

Out-Of-Distribution Detection Is Not All You Need

Joris Guérin,^{1, 2, 4} Kevin Delmas,³ Raul Sena Ferreira,^{1, 2} Jérémie Guiochet^{1, 2}.

¹Université de Toulouse, ²LAAS-CNRS, ³ONERA, Toulouse, France

⁴Espace-Dev, IRD, univ. Montpellier, Montpellier, France

joris.guerin@ird.fr, kevin.delmas@onera.fr, rsenaferre@laas.fr, jeremie.guiochet@laas.fr

Abstract

The usage of deep neural networks in safety-critical systems is limited by our ability to guarantee their correct behavior. Runtime monitors are components aiming to identify unsafe predictions and discard them before they can lead to catastrophic consequences. Several recent works on runtime monitoring have focused on out-of-distribution (OOD) detection, i.e., identifying inputs that are different from the training data. In this work, we argue that OOD detection is not a well-suited framework to design efficient runtime monitors and that it is more relevant to evaluate monitors based on their ability to discard incorrect predictions. We call this setting out-of-model-scope detection and discuss the conceptual differences with OOD. We also conduct extensive experiments on popular datasets from the literature to show that studying monitors in the OOD setting can be misleading: 1. very good OOD results can give a false impression of safety, 2. comparison under the OOD setting does not allow identifying the best monitor to detect errors. Finally, we also show that removing erroneous training data samples helps to train better monitors.

1 Introduction

With the recent progress in machine learning (ML) research, deep neural network (DNN) architectures are now used to address safety-critical tasks, e.g., self-driving cars (Stocco et al. 2020), surgical robots (Haidegger 2019), drones landing (Guerin, Delmas, and Guiochet 2022). Online fault tolerance approaches, or runtime monitors, are promising research directions to improve the safety of such systems. A runtime monitor is a component aiming to identify and reject unsafe data encountered at inference time. As in most of the recent literature, this paper focuses on the unsupervised setting, where we do not have access to examples of “unsafe input data” during monitor training. In other words, to separate safe data instances from unsafe ones, we need to fit a one-class classifier (Khan and Madden 2014) using only the DNN training dataset. Figure 1 illustrates the life cycle of a DNN runtime monitor (training and inference phases).

Many approaches have been proposed to tackle the DNN runtime monitoring problem defined above. However, in the literature, they are found under different names as they adopt different definitions of “unsafe data instances”. On the one

hand, the field of Out-Of-Distribution (OOD) detection aims at identifying input data that are not from the training distribution (Wang et al. 2022; Sun, Guo, and Li 2021; Schorn and Gauerhof 2020). On the other, several works consider directly the problem of identifying input data that lead to errors of the monitored DNN (Granese et al. 2021; Wang et al. 2020; Kang et al. 2018). In this work, we name this second view Out-of-Model-Scope (OMS) detection. In practice, the approaches to address OOD and OMS detection are not different and follow the same workflow: they use the DNN training dataset to build a one-class classifier (the monitor) to characterize safe data instances and use it to reject unsafe samples (Figure 1). These two paradigms only differ in their objectives (definition of normal data samples), and by extension in how new approaches are evaluated. We emphasize that OOD and OMS are evaluation settings and not monitoring approaches per se.

In this work, we claim that the real goal of a DNN runtime monitor is to tackle the OMS problem, i.e., to identify prediction errors before they can propagate through the system. We argue that OOD detection was designed as a proxy task for OMS detection, based on the belief that what the DNN knows is equivalent to the information contained in the dataset that was used to train it. The first objective of this paper is to discuss the conceptual differences between the OOD and OMS detection settings. Then, we conduct experiments to determine whether OOD detection is a good proxy for OMS detection. In other words, we want to know whether the OMS performance of a runtime monitor can be correctly assessed under the OOD setting. Finally, based on the insights gained from these discussions, we formalize a good practice to train better monitors. We show experimentally that, when fitting the monitor, it is better to remove erroneous samples from the DNN training dataset.

This paper is organized as follows. In section 2, notations are introduced and the concepts of OMS and OOD are clearly defined. In section 3, we present the existing literature about OMS and OOD detection. In section 4, we discuss the differences between OOD and OMS and explain why the OOD setting can lead to an inaccurate perceived performance of a monitor. In section 5, we conduct experiments to verify whether these limitations of OOD detection actually occur in practical scenarios. Finally, in section 6, we introduce a new training trick to build better monitors.

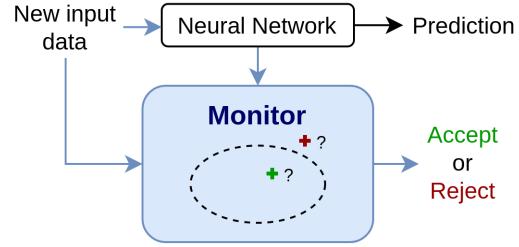
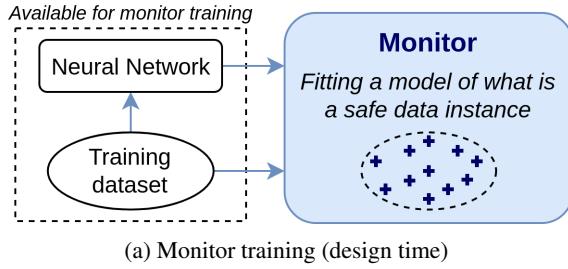


Figure 1: **Neural Network Runtime Monitoring** is a research area aiming to characterize safe data instances, in order to reject unsafe predictions during inference. Different definitions of *safe data instances* gave rise to different subfields in the literature.

2 Notations and definitions

Let T be an ML task, defined by an oracle function Ω , on a domain \mathcal{X} , i.e., $\forall x \in \mathcal{X}$, the ground-truth for T is $\Omega(x)$. This work mostly discusses classification, but could easily be extended to other ML tasks. Let f be a DNN used to solve T , and let m_f be a monitor of f , i.e., a one-class classifier used during inference to reject unsafe predictions of f .

2.1 The Out-of-Model-Scope detection setting

We define the scope of f , \mathcal{D}_f , to be the set of data instances where f is correct:

$$\mathcal{D}_f = \{x \in \mathcal{X} \mid f(x) = \Omega(x)\} \quad (1)$$

Ideally, we want m_f to identify data points that lead to errors of f . The perfect monitor for f , noted m_f^* , is defined by

$$\forall x \in \mathcal{X}, m_f^*(x) = \begin{cases} 0 & \text{if } x \in \mathcal{D}_f, \\ 1 & \text{else.} \end{cases} \quad (2)$$

We call Out-of-Model-Scope (OMS) detection, the task of designing a monitor that reproduces the behavior of m_f^* . OMS detection is defined with respect to a specific model f . Indeed, if another neural network f' (different architecture or weights) is used, the model scope will likely be different ($\mathcal{D}_{f'} \neq \mathcal{D}_f$), and so will the optimal monitor ($m_{f'}^* \neq m_f^*$).

2.2 The Out-Of-Distribution detection setting

In practice, the monitored DNN f is trained using a supervised training dataset, i.e., a small subset of n data points in \mathcal{X} for which the ground truth is known:

$$D_{\text{train}} = \{(x_i, y_i) \mid \forall i \in \{1, \dots, n\} x_i \in \mathcal{X}, y_i = \Omega(x_i)\}.$$

A common practice for DNN monitoring is to define an in-distribution domain \mathcal{D}_{ID} , that comprises all data instances drawn from the same distribution as D_{train} . The Out-Of-Distribution (OOD) detection task aims to build a monitor m to identify data instances that do not belong to \mathcal{D}_{ID} , i.e., the perfect OOD monitor (m^*) is defined by

$$\forall x \in \mathcal{X}, m^*(x) = \begin{cases} 0 & \text{if } x \in \mathcal{D}_{\text{ID}}, \\ 1 & \text{else.} \end{cases} \quad (3)$$

The rationale behind OOD detection is that DNNs trained on D_{train} must be good for input data similar to D_{train} ($x \in \mathcal{D}_{\text{ID}}$), but should not perform well on other data ($x \notin \mathcal{D}_{\text{ID}}$). The evaluations conducted in the OOD setting are independent of f , they only depend on the task and training dataset.

3 Related works

This section presents several monitoring approaches. Conceptually, OOD and OMS detectors are not different, i.e., one-class classifiers fitted to D_{train} , usually using feature representations extracted from f (Figure 1). However, in practice, they differ in the way they are evaluated experimentally.

3.1 OMS in the literature

Several previous works have studied the problem of OMS detection, i.e., they developed DNN runtime monitoring approaches and assessed their performance based on their ability to detect errors of f . Granese et al. (2021) developed a monitor called DOCTOR, which computes an optimal rejection score based on the softmax vector returned by f . Wang et al. (2020) proposed a monitor called DISSECTOR, which rejects inputs with incoherent activations among different layers of f . The monitor proposed by Kang et al. (2018) is specific to object detection. They developed a model assertion technique to specify constraints on the shape of the predicted bounding boxes. Cheng, Nührenberg, and Yasuoka (2019) proposed to store neuron activations of D_{train} in the form of binary decision diagrams, and reject patterns that were not seen previously.

3.2 OOD detection in the literature

The definition of OOD in equation 3 is frequently seen in the literature. However, the boundaries of \mathcal{D}_{ID} are fuzzy and there is no clear definition of whether a data point was drawn from the same distribution as D_{train} . To overcome this issue, OOD detection works consider that the test split associated with D_{train} belongs to \mathcal{D}_{ID} . Then, different approaches exist to construct OOD sets, and the monitors are evaluated based on their ability to distinguish whether a data point is from the test set or the OOD set. In the literature, three main concepts of “OODness” are used to build OOD datasets.

Novelty A data point $x \in \mathcal{X}$ is OOD if the ground-truth $\Omega(x)$ is not among the predefined classes handled by f , i.e., f cannot be correct (e.g., an image of a frog presented to a cat vs. dog classifier). A large body of works was developed and evaluated in this configuration. Liang, Li, and Srikant (2018) proposed ODIN, an approach using input preprocessing and temperature scaling on the softmax outputs to maximize ID/OOD separation. Liu et al. (2020) proposed an en-

ergy score (*logsumexp* of the logits) to detect OOD data. Sun, Guo, and Li (2021) showed that clipping the highest values of the last layer can help to build better logits for OOD detection. Their approach is called ReAct and can be combined with any logits-based scores. Lee et al. (2018) proposed to fit Gaussian distributions to features extracted from f , and to use the Mahalanobis distance as an OOD score. All the above approaches used another dataset, with disjoint label classes, as the OOD set. Another idea to build OOD sets is to remove data samples from certain classes while training f and m_f , and use them as OOD examples to test the monitor. This was done by Henzinger, Lukina, and Schilling (2020), who developed a monitor called Outside-the-Box, using the smallest bounding boxes containing all features representing D_{train} to model the boundary of \mathcal{D}_{ID} .

Covariate shift OOD data instances present different characteristics than D_{train} , but with valid ground-truth. Covariate shift comes from changes in external conditions, sensor failures, or modifications to the environment. Such threats to DNN safety were discussed extensively by Hendrycks and Dietterich (2019). Most works dealing with covariate shift detection have built OOD sets by injecting perturbations to test images. Schorn and Gauerhof (2020) injected different kinds of noise patterns and evaluated their monitor called FACER. Cai and Koutsoukos (2020) used a variational auto-encoder and anomaly detection to detect abnormal rain conditions (injected in simulation). Chen, Yoon, and Shao (2020) applied brightness changes to the test images to build an OOD set to test their approach. Finally, Zhang et al. (2018) used distances in the feature space of a DNN to separate sunny images (used for training) from rainy or snowy images (from YouTube).

Adversarial attack A data sample that was modified to make a DNN fail with high confidence. The difference with covariate shift is in the malicious intent to generate imperceptible perturbations. Several works presented above also tested their approaches for adversarial attack detection. In addition, Kantaros et al. (2021) proposed to detect adversarial attacks by testing the robustness of a prediction against input image transformations. Similarly, Wang et al. (2019) considered the robustness against random mutations of f .

3.3 Approaches considering both OMS and OOD

Hendrycks and Gimpel (2017) proposed to use the Maximum Softmax Probability (MSP) as a rejection score. They evaluated the performance of MSP under both the OOD and OMS settings. Ferreira et al. (2021) conducted a benchmark study of OOD detection methods. Their experiments included novelties, covariate shifts, and adversarial attacks. They also reported the OMS detection performance. However, in both of these papers, OOD and OMS are simply viewed as separate problems. Here, we consider that OOD is simply a proxy for OMS, and we discuss whether both paradigms are useful for the field of DNN monitoring.

4 Limitations of OOD detection

In this work, we claim that a successful runtime monitor m_f should perform well under the OMS setting, i.e., reject data

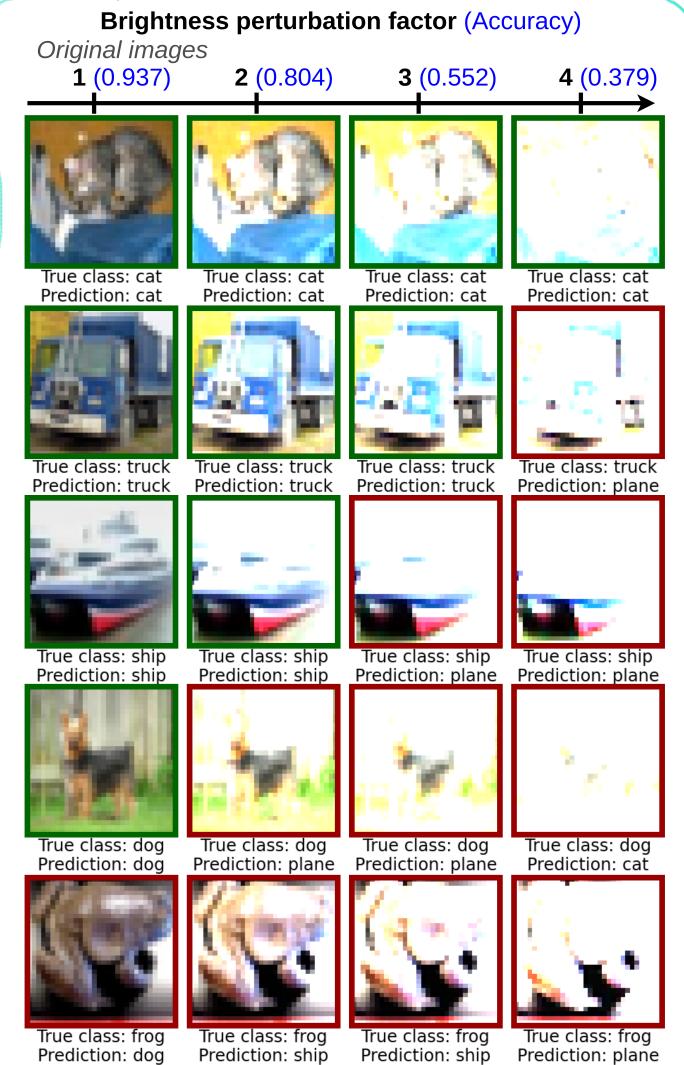


Figure 2: **What is Out-Of-Distribution?** Images from CIFAR10 test set with predictions by a ResNet-34 model. The columns represent increasing brightness perturbation levels.

samples that correspond to errors of f , and accept others. DNNs are usually good on their training dataset (train, validation, and test splits), while their performance decreases when data move away from D_{train} (Hendrycks and Dietterich 2019). These two facts gave rise to the highly studied problem of OOD detection, aiming to reject samples that are far from the training distribution. Studying the alternate OOD detection problem in place of OMS detection presents several conceptual issues, which are highlighted in this section.

4.1 The definition of OOD is ambiguous

The first issue with OOD detection is that \mathcal{D}_{ID} is not well-defined. Figure 2 shows examples of brightness covariate shifts to illustrate this problem. Here, it is not clear how to set a threshold on the perturbation factor to define OODness. At a perturbation level of 2, the accuracy of f has dropped

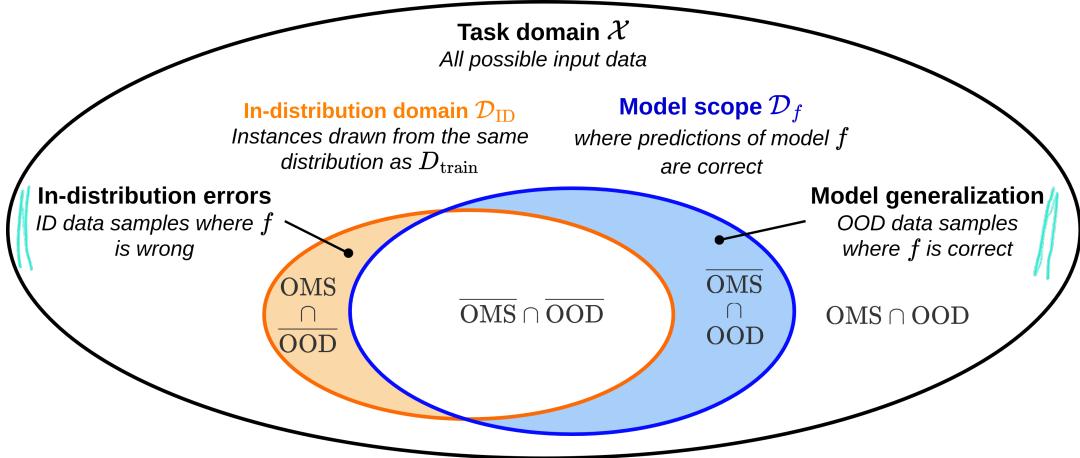


Figure 3: **Differences between OOD and OMS detection.** Schematic representation of the in-distribution domain \mathcal{D}_{ID} for a task, as well as the model scope \mathcal{D}_f of a model f for this task.

about 13%, but it can still produce above 80% of correct predictions. Hence, should the second column be considered OOD? This limitation also applies to adversarial examples, as different attacks lead to different levels of performance drop. For example, if we can generate a few additional misclassifications by changing random pixels, is it an adversarial attack? Regarding novelty detection, the problem is better defined: an image is OOD if its label is not among the output classes handled by f (e.g., an image of a dinosaur is clearly OOD for CIFAR10). However, it is not easy to decide whether a highly corrupted image should hold its label. In figure 2, is the last image of the fourth row still a dog?

There is no simple answer to the questions raised in the previous paragraph. As a matter of fact, Balestrieri, Pesenti, and LeCun (2021) showed that for high-dimensional data spaces, such as images, interpolation almost never happens. Even test data, which are traditionally considered ID, are not in the convex hull of D_{train} and require extrapolation capabilities of f . In other words, the only data samples that are unambiguously ID are the training data themselves. This way, building an OOD evaluation dataset requires making subjective choices regarding the definition of OODness.

Recently, Wang et al. (2022) attempted to build a less ambiguous OOD benchmark dataset by asking two independent human annotators to label OOD images for ImageNet. The resulting OOD set contains all images labeled as OOD by both annotators. Unfortunately, this approach does not really solve the problem, as difficult images for humans are not necessarily what the DNN will struggle to process. In addition, even if this dataset is actually difficult for f , how can we know that it is not too simple for m_f ? What about images that are less clearly OOD?

A less ambiguous way to discuss the concept of OOD is to use ideas from the field of DNN calibration (Guo et al. 2017): a set of images is OOD if the accuracy of f falls below a fixed user-defined threshold on this set. This definition is still threshold dependent, but it describes OODness unambiguously. However, under this definition, OODness

depends on f and is a property of a set, not defined at the individual sample level. This definition could be used to detect a performance drop over time by evaluating the OOD rate in consecutive data, but it is a different problem than the one studied in most OOD detection works.

4.2 OOD does not always represent OMS

Even if we had access to an unambiguous boundary of \mathcal{D}_{ID} , OOD and OMS detection would still remain distinct problems. In the schematic representation of figure 3, we can see that \mathcal{D}_{ID} and \mathcal{D}_f can differ in two different ways.

Model generalization The first difference, represented by the light blue region, is when input data are OOD but correctly classified by f (e.g., the top-right image of figure 2). The limits of the ID domain are often defined by human programmers, independently of the generalization capabilities of f . For such cases, a good OOD monitor will reject perfectly valid inputs (false positives), thus decreasing the availability¹ of the model f . For the special case of novelty detection, this region does not exist because the model cannot be correct on data representing novel classes.

In-distribution errors The second difference between OOD and OMS lies in the orange region in figure 3. In-distribution errors are data samples that are similar to training data, but that are misclassified by f (e.g., the bottom-left image of figure 2). Such cases can have catastrophic outcomes for safety-critical applications, since if an OOD monitor accepts wrong predictions as long as they are similar to D_{train} , it also lets hazardous predictions through the system (false negatives) and decreases its safety. This problem is an issue even when considering novelty detection.

Furthermore, it is counter-productive to expect a monitor to accept misclassified training data. If a data sample is misclassified, its representation by the DNN is likely not similar

¹In the dependability literature, the availability of a system represents its readiness to deliver a correct service.

to other representatives of its class, and forcing the monitor to say otherwise does not make sense.

4.3 Switching to OMS

The above discussion showed that it is hard to come up with an objective definition of what OOD means, and that building good OOD monitors might not be sufficient to ensure the safety of a system using a DNN. The good news is that such a definition is not necessary, as we can simply use OMS as a generic setting for studying DNN runtime monitoring. The OMS paradigm is defined unambiguously and actually corresponds to what we want to achieve, i.e., reject wrong predictions of f . In practice, it is common to define a proxy task (here OOD detection) to address a more complex generic task of interest (OMS detection). However, when the actual performance at the true task is easy to compute, we should use it to evaluate the performance of new techniques. In our case, computing OMS detection performance is easy as we have access to labeled datasets and we can compute the predictions of f . For these reasons, we argue that DNN runtime monitors should be evaluated with respect to the OMS setting, instead of the ambiguous OOD setting.

5 Experimental validation

So far, we have argued that if the ultimate goal is to increase DNN safety, we should always evaluate runtime monitors under the OMS setting. However, many recent works have reported the performance of their approaches under the OOD setting (Section 3.2). In this section, we conduct experiments on common OOD datasets from the literature, to see whether analyzing OOD results can lead to misleading conclusions about the safety of the system.

5.1 Experiments description

Our experiments consist of 27 OOD datasets, 2 DNN architectures (54 OOD scenarios), and 6 monitoring approaches.

We use three popular image datasets as ID: CIFAR10, CIFAR100, and SVHN (Krizhevsky, Hinton et al. 2009; Netzer et al. 2011). For each ID dataset, the train split is used to fit the monitors, while the test split serves as the ID set for evaluation. Each ID set is combined with 9 distinct OOD sets:

- **Three novelty tasks** – For CIFAR10 we use CIFAR100, SVHN, and LSUN (Yu et al. 2015) to represent novel data. For CIFAR100 we use CIFAR10, SVHN, and LSUN. For SVHN we use CIFAR10, LSUN, and Tiny-ImageNet (subset of ImageNet (Deng et al. 2009)). With these choices, ID and OOD classes never overlap.
- **Three covariate shift tasks** – For each ID dataset, we apply three image transformations from the AugLy library (Papakipos and Bitton 2022): Brightness (factor=3), Blur (radius=2), and Pixelization (ratio=0.5).
- **Three adversarial attacks** – For each ID dataset, we apply three adversarial attacks from Torchattacks (Kim 2020), with default parameters: FGSM, DeepFool, and PGD.

For each of these 27 OOD scenarios, we experiment with two DNN architectures: DenseNet and ResNet. We use the pre-trained models from Lee et al. (2018).

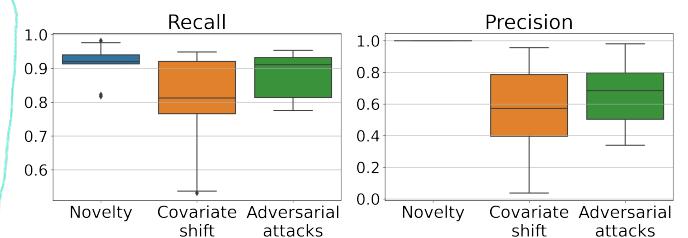


Figure 4: **OMS results for optimal OOD monitors.** Distribution of the OMS recall and precision obtained by the optimal OOD detector m^* across the 27 OOD datasets and two neural networks tested in our experiments.

For monitoring, we test two feature-based approaches: Mahalanobis (**Maha**) (Lee et al. 2018) and Outside-the-Box (**OtB**) (Henzinger, Lukina, and Schilling 2020). We use the last layer before classification as their data representation. We also use four logit-based approaches: Max Softmax Probability (**MSP**) (Hendrycks and Gimpel 2017), Energy (**Ene**) (Liu et al. 2020), and ReAct combined with both MSP (**R-MSP**) and Energy (**R-Ene**) (Sun, Guo, and Li 2021). Except for OtB, the techniques used in our experiments require selecting a rejection threshold on the monitoring scores. This is a complex question that is not studied here. Instead, we consider the optimal F1 threshold, as described by Guérin, de Paula Canuto, and Goncalves (2020).

The complete code to reproduce our experiments can be found at <https://github.com/jorisguerin/neural-network-monitoring-benchmark>

5.2 OOD can give a false sense of safety

Here, we want to answer the following question: if we manage to develop a perfect OOD detector m^* , can we guarantee that (f, m^*) is safe to use in critical applications?

Let's suppose that we are able to build m^* , which rejects all OOD samples and accepts all ID samples (precision and recall of 1 at OOD detection). In our experiments, we can easily simulate the predictions of m^* using the known binary ID/OOD labels. Then, we want to know how well m^* performs the task of detecting OMS data samples. An OMS recall below one means that there exist ID data points for which f is erroneous. It represents a threat to the safety of the system. An OMS precision below one means that there exist OOD data points for which f is correct. It represents a decrease in the availability of the system.

We conduct this experiment for both DNN architectures, across the 27 OOD datasets. The distribution of the OMS performances obtained for the different OOD types is reported in Figure 4. We can see that the OMS recall is not close to 1 for most of the experiments conducted. Even for novelty, the median OMS recall of the 18 experiments is below 0.93. In other words, in most cases, more than 7% of the errors of f are not detected by the perfect OOD monitor m^* , which can lead to catastrophic outcomes for safety-critical applications. This effect is even more pronounced for covariate shifts and adversarial attacks.

These results show that, even if perfect OOD detectors

	Maha	MSP	Ene	R-MSP	R-Ene	
OtB	3e-01	1e-01	1e-01	1e-01	4e-02	5e-07
Maha	3e-01	1e-01	7e-01	5e-02		4e-01
MSP	1e-06	1e-01	1e-01	3e-02		3e-08
Ene		3e-05	2e-02			2e-06
R-MSP		3e-05	2e-02			2e-05

(a) Recall

	Maha	MSP	Ene	R-MSP	R-Ene	
OtB	1e-04	2e-03	3e-03	1e-03	5e-04	1e-09
Maha		2e-01	3e-01	2e-01	5e-01	2e-01
MSP			1e-02	9e-01	2e-03	1e+00
Ene				5e-02	4e-01	5e-01
R-MSP					5e-03	1e-01

(b) Precision

Figure 5: **Pairwise comparison of monitors.** Popular monitors are compared across 54 OOD scenarios using a Wilcoxon test. The obtained p-values are reported. Green cells indicate that the approach corresponding to the row is better than the column, red cells indicate that the column is better than the row, and white cells indicate no statistically significant difference ($p > 0.05$).

are developed, it will not allow us to guarantee the safety of critical systems using ML. Indeed, an OOD recall of 1 does not guarantee the absence of prediction errors. Even worse, reporting very good OOD results can be detrimental as it gives a false sense that the (f, m^*) system is safe.

Regarding OMS precision, we first note that it is always 1 for novelty datasets. This is because f cannot be correct for novel samples, by definition. For other OOD types, we acknowledge an important drop in precision, i.e., an important proportion of correct predictions of f on OOD data.

5.3 OOD can be misleading for comparison

As explained in section 3.2, monitors are often compared using the OOD setting. Here, we want to verify whether comparing monitors at the OOD detection task provides relevant insights into OMS detection performance.

To do this, we compare six monitors from the literature across 27 OOD datasets and two DNNs architectures (section 5.1). For each pair of monitors, we conduct a Wilcoxon signed-rank test across the 54 OOD scenarios to determine whether one is better than the other. The Wilcoxon test is a non-parametric statistical test that can be used to compare pairs of classifiers across multiple datasets (Demšar 2006). The results obtained are reported in figure 5.

From these experiments, we can see that the comparison matrices obtained for OOD and OMS look different. For example, OtB is clearly the most conservative approach (highest recall) when looking at the OMS results, but not when considering OOD. On another note, the benefits of using ReAct for monitoring are best seen in the OMS setting (MSP vs. R-MSP, Ene vs. R-Ene).

From these results, it appears that conducting OOD experiments is not a reliable way to compare monitoring approaches regarding their ability to identify errors of f .

6 Removing training outliers to build better runtime monitors

As explained earlier (Figure 1), runtime monitors are often fitted using the DNN training dataset to represent “what is safe”. The common practice in the literature is to use all available D_{train} samples to fit the monitor. When considering OOD, the objective is to build a binary classifier that can reject “data samples from outside of the training distribution” (dissimilar to D_{train}). Hence, in this context, it makes sense to fit the monitor using all available training data samples.

In practice, DNNs rarely perform perfectly, even on the exact data used for training. For example, the ResNet-34 used in this work misclassifies 127 of the 50,000 training data samples (training accuracy of 99.746%). When considering the OMS setting, the objective is to build a classifier to identify data samples that are misclassified by f . Hence, it is conceptually wrong to use the misclassified training samples to represent the normal class when fitting a one-class classifier for OMS detection. In this section, we propose a simple trick to build better DNN monitors, consisting in removing erroneous training samples to fit the monitor.

To illustrate this idea, we apply a two-dimensional Uniform Manifold Approximation and Projection (UMAP) transformation (McInnes, Healy, and Melville 2018) to the training split of CIFAR10. UMAP is an unsupervised embedding method that produces a lower-dimensional representation preserving the global distances. The 2D representation obtained allows us to investigate visually whether the misclassified training samples are outliers, which change the shape of the one-class classifier boundary. The embeddings of four CIFAR10 classes are shown in figure 6. It shows that correctly classified points form compact clusters while misclassified points appear to be clear outliers. To further illustrate our point, we draw the minimal bounding box containing all training samples, and the one containing only correctly classified samples. These bounding boxes represent the one-class classification boundaries of an Outside-the-Box monitor without K-means pre-processing (Henzinger,

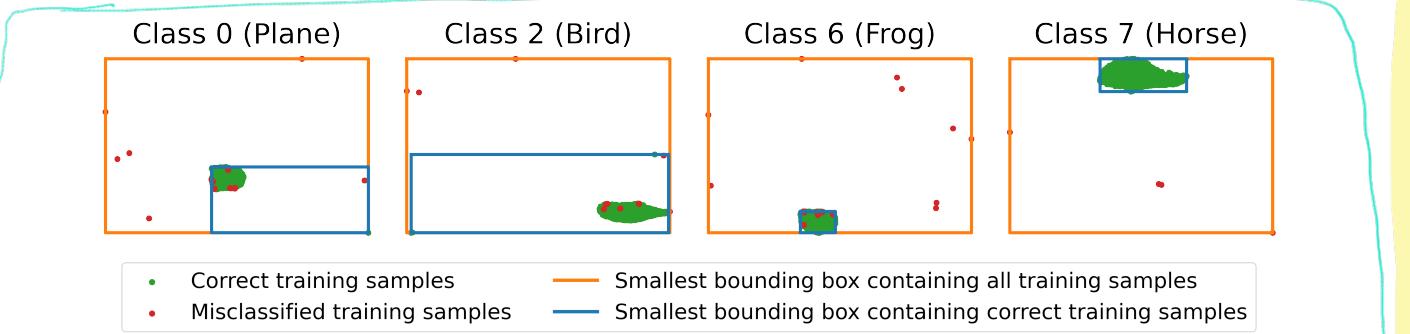


Figure 6: **Should we use misclassified training samples to fit a monitor?** Two-dimensional UMAP visualization of four selected classes from the CIFAR10 training split. Colors indicate whether they were correctly classified by the ResNet-34 model. We also show the minimal bounding boxes containing all training samples (orange) and all correctly classified training samples (blue). Misclassified training samples are often outliers, leading to much larger bounding boxes when they are included.

	Recall Mean (St. dev.)	Precision Mean (St. dev.)
All training data	0.75 (0.19)	0.68 (0.18)
Only correct data	0.77 (0.18)	0.68 (0.18)
Wilcoxon test	Better	Worst
p-value	2.6e-9	1.6e-3

Table 1: OMS results for Outside-the-box

Lukina, and Schilling 2020). The blue boxes, obtained by removing misclassified samples, are much tighter than the orange ones, and appear to better represent the model scope.

Figure 6 suggests that removing misclassified training data helps to fit better monitors. However, UMAP embeddings are not real data and one should not rely on intuition gained from such visualizations. Hence, we conduct more formal experiments and compare the performance of a simple OtB monitor (without K-means) when trained with and without misclassified training samples. The mean and standard deviation (St. dev.) of precision and recall are computed across all 54 OOD scenarios (section 5) and reported in table 1. We also conduct a Wilcoxon test and report the p-values obtained. These results demonstrate that removing misclassified training samples can help to increase the recall (clearly statistically significant) without impacting the precision too much. In other words, the monitor can detect more errors of f without discarding additional valid predictions.

Two previous works already had the intuition that removing misclassified training samples could reduce the false negative rate (Henzinger, Lukina, and Schilling 2020; Cheng, Nürenberg, and Yasuoka 2019). This section explains this intuition and shows experimental evidence of the benefits of this simple trick in the context of OMS detection.

In these experiments, we only used OtB because it is a threshold-free approach. As threshold selection strategies were not discussed in this work, other monitors were not considered here. However, when trying to select an optimal monitoring threshold, we believe that it would also be valuable to not consider misclassified samples as normal data.

7 Conclusion

If we want to use DNN in safety-critical applications, it is paramount to build efficient runtime monitors, which can detect and remove DNN errors from the system before they can have catastrophic consequences. As of today, many research efforts are directed toward solving the problem of out-of-distribution detection, which consists in identifying data samples that were drawn from a different distribution than the DNN training set. In this paper, we discuss the limitations of this setting to enable the safe usage of DNN in critical systems. First, the concept of OOD is not well-defined, which makes it difficult to compare different OOD detection approaches. Second, even a perfect OOD detector can have the undesirable property of discarding valid predictions of the DNN, and even worse, accepting wrong predictions. Extensive experiments conducted on popular OOD detection datasets confirm that these phenomena occur in practical scenarios. Furthermore, we show that the OOD setting cannot be used to compare monitoring approaches accurately, as the best monitor for OOD is not always the best one to detect DNN errors. Finally, we demonstrate experimentally that it is a good practice to remove training data points that the DNN cannot classify correctly before fitting the monitor.

On the bright side, adapting current OOD research to these findings will not require drastic changes. The out-of-model-scope (OMS) setting, defined above and discussed in several previous works, is well-defined and perfectly aligned with the DNN monitoring objectives. In addition, most approaches developed for OOD detection can be used for OMS without any modification, and OMS results can be computed straightforwardly for most OOD datasets used in the literature. Hence, the take-home message from this paper is that instead of evaluating new approaches on their ability to detect samples from other data sources, the OOD detection research community should focus on the ability to detect samples leading to erroneous DNN predictions. We also believe that it is a good idea to include OOD samples in OMS evaluation datasets, which is rarely done in OMS papers. It is worth mentioning that we recently introduced a unified evaluation formalism for runtime monitors, considering the entire system in which a DNN is included (Guerin et al. 2022).

Acknowledgments

This research has benefited from the AI Interdisciplinary Institute ANITI. ANITI is funded by the French "Investing for the Future – PIA3" program under the Grant Agreement No ANR-19-PI3A-0004.

This research has also received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 812.788 (MSCA-ETN SAS). This publication reflects only the authors' view, exempting the European Union from any liability. Project website: <http://etn-sas.eu/>.

References

- Balestrieri, R.; Pesenti, J.; and LeCun, Y. 2021. Learning in high dimension always amounts to extrapolation. *arXiv preprint arXiv:2110.09485*.
- Cai, F.; and Koutsoukos, X. 2020. Real-time out-of-distribution detection in learning-enabled cyber-physical systems. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPs)*, 174–183. IEEE.
- Chen, V.; Yoon, M.-K.; and Shao, Z. 2020. Task-Aware Novelty Detection for Visual-based Deep Learning in Autonomous Systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 11060–11066. IEEE.
- Cheng, C.-H.; Nürenberg, G.; and Yasuoka, H. 2019. Runtime monitoring neuron activation patterns. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy*, 300–303. IEEE.
- Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7: 1–30.
- Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.
- Ferreira, R. S.; Arlat, J.; Guiochet, J.; and Waeselynck, H. 2021. Benchmarking Safety Monitors for Image Classifiers with Machine Learning. In *2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 7–16. IEEE.
- Granese, F.; Romanelli, M.; Gorla, D.; Palamidessi, C.; and Piantanida, P. 2021. DOCTOR: A Simple Method for Detecting Misclassification Errors. *Advances in Neural Information Processing Systems*, 34: 5669–5681.
- Guérin, J.; de Paula Canuto, A. M.; and Goncalves, L. M. G. 2020. Robust Detection of Objects under Periodic Motion with Gaussian Process Filtering. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 685–692. IEEE.
- Guerin, J.; Delmas, K.; and Guiochet, J. 2022. Evaluation of Runtime Monitoring for UAV Emergency Landing. In *International Conference on Robotics and Automation (ICRA)*, To appear. IEEE.
- Guerin, J.; Ferreira, R. S.; Delmas, K.; and Guiochet, J. 2022. Unifying Evaluation of Machine Learning Safety Monitors. In *the 33rd IEEE International Symposium on Software Reliability Engineering (ISSRE 2022)*, To appear. IEEE.
- Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *International conference on machine learning*, 1321–1330. PMLR.
- Haidegger, T. 2019. Autonomy for surgical robots: Concepts and paradigms. *IEEE Transactions on Medical Robotics and Bionics*, 1(2): 65–76.
- Hendrycks, D.; and Dietterich, T. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*.
- Hendrycks, D.; and Gimpel, K. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *Proceedings of International Conference on Learning Representations*.
- Henzinger, T. A.; Lukina, A.; and Schilling, C. 2020. Outside the Box: Abstraction-Based Monitoring of Neural Networks. In *24th European Conference on Artificial Intelligence-ECAI 2020*, 2433–2440.
- Kang, D.; Raghavan, D.; Bailis, P.; and Zaharia, M. 2018. Model assertions for debugging machine learning. In *NeurIPS MLSys Workshop*, volume 3, 10.
- Kantaros, Y.; Carpenter, T.; Sridhar, K.; Yang, Y.; Lee, I.; and Weimer, J. 2021. Real-time detectors for digital and physical adversarial inputs to perception systems. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, 67–76.
- Khan, S. S.; and Madden, M. G. 2014. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3): 345–374.
- Kim, H. 2020. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*.
- Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.
- Lee, K.; Lee, K.; Lee, H.; and Shin, J. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31.
- Liang, S.; Li, Y.; and Srikant, R. 2018. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. In *International Conference on Learning Representations*.
- Liu, W.; Wang, X.; Owens, J.; and Li, Y. 2020. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33: 21464–21475.
- McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.
- Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
- Papakipos, Z.; and Bitton, J. 2022. AugLy: Data Augmentations for Robustness. *arXiv:2201.06494*.

- Schorn, C.; and Gauerhof, L. 2020. FACER: A universal framework for detecting anomalous operation of deep neural networks. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. IEEE.
- Stocco, A.; Weiss, M.; Calzana, M.; and Tonella, P. 2020. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 359–371.
- Sun, Y.; Guo, C.; and Li, Y. 2021. React: Out-of-distribution detection with rectified activations. *Advances in Neural Information Processing Systems*, 34: 144–157.
- Wang, H.; Li, Z.; Feng, L.; and Zhang, W. 2022. ViM: Out-Of-Distribution with Virtual-logit Matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4921–4930.
- Wang, H.; Xu, J.; Xu, C.; Ma, X.; and Lu, J. 2020. Dissector: Input validation for deep learning applications by crossing-layer dissection. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 727–738. IEEE.
- Wang, J.; Dong, G.; Sun, J.; Wang, X.; and Zhang, P. 2019. Adversarial sample detection for deep neural network through model mutation testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 1245–1256. IEEE.
- Yu, F.; Seff, A.; Zhang, Y.; Song, S.; Funkhouser, T.; and Xiao, J. 2015. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.
- Zhang, M.; Zhang, Y.; Zhang, L.; Liu, C.; and Khurshid, S. 2018. 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)*, 132–142. IEEE.