Rule-Based Out-of-Distribution Detection

Rule-based Out-of-Distribution Detection

Giacomo De Bernardi, Sara Narteni, Enrico Cambiaso and Maurizio Mongelli *Member, IEEE*

***Abstract*—Out-of-distribution detection is one of the most critical issue in the deployment of machine learning. The data analyst must assure that data in operation should be compliant with the training phase as well as understand if the environment has changed in a way that autonomous decisions would not be safe anymore. The method of the paper is based on eXplainable Artificial Intelligence (XAI); it takes into account different metrics to identify any resemblance between in-distribution and out of, as seen by the XAI model. The approach is non-parametric and distributional assumption free. The validation over complex scenarios (predictive maintenance, vehicle platooning, covert channels in cybersecurity) corroborates both precision in detection and evaluation of training-operation conditions proximity.**

***Impact Statement*—Many sectors these days address safe AI: automotive (SOTIF), avionics (SAE G-34/EUROCAE WG-114), ISO/IEC (JTC 1/SC 42) and healthcare. Safe AI means understanding under which conditions autonomous actuations may lead to hazards. The impact of research here is to make AI aware of this, thus understanding under which conditions it may operate without detrimental effect to the human or the environment. Examples may involve the prevention of dangerous manoeuvres by autonomous cars, inaccurate clinical diagnosis by artificial doctors, wrong decision making in cyberwarfare and in many other sectors (energy, finance). The theoretical analysis here is empowered by computational and incremental groupwise analysis in order to increase the readiness level of the proposed approach.**

***Index Terms*—Out-of-distribution detection, eXplainable AI, mutual information, open data.**

I. Notation and List of Acronyms

OoD	Out of distribution
ODD	OoD detection
ML	Machine learning
XAI	eXplainable Artificial Intelligence
TR	Training set
OP	Operational set
tr_i	i-th training subset
op_i	i-th operational subset
n_s	number of data samples in a split
N_r	Number of rules
N_{tr}	Number of training splits
N_{op}	Number of operational splits
$\mathcal{R}_{tr}$	Training reference ruleset
r_i	i-th rule
h_i^j	j-th hit for the i-th rule
l_p	l_p norm
μI	Mutual information
$W\mu I$	Weighted mutual information
RBI	Rule based information
H	Entropy

The authors are with the Cnr-Istituto di Elettronica, Ingegneria dell'Informazione e delle Telecomunicazioni (CNR-IEIIT), Corso F. M. Perrone 24, 16152, Genoa, Italy (e-mail: name.surname@ieiit.cnr.it)

G. De Bernardi is also with Università degli studi di Genova, The Electrical, Electronics and Telecommunication Engineering and Naval Architecture Department (DITEN), Genova, Italy

S. Narteni is also with Politecnico di Torino, Department of Control and Computer Engineering (DAUIN), 10129, Turin, Italy

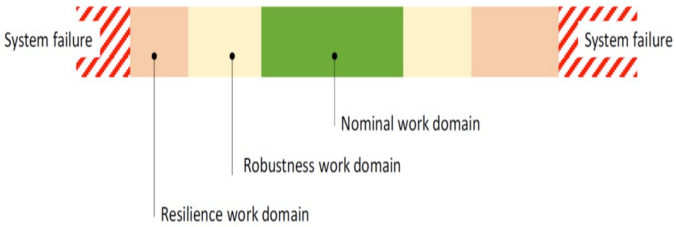


Fig. 1: Illustration of the work domains as reported in [4]. From central green bar to side yellow/orange/red bars, the nominal domain shifts and the severity of the OoD increases in parallel.

II. Introduction

THE problem of out-of-distribution (OoD) detection (ODD) deals with comparing the working conditions of a machine learning model with those considered during the training process. The comparison is performed at the operational level to understand if the new data belong to a probability distribution different from that driving the data collection of the training phase. In case of divergence between training and operation, the system must generate an alarm because the performance of the model may no longer conform to what was measured at the training stage (even in case of successfully passed generalization tests[1]). The problem represents a very important challenge for the secure application of machine learning, and is fundamental in the context of trustworthy AI [2], [3]. The recent standards in avionics [4], [5], automotive [6], [7] and ISO/IEC, as well as other regulatory initiatives in medical informatics [8], pose the problem of identifying all those operating conditions that can have an impact on safety.

The European Union Aviation Safety Agency (EASA) Fig. 1 shows the different levels of severity of the OoD on operational data. The green bar denotes compliance with training data (in-distribution), the yellow color reflects an OoD zone where the autonomous function still produces accurate indications; orange color indicates an OoD area where the autonomous function is fallacious, but the system does not degenerate into dangerous conditions (the surrounding conditions of the environment are still compatible with safe actuations), while red signals that the system may fall into dangerous conditions (if driven by the autonomous function). The tests of autonomous safety-critical actuation should include all the conditions in the mentioned color gradations, at least by simulation analysis.

[1]Generalization bounds, see, e.g., [1], concern the gap that exists between the empirical risk, calculated on the data actually available (on which the model is trained) and the theoretical risk, calculated on the distribution of probability that represents the data; this probability distribution is unknown in closed-form and, in the ODD context, represents the "in-distribution".

Although the literature in the field of OoD already poses solutions based on labelled data or through anomaly detection, as evidenced by [9], [10], the OoD according to distributional assumption-free and OoD-agnostic criteria is still an open problem[2].

A. Contribution

The proposed method is designed under these criteria with the added advantage of avoiding any parameter tuning. It is based on the evaluation of the histogram generated by the frequency of validation of a rule-based model by the data themselves. The histogram generated during the training phase represents a fingerprint to be verified at runtime. If the data at runtime generates a histogram "significantly different" from the training one, it means that the data are OoD. Unlike K-NN [10] and Neural Networks distance [11], where a single distance criterion is defined, the similarity measure can be derived through multiple metrics. This offers support to the tests mentioned by EASA, since the proposed method measures incremental cases of departure from in-distribution. In addition this is a groupwise method which makes the ODD more robust as outlined in Sec VI.

III. Related work

ODD has become an important theme in ML field, since the recognition of unseen data either "similar" or not (in- or out) to the ones the ML system has been trained on may lead to potentially fatal consequences.

Most of the solutions proposed to address the problem of the OoD make strong distributional assumptions of the feature space [12] or suppose they are given a training in and out probability density function (pdf) [13], but this not always holds in practice. What is more, lots of statistic tests fail to estimate the real distribution of training data (data are not enough and the pdf are too coarse) [10]. There are lots of widespread supervised methods used across OoD detection: model-based methods such as the ODIN [14], distance-based methods like OODL[15], density-based methods as the energy-based OoD Detection method [16] and some threshold-based methods, including Maximum Softmax Probability [17] or Autoencoder [18]. Other approaches use outlier detection methods as the Isolation Forest [19] and label shift in deep learning is also considered in [20]. Under distributional assumptions free hypothesis, unsupervised OoD detection methods need the right tuning of some parameters as [10] and [21]. Starting from eXplainability, our solution still maintains the former and does not rely on any critical parameter setting.

IV. Logic Learning Machine

The rule-based model adopted in our work is called Logic Learning Machine (LLM), an efficient implementation of Switching Neural Networks [22], developed and available in Rulex software platform [3]. However, we remark that the methods for OoD detection presented in the paper can be easily extended to any other kind of rule-based model, such as decision tree or tree ensembles like random forests or Skope-Rules [4]. Given some input data, the LLM provides a classification model represented by ruleset $\mathcal{R} = \{r_k\}_{k=1,\dots,N_r}$, with each rule r_k expressed through the following structure: **if** <*premise*> **then** <*consequence*>. The <*premise*> is made up of the logical conjunction (AND) of conditions on the input features and the <*consequence*> constitutes the output of the classification rule. Rule generation process occurs in three steps. First, a discretization and binarization of the feature space is performed by using the inverse-only-one coding. The resulting binary strings are then concatenated into a single large string representing the considered samples. Shadow clustering is then used to build logical structures, called implicants, which are finally transformed into simple conditions and combined into a collection of intelligible rules [23], [24].

V. Rule-based ODD

A. Rule hits histograms

Let us denote with $\mathcal{R}_{tr}$ a set of rules generated from a training set and let N_r be the number of rules composing it. Let N_{tr} and N_{op} be the numbers of splits of the training domain and the operational one, respectively, and let $N_h = N_{tr} + N_{op}$ be the total number of splits. Let n_s be the number of data samples present in a split. For each split, samples [5] may (or not) satisfy each rule a certain number of times. We refer to this number as the *number of hits* for that rule. Therefore we define N_h vectors, considering this number scaled by the split size n_s:

$$\mathbf{h}^j = \{h_i^j\},\ \ h_i^j \in [0,1],\ i = 1,\dots,N_r,\ j = 1,\dots,N_h \qquad (1)$$

Each vector $\mathbf{h}^j$ can be thought as a histogram. Rule hits are the key starting point of the proposed methods. Therefore, particular care should be posed on the quality of the training reference ruleset generating them. For example, feature reduction methods on the input data may provide a simplified ruleset, thus more interpretable. Nevertheless, it would result in less informative rule hits. That is, the less variables we include in the model, the more general would be the support of the resulting rules, which would be more likely to be frequently verified by the data samples. As a result, histograms shape would be flattened and the methods for OoD detection would be less performing.

B. Data splits

At training stage, we exploit N_{tr} splits of the dataset $TR = \{tr_1, \dots, tr_{N_{tr}}\}$. These splits become the baseline for building the in-distribution histograms, as per Eq. 1, representing the numbers of hits obtained by testing the rules in $\mathcal{R}_{tr}$ on each

[2]Distributional assumption-free means no closed-form expressions of in- and out- probability distributions are considered. OoD-agnostic means no information about ODD conditions is considered. Another important issue, which is related with the assumption of probabilistic Gaussian or mixed-Gaussian functions, is to avoid calculating the covariance matrix from data, which can be numerically unstable.

[3]https://www.rulex.ai

[4]https://github.com/scikit-learn-contrib/skope-rules

[5]Samples can satisfy multiple rules and there may be operational samples satisfying none of the rules.

considered split. Two different algorithms are then studied, based on the data organization in operation: $N_{op} = 1$ when only one split is available and $N_{op} > 1$ with more than one split. The first case is suitable for data scarcity in operation and simplifies the calculations.

C. Adopted Metrics

The metrics driving ODD are as follows:

- Weighted mutual information $W\mu I$, used when only one operational split is available ($N_{op} = 1$), as described in Sec. V-D
- Rule-based information RBI, used when the operational data are sufficient to perform multiple splits ($N_{op} > 1$), as per Sec. V-E
- l_p norm, with $p = 1, 2$; their computation is performed in the same way for both scenarios ($N_{op} \geq 1$)

The order of the hits with respect to the rules drives modification to canonical mutual information, $W\mu I$ and RBI, as explained in Appendix IX. For all metrics, the idea is to compare values computed in operation with the ranges achieved in training (*baseline*). Specifically, we expect that histograms generated at training or operational stages have different shapes, providing an indication of OoD. Thus, through the Algorithms presented in the following Sections, we try to quantify such a behavior. Algorithms 1-2 (for $N_{op} = 1$) and 3-4 (for $N_{op} > 1$) both share the same methodological approach. As per the training part, the reference ruleset $\mathcal{R}_{tr}$ is applied to the training data splits to retrieve the rule hits histograms; then the metrics of interest are computed over couples of training histograms and a *baseline* is defined. At operational, rule hits histograms are derived for the couples formed by training and operational data split(s) and the values of the metrics are computed as well. Eventually, an ODD is acknowledged whenever the largest portion of operational values falls outside the training baseline for at least one of the metrics (i.e., through minority voting of the metrics). We finally remark that such methodologies are designed for working at runtime; therefore, adding any pre-processing module, e.g. any instance selection, would make the whole process slower, which would not be always acceptable when dealing with safety-critical scenarios.

D. First scenario: $N_{op} = 1$

1) Training setting: The first scenario deals with a single operational split op_1. We first present the procedure for the training domain and then for the operational one. As to Eq. 1, the training matrix-like structure shown in Table I consequently arises. Based on that table, weighted mutual information and norms are computed as described in Algorithm 1.

2) Operational setting: We now present the procedure when an operational set is considered. As to Eq. 1, we can build the training-operational matrix as in Table II. Weighted mutual information and norms are then computed as described in Algorithm 2.

	tr_1	....	$tr_{N_{tr}}$
r_1	$h_1^{tr_1}$	...	$h_1^{tr_{N_{tr}}}$
r_2	$h_2^{tr_1}$	...	$h_2^{tr_{N_{tr}}}$
.	.	.	.
.	.	.	.
r_{N_r}	$h_{N_r}^{tr_1}$	...	$h_{N_r}^{tr_{N_{tr}}}$

TABLE I: Training numbers of hits table. Each column refers to a training split $tr_i \in TR$ and each row to a rule $r_i \in \mathcal{R}_{tr}$.

Algorithm 1 Weighted Mutual Information and Norms at Training Stage

i, j=1,. . .,N_{tr}, $i \neq j$
Input: Table I
Output: Training baselines $W\mu I_{base}$ and ${l_p}^{base}$

1a. Define the weight associated with tr_i and tr_j, $\alpha_{i,j} \in (0, 1)$, $\forall i$, $\forall j$:

$$\alpha_{i,j} = \frac{1}{N_r} \sum_{r=1}^{N_r} (|h_r^{tr_i} - h_r^{tr_j}|)$$

1b. Compute the weighted entropies $H(tr_i)$, $H(tr_j)$, $H(tr_i, tr_j)$, $\forall i$, $\forall j$

$$H(tr_i) = -\sum_{r=1}^{N_r} [\alpha_{i,j} P(h_r^{tr_i}) \cdot log(\alpha_{i,j} P(h_r^{tr_i}))]$$

$$H(tr_j) = -\sum_{r=1}^{N_r} [\alpha_{i,j} P(h_r^{tr_j}) \cdot log(\alpha_{i,j} P(h_r^{tr_j}))]$$

$$H(tr_i, tr_j) = -\sum_{r=1}^{N_r} [\alpha_{i,j} P(h_r^{tr_i}, h_r^{tr_j}) \cdot log(\alpha_{i,j} P(h_r^{tr_i}, h_r^{tr_j}))]$$

2. Compute the weighted mutual information ($W\mu I$):

$$W\mu I(tr_i, tr_j) = [H(tr_i) + H(tr_j) - H(tr_i, tr_j)], \forall i, \forall j$$

3. Compute the baseline $W\mu I_{base}$:

$$W\mu I_{base} \doteq [\min_{i,j}(W\mu I(tr_i, tr_j)), \max_{i,j}(W\mu I(tr_i, tr_j))]$$

4. Compute l_p (p=1,2) norms:

$$l_p(tr_i, tr_j) = \left[\sum_{r=1}^{N_r} (|h_r^{tr_i} - h_r^{tr_j}|)^p\right]^{\frac{1}{p}}, \forall i, \forall j$$

5. Compute the baseline ${l_p}^{base}$ (p=1,2):

$${l_p}^{base} \doteq [\min_{i,j}(l_p(tr_i, tr_j)), \max_{i,j}(l_p(tr_i, tr_j))]$$

E. Second scenario: $N_{op} > 1$

Again by following the notation of section V-A, several splits of the training domain TR are defined $TR = \{tr_1, \ldots, tr_{N_{tr}}\}$, together with an analogous set for the operational domain, $OP = \{op_1, \ldots, op_{N_{op}}\}$. The training splits are organized in two subsets: $TR1 = \{tr_1, \ldots, tr_k\}$, $TR2 = \{tr_{k+1}, \ldots, tr_{N_{tr}}\}$ with $k = N_{tr} - N_{op} - 1$ and, based on a leave-one-out cross-validation [25], we consider $TR2_m = TR2 \setminus \{tr_m\}$, $m = k + 1, \ldots, N_{tr}$. These sets drive the computation of the baseline according to the rule-based information (RBI) (algorithm 3, table I as a reference). Algorithm 4 defines the inherent ODD by taking table III as a reference.
The estimation of the Gaussian distributions follows the maximum likelihood principle (see 2.5.1 of [26]). Building N_r

	tr_1	....	$tr_{N_{tr}}$	op_1
r_1	$h_1^{tr_1}$	...	$h_1^{tr_{N_{tr}}}$	$h_1^{op_1}$
r_2	$h_2^{tr_1}$	...	$h_2^{tr_{N_{tr}}}$	$h_2^{op_1}$
.	.	.	.	
.	.	.	.	
r_{N_r}	$h_{N_r}^{tr_1}$	...	$h_{N_r}^{tr_{N_{tr}}}$	$h_{N_r}^{op_1}$

TABLE II: Training-operational number of hits table with $N_{op} = 1$. Columns refers to the training splits $tr_i \in TR$ and the operational split op_1, each row to a rule $r_i \in \mathcal{R}_{tr}$.

Algorithm 2 Weighted Mutual Information and Norms at Operational Stage

$i = 1, \ldots, N_{tr}$, $p = 1, 2$
Input: Table II; baseline ranges $W\mu I_{base}$ and ${l_p}^{base}$
Output: ODD through $W\mu I$ and l_p

1. Compute the weighted entropies $H(tr_i)$, $H(op_1)$ and $H(tr_i, op_1)$ as done in the Algorithm 1 (steps 1a-1b) $\forall i$
2. Compute the weighted mutual information ($W\mu I$):

$$W\mu I(tr_i, op_1) = [H(tr_i) + H(op_1) - H(tr_i, op_1)], \forall i$$

3.Compute l_p norms:

$$l_p(tr_i, op_1) = \left[\sum_{r=1}^{N_r}(|h_r^{tr_i} - h_r^{op_1}|)^p\right]^{\frac{1}{p}}, \forall i$$

4.OoD detection:

IF $W\mu I(tr_i, op_1) \notin W\mu I_{base}$ *for the majority of* i THEN flag is *on*
IF $l_p(tr_i, op_1) \notin {l_p}^{base}$ *for the majority of* i THEN flag is *on*,
IF {*at least one flag is on*} THEN op_1 is OoD

separate Gaussian distributions (one for each row of the table) allows to tackle with the curse of dimensionality problem in parameters estimation, in place of building a single, multi-dimensional Gaussian distribution for the entire table (see, e.g., 2.5.7 of [26]). We remark that this methodology still complies with the distributional assumption-free (as stated in Sec. II) because the Gaussian distribution estimation concerns the rule hits and not the data and furthermore thanks to the metodology followed to construct each histogram we can take advantage of the central limit theorem and the law of large numbers.

	tr_1	....	$tr_{N_{tr}}$	op_1	...	$op_{N_{op}}$
r_1	$h_1^{tr_1}$	...	$h_1^{tr_{N_{tr}}}$	$h_1^{op_1}$	...	$h_1^{op_{N_{op}}}$
r_2	$h_2^{tr_1}$	...	$h_2^{tr_{N_{tr}}}$	$h_2^{op_1}$	...	$h_2^{op_{N_{op}}}$
.	.	.	.	.	.	.
.	.	.	.	.	.	.
r_{N_r}	$h_{N_r}^{tr_1}$	...	$h_{N_r}^{tr_{N_{tr}}}$	$h_{N_r}^{op_1}$	...	$h_{N_r}^{op_{N_{op}}}$

TABLE III: Training-operational number of hits table with $N_{op} > 1$. Columns $tr_1, \ldots, tr_{N_{tr}}$ refer to training splits (further organized in $TR1$ and $TR2_m$ sets, see Sec. V-E), columns $op_1, \ldots, op_{N_{op}}$ are the splits in operation. Each row refers to a rule $r_i \in \mathcal{R}_{tr}$.

VI. Groupwise in operation

A. Incremental technique

The method collects a bunch of operational data before processing and classifying them as in or out of distribution.

Algorithm 3 Rule-based Information at Training Stage

$i = 1, \ldots, N_{tr}, p = 1, 2, j = 1, \ldots, N_r, tr_i \in TR2_m$;
Inputs: $TR1$ and $TR2_m$, $m = k+1, \ldots, N_{tr}$, $k = N_{tr} - N_{op} - 1$
Output: Training baselines RBI_{base} and ${l_p}^{base}$

1. Compute $\mu_j^{TR1} = \frac{1}{k}\sum_{i=1}^{k} h_j^{tr_i}$ and $\sigma_j^{TR1} = \sqrt{\frac{\sum_{i=1}^{k}\left(h^{tr_i} - \mu_j^{TR1}\right)^2}{k}}, \forall j$
2. Estimate Gaussian distributions $\{\mathcal{N}(\mu_j^{TR1}, \sigma_j^{TR1})\}$, $\forall j$
3.Compute $\mu_j^{TR2_m} = \frac{1}{N_{tr}-k-1}\sum_{\substack{i=k+1 \\ i\neq m}}^{N_{tr}} h_j^{tr_i}$ and

$$\sigma_j^{TR2_m} = \sqrt{\frac{\sum_{\substack{i=k+1 \\ i\neq m}}^{N_{tr}}\left(h^{tr_i} - \mu_j^{TR2_m}\right)^2}{N_{tr}-k-1}}, \forall j$$

4. Estimate Gaussian distributions $\{\mathcal{N}(\mu_j^{TR2_m}, \sigma_j^{TR2_m})\}$, $\forall j$
5a. Considering $\mathcal{N}(\mu_j^{TR2_m}, \sigma_j^{TR2_m}), \forall j$ and $\forall tr_i$, compute:

$$P_{ij}^{TR2_m} \doteq \mathbb{P}(x \in [h_j^{tr_i} - \sigma_j^{TR2_m}, h_j^{tr_i} + \sigma_j^{TR2_m}]|r_j),$$

$$CP_{ij}^{TR2_m} \doteq 1 - P_{ij}^{TR2_m}$$

5b. Compute the entropy:

$$H(tr_i) = -\sum_{j=1}^{N_r}[P_{ij}^{TR2_m} \cdot log(P_{ij}^{TR2_m}) + CP_{ij}^{TR2_m} \cdot log(CP_{ij}^{TR2_m})], \ \forall tr_i$$

6a. Considering $\mathcal{N}(\mu_j^{TR1}, \sigma_j^{TR1}), \forall j$ and $\forall tr_i$ compute:

$$P_{ij}^{TR1} \doteq \mathbb{P}(x \in [h_j^{tr_i} - \sigma_j^{TR1}, h_j^{tr_i} + \sigma_j^{TR1}]|r_j), \ CP_{ij}^{TR1} \doteq 1 - P_{ij}^{TR1}$$

6b. Compute the weighted conditional entropy, $\forall tr_i$:

$$H(tr_i|TR1) = -\sum_{j=1}^{N_r}\gamma_j^{tr_i} \cdot [P_{ij}^{TR1} \cdot log(P_{ij}^{TR1}) + CP_{ij}^{TR1} \cdot log(CP_{ij}^{TR1})],$$

where $\gamma_j^{tr_i} \doteq \frac{P_{ij}^{TR2_m}}{P_{ij}^{TR1}}$
7. Compute the average entropies:

$$H(TR2_m) = \frac{1}{N_{tr}-k-1}\sum_{\substack{i=k+1 \\ i\neq m}}^{N_{tr}} H(tr_i)$$

$$H(TR2_m|TR1) = \frac{1}{N_{tr}-k-1}\sum_{\substack{i=k+1 \\ i\neq m}}^{N_{tr}} H(tr_i|TR1)$$

8. Measure the similarity between $TR2_m$ and $TR1$ considering $RBI_{TR1-TR2_m}$:

$$RBI_{TR1-TR2_m} \doteq \frac{H(TR2_m)}{H(TR2_m|TR1)}$$

10. Construct the baseline range RBI_{base}:

$$RBI_{base} \doteq [\min_m(RBI_{TR1-TR2_m}), \max_m(RBI_{TR1-TR2_m})]$$

11. Compute the norms baselines ${l_p}^{base}$ as done in Algorithm 1.

For this reason, it falls in the category of groupwise methods [27], [28]. Differently from pointwise, groupwise confirms a type of situation (in or out), without relying on a single point that could be a spike in a steady trend. The collection phase in operation does not imply that one would wait for new n_s samples to register a new split and to provide the

Algorithm 4 Rule-based Information at Operational Stage

$i = 1, \dots, N_{tr}, p = 1, 2, j = 1, \dots, N_r, op_i \in OP$;
Inputs: $TR1$ and OP (Table III); baseline ranges RBI_{base} and l_p^{base}
Output: ODD through RBI and l_p

1. Compute $\mu_j^{TR1} = \frac{1}{k}\sum_{i=1}^{k} h_j^{tr_i}$ and $\sigma_j^{TR1} = \sqrt{\frac{\sum_{i=1}^{k}\left(h^{tr_i} - \mu_j^{TR1}\right)^2}{k}}$, $\forall j$

2. Estimate Gaussian distributions $\{\mathcal{N}(\mu_j^{TR1}, \sigma_j^{TR1})\}$, $\forall j$

3. Compute $\mu_j^{OP} = \frac{1}{N_{op}}\sum_{i=1}^{N_{op}} h_j^{op_i}$ and
$\sigma_j^{OP} = \sqrt{\frac{\sum_{i=1}^{N_{op}}\left(h^{op_i} - \mu_j^{OP}\right)^2}{N_{op}}}$, $\forall j$

4. Estimate Gaussian distributions $\{\mathcal{N}(\mu_j^{OP}, \sigma_j^{OP})\}$, $\forall j$

5a. Considering $\mathcal{N}(\mu_j^{OP}, \sigma_j^{OP})$, $\forall j$ and $\forall op_i$, compute:

$$P_{ij}^{OP} \doteq \mathbb{P}(x \in [h_j^{op_i} - \sigma_j^{OP}, h_j^{op_i} + \sigma_j^{OP}] | r_j),\ CP_{ij}^{OP} \doteq 1 - P_{ij}^{OP}$$

5b. Compute the entropy:

$$H(op_i) = -\sum_{j=1}^{N_r}[P_{ij}^{OP} \cdot log(P_{ij}^{OP}) + CP_{ij}^{OP} \cdot log(CP_{ij}^{OP})],\ \forall op_i$$

6a. Considering $\mathcal{N}(\mu_j^{TR1}, \sigma_j^{TR1}), \forall j$ and $\forall op_i$, compute:

$$P_{ij}^{TR1} \doteq \mathbb{P}(x \in [h_j^{op_i} - \sigma_j^{TR1}, h_j^{op_i} + \sigma_j^{TR1}] | r_j),\ CP_{ij}^{TR1} \doteq 1 - P_{ij}^{TR1}$$

6b. Compute the weighted conditional entropy, $\forall op_i$:

$$H(op_i|TR1) = -\sum_{j=1}^{N_r} \gamma_j^{op_i} \cdot [P_{ij}^{TR1} \cdot log(P_{ij}^{TR1}) + CP_{ij}^{TR1} \cdot log(CP_{ij}^{TR1})],$$

where $\gamma_j^{op_i} \doteq \frac{P_{ij}^{OP}}{P_{ij}^{TR1}}$

7. Compute the average entropies:

$$H(OP) = \frac{1}{N_{op}}\sum_{i=1}^{N_{op}} H(op_i)$$

$$H(OP|TR1) = \frac{1}{N_{op}}\sum_{i=1}^{N_{op}} H(op_i|TR1)$$

8. Measure the similarity between $TR1$ and OP by considering RBI_{TR1-OP} :

$$RBI_{TR1-OP} \doteq \frac{H(OP)}{H(OP|TR1)}$$

9. OoD detection:

IF $RBI_{TR1-OP} \notin RBI_{base}$ THEN flag is *on*
IF $l_p(tr_i, op_j) \notin l_p{}^{base}$ *for the majority of i and j* THEN flag is *on*
IF {*at least one flag is on*} THEN OP is OoD

ODD. Splits are generated continuously, as soon as new samples are collected. Incremental techniques may be also used to accelerate the computation of statistically-based features (mean, variance, skewness and kurtosis), as in the RUL and DNS problems detailed later on [29]. Like in incremental techniques, once a new sample is available, a new (operational) bunch of n_s samples is built, by adding the new sample and by disregarding the most far away point in the past (of n_s positions). In turn, the bunch leads to the split collection, by computing the inherent hits on the ruleset. The process assumes a sample-by-sample incremental time window, over which the following operations are performed. A new data bunch is firstly registered, a new split is calculated and a new ODD is then derived.

B. Computational issues

The computational speed of the bunch building process depends on how fast the data samples are collected by the system (the quantity is denoted by δt_0). The speed of the split building process depends on the time required to compute the hits of the ruleset on the bunch, namely, on the latest n_s data samples (δt_1). The speed of the ODD depends on the computational time of algorithms 2 and 4 above (δt_{a2} and δt_{a4}, respectively).

The computational times of the baselines in algorithms 1 and 3 are less of interest as the algorithms work at design time, in which enough computational resources are assumed to be available; they however follow similar $\mathcal{O}(\cdot)$ as their respective operational versions. On the other hand, δt_{a2} and δt_{a4} are of interest, as algorithms 2 and 4 work over the deployed ML infrastructure, for which limited computational resources may be assumed. The following considerations hold for the δt quantities. δt_0 is outside of the scope of the paper as it depends on the environmental conditions and on the sensing architecture of the system. δt_1 is $\mathcal{O}(n_s)$ (by assuming the time to verify a rule on a data sample a constant, independently to the complexity of the rule). By referring to the computations inherent to the metrics involved in algorithm 2, δt_{a2} is $\mathcal{O}(N_r \cdot N_{tr})$. Analogously, δt_{a4} is $\mathcal{O}(N_r \cdot N_{OP})$.

VII. Case Studies

A. Datasets

Three application scenarios are considered with the inherent datasets and relevance of the ODD problem.

1) RUL: The first dataset concerns damage propagation modeling for aircraft engines and is taken from the NASA repository [30]. It is an important benchmark in predictive maintenance and includes four different subsets of data (tr_1, op_a, op_b, op_c), corresponding to different machines of the same factory family. The problem is interesting in the ODD perspective because one may expect a model trained on a machine (e.g., tr_1) to be applicable (with limited error) to another machine (e.g., op_a). The features are: mean (m), variance (v), kurtosis (k) and skewness (s) of the original 23 physical quantities over time. A preliminary analysis with LLM feature ranking [23] individuated the following set of 7 most important features: s_{os2}, m_{Nc}, v_{Nc}, v_{phi}, $m_{htBleed}$, $s_{htBleed}$, m_{W31}, whose corresponding physical quantities are outlined in table IV.

Symbol	Description
$os2$	Operational setting 2
Nc	Physical core speed
phi	Ratio of fuel flow to $Ps30$
$htBleed$	Bleed enthalpy
$W31$	HTP coolant bleed

TABLE IV: RUL physical quantities.

The target variable is the Remaining Useful Life (RUL), which represents the time before the occurrence of a fault and is binarized to assume either value '0 healthy'(RUL>150) or '1 fault' (RUL≤150). A ML classifier through LLM predicts if the engine would come into a fault state or not; tr_1 constitutes the in-distribution and $\mathcal{R}_{tr_1}$ the reference ruleset.

2) Platooning: The second dataset (platooning [31]) addresses collision avoidance in vehicle platooning, which is one of the most celebrated application in autonomous driving. A group of vehicles is interconnected via wireless, based on the Cooperative Adaptive Cruise Control [32]. The behavior of the platooning system is synthesised by the physical quantities pointed out in table V. The physical quantities correspond to the features of the problem, which identifies potential collision in advance after a sudden brake.

Symbol	Description
N	Number of vehicles
$F0$	Breaking force applied by the leader
PER	Probability of packet loss
$d0$	Initial mutual distance between vehicles
$v0$	Initial speed
d	Communication delay in the inter-connection of vehicles

TABLE V: Platooning features.

We consider two datasets: in the first one (LOW) the communication delay d parameter is bounded by 0.4 s; in the second one (HIGH), d is larger than that threshold. As in the RUL case, we set a training domain: tr_{LOW} (in-distribution) as well as the reference ruleset $\mathcal{R}_{tr_{LOW}}$. A typical ODD problem is thus posed (between LOW and HIGH) as d has a significant impact on performance. The ODD has here a safety preserving role as it recognizes if the delay in operation is larger than the one in training. The algorithms are however not aware that delay is the key for the datasets differentiation and understand the ODD through the operational hits on $\mathcal{R}_{tr_{LOW}}$.

3) DNS: The third dataset (DNS) deals with a DNS tunneling detection problem [33]. The aim is detecting the presence of Domain Name Server intruders by an aggregation-based traffic monitoring. Silent intruders and quick statistical fingerprints generation make the tunneling detection a hard task. Table VI shows the physical quantities of the problem.

Symbol	Description
q	Size of a query packet
a	Size of an answer packet
Dt	Time interval intercurring between query and answer

TABLE VI: DNS tunneling physical quantities.

Again as in the RUL case, mean (m), variance (v), kurtosis (k) and skewness (s) are extracted over the time series of the system, thus leading to 12 features. The target variable is a binary label denoting the 'presence' or 'absence' of a tunneling attack. Two reference datasets are as follows: the first one considers a tunneled peer-to-peer (p2p) application, that is the training (in-distribution) domain tr_{p2p} (with $\mathcal{R}_{tr_{p2p}}$ as the reference ruleset), and the second refers to the tunneled secure shell (ssh) application, which is the operational setting (op_{ssh}). The ODD here is of interest once ssh is used in operation under the trained p2p model. It is a quite realistic situation in cybersecurity as not all attack configurations may be anticipated at design time.

VIII. Results

The first two subsections deal with understanding the ranges of the metrics in OoD conditions. The baseline ranges are reported in the first row of all the tables and represent the reference to infer possible OoD in operation. An even partial overlap between ranges in training and operation leads to a missed detection, i.e., a false negative (FN). A false positive (FP) consists of a wrong ODD for a training (in-distribution) bunch of samples. Secondly, False Negative Rate (FNR) and False Positive Rate (FPR) are reported in subsection VIII-C. An operational sample of table II or table III constitutes a FN in case no OoD is declared; a sample (column) of table I constitutes a FP in case OoD is declared. Tables are built as follows. Each column refers to a bunch of n_s=5000 samples, with values reflecting the number of hits for the reference ruleset. We consider N_{tr}=50 and N_{op}=1 in table II while N_{tr}=50 with $TR1 = \{tr_1, \ldots, tr_{39}\}$, $TR2 = \{tr_{40}, \ldots, tr_{50}\}$ and N_{op}=10 in table III. The total repetitions of the experiments for computing FPR and FNR is 2500. The section ends with subsection VIII-D and outlines incremental groupwise detection in operation.

Example code and data for the experiments are available at the following link: https://github.com/giacomo97cnr/Rule-based-ODD.

A. $N_{op} = 1$

Tables VII and VIII show norms, μI and $W\mu I$ ranges over the RUL datasets. The robustness of algorithm 2 is validated by the fact that all OoD ranges are significantly far away from the training baselines. The values with the norms are closer to the respective training baselines with op_b than with op_a and op_c. This is an important indication about the similarity of in (tr_1) and out (op_b) distributions. Coming back to the EASA introductive figure, it happens that op_b lies in the yellow zone. Namely, the model trained on tr_1 is good on op_b data, with FPR=18% and FNR = 27%, which is quite close to the in-distribution performance (training and test on separate tr_1 samples): FPR = 18% and FNR = 22%[6]. On the other hand, op_a and op_c lie in the orange zone (i.e., the tr_1 model is not good anymore on op_a and op_c, being FPR = 0.03%, FNR = 99.86% and FPR = 0.04%, FNR = 99.88%, respectively)[7]. When the tr_1 model is tested on op_b, a good balance between FPR and FNR is still achieved; the same does not hold for op_a and op_c, which are far away from the tr_1 baseline.

The rationale behind the tr_1 and op_b proximity is beyond the knowledge of the authors (one may argue about some mechanical similarity of the respective engines), but inferring such proximity through ODD is quite an important achievement. In

[6]The mentioned FNR and FPR refer to the original RUL problem, namely, they represent the errors in fault prediction.

[7]Large FNRs here may even lead to the red zone of EASA picture (catastrophic event), depending on system resilience to wrong autonomous decisions.

this perspective, the method should use all the metrics jointly to provide both ODD and a measure of the distance of the in and out distributions.

Couples	l_1	FNR (l_1)	l_2	FNR (l_2)
$tr_1 - tr_1$	[0.09, 0.20]		[0.02, 0.05]	
$tr_1 - op_a$	[3.16, 3.26]	0%	[1.05, 1.06]	0%
$tr_1 - op_b$	[1.16, 1.40]	0%	[0.30, 0.34]	0%
$tr_1 - op_c$	[3.16, 3.26]	0%	[1.05, 1.06]	0%

TABLE VII: Algorithms 1 and 2: RUL. Norms.

Couples	μI	FNR (μI)	$W\mu I$	FNR ($W\mu I$)
$tr_1 - tr_1$	[0.707,1.442]		[0.023, 0.045]	
$tr_1 - op_a$	[0.019,0.027]	0%	[0.291, 0.297]	0%
$tr_1 - op_b$	[0.182,0.278]	0%	[0.159, 0.179]	0%
$tr_1 - op_c$	[0.019,0.027]	0%	[0.290, 0.297]	0%

TABLE VIII: Algorithms 1 and 2: RUL. μI and $W\mu I$.

Couples	l_1	FNR (l_1)	l_2	FNR (l_2)
$tr_{LOW} - tr_{LOW}$	[0.02, 0.12]		[0.01, 0.04]	
$tr_{LOW} - op_{HIGH}$	[3.80, 3.90]	0%	[1.37, 1.39]	0%

TABLE IX: Algorithms 1 and 2: platooning. Norms.

Couples	μI	FNR (μI)	$W\mu I$	FNR ($W\mu I$)
$tr_{LOW} - tr_{LOW}$	[0.87,2.73]		[0.02,0.06]	
$tr_{LOW} - op_{HIGH}$	[0.04,0.97]	8%	[0.51,0.73]	0%

TABLE X: Algorithms 1 and 2: platooning. μI and $W\mu I$.

Couples	l_1	FNR (l_1)	l_2	FNR(l_2)
$tr_{p2p} - tr_{p2p}$	[0.002, 0.090]		[0.002, 0.047]	
$tr_{p2p} - op_{ssh}$	[1.630, 1.770]	0%	[0.820, 0.890]	0%

TABLE XI: Algorithms 1 and 2: DNS. Norms.

Couples	μI	FNR(μI)	$W\mu I$	FNR($W\mu I$)
$tr_{p2p} - tr_{p2p}$	[1.5, 2.2]		[0.01, 0.15]	
$tr_{p2p} - op_{ssh}$	[0, 0.7]	0%	[0.73, 0.96]	0%

TABLE XII: Algorithms 1 and 2: DNS. μI and $W\mu I$.

Couples	l_1	l_2	RBI	FNR
$tr_1 - tr_1$	[0.12, 0.19]	[0.02, 0.03]	[0.927, 0.944]	
$tr_1 - op_a$	[3.22, 3.24]	[1.05, 1.06]	0	0%
$tr_1 - op_b$	[1.24, 1.33]	[0.30, 0.33]	[0.040,0.041]	0%
$tr_1 - op_c$	[3.22, 3.24]	[1.05, 1.06]	0	0%

TABLE XIII: Algorithms 3 and 4: RUL.

Couples	l_1	l_2	RBI	FNR
$tr_{LOW} - tr_{LOW}$	[0.03, 0.09]	[0.01, 0.03]	[0.886, 0.926]	
$tr_{LOW} - op_{HIGH}$	[3.83, 3.90]	[1.37, 1.39]	[0.024,0.025]	0%

TABLE XIV: Algorithms 3 and 4: platooning.

Couples	l_1	l_2	RBI	FNR
$tr_{p2p} - tr_{p2p}$	[0.008, 0.050]	[0.005, 0.020]	[0.821,0.972]	
$tr_{p2p} - op_{ssh}$	[1.670, 1.730]	[0.830, 0.870]	0	0%

TABLE XV: Algorithms 3 and 4: DNS.

As far as platooning and DNS are concerned, good performance are registered, except with μI in platooning (the topic is discussed later through groupwise analysis and in the Appendix).

B. $N_{op} > 1$

This section outlines the performance of algorithms 3 and 4, whose results are shown in tables XIII, XIV and XV. $N_{op} > 1$ allows to exploit more information at the operational level and thus finer separation of the OoD from the baseline.

C. Comparison with canonical methods

This section outlines a comparison with canonical supervised algorithms, such as K-Nearest Neighbours (KNN), Support Vector Machine with a RBF kernel (SVM) and Random Forest as well as unsupervised ones like the unsupervised KNN (u-KNN) and the Autoencoder. In particular, we first present the results considering the pointwise structure and then the groupwise counterpart. Supervised algorithms exploit information about OoD data. A mix of the in and out data are considered for training supervised approaches and then a testing phase got the FPR and FNR values presented in table XVI. In u-KNN we followed [10] yet revisiting it according to the fact that we were not using images; hence, we have first split up the training domain into a training set and a test one and then we have tuned two parameters: the number of neighbours (K) and a distance threshold (λ) used to determine if test data are in-distribution or not; λ was set in order to have a true negative rate of 95% in the training domain. Despite including information about OoD data in their training, supervised algorithms fail the ODD and unsupervised methods work even worse. This may denote that training and operational data are confused in the original feature space. The proposed algorithms, along with the above mentioned supervised and unsupervised techniques, achieve better performance in virtue of looking at in and out separation in a different space, namely, through the ruleset hits in training. Thus, we repeated the same experiments considering the groupwise structure (Table XVII) inducted by the usage of the rule hits; specifically, we used the rule hits as the input features (in place of the original features) and verified a perfect separation between in and out distributions with all the considered methods. Algorithms 2 and 4 are however still preferable for different reasons: first they are not black box methods, secondly they do not need significant tuning of critical parameters and finally for its robustness due to the usage of multiple metrics [34]. In table XVII, Algorithm 2 still experiences some FPR as the weighted version of the mutual information is applied to a smaller portion of operational data than with Algorithm 4. Another rationale behind the sensible level of FPR comes from the declaration of OoD if at least one of the metrics registers an OoD. This minimizes FNR, but may increase FPR. Additional results (not reported here for the sake of synthesis) confirm that the algorithm is even more sensitive to FPR with values of $N_{tr} < 50$. On the other hand, algorithm 4 decreases also FPR (with respect to algorithm 2), in virtue of the (Gaussian) statistical filter applied to several splits of operational data.

	Platooning ($tr_{LOW} - op_{HIGH}$)		DNS ($tr_{p2p} - op_{ssh}$)		RUL ($tr_1 - op_a$)		RUL ($tr_1 - op_b$)		RUL($tr_1 - op_c$)	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
KNN†	0.6%	0.7%	31%	30%	0%	0%	6.5%	7.5%	0%	0%
u-KNN [10]	$\leq 5\%$	0.16%	$\leq 5\%$	94%	≤ 5 %	0%	$\leq 5\%$	96.3%	$\leq 5\%$	0%
SVM†	0.3 %	0.9 %	49%	1.2%	0%	0%	26%	31.5%	0%	0%
Random Forest†	0%	0%	32%	37%	0%	0%	0.8%	5%	0%	0%
Autoencoder	3.6%	19.9%	14.7%	61.7%	4.1%	0%	12.5%	46.8%	4.1%	0%

TABLE XVI: ODD performance comparison considering the point-wise structure, in terms of FNRs and FPRs. Supervised methods are marked with †, the other ones are unsupervised.

	Platooning ($tr_{LOW} - op_{HIGH}$)		DNS ($tr_{p2p} - op_{ssh}$)		RUL ($tr_1 - op_a$)		RUL ($tr_1 - op_b$)		RUL($tr_1 - op_c$)	
	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
KNN †	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
u-KNN [10]	$\leq 5\%$	0%	$\leq 5\%$	0%	$\leq 5\%$	0%	$\leq 5\%$	0%	$\leq 5\%$	0%
SVM†	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Random Forest†	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Autoencoder	3.7%	0%	4%	0%	3.3%	0%	3.3%	0%	3.3%	0%
Algorithm 2	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Algorithm 4	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

TABLE XVII: ODD performance comparison considering the group-wise structure, in terms of FNRs and FPRs. Supervised methods are marked with †, the other ones are unsupervised.

D. Incremental groupwise in operation

By referring to section VI, the following experiments highlight the ODD when replacing in-distribution data with out-of, in a sample-by-sample, incremental, way. The analysis is relevant to the tracking of the OoD drift with both precision and measurement of distributions proximity. Every figure in the following contains the baseline derived at design time; the curves represent the behaviours of the metrics in operation. Increasing time windows with $n_s = 5 \cdot 10^3$ and 10^4 samples are used to emphasize the speed of the drift inference over time. The time size of the windows depends on the time granularity of the arrival of the points in operation; for this reason, the x-axis is not time, but it refers to the progressive identifier of the operational samples. The drift starts at time zero, that means the first operational sample derives from the OoD and previous points (of the window) are compliant with training conditions. As soon as the window collects more data (over the last n_s points), it senses more information about the OoD. As to the $W\mu I$ metric, the results confirm that the shorter the window, the faster the detection. On the other hand, the μI metric experiences a noise that can have different meanings as detailed later on. The following evidence arises for the case studies. In RUL, $W\mu I$ (Fig. 2a) needs at least 200 samples to exit the baseline; this happens with the shortest window ($n_s = 5000$) and with the most divergent OoD (op_a with respect to op_b). The l_1 norm (Fig. 2c) outlines a similar behaviour. μI (Fig. 2b) does not trigger the expected ODD; this seems in contrast with previous results in table VIII, where ODD was successful. This is however due to the limited horizon of the figure; the curves under $n_s = 5 \cdot 10^3$ are actually approaching the baseline and, as expected, op_a reveals to be faster than op_b, being more divergent from tr_1 than op_b. The groupwise progression thus suggests the joint adoption of the metrics to achieve both precision ($W\mu I$) and measure of the distributions similarity (μI). In platooning, $W\mu I$ matches the ODD and, coherently with previous results (table IX), μI is stuck in the baseline. Finally, DNS has good performance with the two metrics as well. The difference between RUL and platooning in μI is remarkable as it is very subtle. In the former case, μI is sensitive to distributions similarity, still being able to slowly proceed in the ODD direction. In the latter, it experiences imprecise calculations (as shown in the appendix), thus complicating the ODD task.

It is finally worth noting that the window of the incremental groupwise should be coherent with the design setting with $n_s = 5000$. Other results may show several counterexamples in RUL with $n_s = 1000$ and $tr_1 - tr_1$, in which, though only points in the baseline would have been expected, many false positives take place.

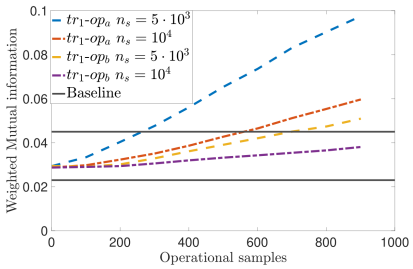
IX. Conclusion and Future Work

The paper deals with the identification of OoD through a distributional assumption free rule-based model. The approach also measures the proximity of in and out-of distributions and is validated in challenging case studies. Future extensions comprise further testing on additional longitudinal datasets, as well as on image data. Alternative ways to the hits of the ruleset to infer in-distribution behaviour are of interest, as well as additional metrics to measure in and out of distribution divergence. Also, a performance evaluation between the complete ruleset and the one obtained on some selected features or the one containing the most understandable rules could be of great interest, as well as the investigation on how to enhance the interpretability of the process, by studying how to properly make the methods suitable for a reduced, more interpretable, set of rules.

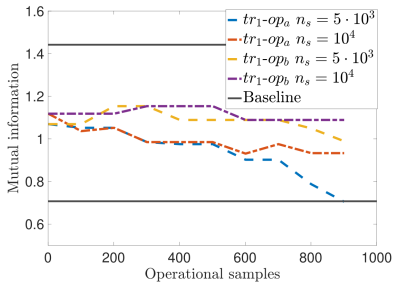
Appendix

Rationale of mutual information modification

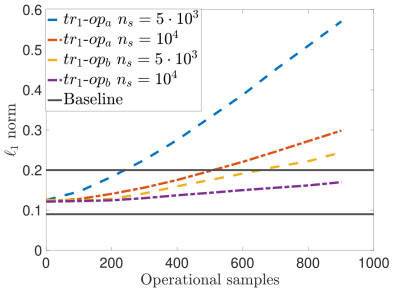
When comparing couples of histograms, (μI) is useful to identify the dependence but it does not capture the differences among their values. Suppose we get these three histograms A, B and C (Table XVIII). Considering the simple μI,



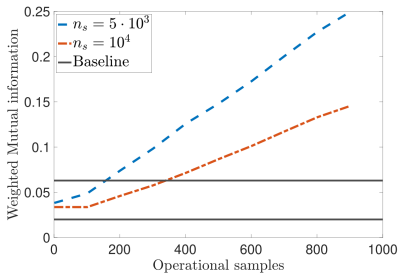
(a) RUL tr_1-op_a and tr_1-op_b with $W\mu I$.



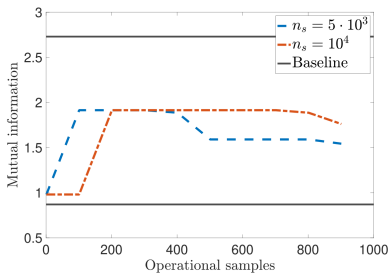
(b) RUL tr_1-op_a and tr_1-op_b with μI.



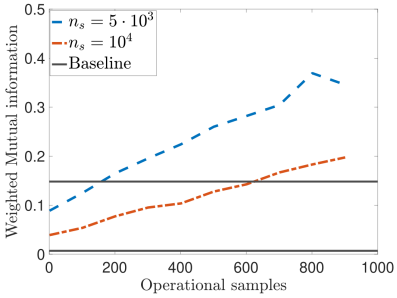
(c) RUL tr_1-op_a and tr_1-op_b with l_1 norm.



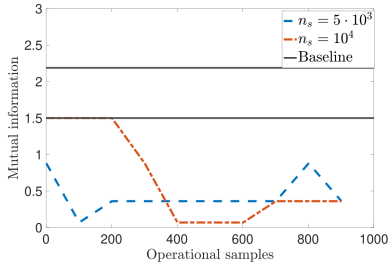
(d) Platooning with $W\mu I$.



(e) Platooning with μI.



(f) DNS with $W\mu I$.



(g) DNS μI.

Fig. 2: Incremental group-wise results

TABLE XVIII: Example of μI

	A	B	C
r_1	0.166	0.211	0.399
r_2	0.182	0.214	0.387
r_3	0.438	0.387	0.214
r_4	0.424	0.399	0.211

histograms A and B are dependent and so A and C are; but B and C are different (they have same values but in different positions). Since each row of the tr and op histograms corresponds to a rule, μI may have a detrimental effect as the rule hits contain the information to the OoD. The correction to overcome this issue consists of weighting the probabilities (used in entropy calculations) through the average of hits differences in each rule/row; this leads to $\alpha_{i,j}$ quantities in Algorithm 1. The more the histograms are dependent and similar, the more $W\mu I$ goes towards zero. Similar considerations hold for RBI, with $N_{op} > 1$, the weights ($\gamma_j^{(\cdot)}$ quantities) are the fractions of the compared probabilities.

ACKNOWLEDGEMENTS

This work was supported in part by REXASI-PRO H-EU project, call HORIZON-CL4-2021-HUMAN-01-01, Grant agreement ID: 101070028. G. De Bernardi PhD is partially funded by Collins Aerospace.

REFERENCES

[1] V. Mirasierra, M. Mammarella, F. Dabbene, and T. Alamo, "Prediction error quantification through probabilistic scaling," *IEEE Control Systems Letters*, vol. 6, pp. 1118–1123, 2022.

[2] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "Improving performance, reliability, and feasibility in multimodal multitask traffic classification with xai," *IEEE Transactions on Network and Service Management*, 2023.

[3] P. Dondio and L. Longo, "Trust-based techniques for collective intelligence in social search systems," in *Next generation data technologies for collective computational intelligence*, pp. 113–135, Springer, 2011.

[4] "EASA concept paper: First usable guidance for level 1 machine learning applications, a deliverable of the EASA AI roadmap," standard, European Union Aviation Safety Agency, Daedalean, AG, Apr. 2021. Also available as https://www.easa.europa.eu/easa-concept-paper-first-usable-guidance-level-1-machine-learning-applications-proposed-issue-01pdf.

[5] "Concepts of design assurance for neural networks codann," standard, European Union Aviation Safety Angency, Daedalean, AG, Mar. 2020. Also available as https://www.easa.europa.eu/sites/default/files/dfu/EASA-DDLN-Concepts-of-Design-Assurance-for-Neural-Networks-CoDANN.pdf.
[6] "Road vehicles safety of the intended functionality pd iso pas 21448:2019," standard, International Organization for Standardization, Geneva, CH, Mar. 2019.
[7] F. Heidecker, J. Breitenstein, K. Rösch, J. Löhdefink, M. Bieshaar, C. Stiller, T. Fingscheidt, and B. Sick, "An application-driven conceptualization of corner cases for perception in highly automated driving," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 644–651, 2021.
[8] F. Cabitza and A. Campagner, "The need to separate the wheat from the chaff in medical informatics: Introducing a comprehensive checklist for the (self)-assessment of medical ai studies," *International Journal of Medical Informatics*, vol. 153, p. 104510, 2021.
[9] V. Sehwag, M. Chiang, and P. Mittal, "{SSD}: A unified framework for self-supervised outlier detection," in *International Conference on Learning Representations (ICLR)*, 2021.
[10] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," *arXiv preprint arXiv:2204.06507*, 2022.
[11] T. Q. Dinh, Y. Xiong, Z. Huang, T. Vo, A. Mishra, W. H. Kim, S. N. Ravi, and V. Singh, "Performing group difference testing on graph structured data from gans: Analysis and applications in neuroimaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 877–889, 2022.
[12] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *Advances in neural information processing systems*, vol. 31, 2018.
[13] J. Bitterwolf, A. Meinke, M. Augustin, and M. Hein, "Breaking down out-of-distribution detection: Many methods based on OoD training data estimate a combination of the same core quantities," in *International Conference on Machine Learning*, pp. 2041–2074, PMLR, 2022.
[14] S. Liang, Y. Li, and R. Srikant, "Principled detection of out-of-distribution examples in neural networks. corr abs/1706.02690 (2017)," *arXiv preprint arXiv:1706.02690*, 2017.
[15] V. Abdelzad, K. Czarnecki, R. Salay, T. Denounden, S. Vernekar, and B. Phan, "Detecting out-of-distribution inputs in deep neural networks using an early-layer output," 2019.
[16] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21464–21475, 2020.
[17] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.
[18] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*, pp. 1–5, 2018.
[19] J. Diers and C. Pigorsch, "Out-of-distribution detection using outlier detection methods," in *International Conference on Image Analysis and Processing*, pp. 15–26, Springer, 2022.
[20] M. Guarrera, B. Jin, T.-W. Lin, M. A. Zuluaga, Y. Chen, and A. Sangiovanni-Vincentelli, "Class-wise thresholding for robust out-of-distribution detection," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2836–2845, 2022.
[21] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*, pp. 1–5, 2018.
[22] M. Muselli, "Switching neural networks: A new connectionist model for classification," 01 2005.
[23] S. Parodi, C. Manneschi, D. Verda, E. Ferrari, and M. Muselli, "Logic learning machine and standard supervised methods for hodgkins lymphoma prognosis using gene expression data and clinical variables," *Health Informatics Journal*, vol. 24, 06 2016.
[24] S. Narteni, V. Orani, I. Vaccari, E. Cambiaso, and M. Mongelli, "Sensitivity of logic learning machine for reliability in safety-critical systems," *IEEE Intelligent Systems*, vol. 37, no. 5, pp. 66–74, 2022.
[25] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.
[26] S. Theodoridis and K. Koutroumbas, "Chapter 2 - classifiers based on bayes decision theory," in *Pattern Recognition (Third Edition)* (S. Theodoridis and K. Koutroumbas, eds.), pp. 13–67, San Diego: Academic Press, third edition ed., 2006.
[27] D. Jiang, S. Sun, and Y. Yu, "Revisiting flow generative models for out-of-distribution detection," in *International Conference on Learning Representations*, 2022.
[28] A. Onan, "Bidirectional convolutional recurrent neural network architecture with group-wise enhancement mechanism for text sentiment classification," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 5, pp. 2098–2117, 2022.
[29] M. Aiello, M. Mongelli, M. Muselli, and D. Verda, "Unsupervised learning and rule extraction for domain name server tunneling detection," *Internet Technology Letters*, vol. 2, no. 2, p. e85, 2019.
[30] "Turbofan engine degradation simulation data set, https://www.kaggle.com/datasets/behrad3d/nasa-cmaps." Accessed: Feb 2023.
[31] M. Mongelli, "Design of countermeasure to packet falsification in vehicle platooning by explainable artificial intelligence," *Computer Communications*, vol. 179, pp. 166–174, 2021.
[32] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis, "Cooperative adaptive cruise control: Definitions and operating concepts," *Transportation Research Record*, vol. 2489, no. 1, pp. 145–152, 2015.
[33] M. Aiello, M. Mongelli, and G. Papaleo, "Dns tunneling detection through statistical fingerprints of protocol messages and machine learning," *International Journal of Communication Systems*, vol. 28, no. 14, pp. 1987–2002, 2015.
[34] A. Onan, "Biomedical text categorization based on ensemble pruning and optimized topic modelling," *Computational and Mathematical Methods in Medicine*, vol. 2018, 2018.

Giacomo De Bernardi Got his Master degree in Mathematics at the University of Milano Bicocca on November 2021. He is currently a PhD student in the PhD programme on Trustworthy AI at university of Genoa, working at CNR-IEIIT Institute and at Collins Avionics. He works on data analytics topics from different fields, such as industry, aerospace and automotive, with specific focus on Explainable Artificial Intelligence, deep learning and machine learning methods and applications.

Sara Narteni Got her M.Sc. in Bioengineering at the University of Genoa on March 2020. She is currently a PhD student in the italian National PhD programme on Artificial Intelligence at Politecnico di Torino, working at CNR-IEIIT Institute. She works on data analytics topics from different fields, such as industry, healthcare and automotive, with specific focus on Explainable Artificial Intelligence methods and applications.

Enrico Cambiaso Enrico Cambiaso, Ph.D in Computer Science, has a background working experience as a computer scientist, for both small and big enterprises. He is currently employed at the IEIIT institute of Consiglio Nazionale delle Ricerche (CNR), as a technologist working on cyber-security topics and focusing on the design of last generation threats. He is author of more than 50 scientific papers on cybersecurity and he's been involved in several financed research projects, at national and European level.

Maurizio Mongelli obtained his PhD. Degree in Electronics and Computer Engineering from the University of Genoa in 2004. He worked for Selex and the Italian Telecommunications Consortium (CNIT) from 2001 until 2010. He is now a researcher at CNR-IEIIT, where he deals with machine learning applied to health and cyber-physical systems. He is co-author of over 100 international scientific papers, 2 patents and is participating in the SAE G-34/EUROCAE WG-114 AI in Aviation Committee.