Shapes are not enough: CONSERVAttack and its use for finding vulnerabilities and uncertainties in machine learning applications

Philip Bechtle[1], Lucie Flek[2], Philipp Alexander Jung[3], Akbar Karimi[2], Timo Saala[1*], Alexander Schmidt[3], Matthias Schott[1], Philipp Soldin[4], Christopher Wiebusch[4], Ulrich Willemsen[3]

[1]Institute of Physics, University of Bonn, Germany.
[2]Bonn-Aachen Institute of Technology, University of Bonn, Germany.
[3]Physics Institute III A, RWTH Aachen University, Germany.
[4]Physics Institute III B, RWTH Aachen University, Germany.

*Corresponding author(s). E-mail(s): tsaala@uni-bonn.de;

Abstract

In High Energy Physics, as in many other fields of science, the application of machine learning techniques has been crucial in advancing our understanding of fundamental phenomena. Increasingly, deep learning models are applied to analyze both simulated and experimental data. In most experiments, a rigorous regime of testing for physically motivated systematic uncertainties is in place. The numerical evaluation of these tests for differences between the data on the one side and simulations on the other side quantifies the effect of potential sources of mismodelling on the machine learning output. In addition, thorough comparisons of marginal distributions and (linear) feature correlations between data and simulation in "control regions" are applied. However, the guidance by physical motivation, and the need to constrain comparisons to specific regions, does not guarantee that all possible sources of deviations have been accounted for. We therefore propose a new adversarial attack – the CONSERVAttack – designed to exploit the remaining space of hypothetical deviations between simulation and data after the above mentioned tests. The resulting adversarial perturbations are consistent within the uncertainty bounds—evading standard validation checks—while successfully fooling the underlying model. We further propose strategies to mitigate such vulnerabilities and argue that robustness to adversarial effects must be considered when interpreting results from deep learning in particle physics.

Keywords: Deep Learning, Adversarial Attack, Data Augmentation, AI in Sciences

1 Introduction

Deep learning has established itself as a powerful tool across a wide range of scientific domains, including medicine [1], autonomous driving [2], and, more recently, the development of general-purpose Large Language Models [3]. High Energy Physics (HEP) is no exception: While machine learning approaches—such as simple multilayer perceptrons—have been used in the field for decades, modern deep neural networks are increasingly employed in generative tasks (e.g., detector simulation), regression tasks (e.g., event reconstruction), and classification tasks (e.g., event

selection and real-time triggering at the Large Hadron Collider (LHC)).

Despite these successes, applying deep learning in precision-driven fields like HEP presents significant challenges. Traditionally, analyses rely on well-defined statistical frameworks to estimate uncertainties by quantifying statistical variations and performing systematic studies of potential mismodellings in the simulated data relative to real observations. The numerical treatment of these uncertainties is included in the statistical frameworks used for model parameter fitting and for the calculation of confidence intervals [4]. These uncertainties are evaluated in hundreds of separate studies, each probing different *physical* sources of discrepancy, and are cross-checked through comparisons between data and simulation in carefully designed "control" and "validation" regions of the data's feature space, as well as through independent analyses. The discovery of the Higgs boson represents a famous and successful example of the thorough application of these techniques [5, 6]. Marginal distributions of features and their linear pairwise correlations—for both real and simulated events—are routinely examined and validated to ensure accurate modelling and well-quantified uncertainties. However, these cross-checks rarely go beyond the inspection of marginals, correlations, and possibly a limited number of two-dimensional distributions. In addition, validation regions are typically used to compare the output distribution of a classifier or regressor between data and simulation.

Neural networks often rely on high-dimensional and non-linear correlations in both data and simulation, making a systematic validation of their predictions using only the techniques described above challenging. Even when substantial effort is invested in producing accurate simulated data for training, the corresponding validation procedures remain limited. In most cases, they rely on comparisons of marginal feature distributions and pairwise correlations between simulated and real events. These checks do not probe the full complexity of the models' decision boundaries.

In this work, we demonstrate that this commonly used validation strategy can potentially become insufficient once we move beyond explicitly studied, physically motivated deviations between data and simulation. As soon as additional hypothetical sources of mismodeling—whether they are forgotten, overlooked, or entirely unknown—are allowed, and once we acknowledge that the "signal region", to which a classifier or regression model is ultimately applied, may be affected by different sources of discrepancy than the control or validation regions, the traditional validation procedure may no longer provide appropriate coverage. To quantify the potential impact of such effects, we construct adversarial perturbations that modify simulated events in a targeted manner, inducing substantial degradation in the performance of downstream deep learning models while keeping all marginal distributions and inter-feature correlations well withing their expected statistical uncertainties. Due to these perturbations remaining invisible to standard validation techniques, they represent a previously unexplored source of systematic uncertainty. The ability to generate such undetectable adversaries thus provides a new means of estimating an upper bound on a model's systematic vulnerability.

The techniques presented here are used to propose a workflow that estimates the maximum susceptibility of a network to these adversarial perturbations, while also outlining steps to reduce the resulting uncertainties to, or below, the level associated with known physically motivated sources. In this workflow, a final result indicating that the model's susceptibility is within the range of established physical uncertainties would suggest that no additional uncertainty from adversarial effects needs to be considered, since it can be assumed that the physically motivated systematics already cover the uncertainty from adversarials. Conversely, if the proposed workflow yields a final fooling ratio exceeding the magnitude of the physically motivated uncertainties, this would prompt either a review for potential omissions in the physical uncertainty estimates or the assignment of an additional uncertainty to account for misclassification due to adversaries of unknown origin.

We further investigate whether—in some cases—discrepancies between model performance on simulated and real events, as sometimes observed in analyses, may be partially explained by the presence of adversarially structured yet statistically consistent events in simulation.

Beyond assessing vulnerabilities and remaining uncertainties, we demonstrate that adversarial

events can be leveraged to improve model performance, even on unperturbed inputs. Specifically, we apply this attack as a form of data augmentation in low-data regimes.

Finally, we explore two adversarial defense strategies aimed at reducing the susceptibility of deep learning models to such attacks. The first, adversarial training, involves augmenting the training set with adversarial events. The second involves constructing an Adversarial Detector network, optimized to distinguish between clean and adversarially perturbed events. In both cases, we show that it is possible to improve the models robustness against these adversaries—consequently lowering the estimation of the uncertainty upper-bound. Additionally, we perform a detailed analysis of the Adversarial Detector's behavior on both simulated and real HEP data, investigating whether certain simulated events exhibit adversarial behavior and how well the detector generalizes to unseen real events.

2 Related Work

Adversarial attacks and defenses are well-established fields of research across a variety of contexts [7], and have been particularly extensively studied in security-critical applications. Many sophisticated methods exist for constructing adversarial examples across diverse types of data, including images [8, 9], tabular data [10], and even natural language [11]. However, these approaches do not necessarily address the specific requirements and constraints relevant to High Energy Physics (HEP).

Within HEP, research on adversarial attacks and, more broadly, adversarial deep learning remains relatively limited [12, 13]. In this work, we propose an adversarial attack specifically designed to address these domain-specific needs. We study the behavior of this attack in a general adversarial deep learning context, as well as its implications for downstream HEP applications and prior event simulations. Furthermore, we discuss the consequences of such attacks on uncertainty estimation in deep learning tasks and simulations within HEP, two areas that have been studied extensively. Nonetheless, we argue that the approach we use here to analyze model uncertainties has not yet been explored sufficiently.

Another line of work addressing robustness to distributional shifts is Distributionally Robust Optimization (DRO) [14]. In DRO, models are trained to minimize the worst-case loss over an uncertainty set of distributions surrounding the empirical training distribution. These uncertainty sets are typically defined through divergence measures, such as Wasserstein or f-divergence balls, allowing the optimization procedure to consider adversarial reweightings or shifts of the data distribution during training. Conceptually, this shares similarities with the adversarial perspective adopted in this work, as both aim to reason about worst-case deviations from the nominal data distribution. However, the assumptions and objectives differ substantially. In most DRO formulations, the distributions considered during optimization are not constrained to preserve specific statistical properties of the dataset, such as marginal feature distributions or pairwise linear correlations. As a result, the adversarial distributions explored by DRO may deviate in ways that would be easily detectable by the validation procedures commonly used in High Energy Physics analyses. In contrast, the perturbations constructed in this work are explicitly constrained to remain consistent with the statistical checks routinely performed in HEP, including agreement of marginal distributions and linear feature correlations within their expected uncertainties. Our approach therefore focuses on identifying adversarial deviations that would evade standard validation strategies, providing a complementary perspective to distributionally robust training methods.

3 Method: CONSERVAttack

The goal of this attack is to construct adversarial perturbations that remain physically invisible under analyses commonly applied in High Energy Physics (HEP). To achieve this, we must ensure that neither the marginal distributions of the input features nor the correlations between them are significantly altered. This requirement motivates a shift in perspective compared to other adversarial attack design. While most attacks typically seek to constrain the perturbation on a per-event basis (e.g., by minimizing the L_∞ norm

of each modified input), our method instead constrains changes at the dataset level, such as over an entire test set.

Similar to state-of-the-art adversarial attacks, such as Projected Gradient Descent (PGD), we aim to optimize a min-max problem. We want to maximize a goal function—such as reaching misclassification—while simultaneously minimizing the perturbations applied to data. For this attack, this minimization step involves two separate constraints evaluated over a batch of inputs: preserving marginal features distributions and preserving the inter-feature correlations. The attack proceeds via an iterative, gradient-based, brute-force–style search over candidate perturbations.

More specifically, for each input we compute the gradient of the model's loss function with respect to the input at the final layer. We then discard the gradient magnitudes, retaining only its sign. Most adversarial attack algorithms use the sign of the gradient to apply either a single large, or many small perturbations to the input, we use it to generate a set of candidate perturbations S_{cp} for each feature.

To this end, we introduce two hyperparameters: the minimal perturbation magnitude $p_{\min} > 0$, which sets the smallest allowed change to an input feature, and a step size ϵ_{step}, which determines the spacing between candidate perturbations. For each candidate in S_{pc}, we evaluate its impact on both the marginal feature distributions and the feature–feature correlations.

Alternatively, the candidate generation procedure can be overridden by explicitly specifying the number of candidates to be considered via the parameter num_C. When this option is used, the attack generates num_C uniformly distributed candidate perturbations between the current feature value and the corresponding minimum or maximum global value of that feature—depending on the gradient sign.

To quantify changes in the marginal distributions, we compute the Jensen–Shannon Distance (D_{JS}), defined as

$$D_{\text{JS}} := \sqrt{\frac{D(\vec{p}\|\vec{m}) + D(\vec{q}\|\vec{m})}{2}}, \tag{1}$$

where D is the Kullback-Leibler divergence. Here, $\vec{p}$ and $\vec{q}$ are the normalized approximate probability distributions (estimated using histograms) and $\vec{m}$ is their pointwise mean.

To quantify changes in correlations, we define Δ_F as the relative Frobenius norm difference between the correlation matrices of the clean and perturbed inputs:

$$\Delta_F := \frac{\|C_{\text{adv}} - C_{\text{clean}}\|_F}{\|C_{\text{clean}}\|_F}, \tag{2}$$

where C_{clean} and C_{adv} denote the correlation matrices of the original and perturbed inputs, respectively. The normalization by $\|C_{\text{clean}}\|_F$ expresses the change relative to the magnitude of the clean correlation matrix.

The Frobenius norm of a matrix A is defined as

$$\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}, \tag{3}$$

where $A \in \mathbb{R}^{m\times n}$ and a_{ij} denotes the entry in the i-th row and j-th column.

After computing D_{JS} and Δ_F for each candidate perturbation of each feature, we select the perturbation that minimizes a custom loss function.

$$\mathcal{L} := \alpha D_{\text{JS}} + \beta \Delta_F, \tag{4}$$

where α and β hyperparameters controlling the relative weight of the two constraints. This is done for a pre-defined amount of iterations n_{it}. Moreover, setting `nc = True` allows the algorithm to skip an update entirely whenever no candidate modification satisfies the required bounds on the D_{JS} variation ¡ $\max_{\text{D}_{\text{JS}_t}}$ and the Frobenius Norm change $\max_{\Delta\text{FN}_t}$, both of which are user-configurable hyperparameters.

Moreover, throughout the optimization we maintain a dynamic boolean mask that tracks whether a given event already successfully fools the model under attack. By default, the flag opt_{af} (optimize already fooled events) is set to `False`. In this configuration, once an event achieves the

adversarial objective (e.g., misclassification), it is excluded from further perturbations in subsequent iterations. However, these events are still included when computing dataset-level metrics such as the marginal feature distributions and inter-feature correlations, ensuring that the overall D_{JS} and Δ_F constraints continue to account for all data. This reduces unnecessary modifications to already successful adversarial examples while preserving accurate distributional monitoring.

Alternatively, setting opt_{af} to `True` keeps such events within the perturbation loop. In this case, the algorithm continues to evaluate and apply candidate perturbations to already successful adversarial examples, aiming to reduce their contribution to D_{JS} and Δ_F terms while maintaining their adversarial property. This optional refinement step can therefore produce adversarial examples that remain effective while exhibiting smaller deviations in the monitored distributional and correlation-based metrics.

Additional optimizations and implementation details—such as more computationally efficient procedures for evaluating D_{JS} and Δ_F—are provided in the Appendix A

4 Data Sets, Tasks and Classification Models Used for the Performance Evaluation

We evaluate CONSERVAttack on two common particle physics tasks. The first task, referred to as Higgs, is based on the popular Higgs Boson Machine Learning Kaggle Challenge [15]. The second task is a jet-tagging problem, where the goal is to distinguish jets originating from a pair of top quarks (TTJets) from those originating from a pair of W bosons (WWJets). The datasets for this task were obtained from the CERN Open Data portal [16], more specifically using simulated samples from the 2012 CMS run [17, 18].

In both tasks, the objective is binary classification: TT vs. WW Jets, and signal (Higgs) vs. background processes. For the studies presented here, we apply simple Multi-Layer Perceptrons (MLP) to both tasks. For the TT vs. WW classification, the architecture of the deep learning model is based on the TopoDNN model [19].

The Higgs dataset contains a total of 30 continuous input features, and the corresponding MLP has a total of 42,163 trainable parameters. The TopoDNN-inspired model used for the jet-tagging task uses 87 continuous input features and has 59,263 trainable parameters. Hyperparameter choice, training details, as well as more technical details for both of these networks can be found in the Appendix C.

5 Impact of CONSERVAttack

Although adversarial robustness is generally not a security-critical concern in HEP, we still evaluate the attack against neural networks to assess its effectiveness. Our goal is to show that it is possible to construct adversarial inputs that, according to standard HEP simulation validation procedures, would be assigned to class A, while the downstream classifier misidentifies them as belonging to another class.

To evaluate the effectiveness of the attack, we consider both parts of the min–max optimization. For the maximization objective—that is, inducing a misclassification—we use the Fooling Ratio ($\mathcal{F}$), defined as

$$\mathcal{F} := \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(f(x_i') \neq f(x_i)) \,, \tag{5}$$

where N denotes the total number of events, $f(\cdot)$ is the classifier, x_i is the original input, and x_i' is its adversarially perturbed counterpart. Here, $\mathbf{1}(\cdot)$ denotes the indicator function, which equals 1 if the condition is true and 0 otherwise. The Fooling Ratio therefore measures the fraction of inputs for which the predicted label changes under the adversarial perturbation. By construction, our attack updates each input along the gradient direction that increases the classifier loss, similar to standard PGD or FGSM approaches; as a result, the Fooling Ratio is implicitly optimized with each iteration without requiring an explicit term in the loss function.

To quantify the minimization component—here, the preservation of the dataset's statistical properties—we use the previously introduced Jensen-Shannon Distance (1) and Δ_F (2). Both quantities are averaged over all input features,

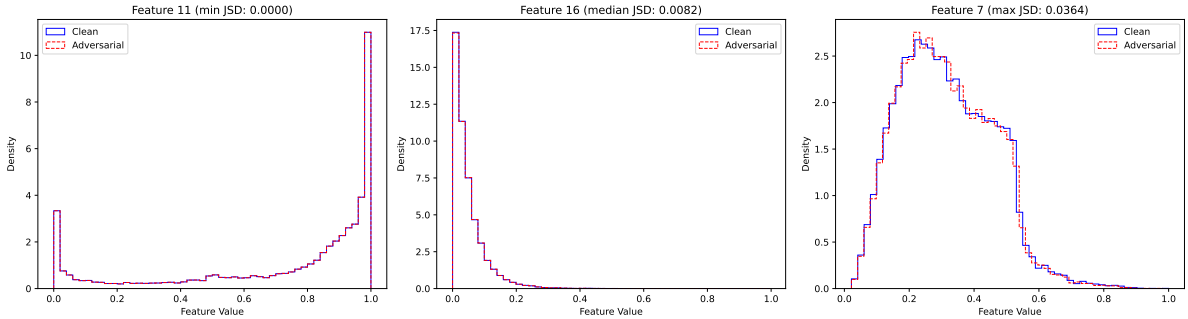

Fig. 1: Comparison of three distinct distributions between clean and adversarial events in an attack setting. The plot on the left shows the feature resulting in the minimal change in Jensen Shannon distance, the one in the middle the feature corresponding to the median Jensen Shannon Distance, and the plot on the right the feature exhibiting the largest change in the Jensen Shannon Distance for the given attack run.

yielding a single value. Smaller values indicate a better match between the perturbed and original distributions or correlations.

In contrast to the Fooling Ratio, these metrics are optimized explicitly via the combined loss function (4) introduced in the Methods section, which penalizes deviations in marginal distributions and inter-feature correlations. While there is no set threshold for interpreting these metrics in terms of statistical significance, visual inspection of the resulting distributions and correlations allows for a good assessment. Example results for the Higgs task, where our attack achieves a Fooling Ratio of roughly 0.9, are shown in Figures 1 and 2.

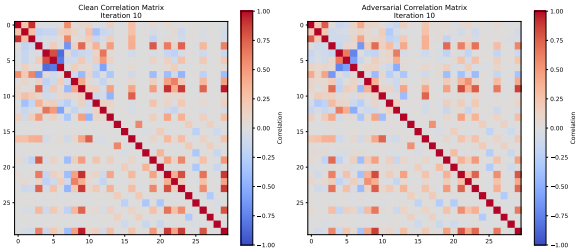

Fig. 2: Comparison of the correlation matrices between the clean events (left) and adversarial events (right) resulting from the attack.

The results indicate that, even in the worst case, the attack introduces only minor perturbations to the marginal distributions, and the resulting correlation matrices remain close to their original values.

To better assess the efficiency of the attack, we apply it ten times for each task. In each run, a new network is trained using identical hyperparameters but different random seeds for initialization. The attack parameters are fixed across all runs and are listed in Table 1.

First, we examine the maximization component of the attack—its ability to induce misclassification. The resulting Fooling Ratios for ten independent runs on the Higgs task are shown in Figure 3.

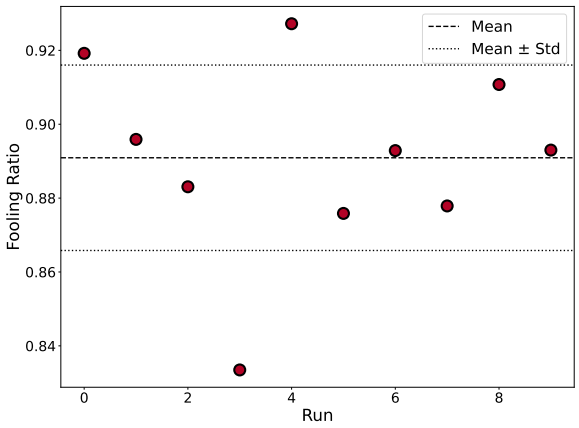

Fig. 3: Fooling ratio of the adversarial attack on the Higgs dataset over 10 runs. The average performance of the attack is shown as a dashed line, and its standard deviation as a dotted line.

Table 1: Attack parameters for both the Higgs and the TT vs. WW jet-tagging tasks.

Task	p_{min}	ϵ_{step}	n_{it}	α	β	$\max_{JSD_t}$	$\max_{\Delta_F t}$	nc
Higgs	0.0005	0.0005	10	4.0	1.0	0.006	0.0002	True
TTvsWW	0.005	0.01	10	6.5	1.0	0.003	0.003	True

Across the ten runs, the average Fooling Ratio is 0.89, indicating that the attack consistently generates adversarial inputs that successfully alter the network's predictions. Next, we need to verify that these adversarial examples achieve this while minimally—ideally imperceptibly—modifying the underlying marginal distributions and feature correlations. The corresponding metrics, the Jensen-Shannon Distance and the Frobenius norm of the difference between correlation matrices, are shown in Figure 4 and 5.

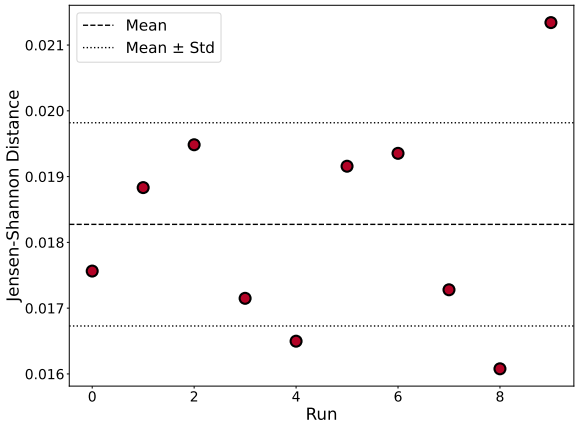

Fig. 4: Average Jensen Shannon distance of the adversarial attack on the Higgs dataset over 10 runs. The average performance of the attack is shown as a dashed line, and its standard deviation as a dotted line.

For the Jensen-Shannon Distance, all but one run remain below 0.02, indicating a good match between the adversarial and clean distributions. Similarly, Δ_F remains below 0.2, demonstrating a low perturbation of the correlations. These results indicate that, across multiple independent runs, it is feasible to generate adversarial samples that induce misclassification while only minimally altering the marginal distributions and correlations.

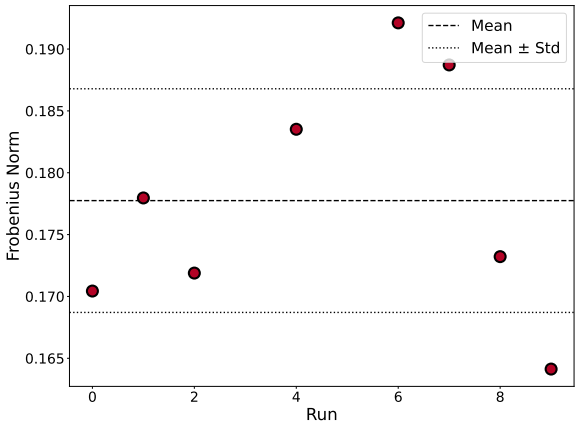

Fig. 5: Average Frobenius Norm of the adversarial attack on the Higgs dataset over 10 runs. The average performance of the attack is shown as a dashed line, and its standard deviation as a dotted line.

The attack shows similar behavior on the TopoDNN network. Across ten independent runs, there, the average Fooling Ratio is 0.675, with an average D_{JS} of 0.045 and an average Δ_F of 0.182. Corresponding plots and additional results are provided in the Appendix D.1.

Additionally, we performed a physically motivated variant of the attack on the Higgs dataset, in which the constrained attack is applied only to the control region, while fully unconstrained perturbations are applied to the complementary region. A more detailed description, along with the results and their discussion, is provided in Appendix B.

6 Data Augmentation using CONSERVAttack

Before addressing how to construct models that are robust to physically undetectable adversaries, we first explore the use of these adversarial attacks as a data-augmentation strategy. Prior work has shown that, in some settings, augmenting the training set with adversarial examples can

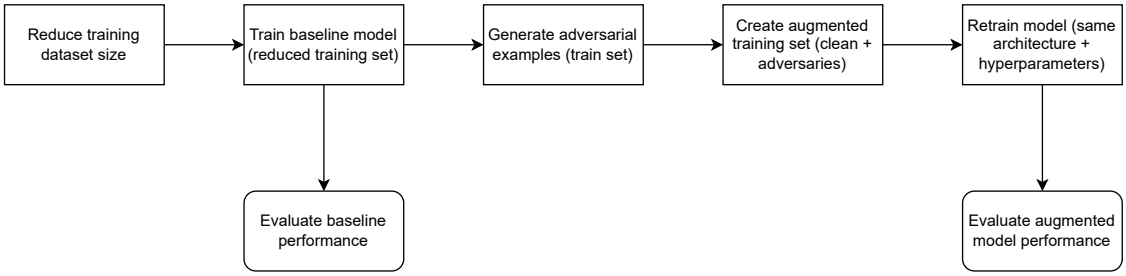

Fig. 6: Flowchart showing the workflow used in the data augmentation pipeline.

improve the generalization capability of neural networks—not only against adversaries but also on clean inputs [20]. However, in cases where a model is already saturated in its learning capacity (for example it has been trained on a large enough dataset), adversarial training can instead degrade the models performance on clean inputs.

In this study, we focus on the situation where the baseline model does not have enough training data to fully saturate. To force this scenario, we intentionally reduce the size of the training data for both the Higgs and the TT vs. WW tasks until the model's baseline performance noticeably deteriorates. For the Higgs network, the training set size is reduced from roughly 150.000 to 5.000 inputs, and for the TopoDNN-like model from about 280.000 to 70.000.

The augmentation procedure is as follows: First, the network is trained on the artificially reduced training set, resulting in an undertrained baseline model. This model is then used within our adversarial attack pipeline, but in contrast to the previous experiments, the adversaries are now generated from the training set. The resulting adversarial inputs are shuffled into the original training data; assuming a Fooling Ratio of 1.0, this yields at most one adversary per clean input, at most doubling the dataset size.

After generating the adversaries, the model's weights and biases are reinitialized, and the network is trained from scratch using the adversarially augmented training set. A flowchart depicting this augmentation approach is shown in Figure 6.

To quantify the performance of this augmentation approach, we compare the AUROC on clean test inputs between the baseline network and the network trained on the augmented dataset, averaging the results over 10 independent runs. The corresponding results for the Higgs dataset are shown in Figure 7.

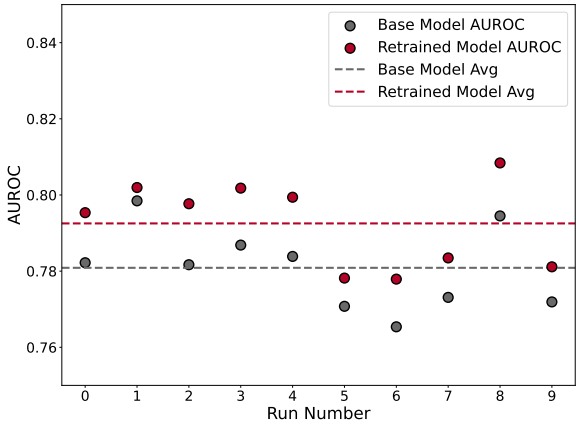

Fig. 7: Comparison of the areas under the Receiver-Operating-Curves—on the base model (grey) and the adversarially augmented model (red)—on the Higgs data set over 10 independent runs. The respective average results are shown in dashed lines.

The results show a consistent improvement in performance on clean test data across all runs, demonstrating the effectiveness of using adversarial samples as data-augmentation. Although the average gain is moderate—about one percentage point—it still indicated that adversarial augmentation can enhance generalization—especially in data-limited regimes.

A key limitation for this study is the need to artificially reduce the training set to place the baseline model in the undertrained regime. For the Higgs task especially, achieving this required a significant reduction in training size. However,

reducing the dataset too aggressively would negatively impact the adversarial attack itself, which relies on sufficient statistics to optimize perturbations over the full dataset. This invoked a balance between reducing the training set enough to benefit from augmentation, but not so much that the attack itself becomes ineffective.

We expect this trade-off to have a significantly smaller impact for more complex tasks that naturally require much larger training datasets, hence resulting larger performance gains through adversarial augmentation.

Results for the TT vs. WW dataset are provided in the Appendix D.2.

7 Robustness studies using CONSERVAttack

A very important field of research within adversarial deep learning—often called adversarial defenses—is studying methods to defend networks against adversarial attacks—e.g. improving a networks adversarial robustness. In this section, we will discuss why it is important in the context of High Energy Physics to conduct such studies, what types of attacks and defenses exist, how they differ and also how they apply to this research. Afterwards, we will apply two well-established defense techniques—adversarial training [8] and an adversarial detector [21]—to increase the robustness of the networks for both the Higgs, and the TT vs. WW datasets.

Within security critical fields, such as self-driving cars, it is immediately obvious why it would be important to design networks that are inherently robust to malicious inputs. In HEP however, such security concerns are of very little to no importance. Still, we believe—especially when considering attacks that will not be detected by current sanity checks—it is important to study the robustness of our deep learning models against such cases. Specifically, by testing and improving the networks robustness against invisible attacks, we can get a better understanding of the upper-bound of a models intrinsic uncertainty.

When it comes to adversarial attacks, they can typically be divided into either two or three classes. White-Box Attacks, Black-Box Attacks, and sometimes even Grey-Box Attacks [22]. These classes divide the adversaries by the knowledge they have about the model—or the data—they are targeting. In White-Box Attacks, one assumes (nearly) full knowledge about the data and the network, for example knowing the gradients of the loss function of the target network, or knowing the exact training, validation, and test data. In Black-Box Attacks, typically one assumes to have no knowledge about the target network or data. In such cases, the attacks, for example, rely on the capability of querying the target network, hence being able to see whether the model correctly or incorrectly processes the input. Another Black-Box approach is trying to reproduce, as closely as possible, the target network, then generating adversaries on the dummy network, and applying these to the target network. This is often possible due to the phenomenon of transferability of adversarial attacks [23]. In Grey-Box Attacks, as the name applies, the information about the network is more nuanced. One assumes to have some knowledge about the network, the data, or the training regime.

In our case, we deploy a Grey-Box Attack. Here, we have knowledge about the network architecture and the data used for training, validating, and testing the model. We do not however, have any knowledge about the gradients, weights, or biases of the target networks. Initially, we deployed White-Box Attacks, however, it is very hard to construct networks that are truly robust against White-Box Attacks. Additionally, the Grey-Box Attack is more realistic for what we are trying the study—namely under the assumption that such adversaries might also arise from the simulations used in HEP—there we also would not expect the "attack" to have full knowledge about the downstream deep learning network.

To be precise, our Grey-Box attacks work as follows: First, we use the exact same approach as in the attack studies. We simply apply our adversarial attack algorithm to the test set using the pre-trained model. Doing this, leveraging the gradients of the respective networks, represents a White-Box Attack. However, instead of testing robustness against these attacks, we instead do this process multiple times, each time using the same model architecture and hyperparameters, also fixing the parameters for the adversarial attack—but using random initialization of weights and biases for each of the independent runs. We

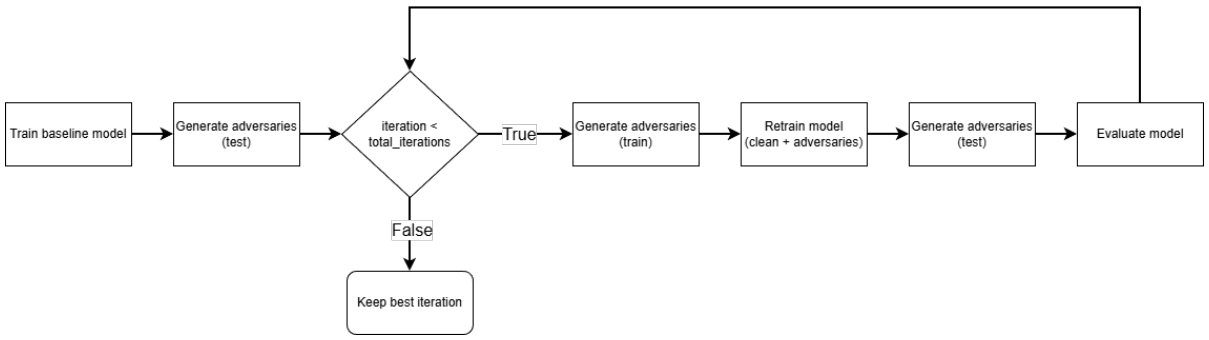

Fig. 8: Flowchart showing the workflow used in the adversarial training pipeline.

can then use the adversaries generated on the disjunct runs to test the robustness of the current run against Grey-Box attacks.

To quantify the effectiveness of our adversarial defense techniques, we will compare the fooling ratio achieved on the non-robust base models to the fooling ratio achieved on our—hopefully—more robust models after applying the defense.

For the Adversarial Training approach, we use mostly the same approach as introduced in Section 6, however, we train the initial model on the full training set instead of the reduced one. We then generate test adversaries on the initial model. These will later on be used to test the robustness of the adversarially trained models by applying the attacks to each of these models. We then deploy a cumulative iterative adversarial training loop, for the Higgs task, we run 10 iterations, and for the TT vs. WW task we run 5. Within each iteration, we generate train adversarial samples on the current runs initial model using a random initialization of the attack parameters, resulting in different setups for each iteration. After generating these train adversaries, we augment the initial training set cumulatively. This means after the second iteration, we would have a total training set containing the initial training set, the training adversaries generated in iteration 0, and those generated in iteration 1. After each iteration, we save the resulting train adversaries, as well as the resulting adversarially trained models. A flowchart depicting this approach is shown in Figure 8.

For the Higgs task, we can see the fooling ratio on the base model, as well as the fooling ratio on our cumulatively adversarially trained models in Figure 9.

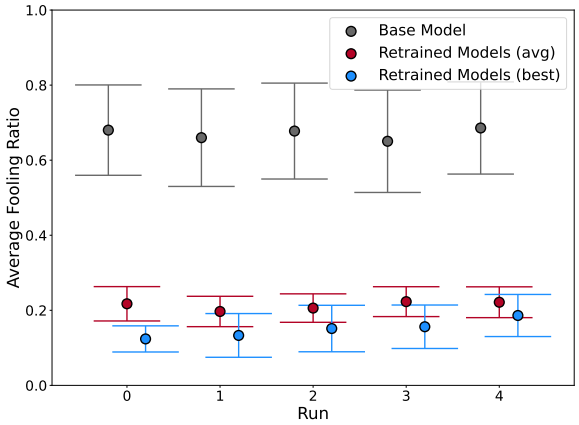

Fig. 9: Comparison of the fooling ratios of adversarial attacks on the baseline model (grey) vs the average-case adversarially trained network (red) and the best-case adversarially trained network (blue) on the Higgs dataset.

We see that—on average- the Grey-Box fooling ratio on the base model is around 70% across all runs. Importantly, here we also include the fooling ratio of the adversaries generated in the same run, which represents a White-Box attack, leading to a slight increase in the average fooling ratio. Additionally, we see that the adversarially trained models exhibit a significantly larger robustness against Grey-Box attacks—both for the average performance across the retraining iterations, as well as choosing the best iteration for each run. We manage to reduce the resulting fooling ratio to about 0.15 when always choosing the best iteration, and to about 0.2 when averaged over all iterations.

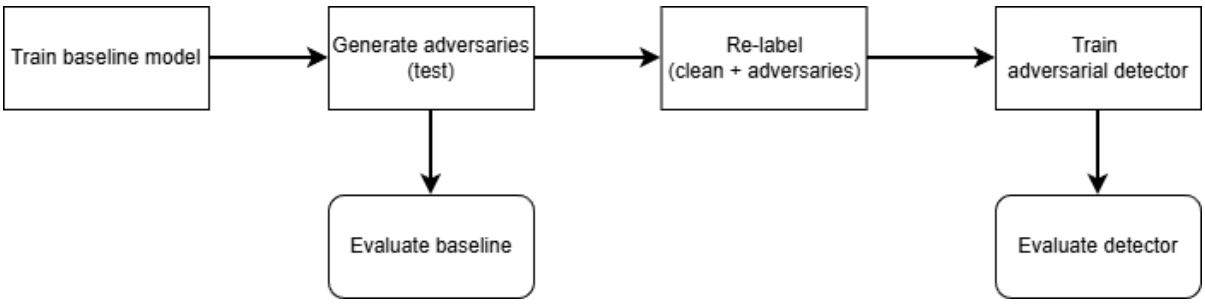

Fig. 10: Flowchart showing the workflow used in the adversarial detector pipeline.

For the Adversarial Detector defense, the pipeline differs slightly compared to the Adversarial Training. While we still train a baseline model with the clean events, instead of generating just adversaries for the train and test sets, here we generate them for all three sets, training, validation, and testing. Then, we re-label both our adversaries, and our clean events, where the adversaries get assigned the label 0, and the clean inputs get 1. Afterwards, the Adversarial Detector network is trained to classify between adversaries and clean events. Once trained, we pass the adversarial attacks generated on every other run to the current runs Adversarial Detector. We can then correct the fooling ratio by removing all adversaries that were correctly classified by our Detector. The flowchar for this pipeline is shown in Figure 10.

For the Higgs dataset, the results can be found in Figure 11.

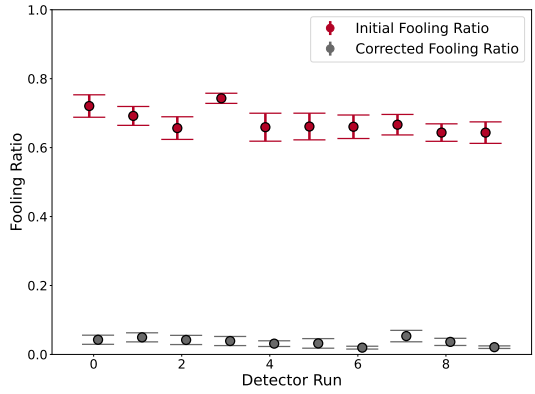

Fig. 11: Comparison of the initial fooling ratios (red) and the corrected fooling ratios (grey) found in 10 independent runs of the adversarial detector pipeline on the Higgs dataset.

There, we see that the corrected fooling ratio is significantly smaller than the initial. Additionally, this Adversarial Detector increases the robustness even more than Adversarial Training, pushing the fooling ratio down to about 0.05 to 0.08.

There are, however, distinct trade-offs for both cases. A drawback introduced—briefly introduced previously—about Adversarial Training is, especially in cases where the model has already seen enough data, that such training can decrease the models efficiency on the clean events.

While the same thing can in theory also happen for the Adversarial Detector approach, here it is much more unlikely. A decrease in clean efficiency would only occur if the Adversarial Detector is very bad at identifying clean inputs correctly, which would lead to filtering out a large amount of training data, consequently reducing the networks performance. However, in the studies here, the detectors were always significantly better than chance guessing at classifying between clean and adversarial events, leading to a negligible reduction in training size.

In both defenses, generating the required adversaries, as well as the training and fine-tuning of the respective models introduces computational overhead, therefore increasing the workload during the production pipeline. The Adversarial Detector also increases inference time, since an additional model query is required: inputs must first be passed through the Detector to filter out events classified as adversarial.

However, applying these techniques provides a better understanding of the worst-case systematic uncertainties of the underlying deep learning models.

8 Adversarial Detector: Systematic Misclassifications and Real-Data Evaluation

In addition to the robustness studies conducted on the Adversarial Detector discussed in the previous section, we further examine two aspects of the Adversarial Detector's behavior.

First, we would investigate whether the clean events that are misclassified as adversaries represent systematic effects or whether they can be attributed to statistical fluctuations within the network. To this end, we model the misclassification of clean events across multiple runs as independent Bernoulli trials. Let each event be evaluated over R independent runs. For run i, we estimate the empirical detector accuracy

$$\mathrm{acc}_i := \frac{N_{\mathrm{correct}} - N_{\mathrm{flagged}}}{N_{\mathrm{correct}}}, \tag{6}$$

where N_{correct} is the number of correctly classified samples and N_{flagged} is the number of detector-flagged misclassifications. Then, the average empirical accuracy over all runs is

$$\bar{a} := \frac{1}{R}\sum_{i=1}^{R} \mathrm{acc}_i, \tag{7}$$

and the per-run misclassification probability is defined as

$$p := 1 - \bar{a}. \tag{8}$$

Under the null hypothesis that misclassification events occur independently for each sample and run with probability p, the number of misclassifications for an event follows a binomial distribution. The probability that an event is misclassified in exactly k of the R runs is

$$P(K = k) := \binom{R}{k} p^k (1-p)^{R-k}. \tag{9}$$

Given N total events, the expected number of events misclassified exactly k times is

$$E_k := N \cdot P(K = k). \tag{10}$$

Let O_k denote the observed number of events that were misclassified exactly k out of R runs. For bins in which E_k is small, we approximate the binomial fluctuation by a Poisson distribution with rate parameter $\lambda = E_k$. The significance of observing at least O_k samples in bin k is assessed using the Poisson upper-tail probability

$$\begin{aligned} p_{\mathrm{val}}(k) &:= P(X \geq O_k \,|\, X \sim \mathrm{Poisson}(E_k)) \\ &= \sum_{x=O_k}^{\infty} \frac{E_k^x e^{-E_k}}{x!}. \end{aligned} \tag{11}$$

Bins with $p_{\mathrm{val}}(k) < 0.05$ are considered to show a statistically significant excess of repeated misclassifications, indicating that there are indeed some clean events exhibiting adversarial-like behavior and are consistently misidentified by the detector beyond what would be expected from random fluctuations. The corresponding plot obtained when applying this procedure to the TT vs. WW data is shown in Figure 12.

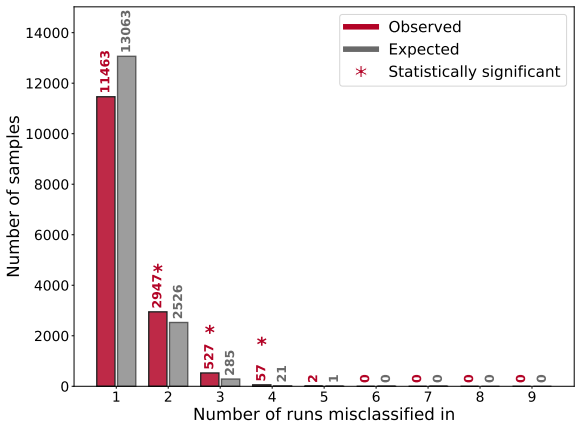

Fig. 12: Comparing the expected amounts of events misclassified across multiple runs (grey) to the observed amount (red). The red stars depict statistically significant differences between observed and expected. This plot was generated using the TT vs. WW dataset.

There, we observe that the number of clean inputs misclassified repeatedly across multiple runs—specifically in two, three, and four out of the ten runs—exceeds the expectation under the binomial-statistical model described above at a

significant level. Consequently, indicating that a subset of clean events systematically appears adversarial to the detector, rather than these repeated misclassifications being attributable merely to random fluctuations during training. In other words, the detector consistently struggles with certain clean events in a reproducible way, suggesting that they might share structural properties with genuine adversaries as perceived by the learned representation.

Next, we evaluate the performance of the adversarial detector on real data rather than on simulated events. For this study, we focus on the TT vs. WW task and use real collision data from the 2012 CMS Single Mu dataset [24]. These events are processes in the same way as the simulated samples, and are labeled as "clean" inputs for evaluation in the adversarial detector. Importantly, none of these real events were used during the training of the detector network.

To assess the detector's behavior under domain shift, we measure its clean-sample detection efficiency—the fraction of clean events classified as non-adversarial—for both real and simulated events across varying detector confidence thresholds, averaging the results over all detector runs. The corresponding comparison is shown in Figure 13.

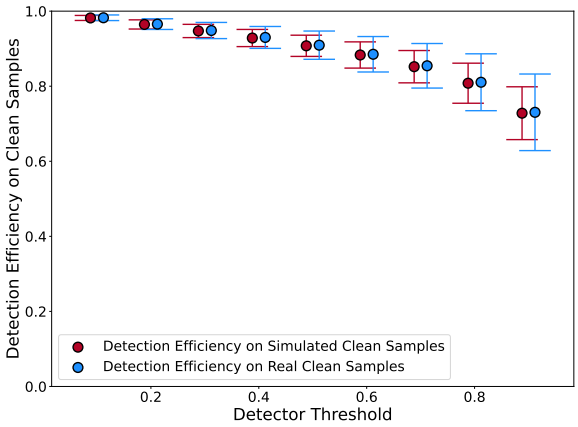

Fig. 13: Detector accuracies on both clean non-simulated clean events (blue) and simulated clean events(red) using an adversarial detector network trained solely on clean and adversarial simulated events, dependent on the classification threshold for the TT vs. WW dataset.

The adversarial detector manages to maintain a high confidence in correctly classifying clean events; even at high decision thresholds (greater than 0.8), the clean-event efficiency remains around 0.8. Notably, the detector performs almost equally well on real events, despite never having encountered them during training. While the mean detection efficiency is nearly identical to that of the simulated events, particularly at higher thresholds, the run-to-run variance increases, indicating a reduction in overall stability for real events.

These results suggest that there is no substantial domain gap between simulated and real clean events—at least not one that causes the detector to misidentify real events as adversaries. Consequently, the well-known degradation in model performance when applying models trained on simulated events to real data likely cannot be explained solely by real events acting as adversaries to the classifier, or vice-versa as simulated events poisoning the networks. Nonetheless, the detector does misclassify a fraction of real clean events as adversarial. Therefore, one cannot assume with certainty that all real clean events behave like their simulated counterparts, nor that simulated clean events are never adversarial relative to the model evaluated on real data.

These observation also suggest that the adversarial detector leverages higher-dimensional structure in the datasets, rather than solely on marginal distributions or correlations. Although the adversarial examples are explicitly constructed to remain indistinguishable from clean events under standard statistical comparisons, the detector still identifies them as anomalous. This implies that the model is sensitive to complex, non-linear features relationships that are not captured by traditional HEP sanity checks, and that clean events—whether real or simulated—occupy a region of the feature space that remains well separated from the adversarial events.

9 Donut: Illustrative Example

In this section, we provide an illustrative example demonstrating how the proposed algorithm constructs adversarial events. Furthermore, as previously highlighted, an adversarial detector—a deep learning network trained to classify between

clean and adversarial events—represents a promising approach to improving a network's robustness against adversarial attacks. We therefore further use this example to gain insight into how the adversarial detector operates.

To this end, we construct a toy dataset consisting of two input variables x_1 and x_2, and two target classes: signal and background. Signal events are sampled from a two-dimensional Gaussian distribution centered at the origin, where each coordinate (x_1, x_2) is drawn independently from a normal distribution with mean 0 and standard deviation σ, i.e., $x_1, x_2 \sim \mathcal{N}(0,\, \sigma^2)$.

Background events are sampled from a donut-shaped distribution, where the radius r is drawn from a normal distribution centered at r_{ring} with standard deviation σ, $r \sim \mathcal{N}(r_{\text{ring}},\, \sigma^2)$, and the angle θ is drawn uniformly from $[0, 2\pi)$, $\theta \sim \mathcal{U}(0, 2\pi)$. The corresponding Cartesian coordinates are given by $x_1 = r\cos\theta$ and $x_2 = r\sin\theta$. For both classes, the input variables are linearly uncorrelated.

The resulting two-dimensional distributions are shown in Figure 14.

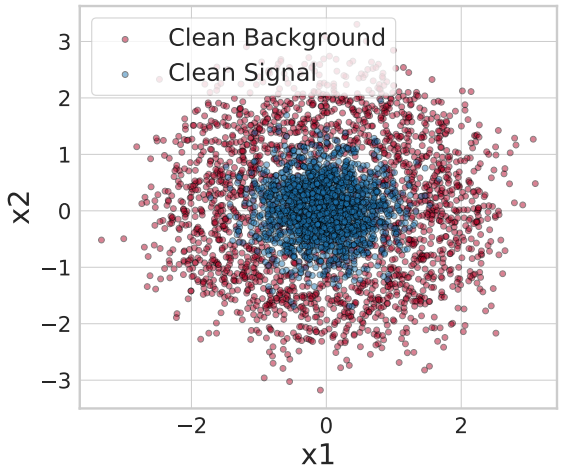

Fig. 14: Point clouds of sub-samples points from both the clean signal (blue) and clean background (red) events for the Donut toy dataset.

The signal distribution forms a circle in the center of the features space, which is surrounded by the donut-shaped background distribution, with a small overlap between the two classes.

To visualize both the attack mechanism and the behavior of the adversarial detector, we apply the same studies discussed in previous sections to this toy dataset. We first generate adversaries on the training and test sets, which are subsequently used to train the adversarial detector network. Here, adversaries are generated exclusively on background events.

Using this procedure, we achieve an initial fooling ratio of 0.354 on the signal-background classifier. After applying the adversarial detector, this value is reduced to a corrected fooling ratio of 0.153, corresponding to a detector efficiency of 0.568 on adversarial events. For clean events, the detector achieves a higher efficiency of 0.788.

Compared to the results presented earlier, both the initial fooling ratio and the detector efficiency on adversarial events are relatively low. We attribute this to the low dimensionality of the dataset: since the data are described by only two input variables, the attack can perturb only these same two features in order to induce misclassification. At the same time, the attack remains constrained to minimally alter the marginal distributions and correlations, making successful misclassification substantially more difficult and thereby limiting the achievable fooling ratio.

A similar limitation applies to the adversarial detector. With only two input variables, the decision boundary that separates clean and adversarial events must also be defined solely in this two-dimensional space, making effective separation challenging. As illustrated in the following, many adversarial events lie well within the signal region, such that correctly identifying them as adversaries would require misclassifying a significant fraction of clean signal events.

To illustrate how the attack modifies the underlying data, we present two complementary visualizations. First, we compare the one-dimensional marginal distributions of the clean background events to those of the adversarial background events. Second, we visualize the corresponding two-dimensional feature distributions as point clouds, allowing for a direct inspection of how the adversarial perturbations reshape the data in feature space.

The resulting marginal distributions are shown in Figure 15. For both clean and adversarial background events, the marginals differ noticeably

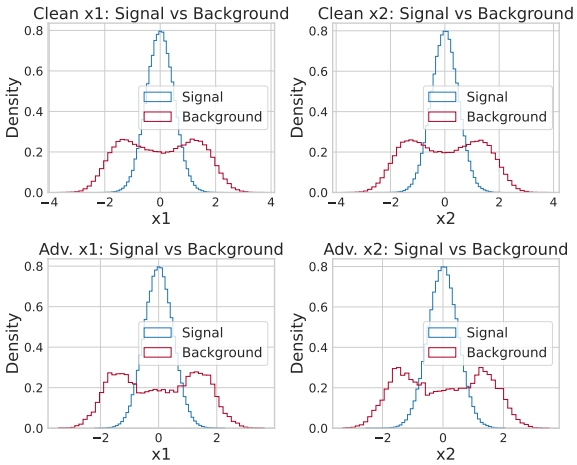

Fig. 15: Comparison of the feature distributions between clean signal vs. clean background events (top row) and clean signal vs. adversarial background events (bottom row).

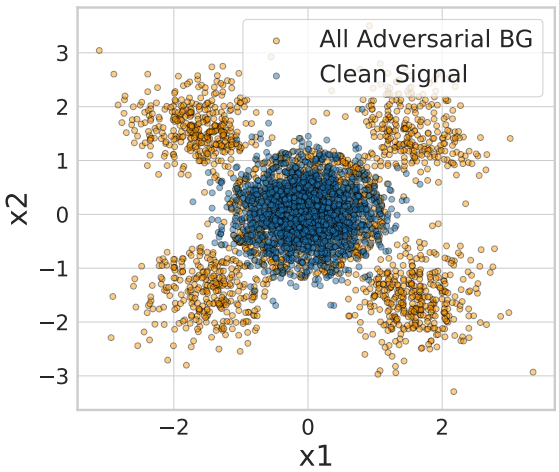

Fig. 16: Point clouds of sub-samples points from both the clean signal (blue) and adversarial background (yellow) events for the Donut toy dataset.

from those of the signal, as expected: the signal is distributed according to a two-dimensional Gaussian centered at the origin, leading to pronounced peaks at zero in both features, whereas the donut-shaped background has a depletion in the central region, resulting in a characteristic dip in the one-dimensional projections.

In this example, the differences between clean and adversarial background distributions are quite prominent. This can likely be attributed to the low dimensionality of the dataset: since the attack can only perturb two features, achieving misclassification requires comparatively larger modifications to each individual variable, as the perturbations cannot be distributed across a higher-dimensional feature space. Nevertheless, the adversarial background marginals preserve the qualitative structure of the original background distribution, maintaining a central depletion surrounded by two smaller peaks.

The two-dimensional point cloud comparing adversarial background events to clean signal events is shown in Figure 16. In this representation, the effect of the attack becomes substantially more apparent than in the one-dimensional marginals. A large fraction of background events is shifted into the central region of feature space that is populated by signal events, effectively overlapping with the signal distribution around the origin.

Notably, many of the background events that are displaced into the signal region originate from areas along the coordinate axes, as evidenced by the depletion of events along these axes outside the central region after the attack. Importantly, however, not all adversarial background events lead to a misclassification by the original classifier. Restricting the visualization to only those adversarial events that successfully fool the network yields the point cloud shown in Figure 17, where, as expected, the fooling adversaries are concentrated almost exclusively within the central signal region.

We now train an adversarial detector network to classify between adversarial background events and clean events, including both signal and background. Applying this detector to the clean signal events and the adversarial background events in the test set yields the point cloud shown in Figure 18.

This point cloud shows clean signal events that are correctly identified as such by the detector, together with adversarial background events that fool the original classifier—i.e., are classified as signal by the initial network—but are not recognized as being adversarial by the detector network. The visualization highlights the strategy employed by the adversarial detector: it learns a decision boundary that balances the correct classification of clean and adversarial events

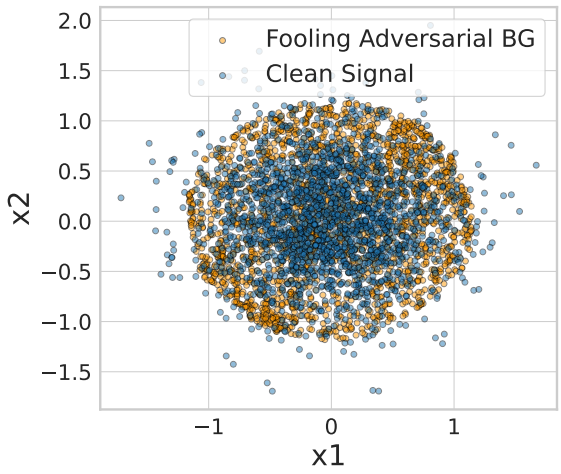

Fig. 17: Point clouds of sub-samples points from both the clean signal (blue) and adversarial background events that successfully fool the baseline network (yellow) for the Donut toy dataset.

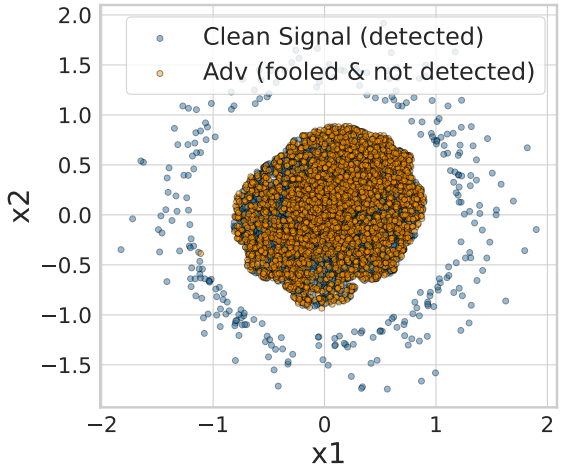

Fig. 18: Point clouds of sub-samples points from both the clean signal correctly classified by the adversarial detector network as such (blue), and adversarial background events that successfully fool the baseline network but are not correctly identified as such by the detector network (yellow) for the Donut toy dataset.

while minimizing the overall misclassification rate. While this approach is standard in deep learning, such decision boundaries are seldom directly visualizable in practice, as most datasets are very high-dimensional.

In this section, we leveraged a simple two-dimensional dataset with clearly separated signal and background classes to illustrate both the mechanism of the adversarial attack and the operation of the adversarial detector network. Although the datasets low dimensionality naturally limits the effectiveness of the attack and negatively impacts the detector's performance, it provides an opportunity to visualize the underlying concepts in a clear and intuitive way. By examining these toy distributions, we gain concrete insight into how adversarial perturbations shift events in feature space and how the detector learns to differentiate between clean and adversarial events, insights that are often difficult to extract from high-dimensional datasets.

10 Distance Correlation Studies

In the previous studies, we focused on preserving linear correlations between features by minimizing changes to the Pearson correlation matrices of clean and adversarially perturbed datasets. While this approach is well aligned with standard validation and sanity checks in high energy physics, it captures only linear dependencies and therefore does not fully reflect the complete statistical structure of the data. Non-linear correlations can also encode important information, both in physics and in other scientific domains.

To address this limitation, we extend the adversarial attack to preserve non-linear dependencies by replacing the Pearson correlation constraint with Distance Correlation [25]. We then study this modified adversarial construction within the same adversarial detector framework used in the previous sections, applying it to the Higgs dataset. The hyperparameter choices for the adversarial attack are summarized in the Appendix E.

Distance correlation is a statistical measure that detects any form of dependence between random variables, linear or non-linear. For two random vectors $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$, the distance correlation $\mathcal{R}(X, Y)$ is zero if and only if X and Y are independent. It is computed by first forming pairwise Euclidean distance matrices for

X and Y, double-centering these matrices (subtracting row and column means and adding the overall mean), and then calculating the covariance between the centered distances, normalized by the respective distance variances. Optimizing adversarial perturbations with respect to distance correlation ensures that both linear and nonlinear feature dependencies are preserved, maintaining the data's statistical structure.

In addition to being more precise at capturing both linear and non-linear feature dependencies, using the Distance Correlation also imposes an additional restrictive burden on the attack algorithm, making it harder to find adversaries compared to the Pearson correlation. Moreover, the computation of distance correlation itself is substantially more expensive.

Let n denote the number of events and d the number of features in the dataset. For our perturbations to the Pearson correlation matrix, we do not recompute the full matrix from scratch; instead, we employ an efficient incremental update procedure, described in the Appendix A.2, which exactly updates the means, variances, and covariances. This reduces the per-candidate computational cost from $\mathcal{O}(d^2 n)$ for a full Pearson recomputation to $\mathcal{O}(d)$. In contrast, evaluating a single candidate perturbation using distance correlation requires $\mathcal{O}(d^2 n^2)$ operations due to the need to compute pairwise distances between all events, and to our knowledge no comparable incremental update scheme exists. Consequently, the distance correlation must be fully recomputed for each candidate perturbation, resulting in a substantially higher computational cost.

Due to the substantially increased computational cost, we severely limit the number of adversarial events generated for the distance correlation study. Specifically, we generate a total of 2500 adversarial events for training and 1000 adversarial events for testing. To maintain a balanced training setup, we also restrict the number of clean events used to train the adversarial detector to the same size.

As this represents a rather drastic reduction in available statistics, and in order to isolate the impact of using distance correlation rather than Pearson correlation, we additionally perform a reference study using Pearson correlations with an equally limited number of adversarial events. This allows us to disentangle effects arising from the correlation metric itself from those induced by the reduced sample size. The detailed configuration of the adversarial algorithm for this comparison study is provided in the Appendix E.

The resulting initial fooling ratios, as well as the corrected fooling ratios obtained after applying the adversarial detector pipeline to these adversaries, are shown in Figure 19.

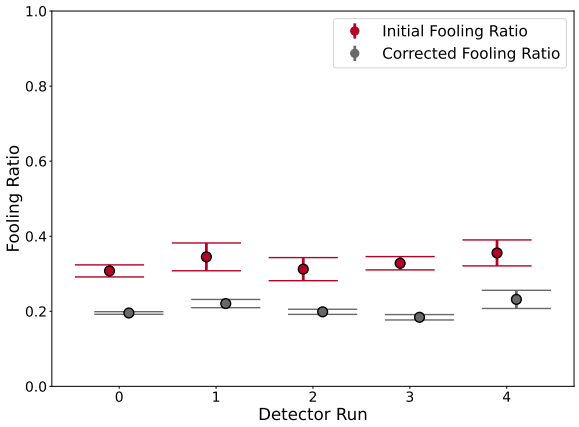

Fig. 19: Comparison of the initial fooling ratios (red) and the corrected fooling ratios (grey) found in 5 independent runs of the Distance correlation restricted adversarial detector pipeline on the Higgs dataset.

As shown there, the initial fooling ratios are—unsurprisingly—significantly lower than those obtained when generating an unrestricted number of adversaries on the same Higgs dataset using the Pearson correlation. Conversely, the corrected fooling ratios after applying the detector network are considerably higher than in the previous Higgs studies. To gain insight into whether the reduced initial fooling ratio and the increased corrected fooling ratio arise from constraining the adversaries using distance correlation, or instead from the limited number of adversarial samples, we additionally present results from a Pearson-correlation-based attack with the same restricted sample size in Figure 20.

From this comparison, we observe that the initial fooling ratios remain significantly lower than those obtained from a Pearson-based attack using a larger adversarial sample size, and that the corrected fooling ratios—consistent with the

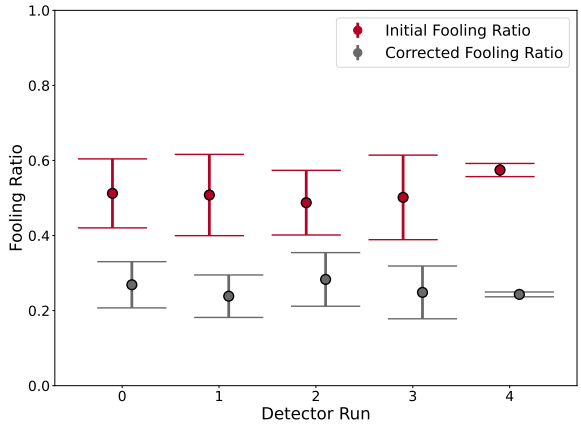

Fig. 20: Comparison of the initial fooling ratios (red) and the corrected fooling ratios (grey) found in 5 independent runs of the limited data Pearson correlation restricted adversarial detector pipeline on the Higgs dataset.

distance-correlation-based approach—are higher. When directly comparing the limited-sample Pearson and distance-correlation studies, both the initial and corrected fooling ratios are higher for the Pearson case: with the difference being substantial for the initial fooling ratio, while it is comparatively small for the corrected fooling ratio.

Focusing for the moment solely on the initial fooling ratios, the observation that distance-correlation-based adversaries achieve noticeably lower fooling ratios is consistent with the more restrictive nature of this constraint. Put simply, it is substantially harder to construct adversarial examples that satisfy the stricter distance correlation requirements than those based only on Pearson correlations.

We further attribute the reduction in initial fooling ratios observed in the limited-sample regime to the statistically grounded nature of the attack. For larger datasets, perturbing individual events has a comparatively small impact on global statistical properties, such as marginal distributions and both linear and non-linear feature correlations. When the dataset size is reduced significantly, however, each individual perturbation has a much larger influence on these quantities, making it inherently more difficult to generate adversarial events that remain statistically consistent.

An even more pronounced effect is the substantially reduced gap between the initial and corrected fooling ratios when the dataset size is limited, an effect that is observed consistently for both the distance correlation and Pearson correlation studies. While the impact of reduced data size on the initial fooling ratios has already been discussed, we now provide insight into why the corrected fooling ratios are in fact higher than those obtained with larger training samples.

To this end, Figure 21 shows the area under the receiver operating characteristic curve (AUROC) of the adversarial detector—trained on Pearson-correlation-based adversaries—as a function of the total number of training samples. In all cases, the detector training sets are constructed using a 50/50 split between clean and adversarial events.

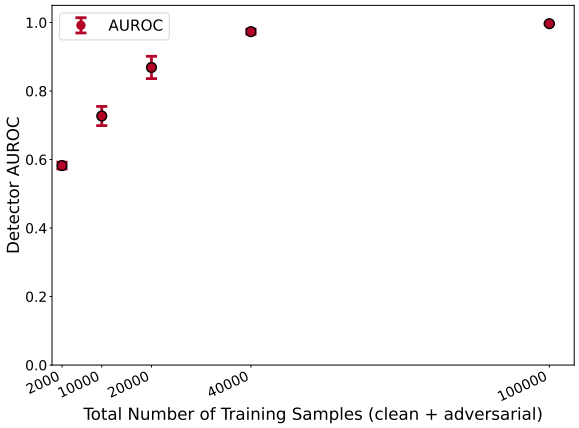

Fig. 21: Area under the Receiver-Operating-Curve for varying sample sizes of training events used for the Pearson correlation restricted adversarial detector network.

There, we see that the adversarial detector's performance improves noticeably for both clean and adversarial samples as the total number of training examples increases. This observation explains why the corrected fooling ratios in the low-sample-size studies are substantially higher than those obtained with larger datasets: the adversarial detector has not yet reached performance saturation when trained on a severely limited number of samples.

Comparing adversarial detectors trained on distance-correlation-constrained and Pearson-correlation-constrained adversaries reveals

virtually no difference in overall performance, as shown in Figure 22.

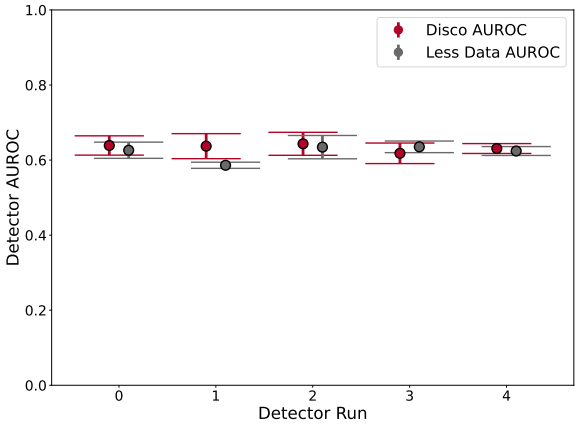

Fig. 22: Comparison of the averageg Areas under the Receiver-Operating-Curve for the distance correlation restricted attack (red) and the limited data Pearson restricted attack (grey) across 5 adversarial detector runs.

This indicates that the adversarial detector does not struggle to identify distance-correlation-constrained adversaries, despite their more restrictive construction, which could in principle have made them harder to detect. A slightly larger gap between the initial and corrected fooling ratios is indeed observed for the Pearson-based adversaries, suggesting a higher adversarial classification accuracy for the detector trained on Pearson perturbations compared to the distance correlation case.

This improved adversarial detection performance, however, comes at the cost of reduced classification accuracy on clean events relative to the distance-correlation-trained detector. We attribute this behavior to the higher initial fooling ratios achieved by Pearson adversaries. Since the detector is trained exclusively on adversarial samples that successfully fool the baseline model, a higher fooling ratio leads to a larger effective representation of adversarial events in the detector's training set. This imbalance likely biases the detector slightly toward classifying ambiguous inputs as adversarial.

11 Toolkit / Workflow

In light of the potential vulnerabilities demonstrated throughout this work, we propose a practical workflow for incorporating adversarial analyses into studies that rely on simulated HEP data. The central idea is to establish a systematic procedure that not only quantifies the susceptibility of a given model to undetectable adversarial perturbations, but also provides a means of mitigating their impact and estimating the associated uncertainty.

The workflow begins by training a baseline model on clean simulated events, following the standard approach used in a typical HEP analysis. This model can then be used to generate adversarial examples—using the attack proposed in this work—on the training, validation, and test datasets. The clean and adversarial samples are then combined into an augmented dataset and assigned corresponding labels indicating whether they are genuinely clean or adversarially perturbed.

Using this augmented dataset, an Adversarial Detector network—implemented, for instance, as a simple MLP—is trained to distinguish between clean and adversarial events. As shown in our studies, such a detector is capable of learning subtle properties beyond classical low-dimensional statistics, and therefore provides a valuable diagnostic tool for identifying events that behave adversarially with respect to a given predictive model.

After training the detector, one can analyze both the clean and adversarial events in the same fashion as in our experiments. In particular, the fooling ratio of the adversarial events with respect to the baseline model can be compared before and after removing events that the detector identifies as adversarial. This allows a quantification of how much of the model's susceptibility to adversarial examples can be mitigated—and consequently how the estimated upper-bound on the model's systematic uncertainty can be reduced—through the use of the detector.

A flowchart showing this workflow is shown in Figure 23.

To support adoption of this workflow, we provide an accompanying GitHub repository containing a working example of the proposed attack, along with a reference implementation of the full pipeline.

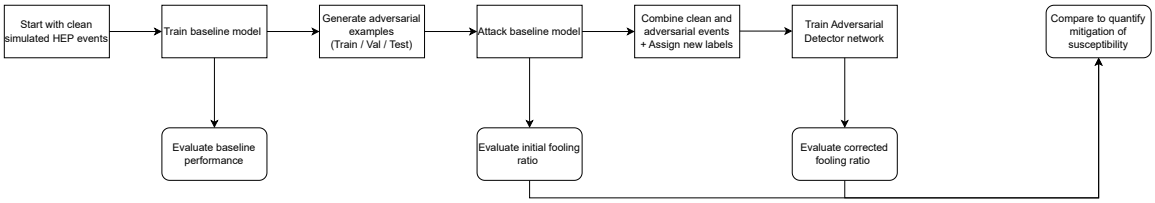

Fig. 23: Flowchart showing the proposed workflow.

12 Limitations and Outlook

In the context of robustness studies, there are numerous promising vectors for further investigation, many of which could improve the estimation of the upper bounds on systematic uncertainties associated with HEP deep learning networks. Furthermore, the adversarial examples constructed in this work open up opportunities to apply—or to develop—methods from explainable AI. Such tools could not only help to better understand why our networks behave as they do, but may also provide insight into potential shortcomings or mismodellings in current HEP simulation pipelines. We believe that a systematic exploration of adversarial directions in feature space could eventually serve as a diagnostic instrument for both ML models and event generators.

Additionally, we believe that the attack proposed could be optimized further. From a computational perspective, more efficient approximations or optimization heuristics may substantially reduce runtime without sacrificing attack performance. Beyond computational efficiency, there is likely room for hyperparameter optimization for each dataset studied here, possibly resulting in even stronger attacks while achieving an even more precise preservation of marginal distributions and their correlations.

A clear limitation of this approach is its reliance on sufficiently large statistics to reliably match correlations and distributions. In low-statistics regimes, the optimization becomes unstable and the attack less well-defined. Moreover, the method as constructed requires continuous or pseudo-continuous features; datasets dominated by categorical variables are not natural candidates for this approach.

13 Conclusions

In this work, we introduce the novel adversarial attack CONSERVAttack designed for deep learning models in High Energy Physics. The attack is optimized to produce inputs that induce erroneous model predictions while keeping perturbations to two key statistical properties of the data—marginal distributions and inter-feature correlations—within their respective uncertainty bounds. This design renders the resulting adversarial events virtually invisible to traditional data validation techniques applied in simulation-data comparisons.

We apply this attack to two common HEP tasks—jet tagging and a signal-background classification problem—demonstrating its ability to achieve high fooling ratios with minimal statistical distortion. Beyond its use as an attack, we show that these events can be repurposed as a data-augmentation tool, improving model performance in low-data scenarios for both tasks.

Conversely, with respect to adversarial defenses, we explore two complementary robustness strategies. First, we perform Adversarial Training using attacks generated on the training set. Second, we introduce an Adversarial Detector: a simple binary classification model capable of distinguishing clean from adversarially perturbed samples. Both defenses, when applied correctly, significantly mitigate susceptibility to our attack and thereby reduce the upper bound on model-induced systematic uncertainty.

To further investigate the robustness of the attack's statistical preservation, we extend CONSERVAttack beyond linear correlation constraints by replacing the Pearson correlation requirement with Distance Correlation, thereby enforcing the preservation of both linear and non-linear feature dependencies. As expected, the stronger

constraint makes the construction of adversarial events substantially more challenging, leading to lower fooling ratios. However, the adversarial detector's performance remains largely unaffected, indicating that it can still reliably distinguish these more tightly constrained adversaries from clean events. This suggests that the effectiveness of the Pearson-constrained attack cannot be attributed solely to the exploitation of non-linear correlations, but instead reflects a broader vulnerability of the model under statistically consistent perturbations.

Finally, we leverage the Adversarial Detector to conduct further experiments, attempting to understand whether there exist simulated events exhibiting "pseudo-adversarial" behavior, and how the performance of the detector generalizes to clean real data. These studies reveal that while most clean events behave consistently across multiple detector runs, a statistically significant subset mimics adversarial characteristics. Furthermore, the detector's performance transfers remarkably well to real events.

We propose a workflow to evaluate the vulnerability of machine learning applications in HEP and beyond to unknown effects that may lead to adversarial attacks. We propose that if the corrected fooling ratio, after applying an adversarial detector, lies within the systematic uncertainties bounds derived from physical sources, and if the fraction of adversarial samples detected in data is consistent with that observed in simulation, then no additional uncertainty needs to be assigned to potential adversarial effects. However, if the above conditions are not satisfied, we propose that further investigations should be undertaken to identify potential unaccounted sources of discrepancies between simulation and data. If such studies do not reveal physical origins for the adversarial behavior, the assignment of additional uncertainties should be considered.

Acknowledgement

This work has been funded in the context of the AISafety Project, funded by the BMBF under the grant proposal 05D23UM1 and supported in the context of the Clusters of Excellence EXC 3037/533607693 "Dynaverse" and EXC 3107/533766364 "Color meets Flavor" under Germany's Excellence Strategy. We thank Markus Prim for very helpful discussions in the context of the example in Section 9. Our work would have been impossible without the services provided by the CERN open data initiative `openscience.cern` and without the contributions from the LHC experiments.

References

[1] Miotto, R., Wang, F., Wang, S., Jiang, X., Dudley, J.T.: Deep learning for healthcare: review, opportunities and challenges. Briefings in Bioinformatics **19**(6), 1236–1246 (2017) https://doi.org/10.1093/bib/bbx044 https://academic.oup.com/bib/article-pdf/19/6/1236/27119191/bbx044.pdf

[2] Chib, P.S., Singh, P.: Recent advancements in end-to-end autonomous driving using deep learning: A survey. IEEE Transactions on Intelligent Vehicles **9**(1), 103–118 (2024) https://doi.org/10.1109/TIV.2023.3318070

[3] Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., Wen, J.-R.: A Survey of Large Language Models (2025). https://arxiv.org/abs/2303.18223

[4] Cowan, G., Cranmer, K., Gross, E., Vitells, O.: Asymptotic formulae for likelihood-based tests of new physics. Eur. Phys. J. C **71**, 1554 (2011) https://doi.org/10.1140/epjc/s10052-011-1554-0 arXiv:1007.1727 [physics.data-an]. [Erratum: Eur.Phys.J.C 73, 2501 (2013)]

[5] Aad, G., *et al.*: Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. Phys. Lett. B **716**, 1–29 (2012) https://doi.org/10.1016/j.physletb.2012.08.020 arXiv:1207.7214 [hep-ex]

[6] Chatrchyan, S., *et al.*: Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC. Phys. Lett. B **716**, 30–61 (2012) https://doi.org/10.1016/

j.physletb.2012.08.021 arXiv:1207.7235 [hep-ex]

[7] Costa, J.C., Roxo, T., Proença, H., Inácio, P.R.M.: How deep learning sees the world: A survey on adversarial attacks & defenses. IEEE Access **12**, 61113–61136 (2024) https://doi.org/10.1109/ACCESS.2024.3395118

[8] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples (2015). https://arxiv.org/abs/1412.6572

[9] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards Deep Learning Models Resistant to Adversarial Attacks (2019). https://arxiv.org/abs/1706.06083

[10] Cartella, F., Anunciacao, O., Funabiki, Y., Yamaguchi, D., Akishita, T., Elshocht, O.: Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data (2021). https://arxiv.org/abs/2101.08030

[11] Zhang, W.E., Sheng, Q.Z., Alhazmi, A., Li, C.: Adversarial Attacks on Deep Learning Models in Natural Language Processing: A Survey (2019). https://arxiv.org/abs/1901.06796

[12] Flek, L., Jung, P.A., Karimi, A., Saala, T., Schmidt, A., Schott, M., Soldin, P., Wiebusch, C.H.: Enforcing fundamental relations via adversarial attacks on input parameter correlations. Computing and Software for Big Science **9**(1), 19 (2025) https://doi.org/10.1007/s41781-025-00148-1

[13] Flek, L., Janik, O., Jung, P.A., Karimi, A., Saala, T., Schmidt, A., Schott, M., Soldin, P., Thiesmeyer, M., Wiebusch, C., Willemsen, U.: MiniFool – Physics-Constraint-Aware Minimizer-Based Adversarial Attacks in Deep Neural Networks (2025). https://arxiv.org/abs/2511.01352

[14] Rahimian, H., Mehrotra, S.: Frameworks and results in distributionally robust optimization. Open Journal of Mathematical Optimization **3**, 1–85 (2022) https://doi.org/10.5802/ojmo.15

[15] Kegl, B., CecileGermain, ChallengeAdmin, ClaireAdam, Rousseau, D., Djabbz, fradav, Cowan, G., Isabelle, joycenv: Higgs Boson Machine Learning Challenge. https://kaggle.com/competitions/higgs-boson. Kaggle (2014)

[16] CERN: CERN Open Data Portal (2026). https://opendata.cern.ch/ Accessed 2026-02-20

[17] CMS collaboration: Simulated dataset TTJets_FullLeptMGDecays_TuneP11TeV_8TeV-madgraph-tauola in AODSIM format for 2012 collision data. CERN Open Data Portal (2017). https://doi.org/10.7483/OPENDATA.CMS.7RZ3.0BXP

[18] CMS collaboration: Simulated dataset WWJetsTo2L2Nu_TuneZ2star_8TeV-madgraph-tauola in AODSIM format for 2012 collision data. CERN Open Data Portal (2017). https://doi.org/10.7483/OPENDATA.CMS.V2C6.O1P4

[19] Kasieczka, G., Plehn, T., Butter, A., Cranmer, K., Debnath, D., Dillon, B.M., Fairbairn, M., Faroughy, D.A., Fedorko, W., Gay, C., Gouskos, L., Kamenik, J.F., Komiske, P.T., Leiss, S., Lister, A., Macaluso, S., Metodiev, E.M., Moore, L., Nachman, B., Nordström, K., Pearkes, J., Qu, H., Rath, Y., Rieger, M., Shih, D., Thompson, J.M., Varma, S.: The Machine Learning landscape of top taggers. SciPost Phys. **7**, 014 (2019) https://doi.org/10.21468/SciPostPhys.7.1.014

[20] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=SyxAb30cY7

[21] Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On Detecting Adversarial Perturbations (2017). https://arxiv.org/abs/1702.04267

[22] Akhtar, N., Mian, A., Kardan, N., Shah, M.: Advances in adversarial attacks and defenses in computer vision: A survey (2021). https://arxiv.org/abs/2108.00401

[23] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical Black-Box Attacks against Machine Learning (2017). https://arxiv.org/abs/1602.02697

[24] CMS collaboration: SingleMu primary dataset in AOD format from Run of 2012. CERN Open Data Portal (2017). https://doi.org/10.7483/OPENDATA.CMS.9A4E.7SIR

[25] Székely, G.J., Rizzo, M.L., Bakirov, N.K.: Measuring and testing dependence by correlation of distances. The Annals of Statistics **35**(6) (2007) https://doi.org/10.1214/009053607000000505

Appendix A Attack Optimization

A.1 Jensen Shannon Distance Calculation

When evaluating the effect of small perturbations on feature histograms (as required for Jensen–Shannon distance calculations), it is not necessary to recompute the entire histogram from scratch for each candidate change. Let n denote the number of events and K the number of histogram bins for a given feature. If a single entry x_{ij} in the dataset is modified, only the bin counts corresponding to the old and new values are affected, while all other bins remain unchanged. Specifically, the histogram can be updated exactly by decrementing the count in the bin containing the old value and incrementing the count in the bin containing the new value. This procedure is mathematically exact because the histogram is simply a count of occurrences in each bin, and a single change only affects the relevant bins.

From a computational perspective, this incremental approach reduces the per-candidate complexity from $O(n)$—required to recompute the histogram over all n samples—to $O(1)$, since only two bins are updated per candidate. This makes it highly efficient for evaluating large numbers of candidate changes, especially in high-dimensional or large-sample settings.

$$k_{\text{old}} := \text{bin}\left(x_{ij}^{\text{old}}\right), \tag{A1}$$

$$k_{\text{new}} := \text{bin}\left(x_{ij}^{\text{new}}\right), \tag{A2}$$

$$h_k^{\text{new}} := \begin{cases} h_k^{\text{old}} - \frac{1}{n}, & \text{if } k = k_{\text{old}}, \\ h_k^{\text{old}} + \frac{1}{n}, & \text{if } k = k_{\text{new}}, \\ h_k^{\text{old}}, & \text{otherwise.} \end{cases} \tag{A3}$$

where h_k denotes the normalized count in bin k. This update can be efficiently vectorized for all features and candidate changes, enabling rapid batch evaluation of Jensen–Shannon distances on the GPU.

A.2 Pearson Correlation Matrix Calculation

When evaluating the effect of small perturbations on the Pearson correlation matrix, it is not necessary to recompute the entire matrix from scratch for each candidate change. Let n denote the number of events and d the number of features in the dataset. If a single entry in the dataset is modified, the corresponding feature mean can be updated directly, followed by an update of the feature variance, and finally an update of the covariance between the modified feature and all other features. By introducing intermediate variables representing deviations from the old and new means, the covariance update can be expressed compactly and exactly as the difference of products of these deviations. The final correlation entries are then obtained by normalizing the updated covariances by the updated standard deviations. This procedure is mathematically exact because the Pearson correlation depends solely on the means, variances, and covariances of the features, all of which are correctly adjusted. From a computational perspective, this incremental approach reduces the per-candidate complexity from $O(d^2 n)$—required to recompute all pairwise covariances from scratch—to $O(d)$, making it highly efficient for evaluating large numbers of small changes.

$$\Delta_j := x_{\text{new}} - x_{\text{old}},$$

$$
\begin{aligned}
\delta_j^{\text{old}} &:= x_{\text{old}} - \mu_j^{\text{old}}, \quad \delta_j^{\text{new}} := x_{\text{new}} - \mu_j^{\text{new}}, \\
\delta_k^{\text{old}} &:= x_{ik} - \mu_k^{\text{old}}, \quad \delta_k^{\text{new}} := x_{ik} - \mu_k^{\text{new}}, \\
\mu_j^{\text{new}} &:= \mu_j^{\text{old}} + \frac{\Delta_j}{n}, \\
\sigma_j^{2,\text{new}} &:= \sigma_j^{2,\text{old}} + \frac{(\delta_j^{\text{new}})^2 - (\delta_j^{\text{old}})^2}{n-1}, \\
\text{cov}_{jk}^{\text{new}} &:= \text{cov}_{jk}^{\text{old}} + \frac{\delta_j^{\text{new}}\,\delta_k^{\text{new}} - \delta_j^{\text{old}}\,\delta_k^{\text{old}}}{n-1}, \\
\rho_{jk}^{\text{new}} &:= \frac{\text{cov}_{jk}^{\text{new}}}{\sigma_j^{\text{new}}\,\sigma_k^{\text{new}}}.
\end{aligned}
\tag{A4}
$$

Appendix B Control Region Attack

In this section, we consider a physically motivated variant of the CONSERVAttack. To define the control and signal regions, we identify the most discriminating feature by performing an exhaustive search over all input variables for the optimal single-variable decision boundary, i.e., the feature and threshold that maximize classification accuracy using a single cut. This procedure selects the reconstructed invariant mass from the Missing Mass Calculator (`DER_mass_MMC`) as the best separator. The control region is defined as the subset of events below this optimal threshold, corresponding to a background-enriched phase-space region where the distributions and correlations are well understood. The signal region comprises all remaining events above the threshold.

In this variant, the marginal distributions and linear correlations are constrained only in the control region. In the signal region—where the corresponding distributions and correlations are typically unknown—large, unconstrained perturbations are allowed.

The effectiveness of the attack is evaluated as follows. For the Fooling Ratio, we consider three subsets: the control region, the signal region, and the full dataset. For D_{JS} and Δ_F, we restrict the comparison to the control region before and after the attack, as this is the only region in which statistical consistency can be reliably validated.

The results are shown in Figures B1, B2, and B3.

For the Fooling Ratio, we observe values close to 1 in the signal (unrestricted) region, as expected given that large perturbations are permitted. In contrast, the Fooling Ratio in the control (restricted) region is noticeably lower, reflecting the constraints imposed on the marginal distributions and linear correlations. Nevertheless, when evaluated over the full dataset, the combined attack achieves a Fooling Ratio of nearly 0.8 across all runs.

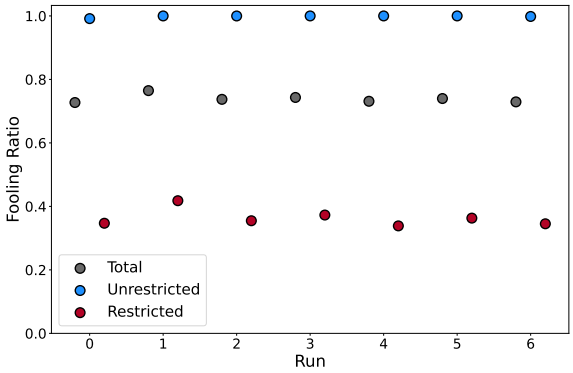

Fig. B1: Fooling ratios of the restricted attack on the Higgs dataset over 10 independent runs. The grey points indicate the total fooling ratio across all events, the blue points correspond to the signal (unrestricted) region, and the red points to the control (restricted) region.

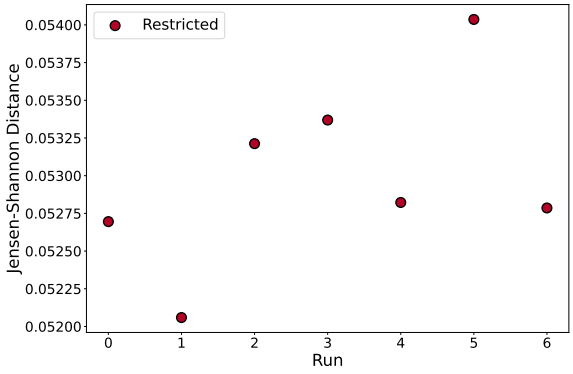

Fig. B2: Average Jensen–Shannon distance of the adversarial attack evaluated on the control (restricted) region of the Higgs dataset over 10 runs.

For D_{JS} and Δ_F—evaluated exclusively in the control region—we obtain small values (approximately 0.05 for D_{JS} and 0.09 for Δ_F). Consistent with the previous attack setting, this indicates that the perturbations introduced in the constrained region remain effectively undetectable

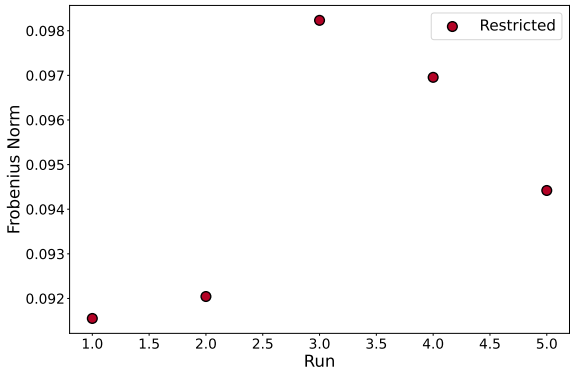

Fig. B3: Average Frobenius norm of the adversarial attack evaluated on the control (restricted) region of the Higgs dataset over 10 runs.

with respect to low-dimensional statistical properties. In particular, this suggests that such an attack would likely pass standard validation checks performed in the control region.

Appendix C Datasets and Networks

C.1 Higgs

As input for the Multi Layer Perceptron used to tackle the Higgs signal–background classification task popularized by the corresponding Kaggle challenge, we use the following 30 features: `DER_mass_MMC`, `DER_mass_transverse_met_lep`, `DER_mass_vis`, `DER_pt_h`, `DER_deltaeta_jet_jet`, `DER_mass_jet_jet`, `DER_prodeta_jet_jet`, `DER_deltar_tau_lep`, `DER_pt_tot`, `DER_sum_pt`, `DER_pt_ratio_lep_tau`, `DER_met_phi_centrality`, `DER_lep_eta_centrality`, `PRI_tau_pt`, `PRI_tau_eta`, `PRI_tau_phi`, `PRI_lep_pt`, `PRI_lep_eta`, `PRI_lep_phi`, `PRI_met`, `PRI_met_phi`, `PRI_met_sumet`, `PRI_jet_num`, `PRI_jet_leading_pt`, `PRI_jet_leading_eta`, `PRI_jet_leading_phi`, `PRI_jet_subleading_pt`, `PRI_jet_subleading_eta`, `PRI_jet_subleading_phi`, and `PRI_jet_all_pt`. Features prefixed with `PRI` are primitive raw quantities describing the bunch collision as measured directly by the detector, while features prefixed with `DER` are quantities derived from the primitive features.

The hyperparameter setup for the baseline classifier is shown in Table C1 and the adversarial detector setup in Table C2.

Table C1: Architecture of the baseline signal-background classifier for the Higgs dataset.

Higgs - Signal-Background Classifier		
Layer	**Nodes**	**Activation Function**
Input Layer	30	-
Dense	300	ReLU
BatchNormalization	-	-
Dense	102	ReLU
BatchNormalization	-	-
Dense	12	ReLU
BatchNormalization	-	-
Dense	6	ReLU
BatchNormalization	-	-
Dense (Output)	1	Sigmoid

Table C2: Architecture of the adversarial detector MLP for the Higgs dataset.

Higgs - Adversarial Detector Model		
Layer	**Nodes**	**Activation Function**
Input Layer	30	-
Dense	300	ReLU
BatchNormalization	-	-
Dense	102	ReLU
BatchNormalization	-	-
Dense	12	ReLU
BatchNormalization	-	-
Dense	6	ReLU
BatchNormalization	-	-
Dense (Output)	1	Sigmoid

C.2 TT vs. WW Jets

As input for this TopoDNN inspired model on the TT vs. WW Jet classification task, we take p_T, η, and ϕ of the first 30 jet constituents, sorted by their momentum. However, we leave out η and ϕ of the first constituent, as well as η of the second constituent, as these take always the same value due to the pre-processing applied, resulting in a total of 87 input features. A more detailed overview of

the variables can be found in the original paper [19].

The hyperparameter setup for the baseline classifier is shown in Table C3 and the adversarial detector setup in Table C4.

Table C3: Architecture of the baseline signal-background classifier for the TT vs. WW Jets dataset.

TT vs. WW - Jet Tagger		
Layer	**Nodes**	**Activation Function**
Input Layer	87	-
Dense	300	ReLU
BatchNormalization	-	-
Dense	102	ReLU
BatchNormalization	-	-
Dense	12	ReLU
BatchNormalization	-	-
Dense	6	ReLU
BatchNormalization	-	-
Dense (Output)	1	Sigmoid

Table C4: Architecture of the adversarial detector MLP for the TT vs. WW Jets dataset.

TT vs. WW - Adversarial Detector Model		
Layer	**Nodes**	**Activation Function**
Input Layer	87	-
Dense	300	ReLU
BatchNormalization	-	-
Dense	102	ReLU
BatchNormalization	-	-
Dense	12	ReLU
BatchNormalization	-	-
Dense	6	ReLU
BatchNormalization	-	-
Dense (Output)	1	Sigmoid

Appendix D Additional Result Plots

D.1 Attack Results (TT vs. WW Jets)

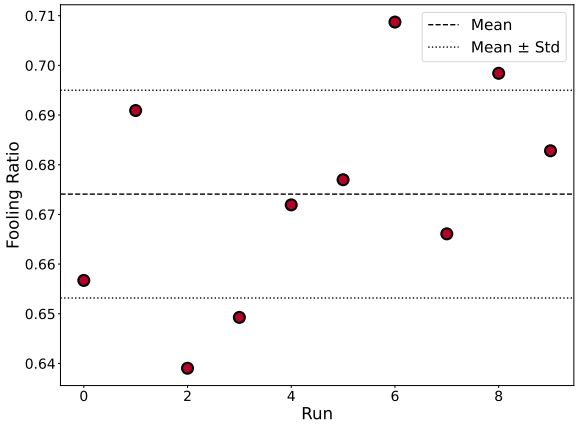

Fig. D4: Average fooling ratio of the adversarial attack on the TT vs. WW dataset over 10 runs. The average performance of the attack is shown as a dashed line, and its standard deviation as a dotted line.

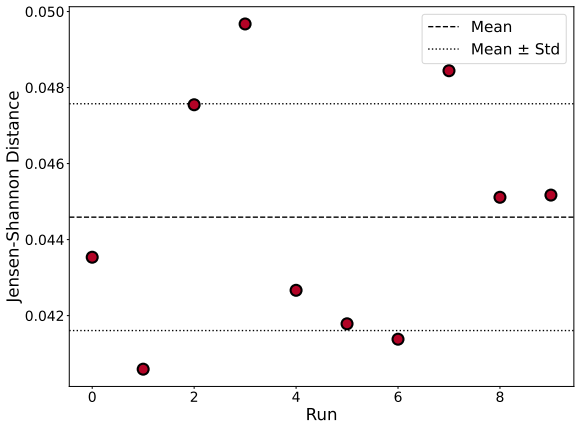

Fig. D5: Average Jensen Shannon distance of the adversarial attack on the TT vs. WW dataset over 10 runs. The average performance of the attack is shown as a dashed line, and its standard deviation as a dotted line.

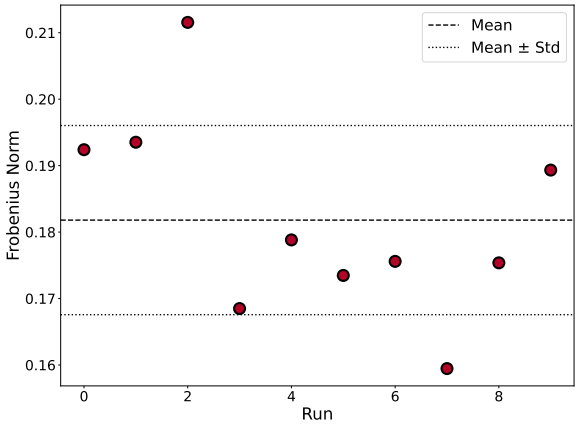

Fig. D6: Average Frobenius Norm of the adversarial attack on the TT vs. WW dataset over 10 runs. The average performance of the attack is shown as a dashed line, and its standard deviation as a dotted line.

D.2 Augmentation Results (TT vs. WW Jets)

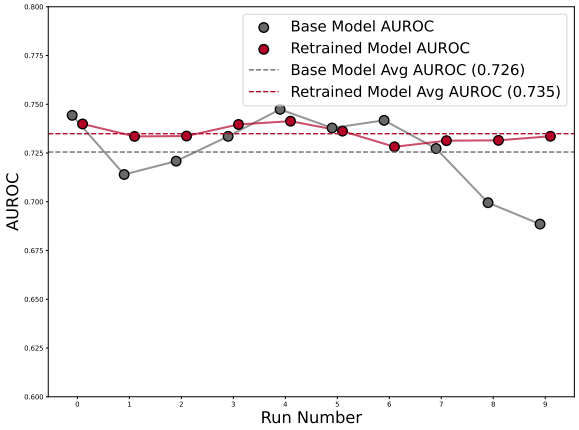

Fig. D7: Comparison of the areas under the Receiver-Operating-Curves—on the base model (grey) and the adversarially augmented model (red)—on the Higgs data set over 10 independent runs. The respective average results are shown in dashed lines.

D.3 Robustness Results (TT vs. WW Jets)

D.3.1 Adversarial Training

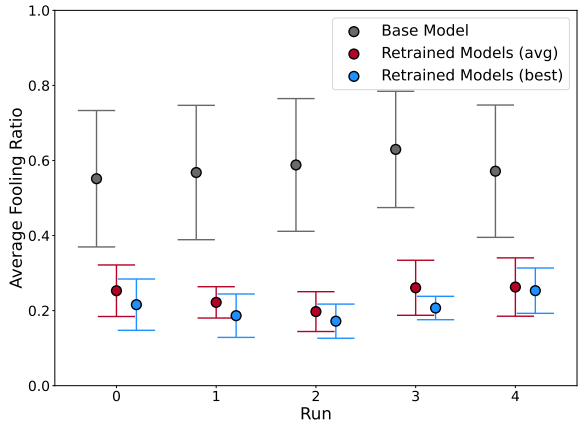

Fig. D8: Comparison of the fooling ratios of adversarial attacks on the baseline model (grey) vs the average-case adversarially trained network (red) and the best-case adversarially trained network (blue) on the TT vs. WW dataset across 5 runs.

D.3.2 Adversarial Detector

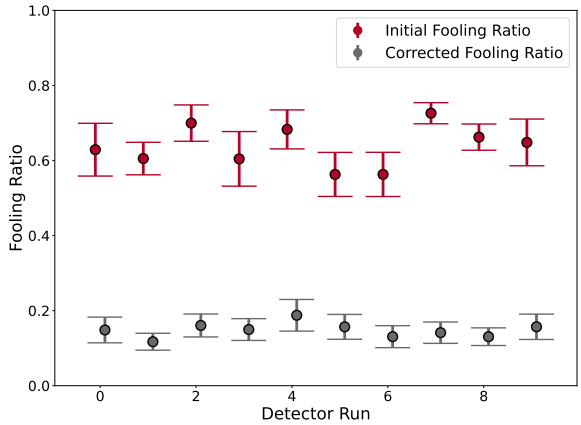

Fig. D9: Comparison of the initial fooling ratios (red) and the corrected fooling ratios (grey) found in 10 independent runs of the adversarial detector pipeline on the TT vs. WW dataset.

Appendix E Attack Parameters

Table E5: User-configurable hyperparameters for the adversarial attack.

Adversarial Attack Hyperparameters		
Parameter Name	**Type**	**Description**
x	Numpy Array	Input data to be attacked.
y	Numpy Array	True labels for the input data.
model	str	Path to the trained model.
min_change	float	Minimum allowed change per feature in each step.
step	float	Step size for candidate feature changes.
num_candidates	int/None	Number of random candidate values per feature (overwrites step).
n_iterations	int	Number of attack iterations to perform.
n_gpus	int	Number of GPUs to use for parallelization.
num_bins	int	Number of bins for feature histograms (D_{JS} calculation).
mask	bool array	Boolean array indicating which samples to target for adversarial modification.
alpha	float	Weight for Jensen-Shannon distance in the cost function.
beta	float	Weight for correlation change in the cost function.
max_jsd_single_change	float	Maximum allowed D_{JS} for a single feature change.
max_frob_single_change	float	Maximum allowed Frobenius norm change for a single feature change.
use_no_change	bool	Whether to allow "no change" as a candidate for each feature.
optimize_already_fooled	bool	Whether to further optimize already fooled samples.
use_disco	bool	Use DisCo correlation instead of Pearson.

E.1 Higgs

Table E6: Parameter values used for adversarial attack for the Higgs dataset.

Higgs Attack Hyperparameter Choice	
Parameter Name	**Value**
min_change	0.0005
step	0.0005
n_iterations	10
num_bins	200
alpha	4.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.006
max_frob_single_change	0.0002
optimize_already_fooled	False
use_disco	False

Table E7: Parameter values used for the augmentation pipeline for the Higgs dataset.

Higgs Attack Hyperparameter Choice	
Parameter Name	**Value**
min_change	0.002
step	0.006
n_iterations	10
num_bins	200
alpha	6.5
beta	1.0
use_no_change	True
max_jsd_single_change	0.01
max_frob_single_change	0.001
optimize_already_fooled	False
use_disco	False

E.2 TT vs. WW

E.3 Donut

Table E8: Parameter values used for the adversarial detector pipeline for the Higgs dataset for training and test samples.

Higgs Attack Hyperparameter Choice	
Parameter Name	**Value**
min_change	0.003
step	0.02
n_iterations	10
num_bins	100
alpha	6.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.01
max_frob_single_change	0.0003
optimize_already_fooled	False
use_disco	False

Table E9: Parameter ranges used for generating training adversaries in the iterative retraining pipeline (randomized per retraining iteration).

Higgs Attack Hyperparameter Ranges	
Parameter Name	**Value / Range**
min_change	[0.001, 0.005] (uniform)
step	[0.005, 0.03] (uniform)
n_iterations	10
num_bins	100
alpha	[3.0, 10.0] (uniform)
beta	[0.5, 2.5] (uniform)
use_no_change	True
max_jsd_single_change	[0.005, 0.03] (uniform)
max_frob_single_change	[0.0001, 0.001] (uniform)
optimize_already_fooled	False
use_disco	False

Table E10: Parameter values used for generating test adversaries in the iterative retraining pipeline.

Higgs Attack Hyperparameter Choice (Test)	
Parameter Name	**Value**
min_change	0.003
step	0.01
n_iterations	10
num_bins	100
alpha	8.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.01
max_frob_single_change	0.0003
optimize_already_fooled	False
use_disco	False

Table E11: Parameter values used for the adversarial attack in the DisCo attack experiment on the Higgs Dataset.

DisCo Attack Hyperparameter Choice	
Parameter Name	**Value**
min_change	0.015
step	0.06
n_iterations	10
num_bins	30
alpha	4.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.02
max_frob_single_change	0.0003
optimize_already_fooled	False
use_disco	True

Table E12: Parameter values used for generating training adversaries in the DisCo detector experiment.

DisCo Detector Attack Hyperparams. (Train)	
Parameter Name	**Value**
min_change	0.07
step	0.08
n_iterations	10
num_bins	30
alpha	4.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.07
max_frob_single_change	0.0025
optimize_already_fooled	False
use_disco	True

Table E13: Parameter values used for generating test adversaries in the DisCo detector experiment.

DisCo Detector Attack Hyperparams. (Test)	
Parameter Name	**Value**
min_change	0.01
step	0.05
n_iterations	10
num_bins	30
alpha	4.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.01
max_frob_single_change	0.00015
optimize_already_fooled	False
use_disco	True

Table E14: Parameter values used for generating training and test adversaries in the for the limited data Pearson correlation experiment.

Limited Data Attack Hyperparameters	
Parameter Name	**Value**
min_change	0.005
step	0.02
n_iterations	10
num_bins	30
alpha	6.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.02
max_frob_single_change	0.002
optimize_already_fooled	False
use_disco	False

Table E15: Parameter values used for adversarial attack for the TT vs. WW dataset.

TTvsWW Attack Hyperparameter Choice	
Parameter Name	**Value**
min_change	0.005
step	0.01
n_iterations	10
num_bins	200
alpha	6.5
beta	1.0
use_no_change	True
max_jsd_single_change	0.003
max_frob_single_change	0.003
optimize_already_fooled	False
use_disco	False

Table E16: Parameter values used for the augmentation pipeline for the TT vs. WW dataset.

TTvsWW Attack Hyperparameter Choice	
Parameter Name	**Value**
min_change	0.003
step	0.02
n_iterations	10
num_bins	100
alpha	6.5
beta	1.0
use_no_change	True
max_jsd_single_change	0.1
max_frob_single_change	0.1
optimize_already_fooled	False
use_disco	False

Table E17: Parameter values used for the adversarial detector pipeline for the TT vs. WW dataset for training and test samples.

TTvsWW Attack Hyperparameter Choice	
Parameter Name	**Value**
min_change	0.01
step	0.01
n_iterations	10
num_bins	100
alpha	6.0
beta	1.0
use_no_change	False
max_jsd_single_change	0.01
max_frob_single_change	0.03
optimize_already_fooled	False
use_disco	False

Table E18: Parameter ranges used for generating train adversaries in the TT vs. WW iterative retraining pipeline (randomized per retraining iteration).

TTvsWW Attack Hyperparam. Ranges (Train)	
Parameter Name	**Value / Range**
min_change	[0.001, 0.005] (uniform)
step	[0.005, 0.03] (uniform)
n_iterations	10
num_bins	100
alpha	[3.0, 10.0] (uniform)
beta	[0.5, 2.5] (uniform)
use_no_change	True
max_jsd_single_change	[0.005, 0.03] (uniform)
max_frob_single_change	[0.0001, 0.001] (uniform)
optimize_already_fooled	False
use_disco	False

Table E19: Parameter values used for generating test adversaries in the TT vs. WW iterative retraining pipeline.

TTvsWW Attack Hyperparameters (Test)	
Parameter Name	**Value**
min_change	0.002
step	0.005
n_iterations	10
num_bins	100
alpha	8.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.01
max_frob_single_change	0.0001
optimize_already_fooled	False
use_disco	False

Table E20: Parameter values used for generating test adversaries in the DonutDummy experiment.

DonutDummy Attack Hyperparameters (Test)	
Parameter Name	**Value**
min_change	0.001
num_candidates	150
n_iterations	10
num_bins	70
alpha	6.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.005
max_frob_single_change	0.05
optimize_already_fooled	False
use_disco	False

Table E21: Parameter values used for generating training and test adversaries in the DonutDummy adversarial detector pipeline experiment.

DonutDummy Detector Attack Hyperparams.	
Parameter Name	**Value**
min_change	0.001
num_candidates	150
n_iterations	10
num_bins	60
alpha	6.0
beta	1.0
use_no_change	True
max_jsd_single_change	0.005
max_frob_single_change	0.05
optimize_already_fooled	False
use_disco	False