# Risk Assessment for Machine Learning Models

**Paul Schwerdtner**[*] Neurocat GmbH ps@neurocat.ai
**Florens Greßner**[*] Neurocat GmbH fg@neurocat.ai
**Nikhil Kapoor**[*] Volkswagen AG nikhil.kapoor@volkswagen.de

**Felix Assion** Neurocat GmbH
**René Sass** Volkswagen AG
**Wiebke Günther** Neurocat GmbH
**Fabian Hüger** Volkswagen AG
**Peter Schlicht** Volkswagen AG

## Abstract

In this paper we propose a framework for assessing the risk associated with deploying a machine learning model in a specified environment. For that we carry over the risk definition from decision theory to machine learning. We develop and implement a method that allows to define deployment scenarios, test the machine learning model under the conditions specified in each scenario, and estimate the damage associated with the output of the machine learning model under test. Using the likelihood of each scenario together with the estimated damage we define *key risk indicators* of a machine learning model.

The definition of scenarios and weighting by their likelihood allows for standardized risk assessment in machine learning throughout multiple domains of application. In particular, in our framework, the robustness of a machine learning model to random input corruptions, distributional shifts caused by a changing environment, and adversarial perturbations can be assessed.

## 1 Introduction

With the deployment of machine learning (ML) models in safety and security critical environments, risk assessment becomes a pressing issue. Failure modes of a given ML model must be identified and the likelihood of occurrence and severity of the damage caused by failure must be assessed. In this work, we focus on failures that result from input perturbations and provide a framework that allows to integrate different sources of input perturbations and to compute general risk scores for a given network and operational environment. These *key risk indicators* (KRIs) can guide the decision on whether it is safe and secure to deploy a given ML model in a specified environment.

For the evaluation of ML risk, we consider *adversarial* input data and *corrupted* input data, which can be used to evaluate ML security and ML safety, respectively. In particular, to qualify as adversarial input data, we assume that a perturbation on the input is specifically crafted to maximize the difference between a ML model's output and the human interpretation of that same input. On the other hand, *corrupted* input data is usually generated ML model agnostic and follows a somewhat *natural* distribution of input data or naturally occurring noise.

In recent years, it has become a well-known fact that neural networks (NN), a subset of ML models, are susceptible to adversarial perturbations Goodfellow, Shlens, and Szegedy (2015) and various algorithms have been proposed to compute such perturbations effectively (known as *adversarial attacks*). It is important to note that due to the transferability of attacks between NN that perform a similar task, an attacker does not need to have access to the attacked NN to successfully craft

[*]Authors contributed equally

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

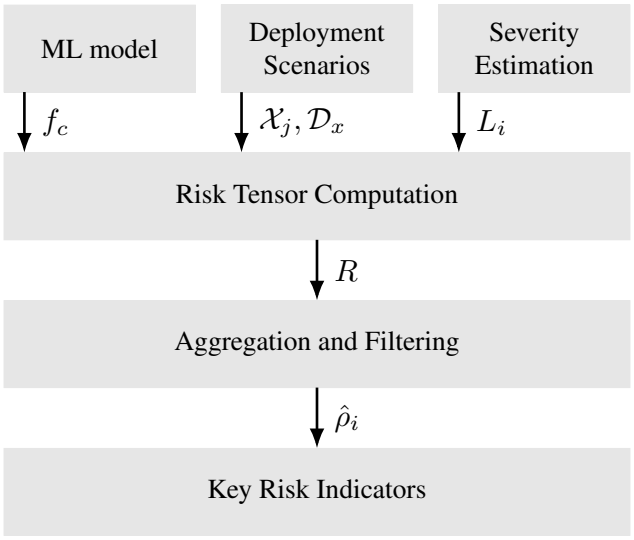



Figure 1: Overview of the proposed key robustness indicator computation method, which is explained in Section 4

adversarial perturbations Liu et al. (2017). Furthermore, adversarial attacks are not merely a fragile phenomenon but can also be planted in the real world to fool NNs Kurakin, Goodfellow, and Bengio (2017).

Alongside adversarial attacks a large number of adversarial defenses that are designed to detect and/or mitigate the effect of adversarial noise have been proposed. However, typically a few months after a defense has been published, an attack that circumvents the detection and mitigation mechanism of that defense is found Carlini and Wagner (2017b); Athalye, Carlini, and Wagner (2018).

This *attack and defense arms race* has led to the introduction of formal verification algorithms of NNs such as the seminal work of Katz et al. (2017). These algorithms are used to verify that around a given set of input points the NN's output does not change for perturbations up to a certain size usually measured in either the $\ell_0$, $\ell_2$, or $\ell_\infty$ norm. However, such formal verification methods do not scale to larger, industry relevant tasks without sacrificing rigor. Furthermore, realistic attack scenarios or image corruptions which are usually not bounded in some $\ell_p$ norm render the formal verification techniques inappropriate in these situations.

In Tian et al. (2018) and Pei et al. (2017) application-oriented robustness evaluation procedures were proposed that explicitly take a more realistic attack and corruption scope into account. As an example, instead of simply limiting the $\ell_2$ norm of a possible perturbation, the adversarial image transformation must be a rotation or change in brightness of the original image. The consideration of realistic image corruptions is key for risk assessment since a highly damaging perturbation that cannot occur in practice or, if at all, with vanishing probability, demands less action than a less problematic but still harmful perturbation that occurs regularly.

Therefore, we propose a framework that lets deployers of ML models define the possible perturbations and their respective magnitude and likelihood to set up realistic test scenarios. Then this scenario dependent robust performance is systematically evaluated by the introduction of KRIs. These indicators allow for comparability of ML models with respect to their robustness in different operational environments. This approach enables well-founded decisions on whether a ML model is fit for application. An overview of our method is given in Figure 1. The input data consists of a ML model that is to be tested, previously designed deployment scenarios, and a severity estimation in the form of a loss function, that computes a damage associated with the ML model's output. From these inputs, we compute a risk tensor that is used as data-storage to be able to extract the required risk indicators by aggregation and filtering.

Next to estimating ML risk, our method can also be used to understand the failure modes of a ML model, or in particular the reason for success of implemented adversarial defenses. Specifically, throughout this paper, we make the following contributions.

- We provide a framework in which the risk associated with deploying ML models in specified environments can be assessed in a standardized way.
- We provide a data-efficient tensor based method for storing robustness information on a given NN that can be queried and filtered to extract KRIs.
- We implement and test our framework on a set of image classifiers for the CIFAR10 dataset Krizhevsky, Hinton, and others (2009) to identify robustifying features of the training process or NN topology.

Our paper is organized as follows. In the next section we describe our setting and compare risk and robustness definitions common in ML to the risk definition from statistical decision theory. After that, we explain how we can apply the latter in the context of ML. For that we present a light-weight data structure that allows for scenario-based risk assessment of a ML model. We illustrate our method in an image classification case study, in which we identify the safest model for classifying images under a set of sensor, weather induced, random, and adversarial perturbations.

## 2 Background

We restrict our presentation of the theoretical background to classification as this allows for a more concise notation. However, it is important to note that our considerations immediately carry over to more complex tasks such as semantic segmentation or object detection.

We denote a classifier as $f_c : \mathbb{R}^{n_x} \to \mathbb{R}^{n_c}$, where $n_x$ and $n_c$ are the length of the input vector (e.g. a vectorized input image) and the number of classes, respectively. Let $\mathcal{X} = (X, \mathcal{F}, \mathbb{P})$ be the probability space of inputs and $\bar{f}_c : \mathbb{R}^{n_x} \to \mathbb{R}^{n_c}$ be the *true* classifier that maps each input $x \in X$ to the correct class (in one-hot encoding).

The most common concept of (adversarial) ML robustness is based on the smallest perturbation necessary to provoke an incorrect classification Fawzi, Fawzi, and Frossard (2018), i.e.

$$\rho_1(f_c, \mathcal{X}, \bar{f}_c) := \mathbb{E}_{x\sim\mathcal{X}}(\Delta_{\mathsf{adv}}(f_c, \bar{f}_c, x)),$$

where

$$\Delta_{\mathsf{adv}}(f_c, \bar{f}_c, x) = \min_{r\in\mathbb{R}^{n_x}} \|r\|_2$$
$$\text{s.t.} \quad \arg\max f_c(x + r) \neq \arg\max \bar{f}_c(x).$$

The value of $\rho_1$ is an important metric for the investigation of ML models Mickisch et al. (2020). However, it only provides the mean distance of $x \sim \mathcal{X}$ to the decision boundary. Neither the severity of the misclassification on the application nor the likelihood of occurrence of the critical perturbation $r$ are considered.

A related property is the so called *cross Lipschitz extreme value for network robustness* (CLEVER) score of a classifier introduced in Weng et al. (2018). For CLEVER, the maximum of the norm of the gradient in a ball around a test input value $x$ is estimated because it can be used to predict the distance of $x$ to the decision boundary. To arrive at the CLEVER score, the mean value over the maximal gradients in balls around $x \sim \mathcal{X}$ is computed.

In Madry et al. (2018) a loss function $L$ is used which allows to quantify the effect of misclassification in the environment of the ML model under test. Using $L$, the authors define the *adversarial risk*

$$\rho_2(f_c, \mathcal{X}, \bar{f}_c) := \mathbb{E}_{x\sim\mathcal{X}}\left(\max_{r\in\mathcal{S}} L(f_c(x + r), \bar{f}_c(x))\right),$$

where $\mathcal{S}$ is a set of admissible perturbations. This definition works well when evaluating ML models in the adversarial setting to assess the mean of the maximal damages an adversary could potentially have on the deployed ML application in a specified environment. However, for general risk assessment this worst case definition does not apply. For that, we need a definition that also takes the probability of each perturbation into account.

To find such a definition, we turn to statistical decision theory and view the risk of deploying a ML model as the risk of a statistical decision making process.

**Definition 1** Risk of a statistical procedure Berger (1985)
*Let $\mathcal{X}$ be a probability space defined as above and let $\mathcal{A}$ be an action space. Furthermore, let $d : X \to \mathcal{A}$ be a deterministic decision function. Then the risk of $d$ with respect to a loss $L$ in the setting of $\mathcal{X}$ is defined as*

$$R(d) = \mathbb{E}_{x\sim\mathcal{X}} L(d(x)) = \int_X L(d(x))\mathsf{d}\mathbb{P}(x).$$

*For a randomized decision function $d^* : X \times \mathcal{D}_x \to \mathcal{A}$ with the parametric probability space $\mathcal{D}_x = (N, \mathcal{G}, \mathbb{P}_x)$ we have that*

$$R(d^*) = \int_X \int_N L(d^*(x, \delta))\mathsf{d}\mathbb{P}_x(\delta)\mathsf{d}\mathbb{P}(x)$$

This definition of risk for a deterministic decision function is well-suited for risk assessment of a ML model on unperturbed (test) data. On the other hand, the double integral formula is a good starting point for general ML risk assessment since it allows to cover both the original data distribution and possible perturbations. In the following we explain how this definition of risk can be applied to evaluate ML models.

## 3 Risk Definition for Machine Learning Applications

To utilize the risk definition from decision theory in ML, we translate all terms from Definition 1 to the ML domain. Our starting point is the randomized setting with the decision function $d^*$. First, we propose to decompose $d^*$ into a deterministic and a stochastic part, which represent the ML model and the input noise, respectively. Note that some ML models include randomization such as in some proposed adversarial defenses (Xie et al. (2018) and Meng and Chen (2017)). This additional randomization that is part of the ML model and that is not caused by input noise can be encompassed similarly by decomposing the ML model into a deterministic and a randomized part. Then for the evaluation of a randomized ML model a third integral is added.

After that decomposition, $\mathcal{X}$ and $\mathcal{D}_x$ immediately carry over to the ML setting. $\mathcal{X}$ represents the underlying data distribution and $\mathcal{D}_x$ represents natural and artificial noise. The interpretation of the loss and the decision function depend strongly on the specific use case. When the ML model is deployed to autonomously take actions, then the ML model is directly the decision function and the loss can simply rate the ML model's decisions. However, if the ML model is used for data analysis and only implies decisions within a more complex system we must either introduce a function that maps the ML model's output to a decision or incorporate the cost associated with worse decisions caused by faulty data analysis into the loss function. We propose to use the latter approach since this reduces the overall complexity of the evaluation.

Using the above considerations, we define the risk of deploying a classifier $f_c$ in an environment $\mathcal{X}$ with perturbations $\mathcal{D}_x$ by

$$\rho(f_c, \mathcal{X}, \mathcal{D}_x) = \int_X \int_N L(f_c(x + \delta))\mathsf{d}\mathbb{P}_x(\delta)\mathsf{d}\mathbb{P}(x), \tag{1}$$

where $L$ is a loss function that maps the classification to the loss of the resulting decision. Note that a possible explicit dependence of $L$ (an thus $\rho$) on $\bar{f}_c$ and $x$ is omitted in (1). Furthermore, in the adversarial setting, $\mathcal{D}_x$ can also depend on $f_c$.

Before explaining how (1) can be approximated, we give a few examples to roughly sketch the scope of our definition of risk.

The adversarial risk from Madry et al. (2018) is encompassed by our framework. This can be seen by choosing $L$ as training loss, and $\mathcal{D}_x$ as space of adversarial perturbations computed as in Madry et al. (2018) that occur for the given target image $x$ with probability one.

When an adversarial defense is proposed, the robustness evaluation is normally performed by checking the decrease in accuracy for different perturbation budgets. In our framework, this translates to a

computation of $\rho$ with

$$L(f_c(x+\delta)) = \mathbb{1}_{\arg\max f_c(x+\delta)=\arg\max \bar{f}_c(x)},$$

for different noise distributions $\mathcal{D}_x$, where $\mathbb{1}$ is the indicator function. Note that in this setting, to address different perturbation budgets, we can compute the risk multiple times for all different perturbation budgets.

We now describe the use case that is the main motivation for this work. When choosing a ML model as vision system for a self-driving car, it must be determined which model leads to the minimal risk when deployed. To assess the risk associated with deploying a ML model, the environment in which it is deployed is described using the natural distribution of input images $\mathcal{X}$ and the noise $\mathcal{N}$. As an example, the model might be deployed in an urban area (which is described by $\mathcal{X}$), in which fog and rain occur regularly and, moreover, there is a $0.1\%$ chance for an adversarial perturbation created with a transfer attack on one of the street signs (which is covered by an appropriate choice of $\mathcal{N}$).

Furthermore, a loss function that estimates the possible damage of a segmentation output is defined. A detailed description of such a loss function is beyond our scope. However, it is important to note that a simple measure such as the sum of misclassified pixels does not necessarily reflect the possible damage. A pedestrian not being detected on non-drivable area is less taxing than a pedestrian being missed on an area that is otherwise classified as drivable.

We emphasize the fact that our risk definition for ML applications via the double integral over the natural data distribution and the (possibly adversarial) noise allows a realistic description of the environment in which the ML model is deployed. On top of that, the loss function within the risk definition can be designed to weight each classification error based on its severity with respect to the given applications.

## 4 The Key Risk Indicator Tensor

We now turn to the computation of $\rho$ for given $L, \mathcal{X}$, and $\mathcal{D}_x$. For that, we propose to approximate the double integral (1) using a Monte Carlo simulation such that we have

$$\rho \approx \hat{\rho} = \frac{1}{n_x n_\delta} \sum_{i=1}^{n_x} \sum_{j=1}^{n_\delta} L(f_c(x_i + \delta_j)),$$

where $x_i$ and $\delta_j$ are samples from $\mathcal{X}$ and $\mathcal{D}_x$, respectively. This straightforward approach works well for fixed $L$, $\mathcal{X}$, and $\mathcal{D}_x$. However, when $\mathcal{X}$ or $\mathcal{D}_x$ are changed (e.g. if new scenarios are added), all computations have to be carried out again which is computationally taxing. Therefore, we propose a light-weight data structure from which $\hat{\rho}$ can be extracted that allows for more flexibility.

The basis for reusing inference results of a classifier $f_c$ for different deployment scenarios in which $\hat{\rho}$ is evaluated is the composition of the scenarios from sets $\{\mathcal{X}_i\}_{i=1}^{n_\mathcal{X}}$ and $\{\mathcal{D}_{xi}\}_{i=1}^{n_\mathcal{D}}$. Then a set of risk values $\{\hat{\rho}_i\}_{i=1}^{n_\mathcal{X} n_\mathcal{D}}$ can be computed and the final risk value $\hat{\rho}$ can be obtained as a convex combination of the elements $\hat{\rho}_i$ as

$$\hat{\rho} = \sum_{i=1}^{n_\mathcal{X} n_\mathcal{D}} \alpha_i \hat{\rho}_i, \text{ with } \sum_{i=1}^{n_\mathcal{X} n_\mathcal{D}} \alpha_i = 1, \tag{2}$$

where $\alpha_i$ can be used to weight different scenario components from which the deployment scenarios are constructed. Note that all different $\hat{\rho}_i$ (and therefore $\mathcal{X}_i$ and $\mathcal{D}_{xi}$) need not be known at the same time. On the contrary, scenario components can be added later to further refine the description of the deployment scenario.

Another advantage of separating $\hat{\rho}$ into different components is a more detailed insight into failure modes of the ML model under test. When $\hat{\rho}$ is directly computed, we obtain no information on which parts of $\mathcal{X}$ or $\mathcal{D}_x$ have caused the risk to increase. However, this information is invaluable for uncovering weaknesses and improving the ML model. As an example, when the evaluation shows that a given segmentation model misses pedestrians in images that contain noise that mimics rain, this can initiate an analysis of whether images of that type are underrepresented in the training set or whether the given ML architecture can in general not deal with that type of noise. For that, we propose to interpret the different $\hat{\rho}_i$ as KRIs of an ML model.

A KRI describes the risk in one particular situation $(\mathcal{X}_i, \mathcal{D}_{xi})$ that may be part of the deployment scenario of the ML model. This can be obtained by modeling a part of the environment. Furthermore, in the adversarial setting, we can view a KRI as an indicator of the susceptibility of a given ML model to a particular adversarial attack. In this way, comparing different KRIs allows to analyze both the mode of action of different attacks as well as the failure modes of the ML model.

When computing $\hat{\rho}_i$, our main objective is the reusability of the inference results of the ML model, since this is the computationally most expensive part. Therefore, we store the computation results in the risk tensor $R$ which is defined by

$$R_{i,j,k,\ell} = L_i(f_c(x_j + \delta_{k,\ell})).$$

$R$ is used to store the results for different loss functions $L_i$, samples of the natural distribution $x_j$, and different samples of a given noise distribution $\delta_{k,\ell}$. The different elements are joined along the different natural distributions and noise types to form the complete risk tensor $R$.

Note that a risk tensor $R^{\text{adv}}$ can be defined for the adversarial robustness use case. Since samples of adversarial noise are typically created for one specific input, we can reduce the dimension of the risk tensor and obtain

$$R^{\text{adv}}_{i,j,k} = L_i(f_c(x_j + \delta_{k,j})).$$

Here we have a one-to-one correspondence of the samples of the noise distribution to samples of the natural image distribution.

Once $R$ is computed, the different $\hat{\rho}_i$ can be obtained by filtering $R$ for distributions relevant for the specific situation which is encompassed by $\hat{\rho}_i$ and aggregating the different tensor elements. When all $\hat{\rho}_i$ have been computed, $\hat{\rho}$ can be obtained as in (2).

## 5 Case Study

We demonstrate the feasibility and utility of our approach by computing the KRIs for a set of neural image classifiers. Note that the KRIs we use in this study are based on well-known and rather straightforward risk measures like the *probability of class change* to keep our results well-aligned with state-of-the-art robustness investigations. In particular, we compare the KRIs of 20 residual neural networks (ResNets), trained on the CIFAR10 Krizhevsky, Hinton, and others (2009) dataset to investigate their respective robustness with respect to image corruptions and adversarial attacks.

We use different robustifying measures alongside changes in the ResNet depth to vary the ResNets under study. In particular, we vary the training data augmentations by adding both Gaussian noise and standard image augmentations implemented in Keras Chollet (2015), adding a regularization proposed in Cisse et al. (2017), changing the training loss function to the robustifying guided complement entropy loss Chen et al. (2019). Furthermore, we obtain a few ResNets from *defensive distillation* as proposed in Papernot et al. (2016) performed at different distillation temperatures and by adversarial training (both *ensemble adversarial training* Tramèr et al. (2018), and *projected gradient descent* Madry et al. (2018) were tested). A description of the setup of each ResNet we study is provided in Table 1.

We evaluate and compare the ResNets' capability to cope with image perturbations induced by sensor corruptions, random noise, weather phenomena, and adversarial attacks. For each image perturbation type, we set up several distributions that represent each corruption scenario. For sensor corruptions, we consider random changes in brightness and contrast. On top of that, we add shadows and rotations of varying magnitude to the test images. Random noise is considered by adding distributions of Gaussian, uniform and salt-and-pepper noise. We incorporate weather phenomena by adding randomly generated layers of rain or fog to the test images. For the creation of adversarial perturbations, we use the adversarial robustness toolbox Nicolae et al. (2019) to generate distributions that contain images created with the fast gradient sign method Goodfellow, Shlens, and Szegedy (2015), the CarliniL2 method Carlini and Wagner (2017a), and the DeepFool attack Moosavi-Dezfooli, Fawzi, and Frossard (2016), respectively.

A key feature of our approach is the hierarchical aggregation of the computed loss values. In our study, the loss values are the probability of class changes, which can be aggregated with a mean value function over the different noise types. As an example, in Figure 2a, we compare the class change risk

| # | label | # layers | reg. | augment. | defense |
|---|---|---|---|---|---|
| 1 | NNet | 72 | - | - | - |
| 2 | NNetLarge | 114 | - | - | - |
| 3 | DistilledT10 | 72 | - | - | distillation |
| 4 | DistilledT10Large | 114 | - | - | distillation |
| 5 | DistilledT100 | 72 | - | - | distillation |
| 6 | DistilledT100Large | 114 | - | - | distillation |
| 7 | DistilledT10Augm | 72 | - | std. image augmentations | distillation |
| 8 | DistilledT100Augm | 72 | - | std. image augmentations | distillation |
| 9 | Gauss03 | 72 | - | Gaussian noise ($\sigma = 0.03$) | - |
| 10 | Gauss09 | 72 | - | Gaussian noise ($\sigma = 0.09$) | - |
| 11 | Gauss09Augm | 72 | - | Gaussian noise ($\sigma = 0.09$) | - |
| 12 | Gauss09Large | 114 | - | Gaussian noise ($\sigma = 0.09$) | - |
| 13 | GCE | 72 | - | gce loss | - |
| 14 | GCELarge | 114 | - | gce loss | - |
| 15 | ParsNet | 72 | parseval frame | - | - |
| 16 | ParsNetAugm | 72 | parseval frame | std. image augmentations | - |
| 17 | ParsNetLarge | 114 | parseval frame | - | - |
| 18 | AdvTrainFGSM | 72 | - | - | adv. training |
| 19 | AdvTrainPGD | 72 | - | - | adv. training |
| 20 | AdvTrainFGSMLarge | 114 | - | - | adv. training |

Table 1: Description of ResNets used for KRI computation

of test images with brightness perturbations for the ResNets we study. These values are computed by

$$\hat{\rho}_{\text{br}} = \frac{1}{n_{\text{samples}}} \sum_{x_i \in X} \sum_{\delta_{\text{br},\ell} \in \mathcal{D}_{\text{br}}} L_{cc}(x_i + \delta_{\text{br},\ell}),$$

where $L_{cc}$ is the indicator for a class change, $X$ is the set of test images, and $\delta_{\text{br},\ell}$ is a sample from the distributions of brightness perturbations.

The individual values $\hat{\rho}_{\text{br}}$ for the different ResNets under study can be considered their brightness corruption KRIs. On the other hand, we can summarize the risk values for all sensor perturbations into a single sensor corruption KRI, by aggregating over all sensor corruption distributions. In this way, the sensor corruption KRIs are computed by

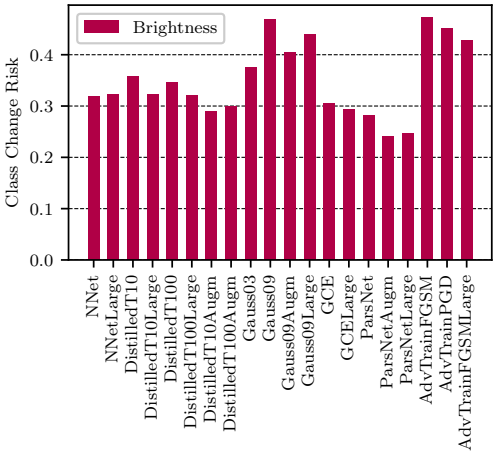
$$\hat{\rho}_{\text{sc}} = \frac{1}{n_{\text{samples}}} \sum_{x_i \in X} \sum_{D_i \in D_{\text{sc}}} \sum_{\delta_{\text{i},\ell} \in D_i} L_{cc}(x_i + \delta_{i,\ell}),$$

where $D_{\text{sc}}$ is the set of all considered sensor corruption distributions. At this stage it is possible to weight the different corruptions in order to mimic their given occurrence probability. In Figure 2b we use these higher level KRIs to understand the effect of data augmentation on the robustness of a ResNet. We can observe that by adding Gaussian noise and standard image augmentations or adversarial noise to the training images, we can increase the robustness of the ResNets with respect to random and adversarial noise by similar amounts. On the other hand, when we compare the standard cross entropy loss to the guided complement entropy loss as in Figure 2c, we can observe that using the guided complement entropy loss, we can significantly increase the robustness with respect to adversarial noise. However, the vulnerability with respect to the other noise types stays approximately the same.

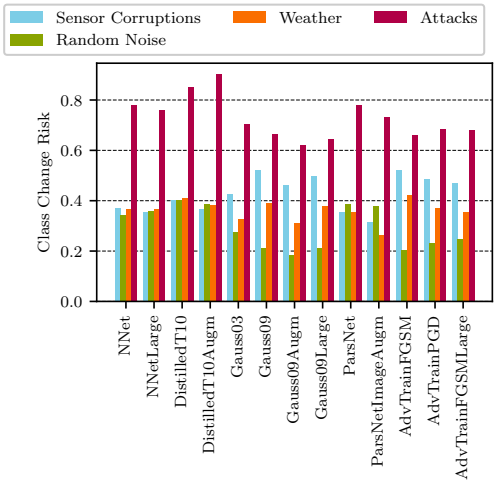
Finally, we can merge all KRIs into the final risk value. In our study, we simply compute the mean value over all KRIs. However, a more involved strategy to study a specific use case can also be implemented. The final risk values are displayed in Figure 2d. On the basis of these values, an informed choice of the ResNet associated with the minimal risk can be made.

## 6 Conclusion and Outlook

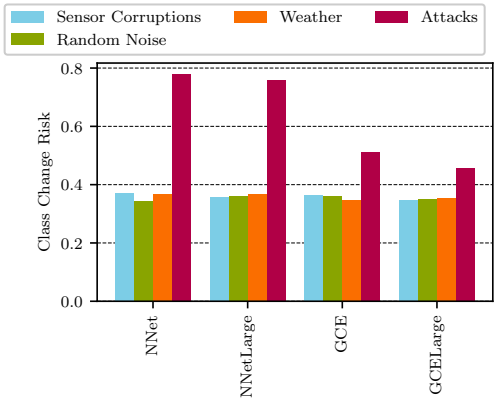
In this work, we have applied the risk definition from statistical decision theory to ML. On the basis of this definition we have developed a framework that allows to specify different deployment scenarios



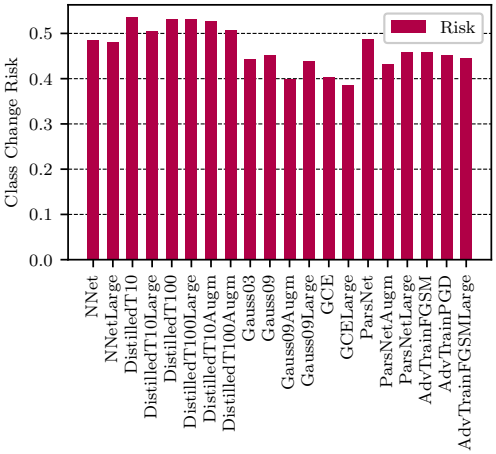

(a) Comparison of class change risk for brightness perturbations




(b) Assessment of the impact of image data augmentation for different noise types




(c) Assessment of the impact of the loss function for different noise types




(d) Final risk values

Figure 2: Aggregated risk values

and penalties associated with failures of the ML model. This allows practitioners to evaluate the risk of deploying a given ML model in a standardized way. Furthermore, the setup of deployment scenarios gives regulatory authorities the chance design certificates for ML models in specified environments.

In our preliminary numerical case study we have provided another motivation for using KRIs to investigate ML model robustness, i.e. the investigation of the effect of different robustifying measures on perturbations of different types. As an example, while adding data augmentations increased the accuracy under random and adversarial perturbations, a change in loss function from cross entropy to guided complement entropy only increased robustness for adversarial perturbations.

The application of the risk definition and the proposal for its efficient tensor based evaluation provide the tools necessary for extensive analysis of ML models. It remains to create meaningful loss functions and data distributions for different applications in which such a detailed analysis is required.

## References


Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. volume 80 of *Proceedings of Machine Learning Research*, 274–283. Stockholmsmässan, Stockholm Sweden: PMLR.

Berger, J. O. 1985. *Statistical Decision Theory and Bayesian Analysis*. Springer Science & Business Media.

Carlini, N., and Wagner, D. 2017a. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57.

Carlini, N., and Wagner, D. 2017b. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*, 3–14. Dallas, Texas, USA: ACM Press.

Chen, H.; Liang, J.; Chang, S.; Pan, J.; Chen, Y.; Wei, W.; and Juan, D. 2019. Improving adversarial robustness via guided complement entropy. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 4880–4888.

Chollet, F. 2015. Keras. https://keras.io.

Cisse, M.; Bojanowski, P.; Grave, E.; Dauphin, Y.; and Usunier, N. 2017. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 854–863. JMLR.org.

Fawzi, A.; Fawzi, O.; and Frossard, P. 2018. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning* 107(3):481–508.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient smt solver for verifying deep neural networks. In Majumdar, R., and Kunčak, V., eds., *Computer Aided Verification*, 97–117. Cham: Springer International Publishing.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial examples in the physical world. *arXiv:1607.02533 [cs, stat]*.

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. *arXiv:1611.02770 [cs]*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Meng, D., and Chen, H. 2017. Magnet: A two-pronged defense against adversarial examples.

Mickisch, D.; Assion, F.; Greßner, F.; Günther, W.; and Motta, M. 2020. Understanding the Decision Boundary of Deep Neural Networks: An Empirical Study. *arXiv:2002.01810 [cs, stat]*.

Moosavi-Dezfooli, S.; Fawzi, A.; and Frossard, P. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2574–2582.

Nicolae, M.-I.; Sinn, M.; Tran, M. N.; Buesser, B.; Rawat, A.; Wistuba, M.; Zantedeschi, V.; Baracaldo, N.; Chen, B.; Ludwig, H.; Molloy, I. M.; and Edwards, B. 2019. Adversarial Robustness Toolbox v1.0.0. *arXiv:1807.01069 [cs, stat]*.

Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; and Swami, A. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, 582–597.

Pei, K.; Cao, Y.; Yang, J.; and Jana, S. 2017. Towards Practical Verification of Machine Learning: The Case of Computer Vision Systems. *arXiv:1712.01785 [cs]*.

Tian, Y.; Pei, K.; Jana, S.; and Ray, B. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, 303–314.

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2018. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*.

Weng, T.-W.; Zhang, H.; Chen, P.-Y.; Yi, J.; Su, D.; Gao, Y.; Hsieh, C.-J.; and Daniel, L. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*.

Xie, C.; Wang, J.; Zhang, Z.; Ren, Z.; and Yuille, A. 2018. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*.