NORTH-AMERICAN
INTERDISCIPLINARY STUDIES

Article

Enhancing Out-of-Model Scope Detection with TRust Your GENerator (TRYGEN)

John Doe, Jane Smith

Zhejiang University of Technology, Department of Computer Science, Zhejiang, China

Abstract: Recent research has drawn attention to the ambiguity surrounding the definition and learnability of Out-of-Distribution recognition. Although the original problem remains unsolved, the term "Out-of-Model Scope" detection offers a clearer perspective. The ability to detect Out-of-Model Scope inputs is particularly beneficial in safety-critical applications such as autonomous driving or medicine. By detecting Out-of-Model Scope situations, the system's robustness is enhanced and it is prevented from operating in unknown and unsafe scenarios. In this paper, we propose a novel approach for Out-of-Model Scope detection that integrates three sources of information: (1) the original input, (2) its latent feature representation extracted by an encoder, and (3) a synthesized version of the input generated from its latent representation. We demonstrate the effectiveness of combining original and synthetically generated inputs to defend against adversarial attacks in the computer vision domain. Our method, TRust Your GENerator (TRYGEN), achieves results comparable to those of other state-of-the-art methods and allows any encoder to be integrated into our pipeline in a plug-and-train fashion. Through our experiments, we evaluate which combinations of the encoder's features are most effective for discovering Out-of-Model Scope samples and highlight the importance of a compact feature space for training the generator.

Keywords: Out-of-Distribution (OOD) recognition; safety-critical applications; generator as distribution approximator; robustness of computer vision systems

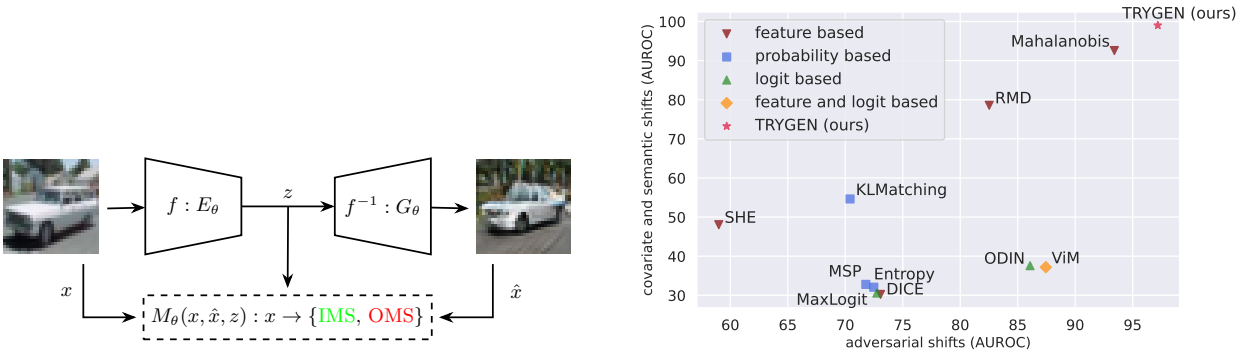
1. Introduction

One of the key factors contributing to the success of deep learning in computer vision is its ability to learn intricate patterns directly from the training data, eliminating the need for manual feature engineering. However, deep learning models are susceptible to overfitting [1], and their performance may degrade when faced with data that lie outside of the training set's distribution [2].

A neural network trained on images of handwritten digits illustrates this issue; while some implementations [3] achieve near-perfect recognition rates on test data, in real-world scenarios [4] where the inputs are unbounded or uncontrolled [5], there is no guarantee that the images will belong to the same distribution as the training data. In this case, how would a network react to an image of a human? Clearly, it would classify it as one of the ten digits. A more tangible example would be autonomous driving, specifically when a car encounters an unknown traffic sign or has to operate in a previously unseen environment. In these cases, we need to develop a monitoring function that recognizes Out-of-Model Scope (OMS) inputs and reacts accordingly.

In this work, we present a novel perspective on the concept of Out-of-Model Scope [6] inspired by the validation processes described in the Safety of the Intended Functionality (SOTIF) norm [7] and in ISO 26262 [8]. We shed light on critical considerations for enhancing

the robustness and reliability of Machine Learning (ML) systems deployed in the computer vision domain. To the best of our knowledge, there is no norm prescribing a specific method for detecting OMS samples. Therefore, the implementation of State-Of-The-Art (SOTA) monitors varies, consisting of hard-coded boundary binary classifiers that use features extracted from different layers [9], input images, or a combination of both [10]. Our approach aims to incorporate diverse information during the prediction phase, as can be seen in Figure 1. We refer to the monitored model as the Encoder and the approximated inverse version of its In-Model Scope (IMS) distribution as the Generator. OMS detection based on the input images, generated images, and latent feature representations is solved by a learnable Monitor. Throughout this article, we use the terms "OMS detection" and "OMS monitoring" interchangeably.



(**a**) (**b**)

Figure 1. (**a**) illustrates our OMS monitoring pipeline, named TRYGEN; the process begins with the input image x, followed by the generation of a synthetic image $\hat{x}$ and extraction of the latent feature representation z from an arbitrary encoder. These three sources of information are integrated by the monitor to classify the sample as either In-Model Scope (IMS) or Out-of-Model Scope (OMS). In (**b**), we plot the AUROC (in percentage) of SOTA Out-Of-Domain (OOD) monitoring algorithms tested on our OMS dataset collections, comparing the covariate and semantic shifts versus adversarial attacks.

Our contributions to the domain of OMS detection are as follows:

- We describe the theoretical background of OMS detection, incorporating a generator of synthetic images.
- We devise a novel multi-input OMS detection pipeline that facilitates the monitoring of any pretrained classifier.
- We demonstrate the robustness of our new method against distributional shifts and adversarial attacks.
- Through a sensitivity analysis, we evaluate the impact of various inputs on the OMS monitor.
- We show that our pipeline achieves SOTA results in the computer vision domain.

2. Related Work

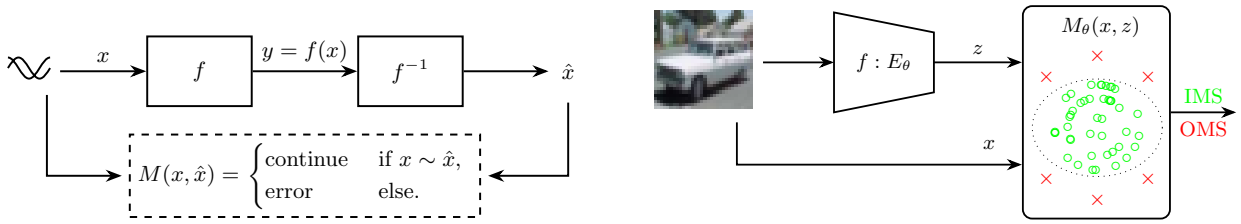
For practical reasons, the objective of our contribution is aligned with industry norms. In the following sections, we review related works to position our contribution in the broader research landscape.

2.1. Functional Safety Versus Computer Vision

As mentioned in the introduction, the uncertainty of machine learning models incurs potential risk in real-world applications. Therefore, we consider the definitions and processes described in EN IEC 61508 [11], which defines a general clustering of safety components in the Safety–Integrity Level (SIL). Additionally, the functional safety norm ISO 26262 defines guidelines that minimize the occurrence and severity of hazardous situations related to components' malfunction or failure. Furthermore, the ISO PAS 21448,

also referred to as SOTIF, builds upon ISO 26262 and extends the development process by addressing potential risks in situations where a system could operate incorrectly or unsafely.

The detection of OMS falls within the scope of SOTIF. Nevertheless, the OMS detection pipeline presented in this paper is derived from the concept of hardware (HW) malfunction detection described in ISO 26262, namely, that in the event of HW failure the objective is to transition the system to a safe state and enable the operator to maintain control. To achieve this, the safety-critical function and its inverse implementation are monitored by a plausibility function, as highlighted in Figure 2. The implementation of such software (SW) measures reduces the risk of potential undetected malfunctions to a predefined and acceptable level.



(**a**) (**b**)

Figure 2. (**a**) The standard approach of plausibility function in contemporary SW and (**b**) how the monitor M, equivalent to a plausibility function, is deployed in the computer vision field.

In Computer Vision (CV), and particularly in deep learning, lossy information compression (pooling Layers and activation functions) prevents the direct implementation of an inverse function. Additionally, in real-world conditions, the monitoring function must contend with the uncertainty of the monitored model, which is caused either by the model's limited generalization capabilities [12] or by missing features in the training data [2]. These factors have led to the design of monitoring functions [13] (Monitor) based on hard boundaries. The Monitor's role is to detect the domain in which the model operates safely. Although the Monitor can predict whether the input belongs to the known data distribution, uncertainty in OMS detection remains [14].

2.2. Out-of-Distribution and Out-of-Model Scope

The terms "Out-of-Distribution" and "Out-of-Model Scope" are often used interchangeably, but their definitions vary [15]. In this work, we adopt the definitions and guidelines provided by [6].

2.2.1. Out-of-Distribution

Let us consider a machine learning task on a domain $\mathcal{X}$, such as classification or detection, defined by an oracle function Ω. The oracle function Ω maps points $x \in \mathcal{X}$ from the task domain to their corresponding ground truths $\Omega(x) = y \in \mathcal{Y}$. Using the oracle function Ω, we can define the OOD scope and OMS of any domain. We denote the operational domain as data points $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$. Considering the predefined OOD settings, we assume that the data follow a probability distribution p_{ID}, where

$$\forall i = 1, \ldots, n; x \sim p_{\text{ID}}. \tag{1}$$

Given a new input instance $x \in \mathcal{X}$, our goal is to determine whether it originates from the same distribution p_{ID} or resides outside of this distribution. This leads to the formulation of an out-of-distribution monitor $M : \mathcal{X} \to \{0, 1\}$, provided by

$$\forall x \in \mathcal{X},\ M(x) = \begin{cases} 0 & \text{if } x \sim p_{\text{ID}}, \\ 1 & \text{else.} \end{cases} \tag{2}$$

2.2.2. Out-of-Model Scope

Referring to the nomenclature established in Section 2.2.1, we define the model scope S_f as follows:

$$S_f = \{x \in \mathcal{X} : f(x) = \Omega(x)\} \tag{3}$$

where f denotes the deployed model.

We aim to develop an OMS monitor $M_f : \mathcal{X} \to \{0, 1\}$ capable of identifying all data points that fall outside the model scope S_f. This monitor function is defined as follows:

$$\forall x \in \mathcal{X},\ M_f(x) = \begin{cases} 0 & \text{if } x \in S_f, \\ 1 & \text{else.} \end{cases} \tag{4}$$

As can be seen in Equation (2), OOD detection focuses on distinguishing among an infinite number of probability distributions within the domain $\mathcal{X}$ and a specific In Distribution p_{ID}. However, capturing all boundaries within a CV domain is only possible if the condition $|\mathcal{X}| < +\infty$ is fulfilled [14]. Consequently, proposed solutions for OOD detection in a stochastic environment are vulnerable to adversarial attacks and shifts within the ID, as demonstrated in [16]. Furthermore, the definition of OOD can be significantly influenced by the domain and the definition of the machine learning task. These ambiguities have led to the implementation of various monitoring functions and the attainment of diverse results, as shown in [17].

In contrast to OOD, the definition of OMS provides a more precise and consistent framework for detecting samples that the model should not classify or make predictions on because they are not drawn from p_{ID}. The conditions for learnable OMS remain the same as for OOD; nonetheless, its clearer definition eliminates the need to differentiate between outlier detection [18,19], anomaly detection [20,21], novelty detection [22], and the open-set recognition problem [23,24], as is the case with OOD. Due to these findings, in this paper we focus only on OMS detection instead of OOD detection. Based on the definition of an OMS Monitor in Equation (4), our OMS definition encompasses covariate and semantic shifts as well as adversarial attacks. These distribution shifts can be introduced into the training data through a variety of methods, including:

- Covariate Shifts: Altering certain image aspects, such as brightness, to a degree that causes the classifier to fail.
- Semantic Shifts: Introducing semantic content that has not been previously encountered in our domain $\mathcal{X}$.
- Adversarial Attacks: Incorporating malicious perturbations in the input data that are imperceptible to human eyes.

2.3. Out-of-Model Scope Metrics and Monitors

Although the monitors and metrics discussed in this chapter were originally developed for detecting OOD samples, they can also be utilized for OMS detection. As is commonly done, we cluster OOD detection methods based on the inputs used. Therefore, Figure 1 highlights the following possibilities:

- Feature-based
- Probability-based
- Logit-based
- Feature- and logit-based.

Our method belongs to the group of novel generative approaches utilizing both features and logits.

2.3.1. Metrics

Among the possible feature-based methods, the distance between any point $\hat{x} = (x_1, x_2, x_3, \ldots, x_N)$ within the operational domain $\mathcal{X}$ and the ID distribution p_{ID}, called the Mahalanobis distance (MAHA), is the most frequently used. Introduced by

P.C. Mahalanobis [25], the Mahalanobis distance was further employed by Lee et al. [16] through a Gaussian discriminant analysis applied to the softmax layer. Lee et al. discovered that abnormal samples are better represented in the feature space of Deep Neural Networks (DNNs) rather than in the "label-overfitted" output space of the softmax-based posterior distribution, as demonstrated in earlier studies [26,27].

The Max Softmax Probability (MSP) method, pioneered by Hendrycks and Gimpel [13], establishes OMS detection based on the hypothesis that correctly classified examples should exhibit higher probability compared to other classes. This involves determining a statistical threshold for a binary classifier using a validation set. On top of this work, Liang et al. built the ODIN (Out-of-Distribution detector for Neural Networks) system [28], which uses temperature scaling to maximize the discriminability of the softmax outputs for In- and Out-Of-Distribution images. Hendrycks et al. further focused on anomaly detection in large-scale datasets, for which they proposed using the negative of the maximum of the un-normalized logits for an anomaly score, which they called MaxLogit [29].

As pointed out by Liu et al. [24], however, relying solely on softmax probabilities leads to high confidence for misclassified samples. To address this, they incorporated an energy-based score that is theoretically aligned with the probability density of the inputs and less susceptible to overconfidence.

2.3.2. Monitors

The Outside-the-Box method introduced by Henziger et al. [23] addresses novelty detection by employing a monitor in addition to the arbitrary hidden layers of a pretrained classifier. During training, the method samples class-related responses and clusters them, thereby improving monitor accuracy. Novelties are recognized using a set of box abstractions that determine whether a new data point is inside or outside of the projected box.

Motivated by activation patterns analysis, Sun et al. [30] noted that activations of OMS samples differ significantly from those of In-Model Scope samples. Their Rectified Activation (ReAct) method further showed that gathering activations from layers other than the penultimate one does not enhance accuracy, as early layers capture less distinctive features.

This latter statement asserted by [30] was negated by Lin et al. [9], who evaluated the possibility of detecting an OOD sample from different intermediate layers. They proposed a new energy-based score to dynamically terminate an early exit during inference. They picked the early exit based on the number of bits needed to encode the compressed image, consequently proving on many datasets that less complex OMS inputs can also be reliably discovered from the early layers.

2.3.3. Other Evaluation Metrics

Class imbalances and other nonuniformities in the number of False Positives (FP) and False Negatives (FN) must be considered during the evaluation of any classifier. To address this, the Area Under the Receiver Operating Characteristic (AUROC) [31] is frequently used in conjunction with the F-score. AUROC evaluates the True Positive Rate (TPR) against the False Positive Rate (FPR), where the TPR is calculated as $TP/(TP+TN)$ and the FPR is analogously calculated as $FP/(FP+TN)$, where TN stands for True Negative samples. Nonetheless, as mentioned in [13], AUROC may not be suitable when the positive and negative classes have notably different base rates. In such cases, the Area Under the Precision–Recall curve (AUPR) is preferred, as it adjusts for the varying positive and negative base rates. Another metric used to combat class imbalance in data is the balanced accuracy $ACC_B = (TPR + TNR)/2$, where TPR is the true positive rate, defined as before, and the True Negative Rate (TNR) is provided by $TN/(TN+FP)$. From now on, we refer to accuracy in the OMS domain as balanced accuracy.

3. Methodology

Our TRYGEN methodology is based on a novel pipeline that enhances the robustness of OMS detection by providing multiple inputs to a learnable OMS monitor.

All components of the OMS monitoring pipeline are depicted in Figure 3. The weights θ_E of the Encoder E represent the feature space extracted from the operational domain $\mathcal{X}$, which in return represents the IMS distribution p_{S_f}. The Generator G produces reconstructed synthesized images $\hat{x}$ based on a combination of latent features z. Ideally, if the input sample x belongs to the model scope S_f, then G generates an image from the same distribution; on the other hand, when an OMS input is presented, the Generator combines different features based on the Encoder's latent feature representation z in the generated image. The Latent Space Wrapper L extracts this representation z, which can contain information from different layers of the Encoder and its softmax prediction $\hat{x}$. In the last step, the OMS binary classification is addressed by a multi-input OMS Monitor M_θ. The Monitor receives three inputs: the original input image x, the reconstructed image $\hat{x}$, and the latent feature representation z. Conceptually, our OMS monitor can be seen as a mapping $OMS(\theta_M, \theta_G, \theta_E, \theta_L) : \mathcal{X} \rightarrow \{0, 1\}$. Algorithm 1 presents the description of the inference process.

Algorithm 1 TRYGEN OMS detection pipeline during inference

Require: $|\mathcal{X}| < +\infty$
  for $x \in \mathcal{X}$ **do**
    $\hat{y}, z = E_\theta(x)$ ▷ $\hat{\mathbf{y}} \in \mathbb{N}^C$, $\mathbf{z} \in \mathbb{R}^D$, $\mathbf{x} \in \mathbb{R}^{3\times M\times N}$
    $\hat{x} = G_\theta(\hat{y}, z)$ ▷ $\hat{\mathbf{x}} \in \mathbb{R}^{3\times M\times N}$
    $\hat{o} = M_\theta(x, z, \hat{x})$ ▷ $\hat{\mathbf{o}} \in \{0, 1\}$
  end for

Preconditions for OMS Problem-Solving

Based on the theoretical and practical expertise we gathered during experimentation, we identified the following properties that the OMS pipeline needs to fulfill:

- The input space $\mathcal{X}$ cannot be infinite, as described in [14].
- The influence of the regularization of the Encoder's feature space on the Generator's performance should be investigated; for this, we use Dropout [1] as well as latent feature space normalization [32], which calculates a class-related cosine distance and maximizes it during the training of the Encoder.
- The Generator needs to have high reconstruction capability for samples within the Encoder's scope S_f, but needs to lose this ability for samples outside of the scope S_f; to achieve this, during the generator's training, we minimize the classification loss between the Encoder's prediction on the original and the generated images from the model scope.
- In order to compare the contribution of our multi-input OMS pipeline, we investigate the performance fluctuation of the Monitor through a sensitivity analysis.
- To increase robustness, we use Dropout directly on the inputs provided to the OMS monitor, namely, x, $\hat{x}$, and z.

In general, the Encoder can be any deep neural network that has been trained on the operational domain $\mathcal{X}$; its implementation must allow for access to intermediate layers. The Generator has to follow an inverse information flow with respect to the Encoder; in our case, the generator produces synthetic images $\hat{x}$ with the same resolution as the input image x, allowing us to directly evaluate the image fidelity with the Fréchet Inception Distance (FID) [33] and the Inception Score (IS) [34]. There are no further constraints on the implementation of the Generator, and its architecture can be similar to that of a Generative Adversarial Network (GAN) [35] or Latent Diffusion model [36]. What distinguishes our approach from its predecessors is that the OMS monitor contains learnable parameters and receives synthetic images generated from the Encoder's approximated density function, referred to as the Generator.

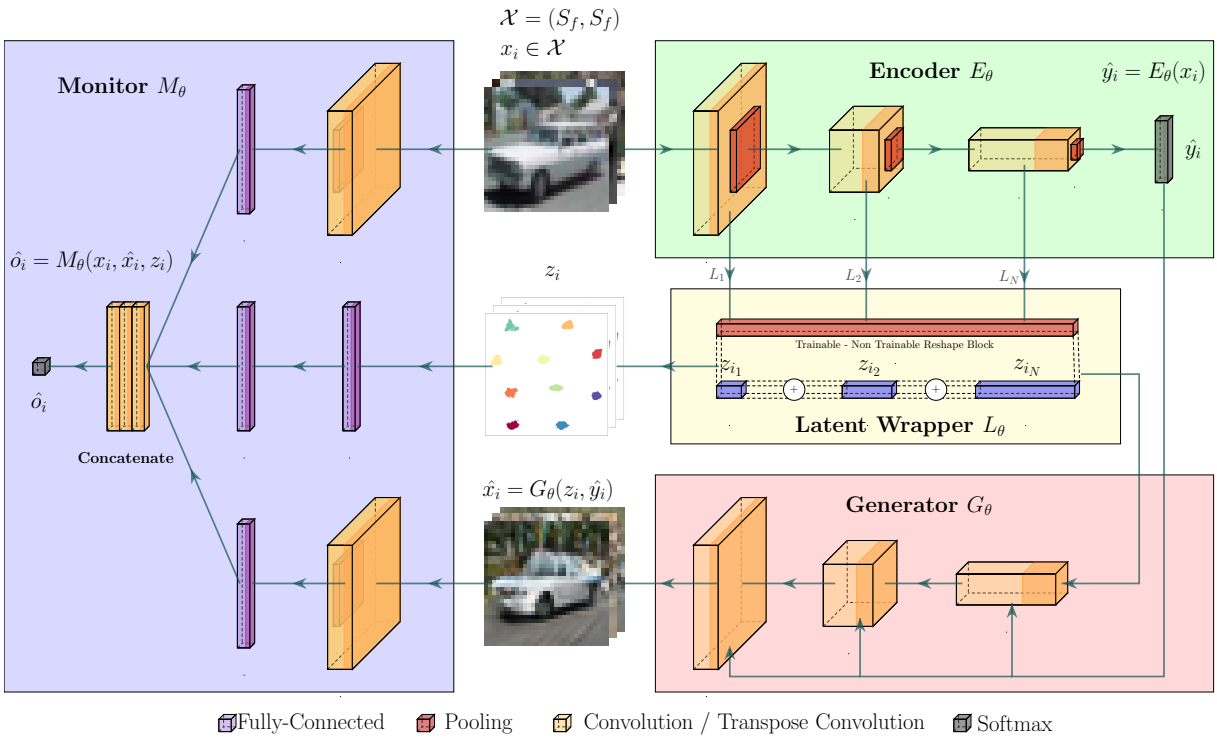


Figure 3. Our proposed TRYGEN pipeline for OMS detection contains the following building blocks: the Encoder E, Generator G, Latent Space Wrapper L, and OMS monitor O. Each block in this diagram is simplified, and can be replaced with more complex or deeper models in order to adapt to a specific domain.

4. Experiments

We designed our evaluation setup based on the methodologies presented in [6,9]. However, instead of using pretrained networks, we explored the influence on the OMS performance by training custom ResNet models with different numbers of parameters and layers.

4.1. In Model Scope and Out-of-Model Scope Datasets

We chose CIFAR-10 [37] as our initial dataset, following common practice in the OOD and OMS community. We split all datasets uniformly: 70% were used for the training set, 20% for validation, and the remaining 10% for testing purposes.

The OMS dataset consists of semantic shifts, covariate shifts, adversarial attacks, and FP samples. We built the set of semantic shifts using publicly available datasets, namely, MNIST [38], fashion-MNIST [39], k-MNIST [40], SVHN [41], and DTD [42]. To test our pipeline's robustness against covariate shifts in the data, we introduced various modifications to the original CIFAR-10 dataset. These modifications were applied until the Encoder failed to recognize otherwise correctly classified IMS samples. The applied modifications included blurring, brightness adjustments, image rotations, and noise addition. Lastly, we introduced imperceptible malicious perturbations to the IMS data, also known as adversarial attacks [43]. These perturbations are designed to be undetectable by human observers. Our approach involved leveraging the Fast Gradient Sign Method (FGSM) [44], Projected Gradient Method (PGD) [45], DeepFool [46], and AutoAttack [47]. The implementation in the Torchattacks [48] library was used in our case. FP samples, which were identified after the training of the Encoder was completed, formed the default OMS dataset. Consequently, the identified TP samples formed the IMS (S_f) dataset. A selection of random samples and transformations from all datasets can be seen in Figure 4. Furthermore, a summary of all the methods used to construct the OMS datasets is highlighted in Table 1.

Table 1. Summary of all the methods employed in our experiments to construct the OMS datasets.

IMS	OMS		
	Semantic	**Covariate**	**Adversarial**
CIFAR-10			
	MNIST	blurring	FGSM
	fashionMNIST	brightness	PGD
	kMNIST	rotation	DeepFool
	DTD	noise	AutoAttack
	SVHN		

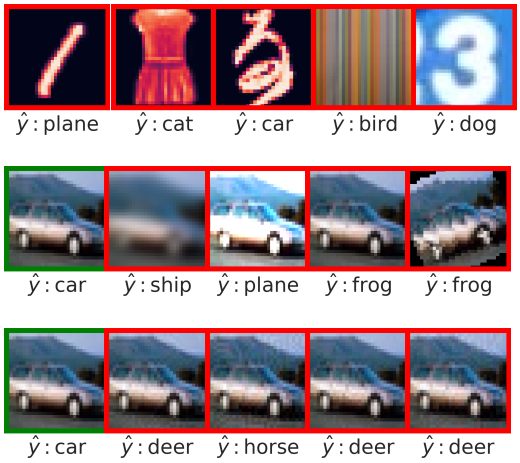


Figure 4. The first row shows examples of the semantic shifts from Table 1, introduced in the same order as described in Table 1. Additionally, the Encoder's predictions for each image are shown. The second row presents examples of covariate shifts performed on the original image in the first column of the corresponding row. The final row shows the adversarial attacks applied to the original images.

4.2. Training

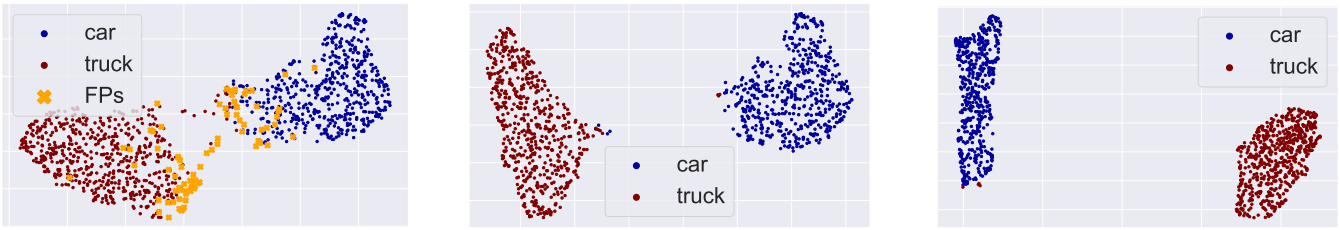
Our OMS monitoring pipeline consists of four main components: the Encoder E, Generator G, OMS monitor O, and Latent Space Wrapper L. Each component is trained independently, which ensures a plug-and-train monitoring system that can be applied to any pretrained encoder (e.g., a ResNet).

4.2.1. Encoder

To establish the foundation for the IMS dataset, we commenced by training the Encoder, which defines the IMS dataset as per Equation (3). We employed ResNet models [49] with 18, 34, 50, and 101 layers, leveraging the CIFAR-10 dataset divided by the guidelines outlined in Section 4.1. Each model underwent training with the Adam optimizer [50], for a total of 500 epochs, employing a learning rate of 0.001 and following a cyclic learning rate scheduler, all while utilizing a batch size of 1024. The Categorical Cross Entropy was used as loss function. Additionally, we applied various data augmentation techniques available in PyTorch [51], including random horizontal flipping, random cropping, and random affine transformations.

Many OMS detection methods assign a score to an input sample based on its representation in the feature space. Given the infinite variations in OMS data, we have to rely on the IMS feature space generated by our Encoder. Should this feature space representation of our IMS data lack relevance, it may falsely contribute to the detection of OMS examples. As a result, we applied regularization techniques related to the latent feature space. This involved implementing a dropout mechanism with a probability of 0.2 for zeroing out and incorporating a Spatial Feature (SF) regularizer. The SF regularizer calculates both inner and outer cosine class similarities. During the training, it maximizes the inner similarity

while minimizing the outer similarity. For a visual comparison of the feature spaces before and after regularization, we refer to Figure 5, which showcases the Uniform Manifold Approximation and Projection (UMAP) [52] results.



(**a**) (**b**) (**c**)

Figure 5. (**a**) Clustered features from the test set, which contains numerous FPs (orange dots). The presence of FPs causes a sparse and enlarged feature space. The compact feature space in (**b**) was achieved by eliminating the FPs. Nonetheless, some samples remain close to or within the other class cluster, even though the Encoder has classified them correctly. In (**c**), the compactness of the inner class subtly changes after applying the Feature Regularizer, making the higher distance between the clusters for the "car" and "truck" classes even more visible.

4.2.2. Generator

We begin with a theoretical description of the Generator's role, using the vanilla GAN architecture as an example. The GAN architecture is comprised of two main components: the Generator G, and the Discriminator D. The Discriminator is trained to differentiate between an original image x and a synthetic image created by the Generator $G(z)$. This is reflected in its loss function, which is $\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$. Concurrently, the Generator aims to produce images that fool the Discriminator, optimizing its loss function $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$. The Generator uses a latent vector z as input, typically sampled from a normal distribution with mean = 0.0 and standard deviation = 1.0. During training, the vanilla GAN model attempts to minimize the KL divergence [53] between the IMS data distribution $p_{S_f}(x)$ and the implicit data distribution $p_G(z)$ of the Generator. This often leads to learning instability and difficulty in capturing structural and geometric features [54], spurring the development of enhanced GAN architectures such as Convolutional GAN [55], Self-Attention GAN [56], BigGAN [57], and Style GAN [58] along with improved training setups such as Wasserstein GAN with gradient penalty [54] and Spectral Normalization [59].

In our pipeline, the Generator aims to densely approximate the IMS distribution $p_{S_f}(x)$. Its task is to reconstruct the input x given its latent feature space representation z and the predicted class $\hat{y} = E(x)$ of the Encoder E. More specifically, the Generator learns $p_{S_f}(x|\hat{y}, z)$. It should be noted that as we can use y in place of $\hat{y}$ during training with the IMS dataset, we can use $p_{S_f}(x|y, z)$ here. Our goal is to minimize the distance between $p_{S_f}(x)$ and $p_{S_f}(x|\hat{y}, z)$; however, instead of using random noise, we utilize the latent feature space from the Encoder E. Consequently, we minimize the classification loss $\mathcal{L}_{\text{cls}}(G, E) = CCE(y, E(G(z))) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$, where $CCE(y, E(G(z)))$ is the Categorical Cross-Entropy (CCE) loss between ground truth label y and the classified generated image $E(\hat{x}) = E(G(z))$. In our case, the representation of z can correspond to the latent feature space of one or more layers. The complete training loss function for the Generator G is detailed in Equation (5):

$$\begin{aligned}\mathcal{L}_{\text{Generator}}(G, D, E) = &\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(D(x))] \\ &+ \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \\ &- \alpha\mathbb{E}_{y\sim P(y)}\left[\sum_{c=1}^{C} y_c \log(\hat{y}_c)\right]\end{aligned} \tag{5}$$

where $\mathbb{E}_{y\sim P(y)}$ denotes the expected value over the probability distribution of the true labels y, C is the number of classes, y_c is a ground truth indicator, $\hat{y}_c$ is the predicted probability, and α is a scaling factor for training stability that is empirically set to 0.1.

We applied the aforementioned training approach to all three models: WGAN [54], BigGAN [57], and the Class-Conditional Latent Diffusion Model [36] (CCLDM). To summarize, the following hyperparameters were explored in terms of image fidelity and contribution to OMS detection:

- The generator's architecture and size,
- The dimension of the latent feature space z,
- The method used for class/feature embedding,
- The combination of intermediate layers used to build the latent feature space z,
- The implementation of the Latent Space Wrapper L,
- Integration of a CCE between the ground truth label y and the classified generated image.

For the first two GAN-based models (WGAN and BigGAN), we used the training loss described in Equation (5) and followed the hyperparameter settings from the original papers. In the case of the CCLDM model, the class embedding information was directly concatenated with the denoising parameters and the original image. During training, the original image with added noise was reconstructed and the Mean Squared Error between the original and denoised image was minimized. The main component of the diffusion architecture was the denoising U-Net [60] model.

By incorporating the latent features z extracted by the encoder as class-conditioning information, we encourage our generator to synthesize images of the same class with similar features learned during the Encoder's training. This hypothesis is guided by the expectation that even in the event of an adversarial attack, where the sample falls outside of the model's scope, the Generator will still reconstruct an image containing features from the IMS distribution. To achieve this, we explored the hyperparameter space and evaluated which combinations were the most effective for OMS detection.

4.2.3. Latent Space Wrapper

As previously mentioned, we use information from the Encoder's latent feature space. As this feature space can stem from any arbitrary deep neural network, we have to devise a mechanism that will aggregate the information from different feature maps of the model and produce a single latent representation z. We deploy both a non-trainable approach by applying Average and Max-Pooling layers directly to the intermediate layers, and a trainable approach with convolutional layers, for which we use a stride with the same size as the kernel. Our Latent Space Wrapper adapts to the size of each given layer and its parameters are tuned during the training of the Generator, as shown in Figure A1 in the Appendix A.

4.2.4. Out-of-Model Scope Monitor

In the final phase, we train the OMS Monitor. The primary task of the OMS Monitor is to perform binary classification based on the similarities between its three inputs. Specifically, the Monitor predicts whether or not the input from the Encoder is OMS.

During the training of the OMS Monitor, we constructed the OMS datasets using the methods and combinations described in Section 4.2. It is worth mentioning that if we solely consider FPs as examples of OMS, then the size of the OMS dataset is contingent upon the performance of the Encoder. In scenarios where the Encoder achieves 100% accuracy, the OMS dataset would consist of an empty set of images. This underscores

the dependency between the size of the OMS dataset and the performance of the Encoder, justifying the need to integrate supplementary OMS data. To address potential issues with dataset imbalance during training, we employed the Focal Loss [61] and weighted random sampling techniques.

Notably, we did not apply any data augmentation to the IMS dataset. This precaution was taken to prevent samples from inadvertently being categorized as OMS. Building on the training setup of the Encoder, we applied dropout before each linear layer and used Binary Cross-Entropy as the loss function.

4.3. Evaluation of the Encoder and Generator

For the Encoder, we trained different ResNet models with 18, 34, 50, and 101 layers. All models consisted of four blocks which served as binding points to our Latent Space Wrapper. The accuracy results of the Encoders on each dataset are presented in Table 2.

Table 2. Accuracy of Encoder-ResNet on different dataset splits; "Dropout" describes a model with a dropout2d layer after each ReLU activation. The best accuracy was achieved by ResNet-34.

ResNet	train.	val.	Test	Dropout	FR
18	98.94	88.64	87.01	88.69	**74.06**
34	**99.48**	**89.88**	**88.18**	**88.99**	73.51
50	98.89	89.19	87.73	87.76	72.12
101	98.35	89.10	86.71	77.72	70.09

Bold: the best results.

As elaborated in detail in Section 4.2, the generator was trained on the IMS dataset. The highest classification accuracy on the generated images ($ACC_{E(G(z))}$) was achieved by the bigGAN model with a latent feature space dimension of 64, combining the features extracted from ResNet34 blocks 1 and output from the softmax layer. This model reached 91.80% for the training set and 91.96% for the validation set. Image fidelity and image variance were evaluated using the Frechet Inception Distance (FID) [33] and Inception Score (IS) [34], which correlate well with human judgment. The same model achieved an FID of 28.04% and IS of 7.92%. To increase the evaluation set, all samples from the IMS dataset were included in the assessment, as recommended in the original FID paper.

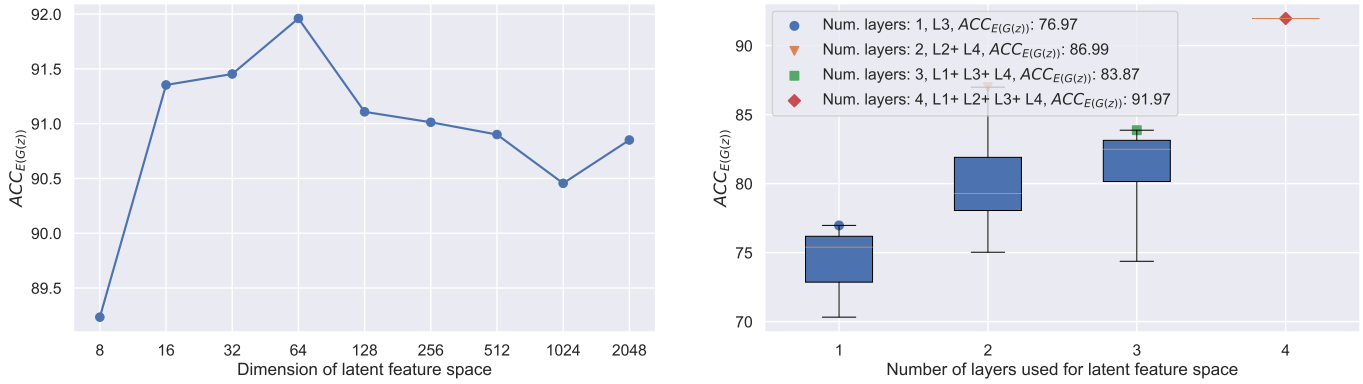
The best model for each Generator architecture can be found in Table 3. Because we conditioned the training loss from Equation (5) using $ACC_{E(G(z))}$, our model finds the optimum between image fidelity and the accuracy of the Encoder. Consequently, our generator achieves lower IS and higher FID scores than reported in the original papers.

Table 3. FID, IS, and $ACC_{E(G(z))}$ achieved only by the **best** Generators with ResNet-34. The best model was the bigGAN model, which we further used as a Generator in the OMS monitoring pipeline.

Model G	↑ IS	↓ FID	↑ $ACC_{E(G(z))}$	$\|z\|$	Layers Combination	E-Regularization	Learnable *L*
W-GAN	5.71	38.21	70.50	64	L1 + L2 + L3 + L4	with Dropout	False-Avg Pool
bigGAN	7.92	28.04	**91.96**	64	L1 + L2 + L3 + L4	with Dropout	False-Max Pool
CCLDM	8.01	19.87	89.89	128	L1 + L2 + L3 + L4	with Dropout	False-Max Pool

Bold: the best results.

As mentioned at the beginning of Section 4.2.2, several hyperparameters influence the Generator's performance, and consequently that of the Monitor as well. The influence of the dimension of the latent feature z on $ACC_{E(G(z))}$ is highlighted in Figure 6a. A sensitivity analysis of different combinations of layers further shows that the most relevant features are contained in the deeper layers. As can be seen in Figure 6b, a generator trained on combinations of features extracted from multiple layers outperforms a generator trained only on features from one specific layer.



(**a**) (**b**)

Figure 6. (**a**) Variations of latent feature space dimensions for the bigGAN generator; the best $ACC_{E(G(z))}$ was achieved with a dimension of the latent feature space equal to 64. (**b**) Features from layers L1 + L2 + L3 + L4 and the Latent Space Wrapper with Max-Pooling layers assured the highest $ACC_{E(G(z))}$ in the case of a combination search for the optimal layers.

Although the highest classification accuracy on CIFAR-10 was achieved by ResNet-34, as highlighted in Table 4, for the OMS detection task the combination of the Generator and ResNet-18 showed similar results. The deployment of more complex encoders with higher numbers of layers did not improve the performance of the Generator. On the contrary, as can be seen in Table 5, applying dropout during training of the Encoder improved the Generator's performance. The final overview, presented in Table A2 in the Appendix A, summarizes all of the hyperparameters and techniques related to the Generator's training.

Table 4. $ACC_{E(G(z))}$ on the IMS test set for all Encoders and all Generator architectures. The bigGAN architecture in combination with ResNet-34 produced the highest accuracy; the WGAN and CCLDM models achieved unsatisfactory results, and were not used for further experiments.

Model E	WGAN	bigGAN	CCLDM
18	69.14	90.15	79.60
34	70.50	**91.96**	78.69
50	65.12	85.25	73.50
101	63.87	86.74	68.28

Bold: the best results.

Table 5. Influence of different regularization techniques applied to the ResNet-34 Encoder on the performance of the bigGAN Generator ($z_{dim} = 64$, LatentWrapper with maxPool).

Regularization	↑ IS	↓ FID	↑ $ACC_{E(G(z))}$
no regularization	6.57	30.01	90.61
dropout	7.92	28.04	**91.96**
feature regularization	5.84	35.85	86.38

Bold: the best results.

4.4. Evaluation of Out-of-Model Scope Detection

In the case of the OMS Monitor, we are dealing with a binary classification problem involving the identification of two classes. However, it is common for the volume of OMS data to be larger than the IMS data in real-world applications as well as in experimental setups [62]. As noted in Section 2.3, we employ metrics that consider a tradeoff between TPs and FPs to address this issue, and calculate the balanced accuracy instead of the vanilla accuracy. Achieved results can be seen in Table 6.

Table 6. Results of the Precision, Recall, and F1-Score evaluated on the OMS and IMS test sets. As can be seen in the "Support" column, the size of the OMS dataset is approximately 14× larger than the IMS dataset.

↑ Precision	↑ Recall	↑ F1-Score	Support OMS + IMS
67.42	98.76	80.14	67,014 + 4914

As mentioned in Section 4.2, we trained the OMS Monitor on each IMS and OMS training set presented in Table 1 and validated it on all the IMS and OMS validation and test sets. In our experiments, we explored various combinations of OMS datasets to discern the individual impact of each dataset on the OMS detection instances. The results are presented in Table 7.

Table 7. Our OMS detection pipeline achieved an overall balanced accuracy of **99.52%** on [1] + [2] + [3]. We further evaluated each test set separately, achieving results close to 100%. Notably, the accuracy in recognizing adversarial attacks [2] reached 98.58%.

Training OMS Datasets	Testset Balanced Accuracy in [%]			
	[1] Covariate	[2] Adversarial	[3] Semantic	[1] + [2] + [3] Combined
covariate	**98.95**	93.04	97.99	97.93
adversarial	99.18	**98.58**	97.28	97.96
semantic	97.18	83.50	**99.73**	95.56
combined	99.21	92.25	97.76	**99.52**

Bold: the best results.

4.5. Sensitivity Analysis of the Out-of-Model Scope Monitor

Sensitivity analysis in machine learning involves the systematic evaluation of a model by removing or modifying its components in order to understand their impact on its performance. In the context of our OMS Monitor, the key elements under consideration are the original input x, the latent feature representation z, and the generated image $\hat{x}$, all of which serve as inputs to the OMS monitor.

The results presented in Table 8 show the contributions of each input to overall OMS detection. The accuracy drops by 25.20% after removing the original input image x, proving that the primary source of relevant information for the OMS monitor still lies with the original input data. On the other hand, incorporating the feature space and generator increases the overall classification accuracy by approximately 4%. This subtle improvement can be explained by insufficient approximation of the IMS distribution $p_{S_f}(x)$. Specifically, the best bigGAN model achieved an accuracy of 91.96% on the IMS training set. Images identified as FP by $E(G(z))$ could be considered for removal from the original IMS training set; however, this action would only superficially boost the $ACC_{E(G(z))}$ to 100% while shrinking the training set of valid IMS samples, thereby impairing the Monitor's ability to generalize.

Table 8. Results of the OMS Monitor sensitivity analysis; the results were achieved by setting the corresponding input [1], [2], or [3] to zero.

Deactivated Branch	Delta acc. [%] to No Deactivation
[1] Input image x	−25.20
[2] Feature space z	−4.32
[3] Generated image $\hat{x}$	−3.20
[1] + [2]	−37.36
[2] + [3]	−22.94
[1] + [3]	−32.43
No deactivation	**97.56**

Bold: the best results.

5. Results and Discussion

Our novel OMS Monitoring pipeline represents an advancement in the development of functional safety monitors. It effectively detects samples that were not part of the training setup, preventing actions that could lead to severe consequences. As demonstrated in Section 4.4, our OMS Monitoring pipeline, inspired by the HW-Malfunction detection, achieves a balanced accuracy of 99.52% on OMS samples from previously unseen test sets. Furthermore, it exhibits the highest AUROC of 99.46% among all evaluated monitors and metrics. As was presented in Table 7, our pipeline excels in detecting adversarial attacks designed to deceive the potentially safety-critical Encoder. The overall detection accuracy is achieved by incorporating OMS distribution shift methods during OMS Monitor training. However, it is important to note that detecting false positives, which were excluded from the original dataset, remains the most challenging aspect of OMS detection. This behavior is not unique to our monitoring pipeline and is consistent across SOTA methods, as shown in Table 9. Our method achieves the best results as quantified by the Area Under the Receiver Operating Characteristic and the probability of a negative (OMS) example being misclassified as positive (IMS) when the TPR of the model is set to achieve 95%, that is, FPR95TPR. Our weaker performance in AUPR compared to the current SOTA methods can be explained by the optimization process of the OMS Monitor via Binary Cross-Entropy, which rewards both TP and TN. This dichotomy can result in a slight decrease in precision, as seen in Table 10.

Table 9. Results on the combined IMS+OMS test set incorporating other SOTA methods.

Monitor	↑ AUROC	↑ AUPR	↓ FPR95TPR
MSP [13]	61.28	95.50	100
ODIN [28]	73.32	97.71	100
Mahalanobis [16]	93.74	**99.52**	39.50
KLMatching [63]	66.17	95.81	83.70
MaxLogit [29]	61.44	95.49	100
EnergyBased [24]	61.30	95.42	100
Entropy [64]	61.60	95.55	100
DICE [65]	61.60	95.50	100
RMD [66]	81.49	98.16	63.93
ReAct [30]	62.05	95.53	99.98
ViM [10]	74.22	97.80	99.80
SHE [67]	56.13	95.19	100
TRYGEN (ours)	**99.46**	93.94	**3.42**

Bold: the best results.

Table 10. Confusion matrix, with terms reflecting the OMS binary classification.

	Total Population 71,928	**Predicted cls.** **Positive (IMS)**	**Negative (OMS)**
True cls.	Positive (IMS)	↑ TP: IMS as IMS 4701	↓ FN: IMS as OMS 213
	Negative (OMS)	↓ FP: OMS as IMS 2289	↑ TN: OMS as OMS 64,725

In Section 4.3, we have thoroughly discussed the contribution of the Generator to the OMS Monitoring pipeline. The approximation of the IMS distribution of the Encoder improved mainly due to the incorporation of the classification loss on the reconstructed images and by utilizing the Latent Space Wrapper. Consequently, the OMS training showed improved stability as well as better accuracy. A notable drawback of using the diffusion architecture as the Generator throughout our experiment was the high inference time caused by the iterative diffusion process. Even though the FID and IS scores of the diffusion model were better than those of bigGAN, the bigGAN model achieved the highest

$ACC_{E(G(z))}$, which, as mentioned in Section 4.2.2, is more relevant in the context of the OMS detection task.

6. Conclusions

Our TRYGEN OMS monitoring pipeline allows any pretrained Encoder to be integrated, regardless of the working domain. We have demonstrated the potential for robust OMS sample detection, with our main focus on utilizing the Generator as a universal approximator of the Encoder's latent space. Thanks to our improvements, the proposed OMS detection pipeline was able to achieve high accuracy of 98.58% in detecting various adversarial attacks, and additionally reached the highest AUROC of 99.46% and lowest FPRTPR of 3.42% among all tested methods. We hope that the potential of our novel monitoring pipeline will open doors to further topics and future work, including:

- Adjustment of our OMS detection pipeline to other tasks in computer vision and other fields of machine learning such as natural language processing.
- Research into the possibility of approximating the intermediate encoder layers and using end-to-end training, which we found to be inapplicable in the current research due to our the definition of IMS.

Author Contributions: Conceptualization, V.D.; methodology, V.D., B.S. and M.H.; software, B.S. and V.D.; validation, B.S., V.D. and M.H.; formal analysis, V.D.; investigation, V.D.; resources, V.D.; data curation, B.S. and V.D.; writing—original draft preparation, V.D.; writing—review and editing, V.D., B.S. and M.H.; visualization, V.D. and B.S.; supervision, V.D. and M.H.; project administration, V.D.; funding acquisition, V.D. and M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Technology Agency of the Czech Republic, project No. CL01000275 and project No. CK03000179. This work was also supported by the grant of the University of West Bohemia, project No. SGS-2022-017.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data supporting the reported results can be found at: https://gitlab.com/divisvaclav/oms-detection-framework. Throughout our work, we used publicly available datasets which are correctly cited: MNIST, https://yann.lecun.com/exdb/mnist/; kMNIST, https://github.com/rois-codh/kmnist; DTD, https://www.robots.ox.ac.uk/vgg/data/dtd/; SVHN, http://ufldl.stanford.edu/housenumbers/; FashionMNIST, https://github.com/zalandoresearch/fashion-mnist, accessed on 20 October 2024.

Acknowledgments: Computational resources were provided by the e-INFRA CZ project (ID: 90254) supported by the Ministry of Education, Youth, and Sports of the Czech Republic.

Conflicts of Interest: Author Bastian Spatz was employed by the company ARRK Engineering GmbH. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

OOD	Out-of-Distribution
OMS	Out-of-Model Scope
IMS	In-Model Scope
SOTIF	Safety of the Intended Functionality
SOTA	State-Of-The-Art
SIl	Safety–Integrity Level
HW	Hardware
CV	Computer Vision
MAHA	Mahalanobis distance
DNN	Deep Neural Network
MSP	Max Softmax Probability

ODIN	Out-of-Distribution Detector for Neural Networks
AUROC	Area Under the Receiver Operating Characteristic
AUPR	Area Under the Precision–Recall curve
FP	False Positive
FN	False Negative
TN	True Negative
TNR	True Negative Rate
TPR	True Positive Rate
GAN	Generative Adversarial Networks
FSGM	Fast Gradient Sign Method
PGD	Projected Gradient Method
UMAP	Uniform Manifold Approximation and Projection
CCE	Categorical Cross-Entropy
CCLDM	Class-Conditional Latent Diffusion Model
WGAN	Wasserstein GAN
IS	Inception Score
FID	Frechet Inception Distance

Appendix A

Appendix A.1. Latent Space Wrapper

Figure A1 depicts a non-trainable computational graph. Within the computational process, we scale each layer's input to the dimension of the tensor z with a given feature map of dimensions (C, M, M). The left part of the graph depicts the latent feature space reduction through Average and Max-Pooling layers. The right branch provides an example of applied layers when the number of channels C is smaller than the required feature space dimension of z. In this chart, operations such as upscaling and reshaping, which are necessary for achieving the desired output tensor shape, are not visualized. The trainable Wrapper replaces the Maximum Pooling layers with Convolutional layers, and uses the calculated K as the kernel size and stride.

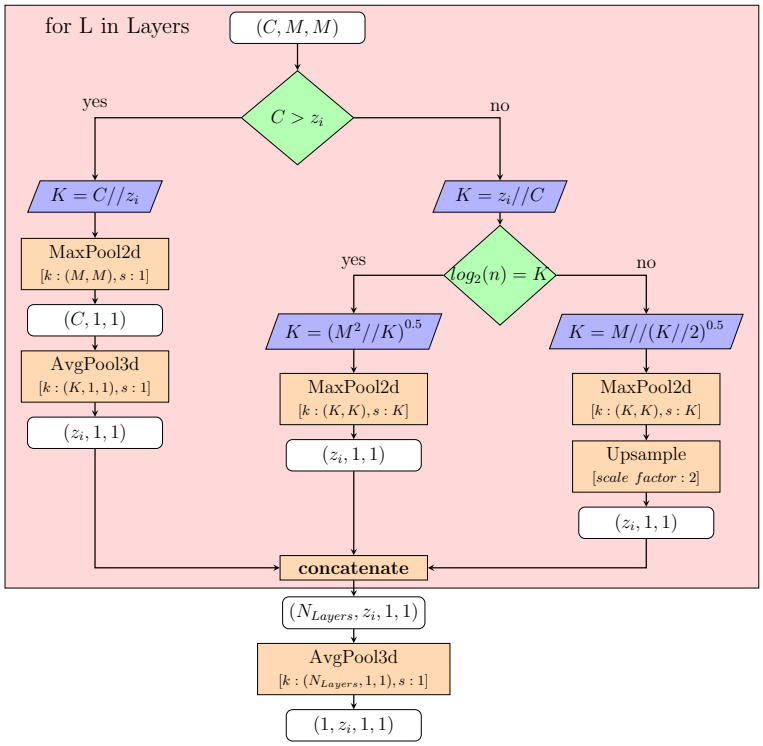


Figure A1. Diagram representing a computational graph of a non-trainable latent space wrapper.

Appendix A.2. Performance of the Generator

During training, the Generator G minimizes the reconstruction loss between the original x_i and the generated image $\hat{x}_i$. This implies that the weights θ_G capture an approximated function inverse to our model E. Consequently, the reconstructed image $\hat{x}_i$ is the outcome of $G_\theta(z_i, \hat{y}_i)$, where z_i generally represents a combination of aggregated feature spaces from various layers. As we want to enhance the robustness of our OMS detection pipeline using the reconstructed image $\hat{x}_i$ from the Generator, it is important to ensure that it has the best possible performance. Therefore, we conducted several additional experiments with learnable and non-learnable latent space wrappers, with the results shown in Table A1. The most robust performance was achieved by extracting the features via Max-Pooling layers. All techniques and their influence are further summarized and listed in Table A2.

Table A1. Variations of the learnable and non-learnable latent space wrapper for the bigGAN generator only, showing the differences in IS, FID, and $ACC_{E(G(z))}$.

$\lvert z\rvert = 64$	↑ IS	↓ FID	↑ $ACC_{E(G(z))}$
avgPool	7.87	29.22	90.78
maxPool	7.92	28.04	**91.96**
learnable	6.89	32.07	78.45

Bold: the best results.

Table A2. Summary of all hyperparameters and techniques along with their influence on training the Generator; FS stands for feature space.

Method/Hyper-Parameter	↑ $ACC_{E(G(z))}$	Reference	Reasoning
Encoders depth	**Small**	Table 4	Deploying more complex ResNet models on the CIFAR-10 dataset did not improve the accuracy of the encoder, nor did it enhance the performance of the decoder.
Encoders FS regularization	**Middle**	Table 5	Training with Dropout and Feature regularization influences FS compactness and consequently improves the Generator's training performance.
Generators Architecture	**High**	Table 3	Similar results were achieved by the bigGAN model and the Stable Diffusion model with latent class embedding. Compared to bigGAN, the performance of the WGAN model was lower by 21%.
FS dimension	**Middle**	Figure 6a	The grid search method settled around the dimension of size 64, with smaller and higher feature space dimensions resulting in lower $ACC_{E(G(z))}$.
FS layers combination	**Middle**	Figure 6b	Combining features from more than one and deeper layers resulted in higher $ACC_{E(G(z))}$.
Trainable Latent Wrapper	**Small**	Table A1	The best results were achieved by fine-tuning the latent wrapper with Max-Pooling layers. The combination with a trainable Conv2d layer did not improve $ACC_{E(G(z))}$.

References

1. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
2. Der Kiureghian, A.; Ditlevsen, O. Aleatory or epistemic? Does it matter? *Struct. Saf.* **2009**, *31*, 105–112. [CrossRef]
3. Byerly, A.; Kalganova, T.; Dear, I. No routing needed between capsules. *Neurocomputing* **2021**, *463*, 545–553. [CrossRef]
4. Rao, Q.; Frtunikj, J. Deep Learning for Self-Driving Cars: Chances and Challenges. In Proceedings of the 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS), Gothenburg, Sweden, 28 May 2018; pp. 35–38.
5. Vaze, S.; Han, K.; Vedaldi, A.; Zisserman, A. Open-Set Recognition: A Good Closed-Set Classifier is All You Need? In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
6. Guérin, J.; Delmas, K.; Ferreira, R.; Guiochet, J. Out-of-distribution detection is not all you need. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 14829–14837.

7. *ISO/PAS 21448:2019*; Road Vehicles-Safety of the Intended Functionality. International Organization for Standardization: Geneva, Switzerland, 2019.
8. *ISO 26262*; Road Vehicles–Functional Safety. International Organization for Standardization: Geneva, Switzerland, 2011.
9. Lin, Z.; Roy, S.D.; Li, Y. Mood: Multi-level out-of-distribution detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 15313–15323.
10. Wang, H.; Li, Z.; Feng, L.; Zhang, W. ViM: Out-of-Distribution with Virtual-Logit Matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 4921–4930.
11. Gall, H. Functional safety IEC 61508 / IEC 61511 the impact to certification and the user. In Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications, Doha, Qatar, 31 March–4 April 2008; pp. 1027–1031.
12. Kendall, A.; Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? *Adv. Neural Inf. Process. Syst.* **2017**, *30*. Available online: https://dl.acm.org/doi/10.5555/3295222.3295309 (accessed on 21 August 2024).
13. Hendrycks, D.; Gimpel, K. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
14. Fang, Z.; Li, Y.; Lu, J.; Dong, J.; Han, B.; Liu, F. Is out-of-distribution detection learnable? *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 37199–37213.
15. Yang, J.; Zhou, K.; Li, Y.; Liu, Z. Generalized Out-of-Distribution Detection: A Survey. *arXiv* **2021**, arXiv:2110.11334. Available online: http://arxiv.org/abs/2110.11334 (accessed on 21 October 2024). [CrossRef]
16. Lee, K.; Lee, K.; Lee, H.; Shin, J. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Brooklyn, NY, USA, 2018; Volume 31.
17. Yang, J.; Wang, P.; Zou, D.; Zhou, Z.; Ding, K.; PENG, W.; Wang, H.; Chen, G.; Li, B.; Sun, Y.; et al. OpenOOD: Benchmarking Generalized Out-of-Distribution Detection. In Proceedings of the Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, Virtual, 28 November 2022.
18. Aggarwal, C.C.; Yu, P.S. Outlier detection for high dimensional data. In Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, USA, 21–24 May 2001; pp. 37–46.
19. Hodge, V.; Austin, J. A survey of outlier detection methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
20. Chalapathy, R.; Chawla, S. Deep Learning for Anomaly Detection: Asurvey. *arXiv* **2019**, arXiv:1901.03407.
21. Pang, G.; Shen, C.; Cao, L.; Hengel, A.V.D. Deep learning for anomaly detection: A review. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [CrossRef]
22. Chen, V.; Yoon, M.K.; Shao, Z. Task-aware novelty detection for visual-based deep learning in autonomous systems. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 11060–11066.
23. Henzinger, T.A.; Lukina, A.; Schilling, C. Outside the Box: Abstraction-Based Monitoring of Neural Networks. In *Proceedings of the ECAI*; Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J., Eds.; Frontiers in Artificial Intelligence and Applications; IOS Press: Amsterdam, The Netherlands, 2020; Volume 325, pp. 2433–2440. [CrossRef]
24. Liu, W.; Wang, X.; Owens, J.; Li, Y. Energy-based out-of-distribution detection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21464–21475.
25. Mahalanobis, P.C. On the generalized distance in statistics. *Sankhyā Indian J. Stat. Ser. A (2008-)* **2018**, *80*, S1–S7.
26. Feinman, R.; Curtin, R.R.; Shintre, S.; Gardner, A.B. Detecting adversarial samples from artifacts. *arXiv* **2017**, arXiv:1703.00410.
27. Lee, K.; Lee, H.; Lee, K.; Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv* **2017**, arXiv:1711.09325.
28. Liang, S.; Li, Y.; Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
29. Hendrycks, D.; Basart, S.; Mazeika, M.; Mostajabi, M.; Steinhardt, J.; Song, D.X. Scaling Out-of-Distribution Detection for Real-World Settings. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022.
30. Sun, Y.; Guo, C.; Li, Y. React: Out-of-distribution detection with rectified activations. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 144–157.
31. Davis, J.; Goadrich, M. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 233–240.
32. Chung, I.; Kim, D.; Kwak, N. Maximizing cosine similarity between spatial features for unsupervised domain adaptation in semantic segmentation. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2022; pp. 1351–1360.
33. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Proceedings of the Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Brooklyn, NY, USA, 2017; Volume 30.
34. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. *arXiv* **2016**, arXiv:1606.03498. Available online: http://arxiv.org/abs/1606.03498 (accessed on 21 October 2024).
35. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]

36. Ho, J.; Jain, A.; Abbeel, P. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Brooklyn, NY, USA, 2020; Volume 33, pp. 6840–6851.
37. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; University of Toronto: Toronto, ON, Canada, 2009.
38. LeCun, Y. The MNIST Database of Handwritten Digits. 1998. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 21 October 2024).
39. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
40. Clanuwat, T.; Bober-Irizar, M.; Kitamoto, A.; Lamb, A.; Yamamoto, K.; Ha, D. Deep learning for classical japanese literature. *arXiv* **2018**, arXiv:1812.01718.
41. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A.Y. Reading Digits in Natural Images with Unsupervised Feature Learning. In Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, Sierra Nevada, Spain, 16–17 December 2011.
42. Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; Vedaldi, A. Describing Textures in the Wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014.
43. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [CrossRef]
44. Goodfellow, I.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
45. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
46. Moosavi-Dezfooli, S.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, 27–30 June 2016; pp. 2574–2582. [CrossRef]
47. Croce, F.; Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In Proceedings of the 37th International Conference on Machine Learning, PMLR, Virtual, 13–18 July 2020; Volume 119, pp. 2206–2216.
48. Kim, H. Torchattacks: A pytorch repository for adversarial attacks. *arXiv* **2020**, arXiv:2010.01950.
49. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
50. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diega, CA, USA, 7–9 May 2015.
51. Paszke, E.A. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NEURIPS*; Curran Associates, Inc.: Brooklyn, NY, USA, 2019; pp. 8024–8035.
52. McInnes, L.; Healy, J.; Saul, N.; Großberger, L. UMAP: Uniform Manifold Approximation and Projection. *J. Open Source Softw.* **2018**, *3*, 861. [CrossRef]
53. Kullback, S.; Leibler, R.A. On Information and Sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86. [CrossRef]
54. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Brooklyn, NY, USA, 2017; Volume 30, pp. 5767–5777.
55. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434.
56. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; pp. 12744–12753.
57. Brock, A.; Donahue, J.; Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
58. Karras, T.; Laine, S.; Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 4217–4228. [CrossRef] [PubMed]
59. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
60. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015; Proceedings, Part III 18; Springer: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
61. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef]
62. Provost, F.J.; Fawcett, T.; Kohavi, R. The Case against Accuracy Estimation for Comparing Induction Algorithms. In Proceedings of the Fifteenth International Conference on Machine Learning, San Francisco, CA, USA, 12–15 June 1998; ICML'98; pp. 445–453.
63. Hendrycks, D.; Basart, S.; Mazeika, M.; Mostajabi, M.; Steinhardt, J.; Song, D. A Benchmark for Anomaly Segmentation. *arXiv* **2019**, arXiv:1911.11132.

64. Chan, R.; Rottmann, M.; Gottschalk, H. Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 5128–5137.
65. Sun, Y.; Li, Y. DICE: Leveraging Sparsification for Out-of-Distribution Detection. In Proceedings of the Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; Proceedings, Part XXIV; Springer: Berlin/Heidelberg, Germany, 2022; pp. 691–708. [CrossRef]
66. Ren, J.; Fort, S.; Liu, J.Z.; Roy, A.G.; Padhy, S.; Lakshminarayanan, B. A Simple Fix to Mahalanobis Distance for Improving Near-OOD Detection. *arXiv* **2021**, arXiv:2106.09022.
67. Zhang, J.; Fu, Q.; Chen, X.; Du, L.; Li, Z.; Wang, G.; Liu, X.; Han, S.; Zhang, D. Out-of-Distribution Detection based on In-Distribution Data Patterns Memorization with Modern Hopfield Energy. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.