EC4

Methodological Guideline for Robustness Functional Set

Document reference number for ANR

contact@confiance.ai | www.confiance.ai

Document reference: XXX

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Mohamed Ibn Khedher	IRT-SystemX	Main author
Scientific responsible	Mohamed Ibn Khedher	IRT-SystemX	Main author
Co-authors	CEMT 1	IRT-SystemX	Help to define Architecture
	CEMT 2	IRT-SystemX	Maturation of component 451

Document Control

Revision	Date	Commentary	Author
v1.0	20/12/2023	Final version	Mohamed Ibn Khedher
v0.3	11/12/2023	Demonstrators	Mohamed Ibn Khedher
v0.2	30/11/2023	Platform architecture	Mohamed Ibn Khedher
v0.1	31/10/2023	Description of Components	Mohamed Ibn Khedher
v0.0	21/09/2023	First draft	Mohamed Ibn Khedher

Contents

A. Introduction and abstract

A.1. General introduction to trustworthy AI challenges

This activity is part of the scientific challenge of the program related to robustness. The Confiance.ai program has identified a set of scientific challenges around all projects. This activity is integrated into the scientific challenges 'SC18: Ensure robustness to variations from the environment' and 'SC8: Ensuring the theoretical robustness of neural networks against misuse and noise'.
As outlined by members of Confiance.ai, the scientific challenge (SC18) aimes to develop AI/ML components that are robust to environment changes. This can be done by augmenting the training set with synthetic data, by adversarial training, by adding external knowledge, etc. This is because machine learning models trained can be sensitive to the environmental conditions of the data used (for example, for images, brightness variation, frame orientation, etc.; for audio data presence of noise, echo conditions, etc.). However, the aim of scientific challenge (SC8) is to obtain guarantees of robustness. In this document, we adopt the definition of robustness as the ability of the system to perform the intended function in the presence of abnormal or unknown inputs. The robustness of Neural Networks (NNs) is a key component of trustworthiness, therefore has to be addressed since there are no easlily obtained guarantees.
For all these reasons, the need for solutions to address scientific challenges is significant. These solutions enable the measurement and enhancement of robustness. In its first two years, the Confiance.ai program has resulted in a set of components and solutions for these robustness challenges. The current objective is to mature these components and, on the other hand, integrate them into a platform that allows users to choose solutions compatible with their data or the technologies of their AI models. In the following section, we describe the platform's objectives and the organization of the document.

A.2. Objectives and organisation

The aim of this action sheet is to develop a platform that empowers AI model designers and evaluators with a structured framework and a collection of components and methods tailored towards robustness. This platform will guide the design and evaluation of AI models against a variety of chosen disturbances. The goal is to identify the models most appropriate for diverse applications, including image detection/classification and time-series analysis.

The platform offers three main functionalities:

- **Formally evaluate the robustness** of AI model (Neural Network based) to adversarial attacks: this functionality allows users to assess the formal robustness of their AI models. It evaluates robustness to all possible perturbations that can be applied to the input. In other words, formal evaluation seeks to identify the minimal perturbation required to alter the model's decision. This step is generally time-consuming, and currently in the literature, it is only applied to small images and small models.
- **Develop robust AI models against adversarial attacks:** this functionality provides users with a set of tools and techniques to help them design and train robust AI models. These

techniques can help to reduce the vulnerability of AI models to adversarial attacks.
- **Empirically evaluate the robustness of AI models against input perturbations including adversarial attacks:** this functionality allows users to evaluate the robustness of their AI models to adversarial attacks or other types of input perturbations.

This document is organized as follows:
- In the first chapter, we provide a brief description of the robustness components categorized into three categories. For each component, we mention the tested use cases, supported models, supported data types, covered problems, input and output formats, and so on.
- In the second chapter, we present the platform's architectural vision and its integration into the Confiance.ai environment. We describe the platform's use cases, as well as its functional and technical specifications.
- In the third chapter, we present demonstrations of the platform, specifically focusing on Component 451, with applications in both Welding (Renault) and Safran (Inspection) use cases.

The main limitation of this platform is that each component has its own constraints regarding data type and model format. Indeed, a user arriving with their AI model in a specific format will only be able to execute components that are compatible with that format. In future versions, we will need to work on the generalization of components as much as possible to accommodate different model types and data types. It is worth mentioning that when we talk about future versions, it is in the context of perspectives and not versions that will be delivered as part of Batch 3. Another note, in the vocabulary of the Confiance.ai program, the robustness platform is also called Functional Set (FS) Robustness. In the rest of the document, we have used the two words FS and platform to refer to the same thing.

B. Description of Robustness Components of confiance.ai

The Robustness Functional Set offers three functionalities for users. The services can be used independently or together. For example, a user may want to use the platform to develop a robust AI model, then use it to formally or empirically evaluate the robustness of the model. The following diagram B.1 shows a recommended evaluation pipeline that users can follow.

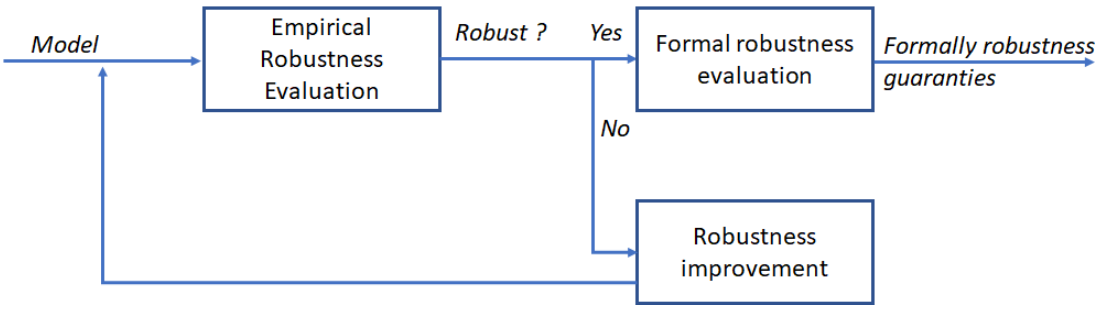

Figure B.1: Recommendation for Formal Robustness Evaluation.

The three use cases of the robustness platform are as follows :

- A user seeking to conduct a formal evaluation of their AI model.
- A user desiring to assess the robustness of their model against various types of perturbation.
- A user seeking to retrain a model that should be more robust than an old training against input perturbation.

In the following sections, we provide a brief description of the robustness components that have been delivered and integrated during batches 1 and 2 of the Confiance.ai program. There will be mainly three major sections to describe the three categories of components. Section B.1 is reserved for components related to the formal evaluation of robustness. Section B.2 is dedicated to components for empirical evaluation against adversarial attacks and other perturbations. Section B.3 is allocated for components related to the improvement of robustness.

For each section, we proceed as follows. First, we provide a brief description of the components. Indeed, we have chosen not to delve into the technical details of the components because they are already described in other program deliverables. We then refer to the relevant deliverables. Next, we present the compatibility of the components with different types of data (inputs) in a first table, the compatibility of the components with different formats of neural networks in a second table, the applicability of the components on the use cases of the program in a third table. Moreover, in a fourth table, a general description (input, use cases tested, maturity level, etc.) of components is provided

In tables related to identify the applicability of components on use cases of the Confiance.ai program, three checkmarks are used:

- ✘ The component is tested on the Use Case.
- ✘ The component is untested with the Use Case but is compatible and testable.
- ✘ The component is incompatible with the Use Case.

B.1. Components of Formal evaluation of robustness

The components of the formal evaluation have been selected from Batch 1 and Batch 2. We invite readers to refer to the deliverable on the state of the art in formal methods Liv (2021), as well as the deliverable on the evaluation of formal methods on specific use cases of the program EC3 (2021a).
Indeed, deliverable Liv (2021) essentially describes the mathematical aspects of formal methods, while deliverable EC3 (2021a) presents the technical results of applying formal methods to the use cases of the program, primarily Welding (Renault) and ACAS-Xu. In batches 1 and 2, these components were tested with the infinity norm.

Component 321: Saimple

Saimple is a verifier for neural network based on abstract interpretation. It was developed by Numalis. The tool delivered by Numalis is a compressed archive composed basically by a set of docker images. Saimple, along with PyRAT, is one of the two industrial tools evaluated. PyRAT and Numalis have demonstrated good results compared to other tools. We invite readers to refer to the deliverable comparing Saimple and PyRAT for more details EC3 (2022b).

Component 322: nnenum

Nnenum is an exact verifier for sequential neural networks supporting fully-connected and convolutional network with ReLU activation functions. It is developed by Stanley Bak Bak et al. (2020) and is considered a state-of-the-art tool, especially on the ACAS-Xu benchmark. Nnenum combines an approach based on zonotopes and star-set overapproximations with an improved path enumeration verification method for case splitting on the ReLU function.

Component 323: α-β-crown

α-β-crown is a formal neural network verifier based on the CROWN Zhang et al. (2018) verifier, GPU relaxation for ReLU and a branch and bound approach for case splitting on ReLU. It supports GPU computation to accelerate the analysis. Combined with its wide support of neural network analysis and thus possibility to run on various benchmarks, this led to very good performance in the VNN-COMP 2021.

Component 3171: PyRAT

PyRAT is a sound abstract interpretation verifier for neural networks. It was developed at CEA after considering the lack of adaptability to neural networks of classical formal software verification tools developed at CEA. Through abstract interpretation techniques, PyRAT proposes multiple abstract domains to verify neural networks, such as boxes or zonotopes. To verify networks, PyRAT propagates the abstract domains through the network in order to obtain the whole set of reachable values from an initial configuration. In addition to this, an input partitioning approach allows PyRAT to improve its precision and prove more general properties on the network, such as the ACAS-Xu properties. For more information on PyRAT, we invite readers to consult the deliverable EC3 (2021a) where PyRAT has been compared with other formal methods and the deliverable EC3 (2022b) where a maturity assessment of PyRAT is proposed.

Table B.1: Components List of Formal evaluation of robustness.

ID	Name	Type	Covered problems	Input format	UCs tested	Technical Maturity	Functional Maturity
321	Saimple	Python Library	Classification	ONNX Keras/TF Images	Welding	1	TBD
322	nnenum	Python Library	Classification	ONNX nnet Images/Tabular	Acas-Xu	2	TBD
323	α-β-crown	Python Library	Classification	ONNX PyTorch Images/Tabular	Acas-Xu	2	TBD
3171	PyRAT	Python Library	Classification	ONNX Keras/TF PyTorch nnet Images/Tabular	Welding Acas-Xu	2	TBD
391	MIP Solver	Python Library	Classification	ONNX Keras/TF Images/Tabular	MNIST CIFAR	TBD	TBD

Table B.2: Components vs supported Data type.

Use Case	Saimple	nnenum	alpha-beta-crown	PyRAT	MIP Solver
Images	✘	✘	✘	✘	✘
Tabular	✘	✘	✘	✘	✘
Time-Series	✘	✘	✘	✘	✘
NLP	✘	✘	✘	✘	✘

Table B.3: Components vs supported Model type.

Use Case	Saimple	nnenum	alpha-beta-crown	PyRAT	MIP Solver
Tensorflow	✘	✘	✘	✘	✘
PyTorch	✘	✘	✘	✘	✘
ONNX	✘	✘	✘	✘	✘
NNET	✘	✘	✘	✘	✘

Component 391: MIP Solver

This component is designed to perform a Robustness Evaluation of neural network models based on an exact formal verification method. As we aim to address all components related to robustness

in this document, we mention component 391. However, we draw the readers' attention to the fact that this component was implemented in an academic framework and evaluated on public databases rather than on the program's use cases. For more information, we invite the readers to refer to deliverable Ramzi Ben Mhenni and Canu (2023).

Table B.4: Applicability of components to use cases.

Use Case	**Saimple**	**nnenum**	**alpha-beta-crown**	**PyRAT**	**MIP Solver**
Welding Inspection (Renault)	✘	✘	✘	✘	✘
Demand forecasting (Air-Liquide-time series)	✘	✘	✘	✘	✘
Scene understanding (Valeo, detection)	✘	✘	✘	✘	✘
Scene understanding (Valeo, segmentation)	✘	✘	✘	✘	✘
ACAS-Xu (Airbus)	✘	✘	✘	✘	✘
Aerial Photograph Interpretation (Thales LAS)	✘	✘	✘	✘	✘
Visual Industrial Control (Safran)	✘	✘	✘	✘	✘
Visual Similarity (ATOS)	✘	✘	✘	✘	✘
NLP Opinion mining (Renault)	✘	✘	✘	✘	✘
Cylinder Counting (Air Liquide)	✘	✘	✘	✘	✘

B.2. Components of Empirical Robustness Evaluation

The components for the empirical evaluation have been selected from Batch 1 and Batch 2. For further details, we invite readers to consult the EC3 deliverable EC3 (2021b).
The main purpose of these components is to assess the robustness of a trained model against a set of perturbations. These perturbations can be specific to images, such as Gaussian blur or geometric transformations. As these perturbations can also serve as adversarial attacks. It is worth mentioning that each component is compatible with only a specific set of data types. All compatibility information is detailed in Table B.6.

Table B.5: Components List of Empirical Robustness Evaluation.

ID	Name	Type	Covered problems	Input format	UCs tested	Technical Maturity	Functional Maturity
331	Adversarial Attack Characterization Component	Python Library	Classification	ONNX Keras/TF Images	Renault Welding	2	1
332	AI Metamorphis Observer Component	Python Library	Classification Surrogate	ONNX Keras/TF nnet Images Time series	Renault Welding Acas-Xu Safran Vis. Indus. Control Air Liquide Demand Forecasting	2	1
333	Amplification Method for Robustness Evaluation Component	Python Library	Classification Corpus Amplification	Time series Images	Renault Welding Air Liquide Demand Forecasting	2	1
334	Non-overlapping Corruption Benchmarker Component	Python Script	Classification	ONNX Images	Renault Welding	2	TBD
335	Time-series Robustness Characterizer Component	Python Library	ONLY Time series	ONNX PyTorch Time series	Air liquide Demand Forecasting	2	TBD
3141	Chiru	SaaS by Kereval (API)	Classification Detection	Any model Images	Thales LAS Renault Welding Valeo Detection	1	TBD
4192	ML watermarking	Python Library	Classification	Keras/TF Images	Renault Welding	2	1

Component 331: Adversarial Attack Characterization Component

The component aims to evaluate the reproductibility level of a decision model: it provides its robustness ratio on the basis of specified interest variables including the variation of their intensity. The component relies on the open-source ART-IBM Nicolae et al. (2018) library and offers the evaluation of a neural network model against a set of adversarial attacks. In this version of component 331, the selected adversarial attacks are:

- Fast Gradient Sign Method (FGSM) Goodfellow et al. (2015)
- Basic Iterative Method (BIM) Kurakin et al. (2017)
- Projected Gradient Descent (PGD)Madry et al. (2018)
- DeepFool Moosavi-Dezfooli et al. (2017)
- NewtonFool Jang et al. (2017)

As mentioned earlier, this component includes only 6 attacks. However, other attacks from the ART-IBM library can be integrated and utilized. A maturity upgrade for upcoming versions could aim to make this component compatible with the ONNX format.
It is worth mentioning that the norm used for the adversarial attack is an important parameter. The most common norms are L1, L2, and L-infinity. For our evaluation, we chose the L-infinity norm, but we invite users to select the norm that aligns with their goals. Moreover, to compare attack strength, it is essential to use the same norm.

Component 332: AI Metamorphis Observer Component (AIMOS)

The component consists in evaluating metamorphic properties on AI models. The toolkit will try to be as agnostic as possible to be able to compare a wide range of model types on different UCs. Different metamorphic properties and transformations can be tested over ranges of values and on different models with some visual displays. On UC ACAS-Xu and UC Welding some transformations and properties have been tested and the code to reproduce the test is available. In batch 2, a maturity upgrade has been proposed. For more information, we invite readers to refer to the deliverable EC3 (2022a). AIMOS includes several attacks such as (non-exhaustive list):

- Gaussian noise (electrically-induced noise)
- Poisson noise (thermally-induced noise)
- Gaussian blur (camera vibration)
- Motion blur, both vertical and horizontal (camera vibration)
- Pixels, columns and lines loss (camera sensor failure)
- Defocus blur (camera focus variation)

Component 333: Amplification Method for Robustness Evaluation Component

The component consists in evaluating the robustness of the models on the Welding and Time Series use case using amplification methods on the corpus with noise functions. The used functions are scripts written in Python and will be provided with the documentation on how to use them. For the Welding use case, the following noise function will be provided : gaussian blur, motion blur (both horizontal and vertical), dead pixels, lines and columns, additive gaussian and multiplicative poisson.

Component 334: Non-overlapping Corruption Benchmarker Component

The component functions as a tool that uses a benchmark of synthetic corruptions that would be relatively similar to a neural network to natural corruptions in order to assess the robustness of a

Table B.6: Components vs supported Data type.

Use Case	331	332	333	334	335	3141	4192
Images	✘	✘	✘	✘	✘	✘	✘
Tabular	✘	✘	✘	✘	✘	✘	✘
Time-Series	✘	✘	✘	✘	✘	✘	✘
NLP	✘	✘	✘	✘	✘	✘	✘

Table B.7: Components vs supported Model type.

Use Case	331	332	333	334	335	3141	4192
Tensorflow	✘	✘	-	✘	✘	-	✘
PyTorch	✘	✘	-	✘	✘	-	✘
ONNX	✘	✘	-	✘	✘	-	✘
NNET	✘	✘	-	✘	✘	-	✘

given model. It is tested on Welding Renault. It evaluates the accuracy drop with the corruptions on the benchmark, and can give impact on the score when the severity of the corruption is modified. The code used to test the benchmark and test some severity modifications is available.

Component 335: Time-series Robustness Characterizer Component

A method to evaluate the robustness of the models on the Welding and Time Series use case using amplification methods on the corpus with noise functions. The used functions are scripts written in Python and are provided with the documentation on how to use them. For the Time Series use case, a frequency keyed noise function is provided. For each of these functions, a sample of noise data will also be included as well as the plot describing the evaluation results.

Component 3141: Chiru

Chiru is a tool developped by Kereval to assess the performance of AI models against perturbation to its input such as Gaussian noise and Gaussian blur. It proposes a graphical interface to visualise these results. For more information, we invite readers to refer to the deliverable Chi (2022).

Component 4192: ML watermarking

The component enables black-box watermarking of ML models in order to reinforce ownership protection. The objective is to investigate how ML watermarking can be used to protect the ownership rights of models creators and ensure traceability of ML models. This component is not about evaluating traditional ML models but rather assessing watermarked models.
The component includes three main categories of attacks against watermarking. The first is the *watermark removal*, where an attacker tries to remove the watermark from the model. The second is the *ambiguity attack*, where an attacker will cast doubt on the legitimate ownership by providing counterfeit watermarks. The third is the *evasion attack*, where an attacker will try to escape watermarks verification and therefore disable the model identification. For more information, we invite the readers to refer to deliverable EC4 (2022).

General comments

For the robustness evaluation components, there are two important comments regarding the criterion of supported models:

- By default, the Chiru component (3141) does not support any models. It is indeed the user's responsibility to add them.
- As for component 333, it only modifies the images. The evaluation of models on the modified images should be conducted separately.

Table B.8: Applicability of components to use cases.

Use Case	331	332	333	334	335	3141	4192
Welding Inspection (Renault)	✘	✘	✘	✘	✘	✘	✘
Demand forecasting (Air-Liquide)	✘	✘	✘	✘	✘	✘	✘
Scene understanding (Valeo, detection)	✘	✘	✘	✘	✘	✘	✘
Scene understanding (Valeo, segmentation)	✘	✘	✘	✘	✘	✘	✘
ACAS-Xu (Airbus)	✘	✘	✘	✘	✘	✘	✘
Aerial Photograph Interpretation (Thales LAS)	✘	✘	✘	✘	✘	✘	✘
Visual Industrial Control (Safran)	✘	✘	✘	✘	✘	✘	✘
Visual Similarity (ATOS)	✘	✘	✘	✘	✘	✘	✘
NLP Opinion mining (Renault)	✘	✘	✘	✘	✘	✘	✘
Cylinder Counting (Air Liquide)	✘	✘	✘	✘	✘	✘	✘

B.3. Improve Robustness

Improving robustness refers to the process of enhancing a system's ability to maintain stable and effective performance in the face of various challenges, uncertainties, or adversarial conditions. The purpose of these components is to demonstrate the effectiveness of retraining the model using these techniques, specifically in terms of robustness compared to traditional training. During the training process, there is nothing preventing the user from using an old pre-trained model as a starting point to retrain their model with these robustness improvement techniques.

Table B.9: Components vs supported Data type.

Use Case	Adversarial Training	Randomized Smoothing (Regression)	Randomized Smoothing (Classification)	DEEL-LIP	Neural-DE
Images	✘	✘	✘	✘	✘
Tabular	✘	✘	✘	✘	✘
Time-Series	✘	✘	✘	✘	✘
NLP	✘	✘	✘	✘	✘

Table B.10: Components vs supported Model type.

Use Case	Adversarial Training	Randomized Smoothing (Regression)	Randomized Smoothing (Classification)	DEEL-LIP	Neural-DE
Tensorflow	✘	✘	✘	✘	✘
PyTorch	✘	✘	✘	✘	✘
ONNX	✘	✘	✘	✘	✘
NNET	✘	✘	✘	✘	✘

Component 451: Adversarial Training Component

Naive deep networks are vulnerable to so called adversarial attacks or evasion attacks. These attacks take place at test time and intent to modify the output of a frozen deep network by modifying the input. The component 451 trains neural network to make them locally robust against adversarial attacks, the result also provides a guarantee over an empirically estimated robustness.The component includes several training methods :

- Classical training (Vanilla): it simply performs the gradient descent on the clean data.
- Auto Attack training: Model training against Auto-PGD (Projection Gradient Descent) attacks.
- Trades: Model training using trade-off between accuracy and robust accuracy. It implements the trades cost function Zhang et al. (2019).
- Fire: Model training similar to Classical training but implementing the Fire cost function .

Component 461/462:

Randomized Smoothing is a technique commonly employed in machine learning to enhance the robustness of models, particularly in the context of adversarial attacks. The fundamental idea behind Randomized Smoothing is to introduce a controlled level of noise or randomness to the input data during both training and inference. This serves to create a more resilient model that is less sensitive to small perturbations in the input. In this context, two components are proposed:

- Component 461: Randomized Smoothing for regression
- Component 462: Randomized Smoothing for Classification

For more information, we invite the readers to refer to deliverable EC4 (2021, 2022).

Table B.11: Components List of improving Robustness.

ID	Name	Type	Covered problems	Input format	UCs tested	Technical Maturity	Functional Maturity	Stage
451	Adversarial Training Component	Python Library	Classification	PyTorch Images	Renault Welding	2	1	Train
461	Randomized Smoothing for regression	Python Library	Demand Forecasting Regression	Time series	Air Liquide Demand Forecasting	TBD	TBD	Inference
462	Randomized Smoothing for Classification Component	Python Library	Classification	PyTorch Images	Renault Welding	2	1	Inference
4111	DEEL-LIP	Python Library	Classification Segmentation Detection	Keras/TF Images	Renault Welding Safran - Vis. indus. control Valeo - Scene understanding	2	1	Train
4205	Neural Differential Equations	Python Library	Classification	Image Time series	Air liquide Demand Forecasting	2	1	Inference

Component 4111: DEEL-LIP

This component trains a neural network that is able to compute, locally and for each input, an associated minimal radius of robustness. The robustness of the network can then be assessed to choose the operational trade-off between performance and robustness. The constructed models as 1-Lipschitz networks. For more information, we invite the readers to refer to deliverable EC4 (2021, 2022).

Component 4205: Neural Differential Equations (NODEs)

Evaluating robustness of NODEs is particularly challenging: their output is computed via iterative optimization schemes and such test-time optimization has shown to prevent the proper evaluation of established robustness methods designed for static networks like autoattack in the adversarial context EC4 (2022); Croce et al. (2022).
This component proposes a purification preprocessing Tool against adversarial and environmental attacks. Currently working on synthetic snow corrupted images with a ConvMixer OOD detection module.

Table B.12: Applicability of components to use cases.

Use Case	Adversarial Training	Randomized Smoothing (Regression)	Randomized Smoothing (Classification)	DEEL-LIP	Neural-DE
Welding Inspection (Renault)	✘	✘	✘	✘	✘
Demand forecasting (Air-Liquide-time series)	✘	✘	✘	✘	✘
Scene understanding (Valeo, detection)	✘	✘	✘	✘	✘
Scene understanding (Valeo, segmentation)	✘	✘	✘	✘	✘
ACAS-Xu (Airbus)	✘	✘	✘	✘	✘
Aerial Photograph Interpretation (Thales LAS)	✘	✘	✘	✘	✘
Visual Industrial Control (Safran)	✘	✘	✘	✘	✘
Visual Similarity (ATOS)	✘	✘	✘	✘	✘
NLP Opinion mining (Renault)	✘	✘	✘	✘	✘
Cylinder Counting (Air Liquide)	✘	✘	✘	✘	✘

B.4. Conclusion

In this chapter, we have presented the robustness components proposed in the Confiance.ai program. We have grouped the components into three categories: formal evaluation, evaluation against adversarial attacks, and improvement of robustness. For each component, we have presented its compatibility with different types of models, various types of AI models, as well as its applicability to the use cases of the program.

C. Robustness Platform Architecture

C.1. Introduction

In this chapter, we present the architecture (Version 1) of our robustness platform. To simplify the understanding of the platform's operation and the services it offers, we begin with a section describing the use cases, inputs, and outputs for each use case. Next, we detail the functional and technical specifications of the platform. Instead of structuring this chapter as a typical software development chapter with conventional diagrams, we have chosen to simplify by using our own diagrams to describe the various components of the platform.

C.2. Platform service descriptions

As mentioned in Chapter B, the robustness platform offers three services or use cases. In the initial version, only the "Robustness Improvement" functionality will be integrated. The other two functionalities will be added in a later version. For each service, we describe the general flow of usage, user requirements, required data, data type, AI model, model type, and the various components that the user could employ. The service descriptions in this chapter are general.

C.2.1 Formal Robustness Evaluation

The formal evaluation of robustness is a relatively new field and has not yet been widely applied to all use cases, i.e., all types of data and all types of models. In this version of robustness platform, only use cases with Image or tabular input formats are supported, and they come with certain constraints.
To test the formal evaluation service, the user should provide an AI model, data, and a robustness property to verify. The service's output is to check if the robustness property is satisfied. In the case of an image input, the property will be to verify if the AI model still classifies correctly even when a perturbation is applied. For example, in the Renault Welding use case, we aim to verify whether the neural network maintains accurate classification of the input image even when subjected to a perturbation. In summary:

- **Input :** AI model, welding image, and perturbation (this is the maximum disturbance that could be applied to the image).
- **Output :** Check if the model is robust or not against the perturbation.

Depending on the model's complexity and input dimension, the user chooses the most appropriate formal method. For users working with image inputs and an AI model, we recommend the following points:

- The AI model should be a neural network model.
- Using a neural network in ONNX format.
- The image dimension is approximately about 100x100 pixels. This is an empirical result based on hardware and software constraints. Indeed, for images of large dimensions (such as those in the Welding use case), formal methods struggle to apply normally. These findings have been discussed in this deliverable EC3 (2021a).

Under these three conditions, users can employ components 321 (Saimple) and 3171 (PyRAT) to formally evaluate their model. The Figure C.1 provides a visual summary of the information mentioned above.

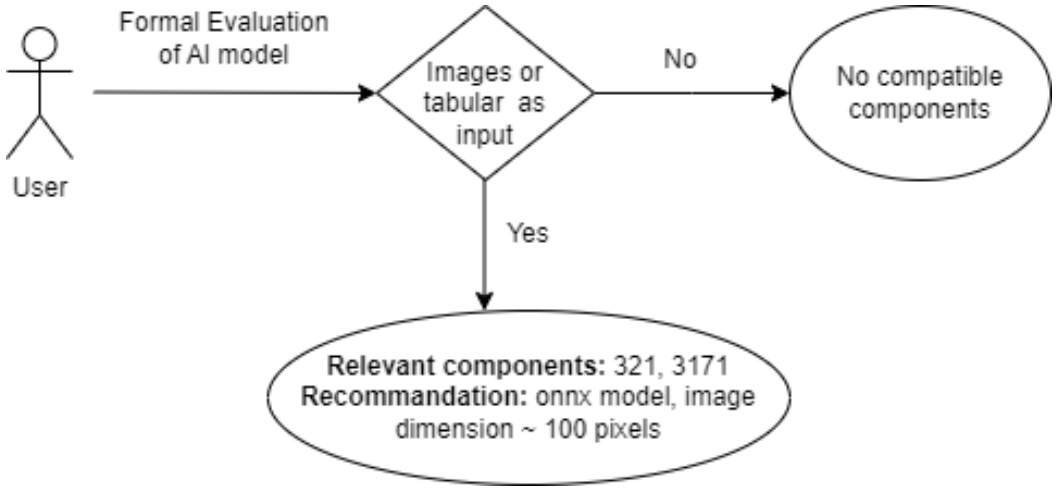

Figure C.1: Recommendation for Formal Robustness Evaluation.

C.2.2 Empirical Robustness Evaluation

The platform introduces a Empirical Robustness Evaluation service designed to assess the resilience of AI models against adversarial attacks and input perturbations. This service aims to comprehensively understand the capacity of robustness of the model. To assess the empirical robustness of an AI model through this robustness platform, users can evaluate two types of data: time series (component 335) and images.
For image input, this platform provides the capability to assess robustness against various types of perturbations, including:

- Adversarial attacks (component 331)
- Metamorphic and geometric perturbations (AIMOS, components 332)
- Synthetic corruption (component 33)
- Input perturbations (Chiru, component 3141)
- Watermarking (component 4192)

Regarding components focused on images, we recommend using models in the ONNX or TensorFlow/Keras format to test a wide range of perturbations. In contrast, for the TimeSeries components, we suggest using the ONNX or PyTorch format.
To test the empirical robustness evaluation service, the user should provide an AI model, data, and a perturbation type. The service's output is to check if the input model is robust against the input perturbation. The Figure C.2 presents a summary of the components of empirical robustness evaluation. It guides the user in choosing the most suitable component for their use case based on the type of data and model.

C.2.3 Improve Robustness

We have identified four components, from the confiance.ai program, to improve robustness of an AI model. In general, we distinguish between components compatible with PyTorch and those

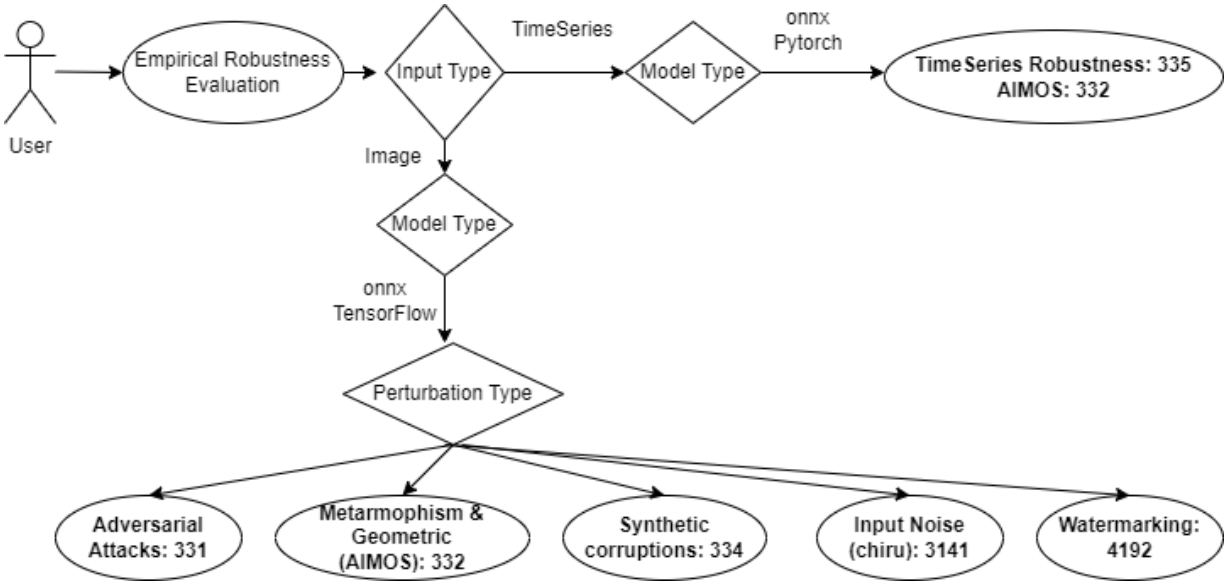

Figure C.2: Recommendation for Empirical Robustness Evaluation.

with TensorFlow. Therefore, the model's format type imposes a constraint when selecting compatible robustness enhancement methods.

- Compatible with PyTorch: 451, 461, and 462
- Compatible with TensorFlow: 4111

For PyTorch models, the platform offers two techniques to enhance robustness: Adversarial Training (451) and Randomized Smoothing (461 for regression tasks and 462 for classification tasks). For TensorFlow models, component 4111 DEEL-LIP is compatible with classification, detection, or segmentation tasks.

With the exception of component 462, which is compatible with regression tasks using time series data as inputs, components 451, 462, and 4111 are exclusively compatible with image inputs. The Figure C.3 provides a visual representation of the information outlined above.

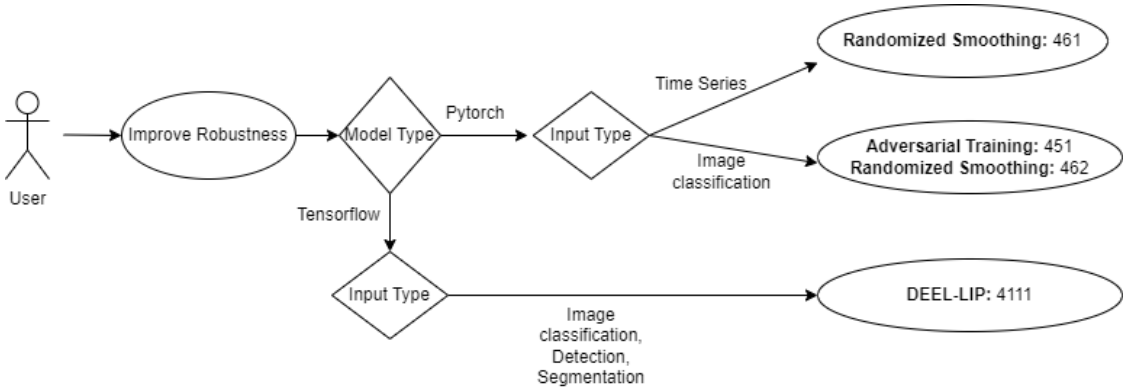

Figure C.3: Recommendation for Improve Robustness Task.

C.3. Functional Specification

A functional specification, commonly referred to as a functional spec, is a document detailing the features and functionalities of a software application or system. Serving as a blueprint for the development team, it provides a comprehensive outline of how the software is expected to behave and enumerates the features it should incorporate.
The platform offers three functioning services (Fig.C.4):

- Robustness improvement
- Formal Evaluation of Robustness
- Empirical Robustness Evaluation

The evaluation results (performances) can be visualized in a graphical interface or another format. In this initial version, we are considering the integration of only the 'Robustness Improvement' service. This decision is primarily driven by resource constraints.

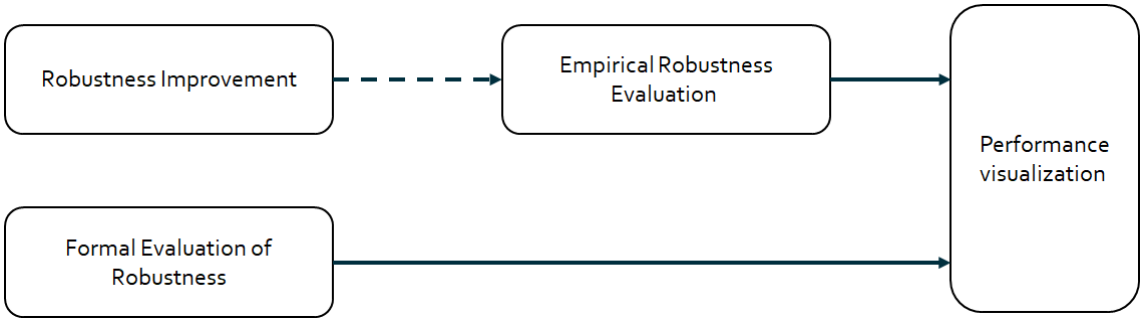

Figure C.4: Robustness Platform Services

The following sections delve into the functional aspects of each service, detailing the various sub-functions that transform the input data into the desired output. This comprehensive analysis encompasses all three services. However, only the 'Robustness Improvement' service has been developed and tested for program use cases.

C.3.1 Details of service: Robustness Improvement

This platform provides a service called "robustness improvement" that allows users to train an AI model based on neural networks to be robust to adversarial attacks. In this section, our focus is on component 451 (Adversarial Training). The service takes a dataset as input and outputs a model that is supposed to be more resistant to adversarial attacks. To execute the service, several subfunctions must be executed, from reading the dataset to fitting the model.
Figure C.5 shows the pipeline of the different stages of the robustness improvement service. In this version of the platform, as we focus on this service , we provide more technical and functional details later in this chapter. The main subfunctions of this service are:

1. **Data preparation:** preprocessing the data through methods specific to ML libraries (such as torch.utils.data.DataLoader) to facilitate access to samples and enable better performance for model execution.
2. **Selection of training method:** Classical, AutoAttack, Fire or TRADE.
3. **Adversarial training:** The application should allow the training of hr AI model.The architecture of the output model must be provided as input for this service. The user should choose an architecture from a list of available architectures. On the frontend, a message should appear to indicate when the training is initiated, in progress, and when it is com-

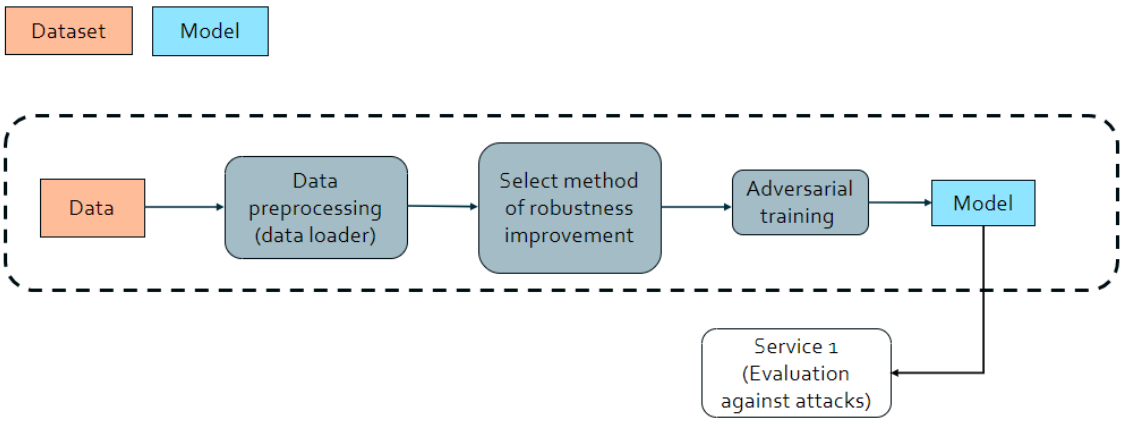

Figure C.5: Details of service: Robustness Improvement

pleted.

C.3.2 Details of service: Empirical Robustness Evaluation against input perturbation including adversarial attacks

The platform offers a second service called "Evaluation against adversarial attacks," allowing users to assess the robustness of an input against perturbations (input perturbations or adversarial attacks). Users can evaluate either the previously trained and returned model from the first service or assess a different model. When users wish to consecutively utilize both "Robustness improvement" and "Evaluation against adversarial attacks" services, compatibility challenges should be addressed.

To execute the service, several subfunctions need be executed, from reading the dataset and the model to visualizing metrics. Figure C.6 illustrates the pipeline of different stages in the Evaluation against adversarial attacks service. In this platform version, the components of this service are not integrated primarily due to resource constraints. The main subfunctions of this service are:

1. **Dataset and model selection:** the user should choose the model to test and the dataset from the frontend.
2. **Model format verification:** The format of dataset and model should be checked.
3. **Adversarial sample generation:** The method, if applied, should allow sampling the dataset to which the evaluation is applied.
4. **Selection of attacks and metrics:** The application should provide the user with a selection of perturbations to apply to the data or the model in order to assess their robustness. Moreover, The user selects the metrics of interest from the list provided. In this phase, the user should select a set of metrics suitable for their use case from the list of possible metrics. It would be beneficial in a future version if the user could easily add their own metrics, even if those metrics were not initially available.
5. **Evaluation of perturbations and generation of metrics reflecting model robustness:** After selecting and applying perturbations to the data or the model, the application should evaluate the effect of these perturbations on the model's performance. Moreover, the application should provide tools for comparing different combinations of perturbations and identifying those that have the most impact on the model's performance.
6. **Export to DebiAI:** The application should allow the user to export data, models, perturbations, and metrics to the DebiAI system. The exportation should support a format compati-

ble with DebiAI to ensure compatibility between the two systems. The user should be able to specify export parameters and configuration options, if applicable.

7. **Graphical user interface:** The application should provide an intuitive graphical interface to facilitate the user's interaction with features and data. The graphical interface should allow the user to navigate between different stages of the process, select configuration options, view results, and manage exports to DebiAI.

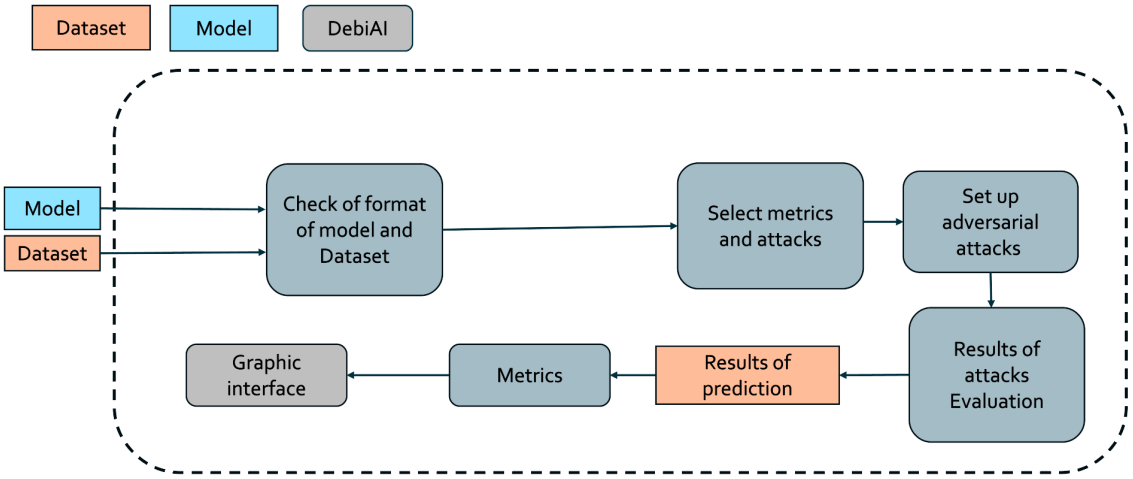

Figure C.6: Details of service: Evaluation against adversarial attack

At the beginning of this activity, we initiated the maturity assessment of component 331 (Adversarial Attack Characterization), which falls under the category "Evaluation against adversarial attacks". Due to resource constraints, the maturation of the component 331 (Adversarial Attack Characterization) has not been implemented. For this version of the platform, only component 451 (Adversarial Training) of the "Robustness improvement" service has been practically applied and implemented.

C.3.3 Details of service: Formal Robustness Evaluation

The platform offers a third service called "Formal Robustness Evaluation," enabling users to assess the formal robustness of an input model. As mentioned in the previous section related to the "Evaluation against adversarial attacks" service, model evaluation is conducted only against specific perturbations known as adversarial attacks or input perturbations. The aim of this service is to formally evaluate the model's robustness. In other words, the objective is to assess the model's robustness against all possible perturbations. Given the infinite nature of potential perturbations, the strength of formal methods lies in their ability to evaluate this infinite set of possible perturbations. For resource constraints, this service will be addressed in a future release.

Figure C.7 illustrates the pipeline of different stages in "Formal Robustness Evaluation" service. In line with the other services, this service requires the execution of a set of subfunctions as follows:

1. Check the format of the model
2. Selection of formal method
3. Metric selection
4. Performance visualization

It is worth mentioning that the proposed components for formal robustness evaluation are only

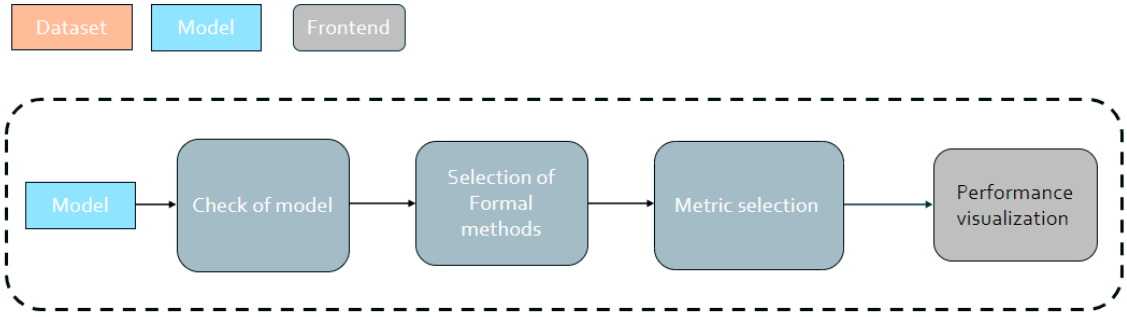

Figure C.7: Details of service: Formal Robustness Evaluation

compatible with images or tabular data. The calculated performance assesses the model's ability to maintain its decision even when inputs are noisy.

C.4. Technical Specifications

C.4.1 Environment Diagram

In this section, we present the robustness platform within its environment. A preliminary version has been proposed, outlining the positioning of the robustness platform in the technical environment of the Confiance.ai program. The Figure C.8 illustrates the interaction of the robustness platform, delivered as a Python library, with other technical components, namely, DebiAI, Algo HUB, the API, and the data server. The architecture has not been sufficiently detailed in this iteration. In this deliverable, our focus is on the maturity enhancement aspects, but in subsequent versions, these mature components should be integrated into the robustness platform.

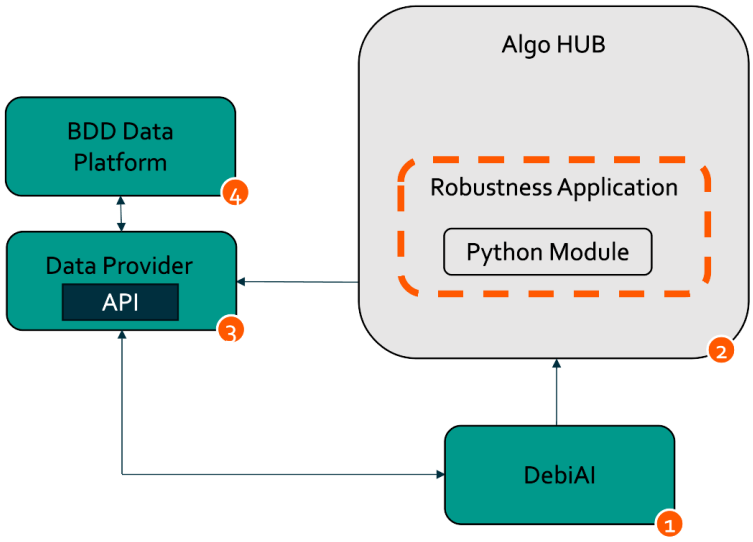

Figure C.8: Diagram Environment of the robustness environment

C.4.2 Details of component 451

The ADVT component (Adversarial Training) aims to enhance the robustness of computer vision models. In this regard, it consolidates a formalized set of training methods against adversarial attacks. The robustness of a model is defined by its capacity to remain unaffected by a modification in its input.

C.4.2.1 Component Pipeline

The challenge is to bring the component to the target level of maturity by providing a Python library that will ensure:

1. Generic data pre-processing
2. Generation of a computer vision model architecture
3. Selection of a training method for adversarial or non-adversarial scenarios
4. Training the model based on the previous choice
5. Visualization of robustness metrics in the form of graphs.

C.4.2.2 Generic data pre-processing

While not entirely generic, we have introduced a generic data pre-processing method tailored for two specific use cases: Welding (Renault) and Inspection (Safran). The complete generic data pre-processing will be included in a future version, with the goal of achieving functional maturity at level 4.

C.4.2.3 Model Architecture Generation

ADVT (component 451) will enable the generation of one of the following four architectures:

- Resnet: residual neural network
- Convnet: convolutional neural network
- Regularized Resnet with the Dropblock method
- Regularized Convnet with the Dropblock method

DropBlock is a structured form of Dropout designed to regularize convolutional networks. Dropout is a technique used to mitigate overfitting during model training.
The current version of the component does not allow the use of other architectures. The choice of architecture is fixed. In a future release, we aim to mature the component so that it can take the user's own architecture as input for training.

C.4.2.4 Model Training

The component 451 (Adversarial Training) provides four learning methods:

- Classical training (Vanilla): Standard non-robust training against adversarial attacks
- Auto Attack training: Model training against Auto PGD (Projection Gradient Descent) attack
- Trades: Training similar to Classical training but implementing the TRADES cost function
- Fire: Training similar to Classical training but implementing the Fire cost function

C.4.3 General Information

The table C.1 displays general information regarding the type of documentation, programming language, maturity levels, and testing framework.
The main outlines of the maturation of the component are as follows:

- Enhance code quality by adhering to the criteria set by the EC1 team.
- Improve documentation quality to facilitate its use.
- Transform the component into a library.
- Apply the component to two program use cases.

Confiance Attribute	Robustness
Technical maturity level	3
Functional maturity level	3
Documentation	Sphinx
Language	English
Test Framework	pytest

Table C.1: General Information of component 451 (Adversarial Training)

C.5. Conclusion

In this chapter, we have presented a general architecture of our robustness platform. Additionally, we have outlined the functional and technical specifications of the platform, as well as the various services offered by it. In the final section, we described the Component 451 (Adversarial Training), the only component reaching functional and technical maturity level 3. The maturity assessment of the remaining components is planned for a future version. In the next chapter, we will showcase the demonstration of Component 451 (Adversarial Training) on the two use cases: Welding Renault and Inspection Safran.

D. Demonstrators

The library requires input data in the dataloader format, a blank model with a specified architecture and other parameters associated with the training phase, as outlined in the Sphinx documentation. As output, the library will return the model in PyTorch format. The applicability of ADVT methods to different use cases (Welding Renault and Inspection Safran) was considered in the component's maturation process.
In this chapter, we present demonstrations of the robustness platform, specifically Component 451, on the two use cases.

D.1. User Guide

D.1.1 Installation

This library requires python 3.8 or later, and is accessible with

```
import advt
```

After having:

```
git clone git@git.irt-systemx.fr:confianceai/ec_4/as05_uc_renault.git
pip install -e <advt_dir>
```

D.1.2 Usage

To import libraries of training (adversarial or classical) stored in the *advt* library, and the available architecture of models in output:

```
from advt.models import ConvNet, ConvNetDropblock, ResNet
from advt.training.adversarial_training import AdversarialTraining
from advt.training.classical_training import ClassicalTraining
```

D.2. Demonstration on Welding (Renault)

D.2.1 Data Pre-processing

The data preprocessing is beyond the scope of this activity. Our library "advt" takes inputs in the form of dataloaders. Currently, this conversion from raw data to dataloaders is compatible with use cases such as Renault Welding and Safran Visual Industrial Control. In a future release, we

aim for functional maturity level 4, and in this scenario, we desire our library to be as generic as possible. We aim for the data reading and conversion to dataloaders to be generic as well. ci-dessous un exemple de conversion de données en dataloader et pour plus d'information nous invitons le lecture à consulter le notebbok sur cette adresse [1]

```
USECASE = "renault"
SCENARIO_1 = "part1"
SCENARIO_2 = "part2"
ROOT_DIR =
    "/home/shared/EC1/UC_Renault_Welding_Inspection/Full_dataset/labelled"

train_dataloader1, val_dataloader1 =
    data_Renault.get_train_dataloader_renault(
    ROOT_DIR,
    path_data+"train_dataset_all.csv",
    path_data+"val_dataset_all.csv",
    batch_size=16,
    train_transform=train_transform,
    test_transform=test_transform,
)
```

D.2.2 Parameters selection

In this phase, we choose the training parameters, essentially including the model architecture in output, the adversarial training method, and whether the learning will take place on a CPU or GPU.

```
DEVICE = "cuda:1"
ARCHITECTURE = "restnet"
LEARNING_RATE = 1e-4
model = model_choices.get(ARCHITECTURE, default_action)(DEVICE)
```

D.2.3 Model training

For the training phase, the user should specify the adversarial training method and invoke the training function as shown in the following command:

```
METHODE="adv_training"
EPSILON = 8 / 255
trainer = AdversarialTraining(model, optimizer, criterion, DEVICE, EPSILON)
```

[1] https://git.irt-systemx.fr/confianceai/Demonstrators/-/blob/develop/Comp451/*UC_Renault_Welding/training.ipynb?ref_type=heads*

E. Conclusion and Future work

In this activity, we proposed a robustness platform with three services: formal evaluation of an AI model, evaluation of the AI model against adversarial attacks and input perturbations, and improvement of robustness.

The goal of the platform is to guide AI model users and designers in choosing methods compatible with their data types and the technology of their AI model.

In this first version of the platform, we integrated component 451 (Adversarial Training). The main activity of the development team was to move from a functional maturity level of 1 and a technical maturity level of 2 to a level of 3, whether technical or functional.

The technology of AI models (TensorFlow, ONNX, PyTorch) and the types of data (image, tabular, time series) remain a limitation of the robustness components. Each tool is only functional with certain technologies.

In future work, we want to generalize the robustness platform to a large number of AI model technologies and data types. Also, in future work, we want to increase the maturity of other robustness components.

Bibliography

(2021). EC4 FA6 FA9 FA10 and FA11: Robust & embeddable deep learning by design. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC4/EC4_Robust_Embeddable_Deep_Learning_by_Design.pdf?csf=1&web=1&e=VBMFEf.

(2021). EC3 FA1 WP2: SoTA Robustness - Data based AI characterization. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC3/EC3.1%20-%20SotA/ISX-EC3-LIV-1337_L3.2.1.1_Etat%20de%20l%E2%80%99art%20sur%20les%20techniques%20de%20caract%C3%A9risation%20d%E2%80%99une%20IA%20%C3%A0%20base%20de%20donn%C3%A9es%20(EC3.1%20-%20WP2%20-%20SotA%20Robustness%20-%20Data%20based%20AI%20characterization).pdf?csf=1&web=1&e=9Pn2Uq.

(2021a). EC3 FA2: Exploitation of formal methods for characterization and safety. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC3/EC3_FA2_m%C3%A9thodes_formelles_v1.pdf?csf=1&web=1&e=wa70uc.

(2021b). EC3 FA3: Local characterization of white/black/grey-box ai. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC3/EC3_FA3_Caracterisation_locale_IAs.pdf?csf=1&web=1&e=S7KbAJ.

(2022a). EC3 FA14: Report on the maturation of local caracterisation tools. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC3/EC3_N14_Maturation_of_local_caracterisation_tools.pdf?csf=1&web=1&e=w1zdF1.

(2022). EC3 FA14: Résultats finaux de chiru. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC3/EC3FA14_Report_on_Chiru_Evaluation.pdf?csf=1&web=1&e=Fw4lgV.

(2022b). EC3 FA17: Report on pyrat maturation and evaluation. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC3/EC3_N17_Report_on_PyRAT_Evaluation.pdf?csf=1&web=1&e=utM8UM.

(2022). EC4 Macro FA Robustness: Methods and evaluation tools for robust ai in industrial applications. https://irtsystemx.sharepoint.com/sites/IAdeConfiance833/Documents%20partages/Forms/AllItems.aspx?id=%2Fsites%2FIAdeConfiance833%2FDocuments%20partages%2FGeneral%2FLivrables%20Documentaires%2FEC4%2FMethods%20and%20Evaluation%20tools%20for%20Robust%20AI%20V1%2Epdf&parent=%2Fsites%2FIAdeConfiance833%2FDocuments%20partages%2FGeneral%2FLivrables%20Documentaires%2FEC4.

Bak, S., Tran, H.-D., Hobbs, K., and Johnson, T. T. (2020). Improved geometric path enumeration for verifying relu neural networks. In *Proceedings of the 32nd International Conference on Computer Aided Verification*. Springer.

Croce, F., Gowal, S., Brunner, T., Shelhamer, E., Hein, M., and Cemgil, T. (2022). Evaluating the adversarial robustness of adaptive test-time defenses.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Jang, U., Wu, X., and Jha, S. (2017). Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC 2017. ACM.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2017). Adversarial machine learning at scale.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94.

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I., and Edwards, B. (2018). Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069.

Ramzi Ben Mhenni, M. I. K. and Canu, S. (2023). Robustness of neural networks based on mip optimization. https://irtsystemx.sharepoint.com/:b:/r/sites/IAdeConfiance833/Documents%20partages/General/Livrables%20Documentaires/EC3/EC3FA9_Robustness_on_MIP_Optimization.pdf?csf=1&web=1&e=o417OK.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31:4939–4948.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. (2019). Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*.

Confiance ai

Title: Methodological Guideline for Robustness Functional Set

Keywords: Robustness, Formal guarantees, Formal Robustness evaluation, Empirical Robustness Evaluation, Robustness improvement, User recommendation, User Guideline

The objective of this activity is to provide an evaluation environment to AI model designers and evaluators. This environment consists of a set of components and methods related to robustness. This helps guide users in the design and evaluation of AI models in the presence of various selected disruptions. The aim is to choose the ones that are most suitable for a wide range of use cases, including tasks like image detection/classification and time-series analysis. The initial release of this platform incorporates component 451, namely adversarial training, to enhance robustness. Specifically, this component is employed to strengthen an AI model's resilience throughout its training.

Our partners:

Air Liquide AIRBUS Atos cea Inria
NAVAL GROUP GROUPE RENAULT SAINT EXUPÉRY SAFRAN sopra steria
SystemX THALES Valeo