As a general rule, the ART implementation does not handle properly customized attacks, in which the attacker is free to craft a tailored list of attacks in order to maximally inflict damage to the model. This is also a major drawback for the ART implementation of AutoAttack since most of the state-of-the-art protocols for attacking specific defenses are indeed custom. It is not only a matter of choosing a named attack but already simply varying the number of restarts in APGD can lead to further increasing the attack success rate of AutoAttack.

It is worth noticing that RobustML *still* supports ART in that it provides a centralized dictionary of attacks which are superior in number (rather than strength) than those contained in RobustBench.

B.3.0.1 On the advantage of customized attacks in RobustML for adaptive defenses

Adaptive test-time defenses, as illustrated in Figure B.1 operate by adapting their computation to the input. A first category of adaptive test-time defenses operate in the input space and aim to "purify" inputs before they are fed to the standard pre-trained model. A second category aims at adapting model parameters or intermediate activations.

Such defenses are not new and the consideration of well-suited adversarial attacks designed to evaluate these defenses already aims towards the aim of addressing worst-case analysis for In-Distribution Robustness at the ML Component's level. Simply, the ML Components here are ex-

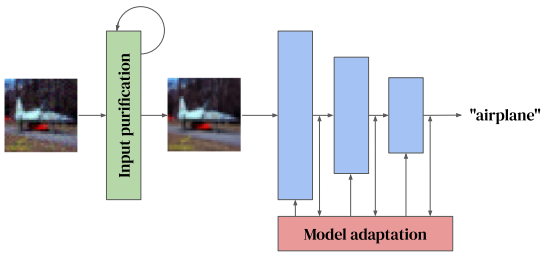


Figure B.1: Adaptive test-time defenses.

Table B.2: Summary of the nine *adaptive test-time defenses* evaluated in the Croce et al. (2022) case study. They measure the robust accuracy of each defense (and in parenthesis that of its underlying static model) against ℓ_∞-norm bounded perturbations of size $\varepsilon = 8/255$ on CIFAR-10 along with the robust accuracy measured by the respective papers ("Reported"). * This evaluation uses $\varepsilon = 2/255$. ** This evaluation uses batch size 50 instead of the original 512.

Defense	Venue	Principles		Building blocks				Inference time	AutoAttack Custom Version	Robust Accuracy		
		IP	MA	IA	AN	R	ED			Reported	Croce et al. (2022)	
Kang et al. (2021)	NeurIPS		•	•	•			2×	Transfer APGD	57.76%	52.2%	(53.9%)
Chen et al. (2021)*	ICLR		•	•	•			59×	APGD+BPDA	34.5%	5.6%	(0.0%)
Wu et al. (2021)	ArXiv	•		•		•		46×	Transfer APGD+BPDA+EoT	65.70%	61.0%	(63.0%)
Alfarra et al. (2022)	AAAI	•		•				8×	RayS (decision-based)	79.2%	66.6%	(66.6%)
Shi et al. (2020)	ICLR	•		•	•			518×	APGD+BPDA (traj.)	51.02%	3.7%	(0.0%)
Qian et al. (2021)	ArXiv	•		•	•			4×	APGD	65.07%	12.6%	(7.7%)
Hwang et al. (2021)	ICML(W)	•		•	•			40×	APGD+BPDA	52.65%	43.8%	(49.3%)
Mao et al. (2021)**	ICCV	•		•	•	•	•	407×	APGD+BPDA+EoT	63.83%	58.4%	(59.4%)
Yoon et al. (2021)	ICML	•		•	•	•		176×	APGD+EoT	69.71%	33.7%	(0.0%)

tremely little and do not yet integrate complex independent auxiliary modules. Still, they exemplify perfectly the importance of the custom version of AutoAttack, which is absent in the ART's version. In the paper Croce et al. (2022), they evaluated a certain number of defenses. Each defense is categorized by its principles—input purification (IP) and model adaptation (MA)—and its building blocks: iterative algorithm (IA), auxiliary network (AN), randomization (R), and external data (ED). We reproduce their results in table B.2.

An important fact about the above evaluation is that none the custom choices of attacks rely on techniques proposed after 2020. In other words, these defenses were not broken with newer attacks but with correctly applied already available and older attacks.

- **RayS** (Chen and Gu, 2020) is a decision-based attack designed for ℓ_∞-norm bounded perturbations. It only requires the label predicted by the model.
- **Backward Pass Differentiable Approximation (BPDA)** (Athalye et al., 2018) permits the attack of non-differentiable defenses by approximating them with differentiable functions during gradient computation. The identity is a common approximation.
- **Expectation over Transformation (EoT)** (Athalye et al., 2018) permits the attack of randomized defenses. The predictions and gradients are computed in expectation over the randomness of the model, approximated by averaging the results of multiple runs with the same input.

- **Transfer attacks** generate adversarial perturbations on a surrogate model and use them on the target model.

Last but not least, **none** of the above methods are listed in the ART library, which means that such evaluation is, at present, impossible within the perimeter of ART. Notice that the correct implementation of AutoAttack induces major revisions in the evaluation and benchmarking of the defense methods. For some of them, one can see that the undefended model actually had a better robust accuracy than the defended model. None of this could have been evaluated with the ART's version of AutoAttack but is very much possible to evaluate with the original version of AutoAttack that is supported by RobustML.

B.4. Conclusion of the Evaluation Report

In conclusion, RobustML provides the best of both worlds while compensating against their respective disadvantages, and being *industry-oriented*. On the one hand, it is build on top of the ART library that contains a wide variety of adversarial attacks, yet is incomplete with regards of the state-of-the-art attacks in the literature and does not contain any significant worst-case in-distribution analysis protocol for ML model's robustness. On the other hand, RobustML compensates this precise insufficiency from ART by integrating the original AutoAttack methodology used in RobustBench, containing the state-of-the-art evaluation protocols and associated attacks in order to best address worst-case in-distribution analysis for ML model's robustness. And contrary to RobustBench, RobustML does not arbitrarily fix a number of attack parameters which might be useful to measure scientific progress but little to no interest or evaluating and characterizing adversarial robustness in an industry-oriented approach.

B.5. Perspectives on the evolution of RobustML

We have argued in favor of the added value of RobustML as an asset developed within the Confiance.ai program. Nevertheless, this library is still far from addressing many evaluation problems on industrial use-cases.

- RobustML limits at the moment on robustness for ML Components whose central model is an image classifier. A promising avenue would be to extend the evaluative functions of the library to address worst-case in-distribution robustness analysis for ML Components whose central model consists on object detectors, semantic segmentators or even anomaly detectors. Part of this program was initiated when addressing robustness for regression tasks and for novel data typologies such as audio data which ca be found in Automatic Speech Recognition Systems. Developing RobustML along this dimension would be promising.

- RobustML, as an "adversarial robustness" evaluation library, has also started to address the challenging question of LLM safety. Recent work has managed to formalize the already challenging task of providing a discrete optimization procedure as an adversarial objective against LLMs. The first practical implementation of this is known as the Greedy Coordinate Gradient attack method, which ranges among "Jailbreaking" methods for LLMs. That being said, many challenges around LLM safety are still unsolved by the scientific community to rigorously evaluate adversarial robustness for LLMs. First, it has become clear that attacks in this scenario are unbounded contrary to the classification case, the

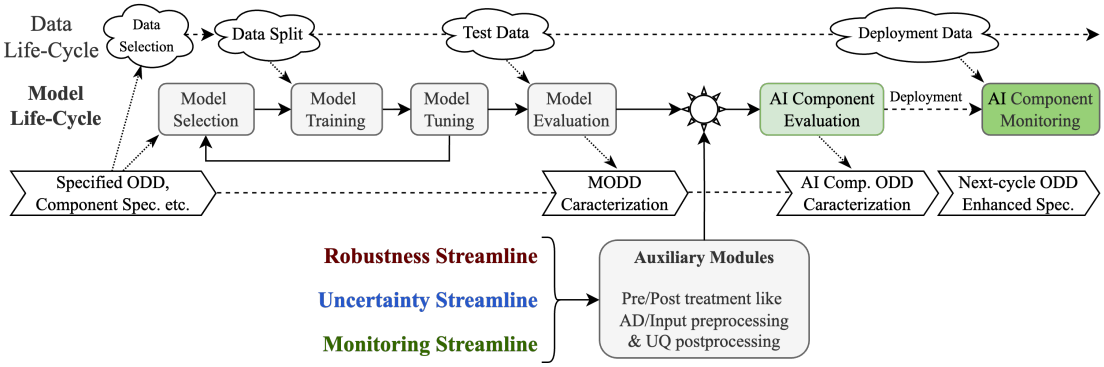
space where attackers implement their algorithms is also challenging to determine, since role-playing can jailbreak LLMs and these kinds of attacks are difficult to compare with prompts for which LLMs have guardrails. Next, the attackers objectives are also elusive since the notion of "harmful content" remains subjective. RobustML should follow closely the ongoing developments of this area in order to extend its functionalities to LLM Safety Evaluation.

B.6. Appendix: RobustML within the RUM Methodology

This section provides an account on an indirect benefit of the RobustML asset among other libraries, as it needs to be interpreted as an instantiation of a part of the RUM Methodology. This section does not constitute part of the evaluation report for RobustML so the reader may skip it. Nevertheless, one should point out that the present evaluation report can itself be considered as an application of the RUM methodology to the particular example of differentiating and positioning robustness evaluation assets in the spectrum of robustness as a trustworthiness attribute at a system's engineering level.

B.6.0.1 From ML Model Robustness Evaluation to ML Component Robustness Evaluation

Ensuring a model's robustness during development, quantifying uncertainties during inference, and monitoring its performance in production are continuous processes. During the Confiance.ai's program, it was clear that a ML Model's life-cycle had to be extended to at least 3 parallel life-cycles, one for Robustness, one for Uncertainty Quantification and one for Monitoring. The key point was that an ML model cannot be an all-in-one solution to address these matters, and that in order to address them effectively, Robustness, UQ and Monitoring approaches had to be designed with a unified and principled overall vision of the ML Component they will constitute. A first architecture of this process is presented in the following figure :



First, the Specified ODD, the component specifications and other higher-level requirements have an impact on the selection of data and model to be used during training. After these have been set, and a data split has been done, the model is trained.

The model tuning phase consists on intermediate evaluations of the trained model that may have a retroactive impact on possible chances in the model architecture, the training algorithm or the data selected and/or its split.

Once this loop gives a satisfactory trained model according to the higher level specified ODD & specifications, the model is evaluated on chosen test data, which might be the same i.i.d. test data used during the intermediate evaluations of the model in the tuning phase, or it can consist on completely different test data such as OOD data (synthetic or real), unbalanced and non-i.i.d data and so on. The choice of the test data reflects the specified ODD requirements.

Overall, evaluating the model on such test data gives a Model-ODD characterization, usually very close but never completely identical to the part of the Specified ODD characterization concerning *only the model*.

In parallel to the streamline concerning these stages, three other streamlines, each one concerning robustness measures, Uncertainty quantification modules and monitoring systems is developed in parallel. Some of these might have dependencies with the model's streamline as they will be *inserted* into the model itself. Otherwise, they will consist on auxiliary modules that are plugged to the initial model, which we will call the *central model*, and which will usually take the form of pre-treatment modules such as anomaly detectors that will filter out anomalies so that the central model dies not make inference on them, or post-treatment modules such as Enriching the central model's outputs with information about its uncertainties.

The AI component will then be the resulting aggregate of the central model along with all the implemented auxiliary modules around it.

Now that we have composed such AI component, a further evaluation must be made concerning its trustworthiness attributes, but this time, *as a component*. Indeed, the trustworthiness attributes of the AI component will in general be different from those of the central model alone, and while one can expect that the robustness or uncertainty of the AI component will be better than that of the central model alone, *there might be cases in which implemented robustness and UQ measures will harm each other,* creating trade-offs between robustness and uncertainty that could not have been foreseen or evaluated on the central model alone.

The AI component's evaluation will then incur into an AI component ODD characterization that will then be compared to the initial specified ODD. In case the resulting AI component is considered to comply with the requirements, it will be embedded in the system that will host this component. We emphasize that although many evaluations are of absolute necessity in this stage, there are not the subject of the RUM methodology and will not be treated here.

Finally, once the system hosting the AI component is deployed, the monitoring functions that are included in the AI component will allow to online monitor many aspects of the deployed and hosted AI component, either in a sample-wise or batch-wise fashion, and which may use informative feedback coming from the UQ and Robustness streamlines to do so.

B.6.0.2 An example of a system-level adversarial attack

In order to keep this survey as simple as possible, we address the necessity to think about the robustness of the whole ML Component in light of a recent adversarial attack Ledda et al. (2023) that can simultaneously damage a semantic segmentator's performance, make its UQ unreliable and ultimately damage any OOD Detection system that was based on UQ estimates.

Here, the focus is on a specific adversarial scenario in which the attacker is interested in manip-

ulating the uncertainty estimate, regardless of the correctness of the central model's prediction. Its aim is to undercut the use of UQ techniques for ML models when their results are consumed by a downstream module or by a human.

Let us survey the formalization of such attacks as well as analyzing its evaluation protocol. Denote $\mathbf{x}_{\text{adv}}$ for an adversarial example for the central model f. The objective of the attacker crafting $\mathbf{x}_{\text{adv}}$ is to lower the model's accuracy $\mathscr{A}_f$, and denote $\mathscr{U}(x)$ for the uncertainty estimate of f at an input x. The objective of a UQ attacker is formalized as follows:

$$\underset{\delta}{\text{argmin}} \quad \gamma \cdot \mathscr{U}(x+\delta) \qquad \text{such that } \|\delta\| < \varepsilon \tag{B.1}$$

where $\gamma \in \{-1,+1\}$ controls the attack objective. Concretely, taking $\gamma = -1$ leads to abnormally increasing the uncertainty measure, making the model underconfident, and taking $\gamma = 1$ leads to abnormally decreasing the uncertainty measure, making the model overconfident. Robustness analysis over such protocol leads to induce that crafting such adversarial attacks address a **worst-case analysis of the UQ module's *integrity*** (in the case $\gamma = 1$) **and *availability*** (in the case $\gamma = -1$). Concretely, the model will not say "I don't know" when it should in the first case, and will say "I don't know" when it shouldn't in the second case

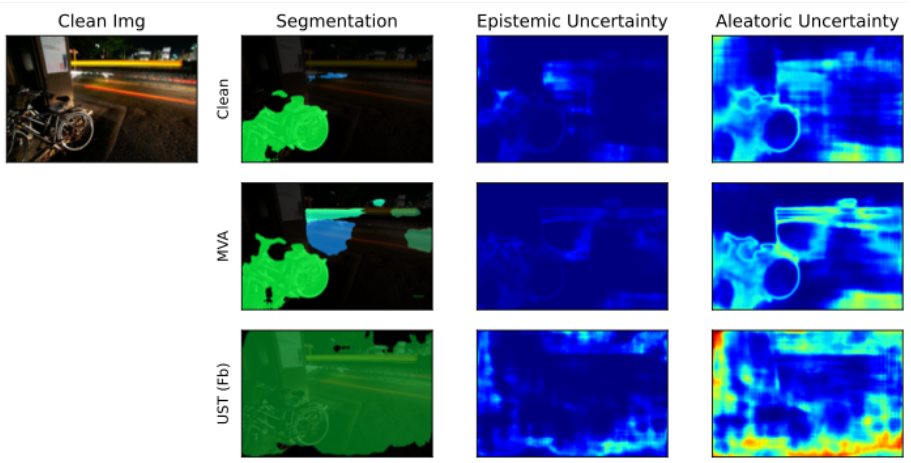


Figure B.2: (From Ledda et al. (2023)) Semantic Segmentation degradation and UQ degradation under UQ-aware adversarial attacks.

Implementing the attacks in equation (B.1) will be different, depending on if the underlying UQ models are of probabilistic or deterministic nature. The work Ledda et al. (2023) develops both cases although we will here only concentrate on the probabilistic case. On the one hand, they obtain *minimum variance attacks* for probabilistic models, that will mainly leave the central model's prediction unharmed while maximally decalibrating the UQ modules associated to them. On the other hand, they obtain *stabilizing attacks*, which leverage the stabilization of predictions, usually used as a robustness method for the central model, as direct attack against the UQ module. The idea is simple: stability in predictions result in lower variance and average prediction's entropy. We reproduce an illustration of their obtained results for stabilization overconfidence attacks in Figure B.3.

The first striking fact is that, while the "robust accuracy" of the model remains unchanged at increasing noise budget ε, it sharply becomes overconfident on the attacked dataset.

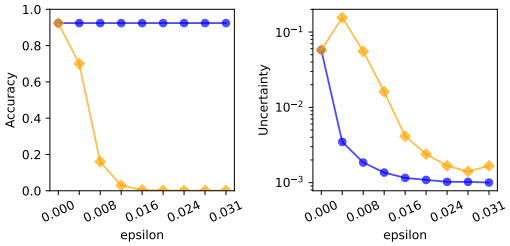


Figure B.3: (From Ledda et al. (2023)) Classification accuracy and uncertainty of a ResNet18 (endowed with a UQ-module) on CIFAR10, under two different UQ-based attacks, as a function of ε. The blue line corresponds to the stabilization overconfidence attack.

The second striking fact is that this attack has a direct incidence on any monitoring system extracting and leveraging uncertainty quantities from such UQ module. For instance, such uncertainties are usually used for OOD detection at the level of the monitoring system. The following show that *the same* stabilization overconfidence attack acts as a *direct attack* on such OOD detection module, greatly lowering its detection accuracy:

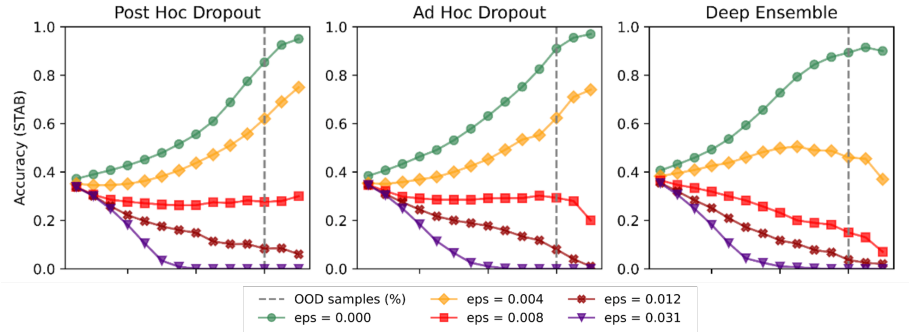


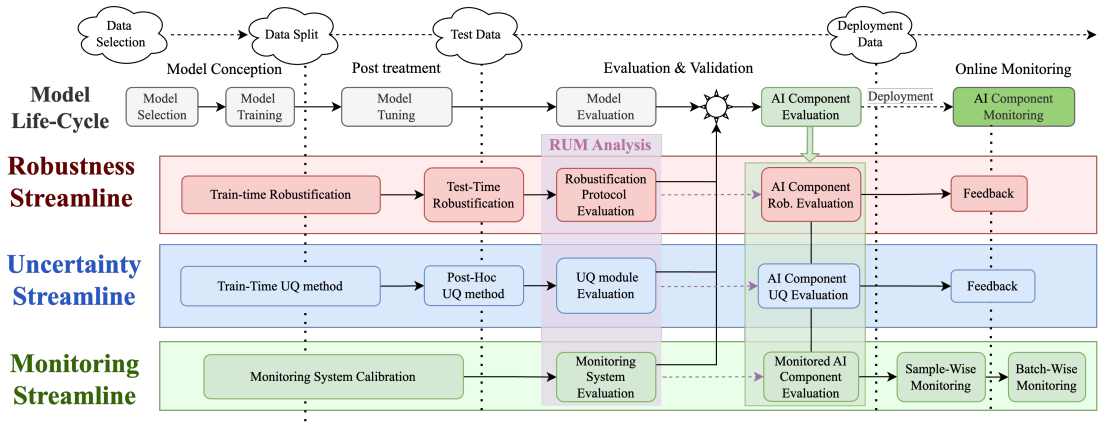
Figure B.4: (From Ledda et al. (2023)) Detection accuracy of UQ-based OOD detection modules in the monitoring system underlying the AI component. The green curve consistently shows how 3 different UQ modules exploited as OOD detectors provide good OOD detection accuracy, and how such accuracy sharply drops while the attacked UQ module becomes overconfident.

The conclusion here provides evidence of a RUM aggregate attribute: UQ-based adversarial robustness, provides also a concrete evaluation protocol on how to measure it at the 3 different RUM levels: a measure on the "robust accuracy" (left almost untouched by the stability attack), a measure on the UQ uncertainty and on the resulting UQ-based OOD detection accuracy.

B.6.0.3 Back to RobustML : Towards a RUM Compliant library

Robustness, uncertainty quantification, and monitoring are interconnected throughout the machine learning life-cycle. Ensuring a model's robustness during development, quantifying uncertainties during inference, and monitoring its performance in production are continuous pro-

cesses.



In light of the above reflection, RobustML, even for a ML Component with a central model being an image classifier, needs to take into account such modern approaches of "UQ-aware" adversarial attacks in order to start implementing worst-case in-distribution robustness analysis for ML Components that have at least this level of complexity.

Bibliography

Alfarra, M., Perez, J. C., Thabet, A., Bibi, A., Torr, P., and Ghanem, B. (2022). Combating adversaries with anti-adversaries. In *AAAI*.

Athalye, A., Carlini, N., and Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *Int. Conf. Mach. Learn.*

Chen, J. and Gu, Q. (2020). RayS: A ray searching method for hard-label adversarial attack. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1739–1747. ACM.

Chen, Z., Li, Q., and Zhang, Z. (2021). Towards robust neural networks via close-loop control. In *Int. Conf. Learn. Represent.*

Croce, F., Gowal, S., Brunner, T., Shelhamer, E., Hein, M., and Cemgil, T. (2022). Evaluating the adversarial robustness of adaptive test-time defenses. In *International Conference on Machine Learning*, pages 4421–4435. PMLR.

Hwang, D., Lee, E., and Rhee, W. (2021). Aid-purifier: A light auxiliary network for boosting adversarial defense. In *ICML Workshop on Adversarial Machine Learning*.

Kang, Q., Song, Y., Ding, Q., and Tay, W. P. (2021). Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *Adv. Neural Inform. Process. Syst.*, 34.

Ledda, E., Angioni, D., Piras, G., Fumera, G., Biggio, B., and Roli, F. (2023). Adversarial attacks against uncertainty quantification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4599–4608.

Mao, C., Chiquier, M., Wang, H., Yang, J., and Vondrick, C. (2021). Adversarial attacks are reversible with natural supervision. In *Int. Conf. Comput. Vis.*

Qian, Z., Zhang, S., Huang, K., Wang, Q., Zhang, R., and Yi, X. (2021). Improving model robustness with latent distribution locally and globally. *arXiv preprint arXiv:2107.04401*.

Shi, C., Holtz, C., and Mishne, G. (2020). Online adversarial purification based on self-supervised learning. In *Int. Conf. Learn. Represent.*

Wu, B., Pan, H., Shen, L., Gu, J., Zhao, S., Li, Z., Cai, D., He, X., and Liu, W. (2021). Attacking adversarial attacks as a defense. *arXiv preprint arXiv:2106.04938*.

Yoon, J., Hwang, S. J., and Lee, J. (2021). Adversarial purification with score-based generative models. In *Int. Conf. Mach. Learn.*

Title: Evaluation Report on RobustML

Keywords: Adversarial Robustness Evaluation, System worst-case analysis, Trustworthiness

This document presents an account on the positioning of the RobustML library among other robustness evaluation libraries, highlighting its advantages over them.