AIMOS: Metamorphic testing of AI - An industrial application

Augustin Lemesle, Aymeric Varasse, Zakaria Chihani, Dominique Tachet

AIMOS: Metamorphic Testing of AI - An Industrial Application

Augustin Lemesle[1], Aymeric Varasse[1], Zakaria Chihani[1], and Dominique Tachet[2]

[1] Université Paris-Saclay, CEA, List F-91120, Palaiseau, France
firstname.surname@cea.fr
[2] Production Engineering Industrial System Renault, F78280, Guyancourt, France
firstname.surname@renault.com

Abstract. Despite initial fears of a renewed AI winter, the current summer seems to have no end in sight. AI in general and Deep Learning in particular are permeating a growing number of our daily applications with an undeniable added value. As with Computer Science itself, this increasing entanglement with our lives calls for a special attention to safety and demands adequate verification and validation methods and tools to assist in the development of reliable AI. In this paper, we present the AIMOS tool as well as the results of its application to industrial use cases. Relying on the widely used Metamorphic testing paradigm, we show how the process of verification and validation can benefit from the early testing of models' robustness to perturbations stemming from the intended operational domain.

Keywords: metamorphism · testing · artificial intelligence · neural networks · support-vector machines · verification and validation

1 Introduction

Advocating for the added value of AI in many industrial applications is becoming as unnecessary as justifying the usefulness of computer science itself. Indeed, while the jury is still out on the future of some aspects such as Artificial General Intelligence, the stakeholders in general and the AI community in particular seem convinced that AI winters are a thing of the past. As a matter of fact, the recent developments, particularly in the subfield related to Neural Networks (NN), have shown a phenomenal ability to assist humans in numerous, albeit specific, tasks such as image recognition and some limited forms of command and control.

While the efficiency in these tasks is far from sufficient to completely replace all systems, they can still be of great value as subcomponents in more traditional cyber-physical systems such as vehicles and electronic devices. In some of these systems, reliability is more than a desirable feature, it can be a critical sine qua non for the adoption of an AI-based component. This makes any method and tool able to aid in the verification and validation process a useful commodity.

In this paper, we report on one such tool, AIMOS, specialized in metamorphic testing for AI, and its application on two industrial use cases.

1.1 Our contribution

Our contributions can be summed up by the following:

1. We propose a methodology to assess the stability of an AI system on a given dataset based on the specification and the test of metamorphic properties.
2. We provide a model-agnostic tool that implements and automates the entire process of applying these metamorphic properties on the inputs and outputs of an AI model, and of comparing and compiling the result of the subsequent test into a stability score which can then be presented in a graphical way.
3. We apply our method and tool to two industrial use cases with different metamorphic properties to show their usefulness.

1.2 Related work

Invented by T.Y. Chen and others in a technical report in 1998, later republished [2], metamorphic testing was then applied to many domains, from embedded systems [13] to bioinformatics [9], including machine learning [4]. A comprehensive survey [10] of these applications was conducted, closely followed by another [3], this time by the original authors.

As traditional testing methods are inherently limited when applied to machine learning models, metamorphic testing has been considered as a viable alternative, particularly for critical uses of AI-based components. One of the predominant use case where metamorphic testing has been used is autonomous driving, with [5], which focuses on identifying implementation bugs in machine learning based applications, [12] and [14] that concentrate on deep neural networks for autonomous driving and [15] that uses metamorphic testing on Apollo (Baidu's self-driving vehicles on-board software).

From there, various other use cases have also been tackled, such as autonomous drones, in [8], where metamorphic testing is used in combination with model based testing to test autonomous drones. Another use case is neural machine translation (NMT) models, where in [**?**], another metamorphic testing approach is proposed, called structure-invariant testing, specially designed for NMT models. In [**?**] mutation is combined with metamorphic testing to detect inconsistency bugs in NMT models, to provide tests for real-world machine learning translation models and repair the mistranslations found during the testing phase.

While also based on metamorphic properties, and more specifically on geometric transformations, the work in [1], specific to neural networks, tries to formally verify a neural network against these metamorphic transformations. By computing linear relaxations for these transformations, they are able to prove, by using external formal verification tools, the robustness of the model around a set of selected inputs.

Overall, while there are plenty methods and tools to apply metamorphic testing to AI models, all of them have been designed either for a specific use case or for a specific technology (*e.g.* deep neural networks built with Keras), and therefore they lack adaptability for other AI systems or metamorphic properties.

In contrast to these methods, the tool presented here is self-sufficient, does not rely on other provers and is completely task agnostic. In a sense, AIMOS allows to rapidly test a much larger set of inputs in a much shorter time, which offers a simple and rapid early problem detection mechanism. In layman's terms, our testing is useful to cover a much larger ground surface, detecting unwanted mines, so to speak, with little cost and in little time, whereas the formal verification techniques such that of [1] can verify at a much deeper level a much smaller space.

2 Background

2.1 AI in this document

As the aim of AIMOS is to be as agnostic as possible, the necessary background to understand it and the work presented in this paper is henceforth quite small. In this document, we simply consider any type of AI model to be a function $p : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X}$ its input space and $\mathcal{Y}$ its output space. We call this function the inference function of the AI model. In that sense, we consider any AI model such as Neural Networks, Support Vector Machines (SVM), Decision Trees, ... as a black box to be tested by AIMOS. The inherent specificities of each model type will not impact our testing procedure.

2.2 Metamorphic property

Simply stated, the main idea behind metamorphic testing is that certain relationships (*e.g.,* R_1, R_2) on some inputs (*e.g.,* a, b, c) should induce, in a sound software S, other relationships (*e.g.,* R'_1, R'_2) on the outputs. For example:

$$\forall a, b, c, R_1(a, b) \wedge R_2(b, c) \to R'_1(S(a), S(b)) \vee R'_2(S(b), S(c))$$

Consider the most common example: a software S that computes the minimal cost of travel between two points, a and b, in an undirected graph. Even if the actual result of this operation is not known, it is possible to generate an arbitrary number of tests using the knowledge that the result should be impervious to symmetry. Here, the relation between the inputs (a, b) and (b, a) is the symmetry, and the relationship on the outputs $S(a, b)$ and $S(b, a)$ is the equality. Such a process can be considered a generalization of the data translations techniques mentioned above (rotations, flip, *etc.,*), since they can all be mathematically described, making them a subset of metamorphic properties. Crucially, focusing on metamorphic properties opens the possibility of more general data generation. Indeed, the relationships can be more complex. Consider an evolution S' of the software S above, that takes as input a point of origin o and a list L of points and computes the lowest cost for package delivery to all of them and come back to the origin. One can generate numerous partitions of L into lists $L_1, L_2, ...$, and the sum of the results of calls $S'(o, L_i)$ should be greater or equal than the result of $S'(o, L)$. (For the mathematically inclined: $\forall L_1, \ldots, L_n, L = \bigoplus_{i=1}^{n} L_i \to S'(o, L) \leq \sum_{i=1}^{n} S'(o, L_i)$, where $\bigoplus$ is the list concatenation).

To apply this concept, a metamorphic property will rely on the following set of functions:

- the input transformation $f_i : \mathcal{X} \to \mathcal{X}$, (*e.g.,* if f_i is used for R_1 above, then $f_i(a) = b$)
- the output transformation $f_o : \mathcal{Y} \to \mathcal{Y}$,
- and the inference transformation $f_{pred} : (\mathcal{X} \to \mathcal{Y}) \to (\mathcal{X} \to \mathcal{Y})$.

In that definition, f_i will be a transformation on the input space. This can be seen as a function that produces a perturbation on the inputs, either through adversarial attacks, symmetries, rotations or noise. Similarly, f_o represents the expected transformation on the outputs which could be the identity, a symmetry, *etc*. When checking for an unchanged classification for example, f_o will be the identity. These two functions are completed by a transformation on the inference function of the model, as this function can also depend on the setting, as it will be shown later with the ACAS Xu use case (Section 5.3). We note $p' = f_{pred}(p)$ to ease the notation, most of the time, however, f_{pred} will be the identity function as we infer the result keeping the same function. This function only targets specific use case with multiple decision models. From there, this metamorphic property can be applied to any given subset of $\mathcal{X}$ and a model as defined by its inference function $p(x)$.

3 Metamorphic testing

As mentioned, our approach aims at testing a given AI model on a dataset using metamorphic properties. As such, we present a tool named AIMOS, standing for AI Metamorphism Observing Software. This approach and tool are black-box as we do not use in any way the intrinsic characteristics of an AI model, we are treating the model as an inference function.

Our approach with AIMOS improves the usual approaches which only test a model against a perturbation without considering potential change to the output. Indeed, a transformation on the input space can also be linked to a transformation on the output space, *e.g.,* a symmetry on the input can mean a symmetry on the outputs as well (*e.g.,* a left arrow becoming a right arrow). A rough schema of the principle behind AIMOS is shown in Figure 1.

AIMOS is implemented in Python as it is the most commonly used language for AI nowadays to allow an easy interface with numerous AI frameworks, such as TensorFlow/Keras, PyTorch, scikit-learn, and others. By default, AIMOS supports different format for inputs (*e.g.,*.png, .jpg, .csv, *etc*) and for models (.h5, .pb, .onnx, .nnet) and implements different classical metamorphic properties

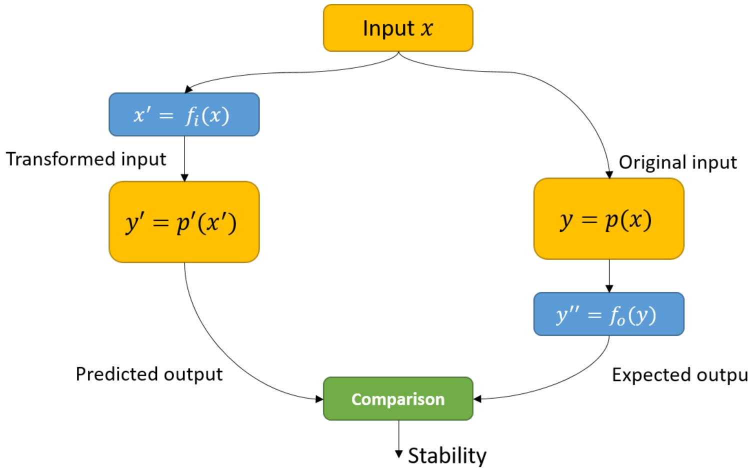
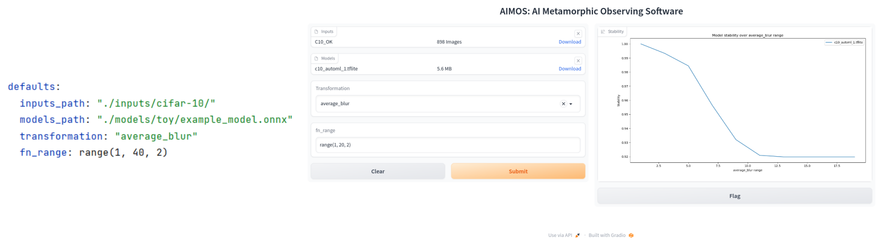


Fig. 1. AIMOS principle

(rotation, brightness, dead pixels, *etc.*,). It can be easily extended with specific loader or properties with simple Python functions.

Figure 2 shows an example configuration file of AIMOS to launch a simple test on CIFAR-10 easily. AIMOS aims to allow the user to provide a more flexible and complete framework for property testing thus specific efforts were made in order to simplify the various APIs in AIMOS such as this config file. In the same figure we show a simple developed graphical interface for AIMOS, once again to ease its usage.



```
defaults:
  inputs_path: "./inputs/cifar-10/"
  models_path: "./models/toy/example_model.onnx"
  transformation: "average_blur"
  fn_range: range(1, 40, 2)
```

AIMOS: AI Metamorphic Observing Software

Inputs: C10_OK 898 Images

Models: c10_automl_1.tflite 5.6 MB

Transformation: average_blur

fn_range: range(1, 20, 2)

Fig. 2. AIMOS user interfaces: a configuration file (left) and its GUI (right)

As we are agnostic of the model type used, AIMOS can compare different types of models or architectures and, in turn, provide incentives for a choice between one type or another. As such, this permits comparisons between similar models types but with different architectures (*e.g.*, one Convolution vs two Convolution layers) but also between very different architectures (*e.g.*, Neural Networks vs SVMs vs Decision Trees) in similar settings, or also of simply differently trained models.

The aim of this approach is to rapidly and widely test a given AI model on a given dataset. This testing can serve to exhibit inefficiencies of the models, compare them to gauge the most stable ones, *etc*. Thus, while being far from a panacea for validation (but, indeed, no tool can pretend to that status) AIMOS can be an important part of an AI development process, providing a framework to facilitate and automate a rapid testing procedure. The metamorphic properties themselves, as part of the testing procedure, should be carefully selected to provide a good criterion of evaluation for the model when tested with AIMOS. For instance, if the problem represented by the AI model does not present any axe of symmetry, it would be of little value to test it against that.

4 Use cases

4.1 Welding use case

The first use case considered for the testing of AIMOS is the control of the conformity of welds of rear axles on a vehicle production line of Renault at a factory in Le Mans. This control is realized by the analysis of the image of the weld by an algorithm which has been trained on weld images labelled by a professional operator.

In Renault's quality process, the need was expressed to test the performance of the algorithm beyond its accuracy. To assist in this task, test cases have to be defined in which the initial test images will be degraded following some specific criteria. These degradations were selected by Renault by analyzing their Operational Design Domain (ODD) for this system. Defined initially in the context of autonomous vehicles in the standard J3016, ODD can be extended to any autonomous system such as this algorithm. Therefore, the ODD defines the operating conditions under which a given system is designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and any other characteristic of interest.

In Figure 3, we can see the result of Renault's context analysis on the problem with both the fixed input parameters (AD) and the ODD parameters to take into account for testing.

Through the analysis with AIMOS of the algorithm results on the original quality image and on the degraded image, Renault aims to gauge the performance of a given model but also to define the tolerance of the system with a confidence indicator. This tolerance will then be used in the scope of other works in the monitoring of each characteristic of the ODD. Indeed, it bears repeating that such metamorphic testing is far from being a comprehensive answer to any and all validation process, and the properties it tests are not the be-all and end-all of safety assurances. But such a process remains an essential part of software validation and its extension to AI is actively pursued by industrial and academic actors alike.

Two models are currently being used by Renault depending empirically on where they achieve the best results:

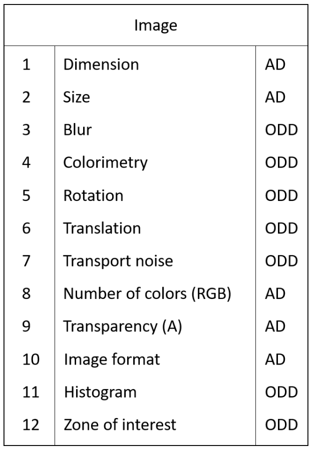


Image		
1	Dimension	AD
2	Size	AD
3	Blur	ODD
4	Colorimetry	ODD
5	Rotation	ODD
6	Translation	ODD
7	Transport noise	ODD
8	Number of colors (RGB)	AD
9	Transparency (A)	AD
10	Image format	AD
11	Histogram	ODD
12	Zone of interest	ODD

Fig. 3. ODD Renault Welding

– Models generated automatically through Google's AutoML framework.
– Specific models created by data scientists at Renault internal research and development lab, henceforth named RD.

These models have different architectures. The AutoML models are using a succession of 52 convolution layers including 17 depth-wise convolution and some residual connections. The RD models are built by combining a pre-trained mobilnet-v1 neural network on the ImageNet dataset to work as a feature extractor and Support Vector Machine (SVM) trained on the output of the first NN. Both models take the image as input after a resizing and have two outputs: whether the weld is OK or it needs to be reworked.

4.2 ACAS Xu use case

The second use case we will tackle is the widely known ACAS Xu as presented by [7] as an Airborne Collision Avoidance System for unmanned vehicles. As opposed to regular and large lookup tables producing advisory, this system uses a deep neural network to produce the advisory to avoid midair collisions. The approach was partly introduced to reduce the memory requirements for hardware of the original 2 GB tables down to 3 MB. The authors in [7] express concerns on the certification of such a system based on neural networks and, indeed, in such safety critical systems a basis of trust in those systems should be built.

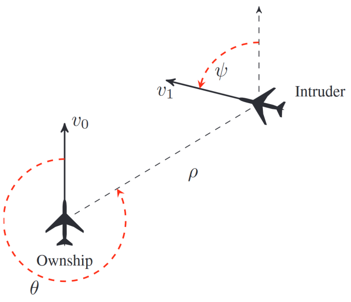


Fig. 4. ACAS Xu setting

Figure 4 shows the different parameters of the problem. Two additional parameters, τ (s), the time until loss of vertical separation and a_{prev} (°/s) the previous advisory, complete the 7 input parameters of this use case. Five advisories are possible for the system: "Clear-of-Conflict", "Weak right", "Strong right", "Weak left", or "Strong left", corresponding respectively to the heading rates 0°/s, -1.5°/s, -3.0°/s, 1.5°/s and 3.0°/s. By discretizing the last two inputs τ and a_{prev} with respectively 9 and 5 values, 45 networks are created, one for each combination. Each network is composed of five inputs and 6 hidden layers of 50 neurons each with ReLU activation functions. We denote their respective inference functions as $p_{\tau,a_{prev}}$.

As mentioned, the concerns on such a system are related to certification and how to show that the DNNs are well representing the properties of the original problem and tables. For example, the setting and the original tables in this use case are symmetric alongside the ownship heading direction. In that sense, we should observe here the same symmetry on the DNNs trained from the tables.

5 Experimentation

As AIMOS does not require more than the inference function of the selected AI model and its execution in terms of computing power, it can be run on a similar architecture as the one normally used for the inference of the model. In this section, all the experimentations were run on a 16 cores and 64 GB RAM server.

5.1 Renault Welding use case

For the Renault Welding use case, we will consider 3 different production lines called C10, C20 and C34 and their corresponding weld. As seen in Figure 5, each weld has different characteristics such as lighting conditions or orientation. All the images are initially Full HD images which are then resized for the inference with the models. Our test set is composed of several hundreds of images depending on the weld (898 for C10, 1069 for C20 and 1070 for C34).

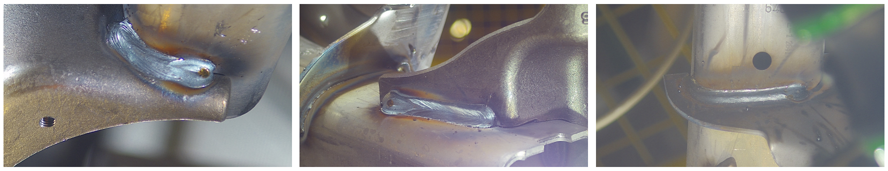


Fig. 5. Welds C10, C20 and C34 of a rear axles production line. (from left to right)

For the purpose of this use case, we compare the two available types of models on each of these production lines with five different AutoML trained models (TFLite) and one Renault own in-house model (pickle format). The accuracy of all models is on average over 97% and is thus sufficient to be able to correctly consider their stability to metamorphic properties.

For each of these models, we will focus on the blur perturbation as it is included in the ODD defined by Renault. The range chosen to test this perturbation is purposefully quite large to include the ODD of Renault but also

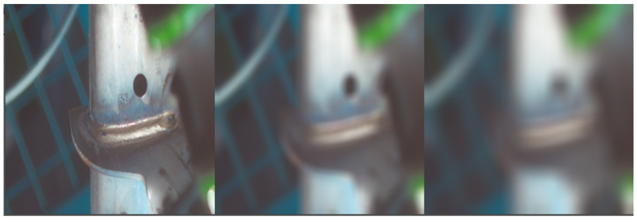


Fig. 6. Example of blur applied on a C34 weld image. Left is original image, middle has a kernel size of 10 and right of 20.

show results outside this zone. The resulting perturbations can thus also imply that the image should not be recognizable by the human or the model. Following this, the metamorphic property tested is the absence of classification change after applying the perturbation and are defined more formally as follows:

- $f_i(x) = blur(x, k)$ where blur is a convolution by a kernel with $k \in [1, 20]$ the kernel size chosen for the blur with a step of 1.
- $f_o(x) = x$ and $f_{pred}(x) = x$

5.2 Results on the Renault Welding use case

The blur modification is created using a normalized box filter which is convoluted to our original image through the OpenCV library. Given the size of the chosen filter each pixel is averaged with its neighbors and thus blurred. In Renault's setting this blur can come either from the sensors themselves, *i.e.,* the camera settings are slightly off, or from vibrations caused by the production line operation. For our testing, we will consider a filter size ranging from 1 to 20. The effect on the intensity of the blur can be seen on Figure 6.

As shown in Figure 7, the different AutoML models, which have the same architecture, present a similar trend in their response, *i.e.,* the stability of the model drops quickly for low perturbation then plateaus at a lower level. Some variations can nevertheless be seen between the models, such as with the first and third model of C20 or the fifth of C34, showing the differences here that the training can bring even at similar accuracy.

On the contrary the RD model on each weld presents greater stability than AutoML models up until a certain point, then drops drastically in stability. This drop, at varying kernel sizes for each weld, shows the limit of the model before it fails to even identify the weld and thus infers a rework is needed for everything (here the test set presents a majority of OK labelled welds). This clearly contrasts with the AutoML models which tend to classify everything as OK at high blur values. After further discussion with Renault, the results of the AutoML models is indeed problematic as it is overly robust and is not desirable for them. Indeed, at high blur level, which are not recognizable by a human, the default action should be to ask for a rework and not the validate the weld.

The different models' responses between the different welds also vary in intensity, showcasing that the setting of each weld can bring different responses.

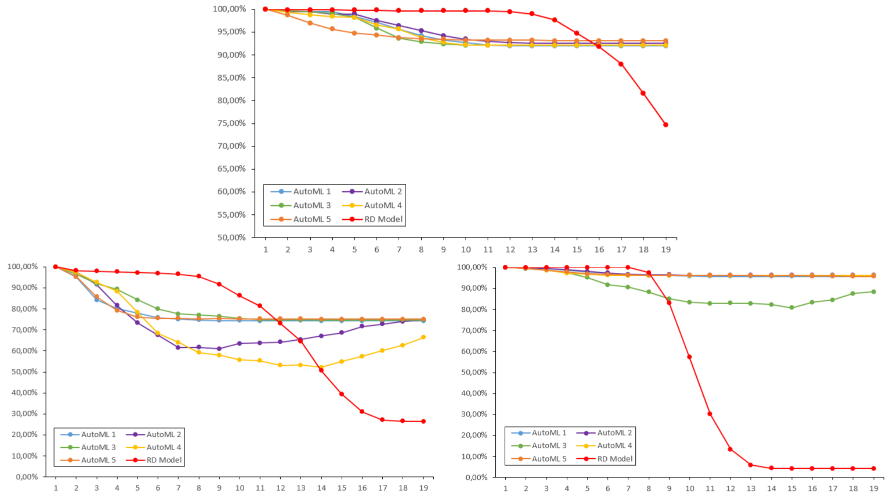


Fig. 7. Stability of the models on the C10, C20 and C34 welds for the blur property with kernel size ranging from 1 to 20. Value points are linked *only* for aesthetic consideration to improve readability as AIMOS does not perform interpolation.

C10 drops in stability are the most moderate with AutoML models staying above 90% whereas it goes down as low as 53% for C20's fourth model and 80% for C34's third model. A possibility here is that the blur is more present in the training dataset for certain production lines as it occurs more frequently there, leading to a better stability.

With these results from AIMOS, metrics can be defined on the stability at a given blur kernel and thus include this indicator in a quality process at Renault to validate or not a model. This could be further linked with other results on different perturbations such as brightness, rotation, *etc*. Of course, all of this should remain coupled with some performance indicator like the accuracy of the model to avoid selecting models outputting constant values (which would always be stable).

As a preliminary work, further testing were also made with AIMOS to compare different type of robust training for the models. Randomized smoothing or 1-Lipschitz models were thus compared to original models. Nevertheless, no clear conclusion were found as some of them appear overly robust like the AutoML models and other present varying degree of stability. A more complete study should be pursued and linked to the full ODD of Renault.

Overall, the chosen perturbation for this paper was kept simple for pedagogical concerns, and more complex perturbation (rotation on top of blur, *etc*) could be used. Even so, we already see here some notable differences between the various model types and architectures in the way they respond to these metamorphic properties. In fact, the results on these properties have shown the utility of this tool to Renault. We are currently collaborating to extended our work on the rest of the control elements as defined in Renault's Operational Design Domain. AIMOS would then be able to be integrated in Renault's quality process.

5.3 ACAS Xu use case

For the ACAS Xu use case, we tested the symmetry of all 45 models to a symmetry alongside the ownship heading axis. The models are available under the NNet format and AIMOS is able to directly read and infer from this format.

For our tests, we defined two test scenarios on different ranges of input. The first one is defined on the full range of the five inputs, and we pick 100 000 test points uniformly on this range. We remarked that more than 85% of these points' output was "Clear-of-conflict". As such while it should remain unchanged by the symmetry it did not allow to fully test this transformation on the output right and left. We then defined a second scenario and selected 100 000 uniformly on a more restricted part of the input space. This subspace was chosen so that a majority of the inputs in that subspace are predicted as either "Strong right" or "Strong left" or if τ is high as "Clear-of-conflict". For smaller τ, around 95% of the inputs is for each network either "Strong right" or "Strong left"

Finally, as explained previously, the 45 networks were discretized through the previous advisory as such when considering a symmetry on the system, the previous advisory should also be symmetrized, leading to inferring the result with a *different* network. This is the incentive behind the extension (mentioned in Section 1.1) of metamorphic testing from the usual transformations on the inputs and expected transformations on the output to include also transformations on models themselves. In that sense, we defined the metamorphic property tested here as follows:

- $f_i((\rho,\ \theta,\ \psi,\ v_{own},\ v_{int})) = (\rho,\ -\theta,\ -\psi,\ v_{own},\ v_{int})$
- $f_o(a) = -a$ with a in °/s
- $f_{pred}(p_{\tau,a_{prev}}) = p_{\tau,-a_{prev}}$.

5.4 Results on the ACAS Xu use case

Overall, the stability of the models *w.r.t.* to our defined symmetry metamorphic property is quite high on the full input space as the average stability over all 45 networks is above 97%, even the worst network drops only to 94%. Nevertheless, on a more restricted space we can see more difference *w.r.t.* to the previous advisory. Indeed, on average over the previous advisory we are at 89.7% stability with "Clear-of-Conflict" and with Left advisories we are at 95% while we are above 99.6% for Right advisories. This shows a clear imbalance between the models of left and right previous advisory. On the other hand, there does not seem to be any correlation between τ and the stability.

Figure 8 shows the differences between networks in function of τ when the previous advisory is "Clear-of-conflict", where the stability is the lowest. In fact, this comes from the 3 models with intermediate values of τ that show only 66%, 67% and 76% of stability to the symmetry. As the network should have trained on fully symmetrical tables, this results shows a discrepancy between the networks and the table and that further training might be necessary for these networks.

This second use case and the metamorphic property tested with AIMOS are presenting more complex and essential properties that one can test using

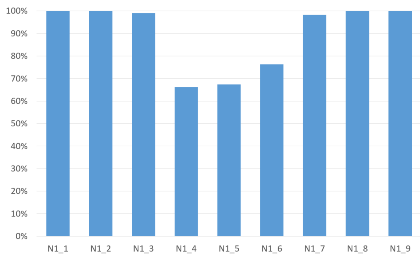


Fig. 8. Stability of the ACAS Xu models with ordered by τ with previous advisory being "Clear of Conflict" on the restricted space.

our tool. Indeed, such considerations, like the symmetry of the model trained, should certainly be considered when the dataset and what it represents are also symmetrical. In the same idea, rotation or translation properties can also be considered by AIMOS. All these types of properties can in turn help to design a more comprehensive dataset for further training if needed, *e.g.,* in case some configurations were underrepresented. As an additional note, such properties have also been tested in two other settings in maritime and avionics use cases (*e.g.,* with our Technip Energies partners, as a part of a larger reliability assessment, mentioned in [11]).

6 Conclusion

We presented AIMOS, a metamorphic testing tool specialized in AI applications, as well as the results of its usage in industrial contexts. Through these results, we hope to advocate for the inclusion of such tools in the verification and validation process of AI-based components. AIMOS will be made freely available for teaching and research purposes as a part of an ongoing effort to increase the awareness of reliability issues in the future generations of AI practitioners. It will also be integrated in more holistic open-source platforms for characterizing safety in AI systems such as CAISAR [6].

This participation in a greater effort for validation will allow AIMOS to collaborate more closely with other tools, such as formal provers. By integrating in a wider platform, we also hope to facilitate the application of AIMOS to the AI-based components of software products that also have non-AI-based components, which is essential for system-wide verification and validation.

Further future work, more intrinsic to AIMOS, is the coverage of more transformations. Several of our industrial partners have expressed the need for specific transformations, beyond the usual ones such as rotation and luminosity. These are related to the operational domains of our partners which prevents us from detailing them further.

Finally, we are investigating application to another type of use cases, very prevalent in the industry, which is time series. Here again, several partners are particularly interested in time-series-specific transformations and this is part of our medium-term plans.

Acknowledgments This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute. The AIMOS tool is also funded under the Horizon Europe TRUMPET project grant no. 101070038 and the European Defence Fund AINCEPTION project grant no. 101103385.

This preprint has not undergone peer review (when applicable) or any post-submission improvements or corrections. The Version of Record of this contribution is published in Computer Safety, Reliability, and Security. SAFECOMP 2023 Workshops, and is available online at https://doi.org/10.1007/978-3-031-40953-0_27.

References

1. Balunović, M., Baader, M., Singh, G., Gehr, T., Vechev, M.: Certifying geometric robustness of neural networks. In: Advances in Neural Information Processing Systems 32. vol. 20, pp. 15234 – 15244. Curran (2020). https://doi.org/10.3929/ethz-b-000395340
2. Chen, T.Y.: Metamorphic testing: A simple method for alleviating the test oracle problem. In: Proceedings of the 10th International Workshop on Automation of Software Test. p. 53–54. AST '15, IEEE Press (2015)
3. Chen, T.Y., Kuo, F.C., Liu, H., Poon, P.L., Towey, D., Tse, T., Zhou, Z.Q.: Metamorphic testing: A review of challenges and opportunities. ACM Computing Surveys (CSUR) **51**(1), 1–27 (2018)
4. Ding, J., Hu, X.H., Gudivada, V.: A machine learning based framework for verification and validation of massive scale image data. IEEE Transactions on Big Data (2017)
5. Dwarakanath, A., Ahuja, M., Sikand, S., Rao, R.M., Bose, R.P.J.C., Dubash, N., Podder, S.: Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM (jul 2018). https://doi.org/10.1145/3213846.3213858
6. Girard-Satabin, J., Alberti, M., Bobot, F., Chihani, Z., Lemesle, A.: Caisar: A platform for characterizing artificial intelligence safety and robustness. In: AISafety. CEUR-Workshop Proceedings (Jul 2022), `https://hal.archives-ouvertes.fr/hal-03687211`
7. Julian, K.D., Lopez, J., Brush, J.S., Owen, M.P., Kochenderfer, M.J.: Policy compression for aircraft collision avoidance systems. In: 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). pp. 1–10 (2016). https://doi.org/10.1109/DASC.2016.7778091
8. Lindvall, M., Porter, A., Magnusson, G., Schulze, C.: Metamorphic model-based testing of autonomous systems. In: 2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET). pp. 35–41 (2017). https://doi.org/10.1109/MET.2017.6
9. Pullum, L.L., Ozmen, O.: Early results from metamorphic testing of epidemiological models. In: 2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom). pp. 62–67. IEEE (2012)
10. Segura, S., Fraser, G., Sanchez, A.B., Ruiz-Cortés, A.: A survey on metamorphic testing. IEEE Transactions on software engineering **42**(9), 805–824 (2016)

11. Serge, D., Augustin, L., Zakaria, C., Caterina, U., François, T.: Reciph: Relational coefficients for input partitioning heuristic. to be presented at ICML's workshop on Formal Verification of Machine Learning (WFVML 2022) (2022)
12. Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars (2018)
13. Tse, T., Yau, S.S.: Testing context-sensitive middleware-based software applications. In: Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004. pp. 458–466. IEEE (2004)
14. Zhang, M., Zhang, Y., Zhang, L., Liu, C., Khurshid, S.: Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 132–142 (2018). https://doi.org/10.1145/3238147.3238187
15. Zhou, Z.Q., Sun, L.: Metamorphic testing of driverless cars. Communications of the ACM **62**, 61 – 67 (2019)