Complex & Intelligent Systems (2023) 9:3717–3738
https://doi.org/10.1007/s40747-022-00649-1

ORIGINAL ARTICLE

Dual-filtering (DF) schemes for learning systems to prevent adversarial attacks

Dipankar Dasgupta[1] · **Kishor Datta Gupta**[1]

Received: 2 April 2021 / Accepted: 3 December 2021 / Published online: 21 January 2022

Abstract
Defenses against adversarial attacks are essential to ensure the reliability of machine-learning models as their applications are expanding in different domains. Existing ML defense techniques have several limitations in practical use. We proposed a trustworthy framework that employs an adaptive strategy to inspect both inputs and decisions. In particular, data streams are examined by a series of diverse filters before sending to the learning system and then crossed checked its output through anomaly (outlier) detectors before making the final decision. Experimental results (using benchmark data-sets) demonstrated that our dual-filtering strategy could mitigate adaptive or advanced adversarial manipulations for wide-range of ML attacks with higher accuracy. Moreover, the output decision boundary inspection with a classification technique automatically affirms the reliability and increases the trustworthiness of any ML-based decision support system. Unlike other defense techniques, our dual-filtering strategy does not require adversarial sample generation and updating the decision boundary for detection, makes the ML defense robust to adaptive attacks.

Keywords Adversarial machine learning · Computer security · Bio-inspired algorithm · Negative selection algorithm

Introduction

Adversarial attacks (AA) manipulate input data by adding traits/noises in various trickier ways and such AAs to deep learning models reduce trustworthiness of their use. It is to be noted that the rationale behind AA's success are inconclusive and do not provide clear explanation for real-world applications. Szegedy et al. [78] stated the reason for attack success is non-linearity of ML models; on the other hand, Goodfellow et al. [34] argued that AAs take advantage of linearity in some ML models. Another theory [80] proposed a tilted boundary theory to explain that it is never feasible to fit a model completely, and that is why AAs exist. Some MIT researchers stated that all adversarial features are not additive noise, rather these data cannot be properly classified because human sensors are not sophisticated enough to associate a class for these data, however this argument is disputed by other researchers [41].

✉ Kishor Datta Gupta
kgupta1@memphis.edu

Dipankar Dasgupta
ddasgupta1@memphis.edu

[1] University of Memphis, Memphis, TN, USA

To build a robust ML/AI-based system against malicious adversaries, we designed a dual-filtering scheme, (which employs end-to-end defense mechanism) one at the input stage (before samples are fed to the core learning model) and other at the output of ML (before the decision component). These two filter sets can function independently as well as dependently (i.e., in a commutative fashion). Specifically, the input filtering layer's main aim is to drop misleading and out of distribution inputs (e.g., image of animal but not a human face in a face recognition system). The output filtering layer's goal is to handle larger variations and restricting mis-classification to improve overall accuracy of the learning system. The proposed dual-filtering strategy can be used both in training and testing phases of ML-based systems. For instance, the independent input filters may be used to detect and deter the poising attacks in a supervised ML. Likewise, dual-filtering strategy helps in addressing adversaries both in supervised and unsupervised ML. A machine learning (ML) framework usually consists of four main modules: feature extraction, feature selection (optional), classification/clustering, and decision. As depicted in Fig. 1, input filters are placed after pre-processing of data stream/feature selection to feed to the learning model and the output filters

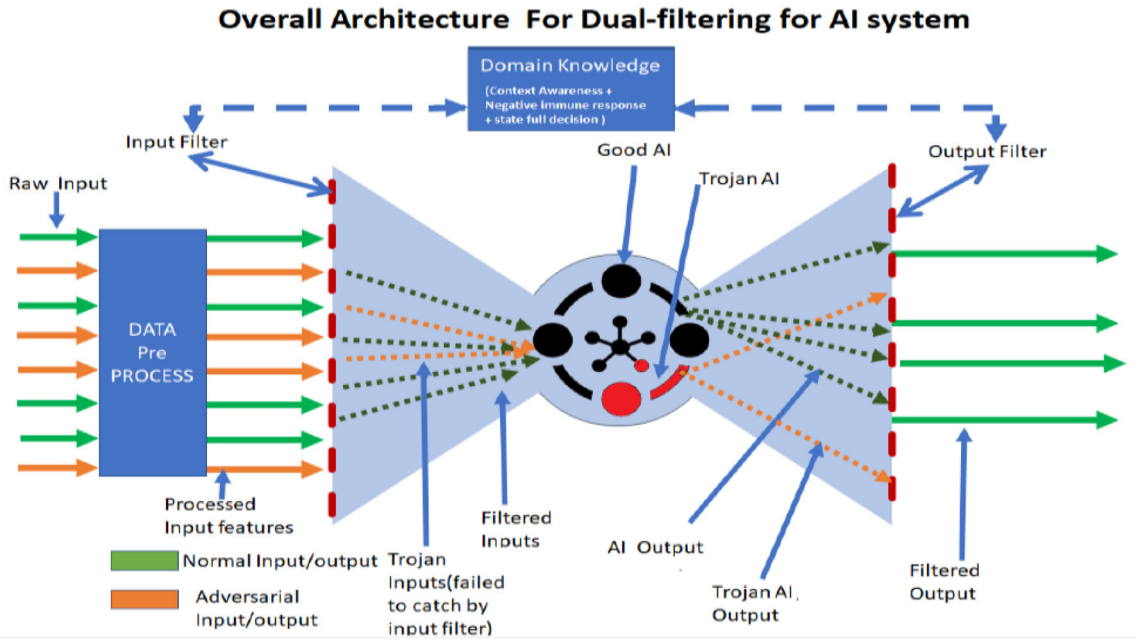


Fig. 1 Schematic diagram of the proposed dual-filtering (DF) framework

are placed after classification/clustering/raw decision module, respectively.

As can been seen in the Fig. 1, the raw input sample is first pre-processed and then fed to the input filter to determine if the received feature/sample is either clean or noisy/adversarial, and accept or reject accordingly. The outcome by ML module is given to the output filter for further scrutiny. The output filter uses context-information and/or communicates with the input filter bank to make the correct final decision. An ensemble of different noise removal or AA detection filters was successfully applied in a recent work [29]. Other techniques focused mostly on adding extra layer on a ML module by adversarial sample training or modification of deep learning models; these defense methods have some constraints, and exposed ML models to new vulnerabilities [36].

In 2019, some works reported launching adaptive attacks where they could bypass known defenses [17]. To alleviate the situation, we consider a non-deterministic (white-box) approach where the attackers cannot perceive our defenses to launch adaptive attacks. Accordingly, we investigated an active learning [73] based dual-validation scheme which work as an extra security (filtering) layer and improve the learning model's trustworthiness.

Accordingly, our defensive measures for machine learning model (MLM) have the following tasks:

- Input filter before MLM: The primary purpose of input filters is to prevent adversarial input data in such a way that can differentiate data manipulation from the trained data. It will be examining the input by deploying application-specific filter sequence. A set of filter sequences are selected (from a given library of filters) using an efficient search and optimization algorithm, called multi-objective genetic algorithm (MOGA). The MOGA can find a sequence of filters (where each filter can detect adversarial traits/noises) satisfying constrains and three objectives: detection of the maximum number of attacks with higher accuracy (above a specific threshold), with minimum processing time, and shorter sequence of ensemble filters. By utilizing the Pareto-set from MOGA runs, and picking a filter sequence dynamically at different times, make filter selections unpredictable and use an active learning approach to protect the ML from adaptive attacks.
- Output filter after MLM: Employ several class-specific latent space-based transformation for outlier detection. After MLM provides an output class label, it is then verified if the output falls in that class's latent space or not. We will make an ensemble of different outlier detection methods and sequence dynamically and also retrain the outlier methods during runtime.

The rest of the paper is organized as follows. "Preliminaries" and "Defense objectives" provides literature review and highlight research challenges. In "Our proposed methodology" and "Experiments", we described our approach with experimental results and analysis. In the following section, we reported advantages and limitations of our defense technique. Next, we gave conclusion with prospects of our future work.

Preliminaries

In this section, we detailed the adversarial properties we studied for our defense techniques and highlighted related works.

Adversarial machine learning (AML) attacks

Based on NIST [79] definition, AML is the manipulation of training data, ML model architecture, or manipulate testing data in a way that will result in wrong output from ML.

Generally speaking, adversarial examples are input data which get miss-classified by an AI method but not by a human eye. In mathematical definition:

Theorem 1 *For a ML model M, if A is Non adversarial input and right class label is C_R, added noise is ϵ, Now,adversarial example $A_x = A + \epsilon$, A_x classify by M as class C_W where $(C_W \neq C_R)$, But if in human eye $A_x \approx A$ and A_x classify as C_R,*

Adversarial defense

Goodfellow et al. [34] tried to training on adversarial inputs pro-actively, Papernot et al. [66] performed defensive distillation and Miyato et al. [57] training the network with enhanced training data all to create a protection against adversarial example. Grosse et al. [35] did statistical tests using a complementary approach to identify specific inputs, that are adversarial. Wong et al. showed convex outer adversarial polytope can be a proven defense [92]. Lu et al. [52] checked whether the depth map is consistent or not (only for image) to detect adversarial examples. Metzen et al. implemented deep neural networks with a small "detector" sub-network were trained on the binary classification task of distinguishing factual data from data containing adversarial perturbations [56]. The same year, Madry et al. [55] published a paper on adversarial robustness of neural networks through the lens of robust optimization. Chen et al. tried to devise adversarial examples with another guardian neural net distillation as a defense from AAs [24]. Wu et al. [93] developed Highly confident near neighbor (HCNN), a framework that combines confidence information and nearest neighbor search, to reinforce adversarial robustness of a base model. Also Paudice et al. [67] applied anamoly detection and Zhang et al. detected adversarial examples by identifying significant pixels for prediction which only work for images [98]. Other researchers such as Wang et al. tried with mutation testing [89] and Zhao et al. developed key-based network, a new detection-based defense mechanism to distinguish adversarial examples from normal ones based on error correcting output codes, using the binary code vectors produced by multiple binary classifiers applied to randomly chosen label-sets as signatures to match standard images and reject adversarial examples [99]. Later that year Liu et al. tried to use steganalysis [50] and Katzir et al. implemented a filter by constructing euclidean spaces out of the activation values of each of the deep neural network layers with k-nearest neighbor classifiers (k-NN) [44]. A different notable strategy was taken by researchers Pang et al. They used thresholding approach as the detector to filter out adversarial examples for reliable predictions [63]. For an image classification problem, Tian et al. did image transformation operations such as rotation and shifting to detect adversarial examples [82] and Xu et al. [95] simply reduced the feature space to protect against adversary. Monteiro et al [59] developed inputfiler which is based on bi-model decision mismatch of image. Sumanth Dathathri showed whether prediction behavior is consistent with a set of fingerprints (a data set of NN) named NFP method [31]. Same year, Crecchi et al. used non-linear dimensionality reduction and density estimation techniques [27] and Aigrain et al. tried to use confidence value in CNN [3]. Some other notable works in that year were meta-learning based robust detection method to detect new AAs with limited examples developed by Ma et al. [54]. Another important and effective work was done by Chen et al., where they tried to keep the records of query and used KNN to co-relate that with adversarial examples [25]. In the Table 1, we summarizes the adversarial defenses.

Nature of adversarial attacks

From our extensive literature review and empirical examination, we observed five basic adversarial nature. These are:

Advanced AAs are ineffective in the physical environment

Advanced adversarial methods are the method which adds fewer noises/perturbs than other methods. In 2017, [47] showed that in the digital version and printed version success of adversarial methods effectiveness decline. They tried to justify their argument with FGSM, BIM, and other iterative methods. [53,61] experimented with FGSM, BIM, and LBFGS methods and showed the destruction rate up of 100% based on distances invalidating these attacks.

Clean and adversarial inputs have identifiable noise difference

Researchers [5,37,68] demonstrate adversarial and clean images have a comparable difference in their noise value which are identifiable for attacks such as FGSM, BIM etc. It was also illustrated that normal filtering technique highlighted the noise part after pixel difference method [37], and these noises could be detected using other metrics such as the histogram average/ local binary pattern, signal to noise ratio (SNR) and etc.

Table 1 Summary of countermeasures against adversarial examples [29]

Defense technique	Approach/scheme
Adversarial training	Ensemble adversarial training, a training methodology that incorporates perturbed inputs transferred from other pre-trained models [86]
	Extended adversarial and virtual adversarial training as a means of regularizing a text classifier by stabilizing the classification function [58]
	Training the state-of-the-art speech emotion recognition on the mixture of clean and adversarial examples to help regularization [21]
Defensive distillation	The main idea used is training the model twice, initially using the one-hot ground truth labels but ultimately using the initial model's probability as outputs to enhance robustness [65,75]
Pre-processing defense	Using PCA, low-pass filtering, JPEG compression, soft thresholding techniques as pre-processing technique to improve robustness [74]
	Use of use two randomisation operations: (1) random resizing of input images and (2) random padding with zeros around the input images [94]
Architecture alteration	Synonyms encoding method that inserts an encoder before the input layer of the model and then trains the model to eliminate adversarial perturbations [90]
	An architecture using Bayesian classifiers (Gaussian processes with RBF kernels) to build more robust neural networks [12]
Network verification	A verification algorithm for DNNs with ReLU function was proposed in [43] verified the neural networks utilizing satisfiability modulo theory (SMT) solver
	The method in [43] was modified in $\max(x, y) = \text{ReLU}(x - y) + y$ and $\|\|x\|\| = \text{ReLu}(2x) - x$ to reduce the computational time
Ensembling countermeasures	The proposed strategy used an ensemble of classifiers with weighted/unweighted average of their prediction to increase robustness against attacks [76]
	A probabilistic ensemble framework against adversarial examples that capitalizes on intrinsic depth properties (e.g., probability divergence) of DNNs [1]
Adversarial detection	First, the features are squeezed either by decreasing each pixel's color bit depth or smoothing the sample using a spatial filter. Then, a binary classifier that uses as features the predictions of a target model before and after squeezing of the input sample [95]
	A framework that utilizes ten nonintrusive image quality features to distinguish between legitimate and AA samples [4]
	Multiversion programming based an audio AE detection approach, which utilizes multiple off-the-shelf automatic speech recognition systems to determine whether an audio input is an AE [97]

Table 2 List of the filters and their accuracy against different attack (250 adversarial inputs for each attack type) on MNIST dataset (ACOORD.net library used for experiment)

Filter family	Code	Filter name	Attack method			
			FGSM	BIM	PGD	JSMA
ANALYTICAL	FT4	Distance	50	70	70	50
	FT10	Morph	75	70	70	60
EDGE Base	FT5	Canny	75	75	75	70
	FT11	Sobel	50	75	50	75
	FT16	Gaussian edge	75	70	75	75
Noise add		Median Blur	65	60	65	55
		Average Blur	70	70	70	70
	FT1	Gaussian Blur	70	65	70	60
	FT7	Gaussian Noise	60	50	60	65
		Dilation	70	70	70	75
		Opening	75	70	75	65
		Closing	70	50	70	75
		SaltAndPepper	75	75	75	75
	FT13	SierraDithering	70	70	50	**70**
Noise reduce	FT12	Erosion	75	55	75	70
	FT0	Sharpen	70	70	75	50
	FT6	Shrink	50	50	55	55
Texture		OilPainting	75	50	75	70
		Pixellate	50	70	50	50
	FT14	Wavelet	70	75	70	50
	FT2	Gabor	50	70	50	50
	FT8	Census	55	55	55	70
Transform		Top_Hat	70	50	70	75
		BlackHat	70	50	75	55
	FT9	Lapalce	70	75	60	75
	FT3	Fourier	55	55	75	75
		Exponential	50	50	50	75
	FT15	Log-polar	50	55	75	65
		Mirror	55	50	75	60
		TopHat	55	50	55	75
		WaterWave	75	50	75	70

Here we only provided successful detection rate

Same filtering technique will work for all ML model for a specific dataset

Filters can detect AAs in data preprocess stage [5,37,68]. That means this technique will work for the black-box model, which means defense is not required to access or modify the ML. So, if the ML changes, for example, from Resnet to VGG or SVM to a Random Forest, the defense technique needs no changes. It will be completely independent of the ML changes.

Different filters have different effectiveness to detect AAs

We have experimented with different filters, as presented in Table 2. Here, we can see that noise addition and canceling filters are working better in the gradient-based attack, and texture-based filters are working better for boundary-based attack types. For example, FGSM and BIM are both gradient-based attacks, and we find out blur works against both of these attacks. This result is expected as AA noises have a distinct nature related to the attack method. This phenomenon proves that picking one filter from each filter family will have more effectiveness than selecting all the filters from the same filter family class. In this experiment, We generated FGSM [34], BIM [55], PGD, JSMA [91] using Pytorch [33], IBM-ART-Toolbox [62] and Cleverhans adversarial library [64]. We noticed that the destruction rate (i.e., the rate of failure of AA when it is converted to physical form) [47] is presented in some attack samples. We disregarded those from attack samples. Also, due to our restriction of $\epsilon = 0.03$

Table 3 List of outlier detection algorithm and their accuracy to detect adversarial (FGSM) input of class label 'O'

Abbr	Algorithm	Accuracy
OCSVM [26]	One-class SVM	99
LMDD [7]	Deviation-based	98
LOF [14]	Local outlier factor	98
COF [81]	Connectivity-based	91
CBLOF [40]	Clustering-based	92
HBOS [32]	Histogram-based	91
kNN [70]	*k* nearest neighbors	91
ABOD [46]	Angle-based	62
COPOD [49]	Copula-based	75
SOS [42]	Stochastic selection	66
IF [83]	Isolation forest	99
FB [48]	Feature bagging	99
XGBOD [100]	Extreme boosting based	26
AutoEncoder [2]	Fully connected AutoEncoder	43
VAE [45]	Variational AutoEncoder	41
SO_GAAL [51]	Single-objective GAN	40
MO_GAAL [51]	Multiple-objective GAN	35
Vdetector [102]	Variable size NSA	99
RNSA [30]	Random real value NSA	75

as maximum noise value, we had to discard some examples from our dataset for having higher noises.

Outlier detection methods can detect AAs as outliers

The work of Ruff et al. [71] shows that outlier detection methods can classify class label from outlier samples. In multi-class classification, each class separately generate their own latent space and outlier detected there as negative class and inliers are detected as positive class and able to achieve 95%+ accuracy for MNIST class classification. Similar approach we experimented with adversarial samples for single class classification. We took class label '0' as positive class or inlier, all other 9 classes and adversarial samples for class 0 are considered negative class or outlier. We trained with 1000 positive class. We used [101] developed outlier library in our experimentation. We tested with 500 positive data, and 500 adversarial sample (FGSM samples generated using [33,62]) of class label 0. The accuracy was presented in the Table 3. We can see that one class support vector machine and V-detector based negative selection algorithm does better than other.

Static defenses can by-passed by adaptive attacks

Carlini el al [17] exhibited an adaptive attack where the attacker can bypass the known defenses. So, if the defense is not changes or remain static it will be vulnerable to adaptive attacks. Also, more recent works showed that dynamic defense mechanism which claims effectiveness against adaptive attacks fails against gradient based adaptive attack [85].

Above natures of adversarial attacks helps us to conclude that filter-based techniques can detect noises and outlier detection method can distinguish between a adversarial and clean input but an adaptive attack can be designed to bypass these defense techniques.

Defense objectives

Researchers have evaluated several defense techniques [8, 16,17,19,20,85] but these evaluations focused on how many different types of attacks a defense technique can defend specially prioritized effectiveness against adaptive attacks. However, many of these defense techniques have implementation issues, such as needing prior knowledge of the ML model and dataset. Some of these techniques require modifying the ML model layers or retrain the model. Retraining the model also can reduce the efficiency of the model. Some defense techniques have high computational costs and are not suitable for ML model's different domains. Yuan et al. [96] suggested making threat models consist of Adversarial Falsification (False negative, False Positive), white-box, BlackBox, targeted, non- targeted, onetime and iterative attacks. Carlini et al. [17], suggested that adversarial attack and defense models need to be tested against a diverse set of attacks. Also, they need to be evaluated against adaptive attacks. Moreover, Tramer et al. [85] suggested different themes to evaluate a defense model. Keeping these guidelines in mind, we developed our threat model inclusive of basic, advanced attack and adaptive attack (against our defenses) types. Carlini et al. [19] also recommended using at least one gradient-free and one hard-label attack. To address that concern, we evaluated our proposed method with gradient-free attacks such as local search attack [60] and hop-skip-jump attack [23]. For testing against an adaptive attack, we used BPDA (Backward Pass Differential Approximation [9]), which can be used to attack non-differential prepossessing-based defenses. Uesato el al. [87] advised to consider obscurity of adversarial attack when considering the defenses. [17] pointed out that testing a defense in one dataset is not enough, therefore we chose multiple datasets (i.e., MNIST, CIFAR-10, and ImageNet). We considered a standard distortion $\epsilon = 0.3$ for MNIST and $\epsilon = 8/255$ for CIFAR-10, as current state-of-the-art [85] recommended. Thus, our threat model is a combination of gradient-based, gradient-free, and adaptive evasion based adversarial attacks on multiple datasets. These attacks studied in this work are a combination of White-box, Black-Box, targeted and non-targeted attacks. Also, the presented defense will be able

to defend against attacks that are completely unknown to the proposed defense scheme. Considering above researchers suggestions in mind, we aim to provide an adversarial defense system which will meet the below objectives:

- Defense needs to work against a diverse set of attack types. Our provided defense technique should work against Gradient or no-gradient, white-box or black-box, targeted or no targeted, adaptive attacks [17].
- Defense should not reduce the accuracy of ML models. The model accuracy should not get effected after deploying our defense technique.
- Defense needs to identify threats faster. If a defense system takes sizeable computational time and resources, it will lose the practicability. For example, if the defense is employed in an autonomous car sensor, the input responses need to evaluate first. Otherwise, an accident can happen.
- Defense should not modify ML architecture. Defense should work for both the white-box and black-box models. A trained ML architectural information is usually black-box. So, it is expected that the defense framework will comply with that.
- Defense should be adaptive in nature and dynamic to prevent the adaptive attacks.
- Defense should not need to update if ML changes (Resnet to VGG or ANN to RNN), and it should be cross-domain (image, audio, text) supported.

Our proposed methodology

In the Fig. 2, we illustrated the basic concept of our proposed solution. As we know that it is possible to detect adversarial input noise using different filters, we will apply filters to detect noise. We need to know which filter we need and the difference between the clean and adversarial noise threshold. that is why we first use the information from the ML model to determine the input is an outlier for the class label the ML model is classified or not. If it is an outlier, we will send it to the adversarial dataset. If not, we will send it to the clean dataset and update the outlier methods decision boundary and determine the required filters and the noise thresholds. Before update/retrain the output and input learning model, we will inspect the data for adaptive attack patterns in the adaptive attack detection module.

The Fig. 3 demonstrated our proposed dual inspection strategy. It is shown that the inspection before and after ML are independent and can be deployed as a plugin. As in active learning, when the clean dataset has some data, it will train the outlier detection techniques, and the 'inspection after ML' module will start to work. After the outlier finds some adversarial examples, the adversarial dataset receives some data. When the adversarial dataset has sufficient data, our multi-objective Genetic algorithm started the genetic search for filter sequences that are effective against the adversarial noises and the differentiating noise thresholds for these sequences. As time progresses, MOGA will detect more adversarial samples, and the knowledge of the outlier detection technique will transfer to noise detection techniques. This way ML model has to process fewer adversarial examples. We will select different filter sequences for each input and different outlier detection methods for each input to make the defense dynamic. After each (or a specific amount of input), outlier methods will retrain, and it will update the outlier detection decision boundary. Similarly, MOGA will update the filters library subsequently. This way, both outlier and filter-based defense technique will keep themselves updated as time progress. As this method can be vulnerable by adaptive attack, we will store the data and inspect for adaptive attack pattern before update the filters and outlier detection methods. We detailed that method on "Adaptiveness and dynamic selection".

The basic workflows from the Fig. 2:

1. Input will be sent for filters to extract different metrics (SNR, Histogram etc). There will be a dynamic selection of the filter set from the filter library.
2. Extracted filter metrics value will check for perturb, if it is above certain threshold switch S1 will open or other wise switch s2 and s3 will open.
3. S1 open:
   - input will be sent to adversarial dataset and the process will terminate.
   - Adversarial dataset will retrain the filter sequence search for noise detection and change the threshold value.
4. S3 and S2 open:
   - If S3 open, extracted filter metrics value will be sent to outlier detection system.
   - If S2 open, input data will be sent to ML model and Switch S5.
5. ML model will deliver the output class to S4 and outlier detection system.
6. Outlier detection system will randomly pick one outlier detection method. If it detected as outlier witch s1 will open, otherwise S4 and S5 will open.
7. S1 open:
   - input will be sent to adversarial dataset and the process will terminate.
   - Adversarial dataset will retrain the filter sequence search for noise detection and change the threshold value.

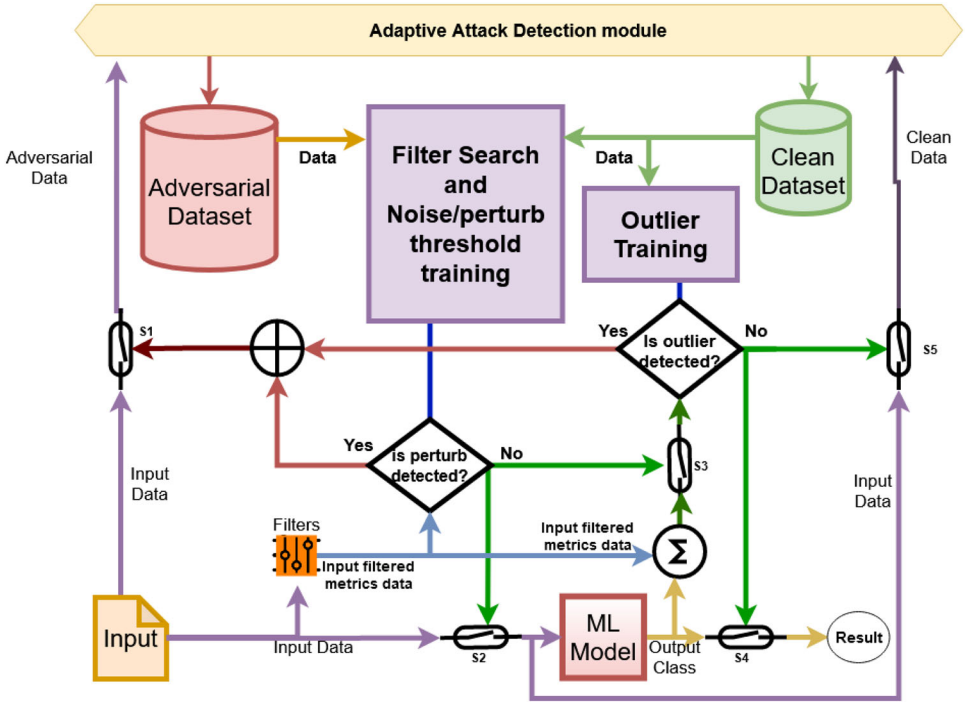


Fig. 2 Illustration of basic flow concept for proposed dual inspection framework. If the input is not adversarial, the original input (not the processed) be sent to the learning model/ML and after ML produce class label, that labels latent space will be used in outlier method. The outlier decision boundary and the threshold of noise will change as the dataset of adversarial and clean data set are updated by each input

8. S4 and S5 open:
   - S4 will provide the final output class and S5 will send the input to clean dataset which will trigger the retrain of outlier methods and change the outlier decision boundary.

Multi-objective genetic search for filters

We need multiple filter sequences because we cannot use the same sequence of filter for every input. A sequence could be any length. Search for optimal set of sequences require significant computational time, if we do exhaustive search, considering multiple objectives. that is why we will employ a multi-objective GA to search for the optimal set of sequences as pareto-front solutions. For search filters, we need to consider different factors besides their accuracy. Based on our objective, our filters need to be fast, that's why order of filters are important because different order of filters require different amounts of processing times. It is preferable to have our solutions time efficient. If we have N number of filters, then total possible number of sequences will be our search space. If we do not consider time efficiency, then we do not need to order in a combination of sequence (For different order a sequence accuracy remain static but time efficiency change). We can optimize our search space by limiting the minimum sequence length and maximum sequence length. So, our search space equation will be:

$$\text{Optimized}_{\text{Search}_{\text{space}}} = \sum_{k=\min}^{N-\max} \frac{N!}{N!-k!} \tag{1}$$

Here, min and max are the minimum and maximum length of a filter sequence. Suppose, we have 17 filters and minimum length were 6, then our optimized experimental search space has consists of 9.66^{14} search item. It justifies the necessity of using heuristics search method like GA. In summary, we need a time-efficient, to produce a reliable performance and a unique set of sequence. Our designed multi-objective GA can achieve all these criteria. The purpose of using a genetic search is to find a diverse sequence of filters detecting AAs with maximum accuracy while each filter is having distinctive characteristics and capabilities when deployed such a sequence adaptively (interchangeably) in a ML that will be unpredictable to attackers compared to a static ensemble of well-known filters. So, the GA will find not only the best filter ensemble, but also a set of diverse filter sequences in multi-objective Pareto-front.

Perturb range/threshold determination of filters

First, we need to process the clean dataset and run all the filters and calculate their metrics value and gather their Mean, Std Dev, Max. Using algorithm for each filter. Now, if Mean is $\bar{\mu}$, standard deviation is σ, Our lower range (L_r) and upper

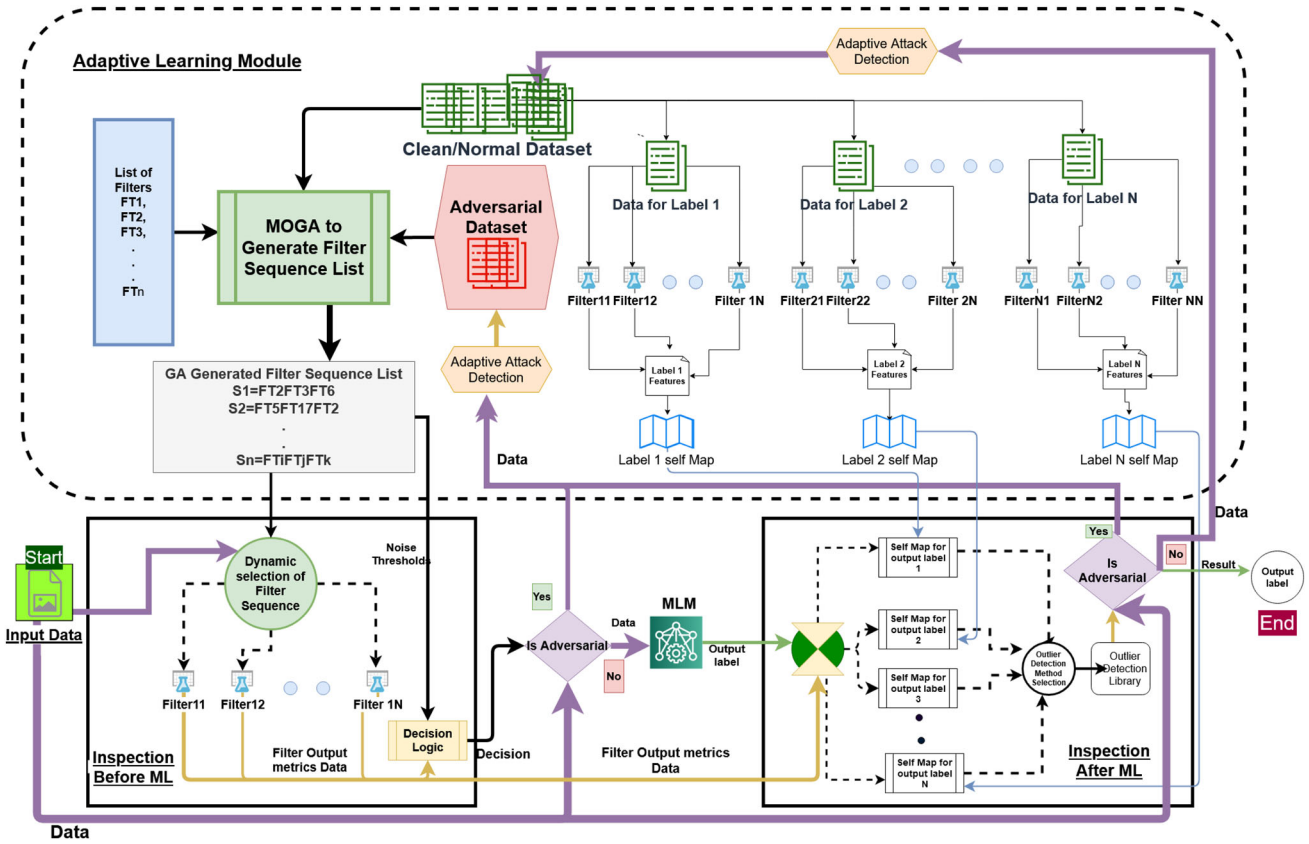


Fig. 3 Illustration of proposed dual inspection framework. If the input is not adversarial, the original input (not the processed) be sent to the learning model/ML and after ML produce class label, that labels latent space will be used in outlier method. Selection of outlier and filer sequence will be dynamic

range(U_r) calculation denoted by Eqs. 2 and 3

$$L_r = \text{Min} - \frac{\sigma}{\bar{\mu}} \tag{2}$$

$$U_r = \text{Max} + \frac{\sigma}{\bar{\mu}} \tag{3}$$

Using Eqs. 2, 3 for 17 filters, we generated list of upper and lowerange

Encoding

First, we encoded all filters according to Table 2. Here, 17 algorithms were assigned sequence number $FT1, FT2 \ldots FT17$. These filters are our genes. We will create our individuals/chromosome using these genes. We generated the population by random sequence generation using the genes. So, each sequence is consists of different length of filters. We remove multiple occurrences of filters in a single sequence. As example a random sequence $FT2FT5FT11FT12$ means **Blur -Census - Morph - Canny**. That way, we generated multiple lengths of sequences as our initial population.

Fitness function

We have three objectives. they are accuracy (α), time to detection (β) and insider diverseness (γ) of filters in sequence. Accuracy is the success rate of detection by the filters. The filter sequence takes adversarial and clean samples from the dataset. Based on the filter sequence's metrics value range, we check how many adversarial samples we can detect and how many we falsely detected. We used F1 score as accuracy value.

$$\alpha(S) = F1 \tag{4}$$

For time, if each filter take t_i time, then total time to detection δ_t can be calculated by

$$\beta(S) = \sum_{i=0}^{n} t \tag{5}$$

For insider diverseness, for each filer $f_i \in F_i$, here F is the filter family and f is the filter, S is the sequence.

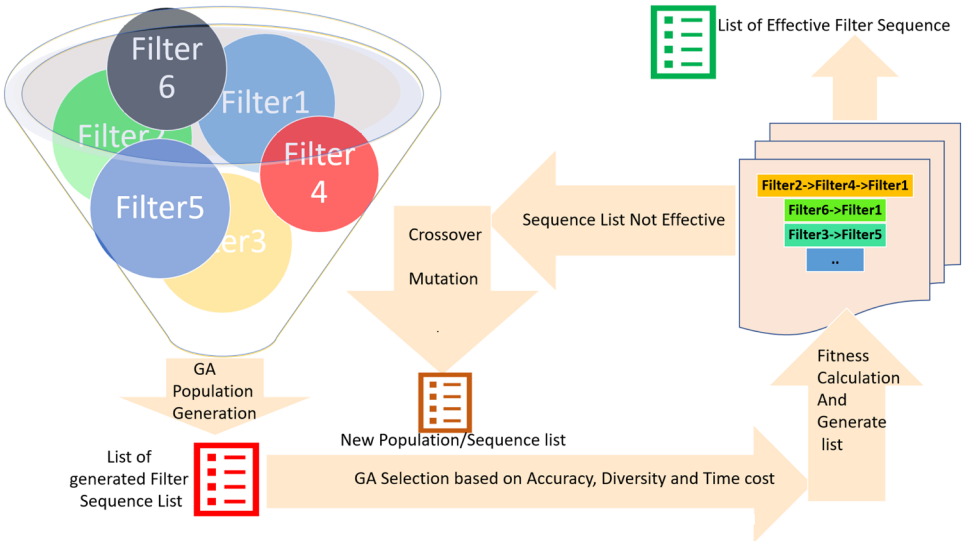


Fig. 4 GA to search appropriate filters

$$\gamma(S) = \frac{\sum f \in F | f \cap S| > 0}{\sum F} \tag{6}$$

We normalized all three objective data using equation 7

$$X_{sc} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}. \tag{7}$$

We inversed the Time data, so, we have to maximize all of the objectives. Our fitness function denoted by

$$\max(f(S)) = ((\alpha_n(S)), (\beta_n(S)), (\delta_n(S))), \tag{8}$$

where α_n is normalized accuracy, $\beta_n$is normalized inverse time, δ_n is diverseness factors.

We have penalty function to prioritize simpler solution and weight values to speed up the GA process. We observed that, in the beginning α is low and after a certain iteration γ gets lower. We use W_0 for α and W_1 for γ as weigh value.

For penalty functions we need below parameters
Length of Best fitted individual in previous iteration $|\max_{f(S_i)} \in \forall(S)|$
Size of current Sequence = $|S|$
Total number of filters = $\sum_{i=0}^{n} |f| \in \forall F$
Equation for penalty function value can be denoted by

$$pf(S) = \frac{f(S)}{100} \times \frac{|S| - |\max_{f(S_i)} \in \forall(S)|}{\sum_{i=0}^{n} |f| \in \forall(F)} \tag{9}$$

So, from Eq. 8, fitness for S is

$$f(S) = \sqrt{(\alpha_n(S)^2 \times W_0 + \beta_n(S)^2 + \delta_n(S))^2 \times W_1} - pf(S). \tag{10}$$

Crossover, mutation and selection

We used an elitist strategy with rank selection [28] and kept the best performing filters for the next generation in steady-state genetic search. We used PMX crossover as the order of the filter sequence are important optimization criteria in a sequence [88]. In the Fig. 4, We illustrated the genetic search of near-optimal filter sequences where search terminated after specific iterations or if the fitness values do not improve for a long period i.e. threshold number of iterations.

One class classifications outlier method

There will be different latent spaces for each class to detect that class's outlier. In the Fig. 5, we can see that MNIST digits have their clusters for each class label, and these are well separable. In the Fig. 6, we can see filter-based metrics can very easily differentiate between adversarial and clean sample. We suggest using an ensemble of different outlier detection methods—for example, a combination of one-class SVM, isolation forest, and negative selection algorithm. Our experimental results shows that negative selection algorithm is random nonlinear learning system, which is applicable for adversarial detection and it randomness made is easy to make the system adaptive by regular updating the learning model.

In the Table 4, we experimented the attack sample accuracy rate with v-detector generated using different number of clean samples. It is observed that after each iteration v-detector performance slightly increased. In the Table 5, we compare v-detector NSA results with other techniques; we can see that OCSVM and IF performs better than NSA for gradient-based attacks for low noise attacks NSA outperforms both of them. Variable Autoencoder did not perform well due to a low number of samples. SOGAL and MOGAL based techniques were also failed to work with low models.

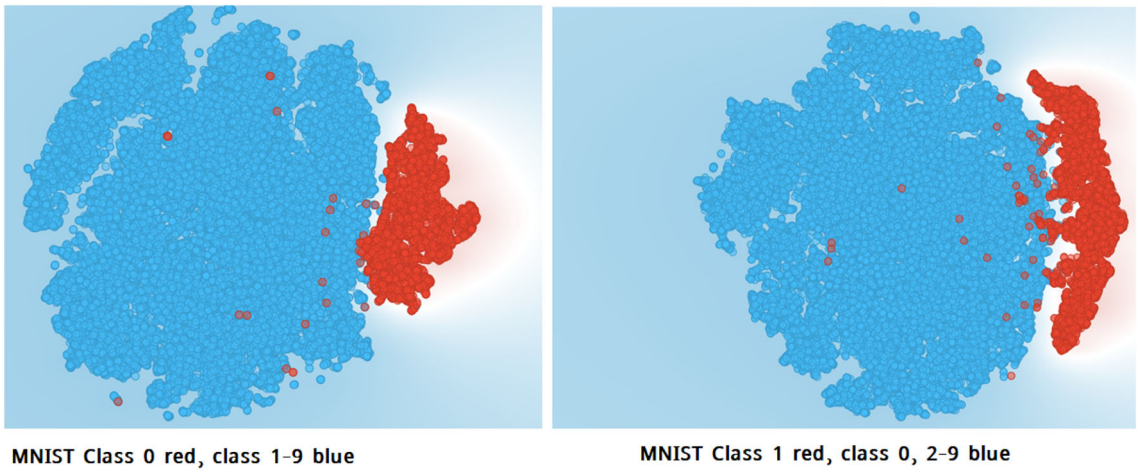


Fig. 5 PCA based clustering for class label 0, and 1 from MNIST dataset using each class own latent space

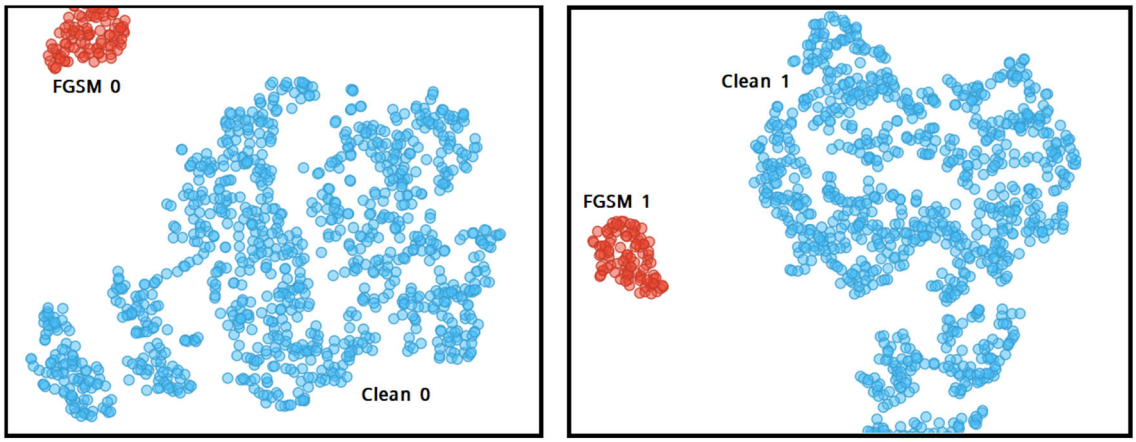


Fig. 6 FGSM based adversarial input differs from their target class using filtered metrics presented using PCA for dimensional reduction

Table 4 Detection of adversarial inputs which classified as MNIST class label '0' and with clean input of class label '0', after sample size increased 100 in each step

Attack type	Step1	Step2	Step3	Step4
FGSM	0.86	0.92	0.902	0.93
BIM	0.89.0	90.0	90.0	0.90
PGD	0.92	0.94	0.95	0.95

Adaptiveness and dynamic selection

We randomly choose different filter sequence and other outlier methods to keep the system dynamic for each input. After each input, the outlier detection modules are updated by changing their decision boundaries, making the detection filters adaptive. However, the filter sequence can change, and the noise threshold value gets updated after each MOGA run. This makes common adaptive attacks ineffective as each input continuously updates the defense strategy. An adaptive attacker will first send random clean input. And started to add some noise in these inputs and send repeatedly until the classification result changes. That way adaptive attackers will know the decision boundary of the learning model. Then adaptive attacker will start creating input that is close to the decision boundary in the representation space.

In our method, attackers have to bypass our dynamic and changing adversarial detection method, which decision boundary is not affected by the actual learning model. If our filter set or outlier detector method was static this method would work, but dynamic selection of filterset and outlier detection method will make it hard to formulate the adaptive attack. Additionally, after a certain set of inputs, we will regenerate negative detector sets by considering these new inputs as self data. So, entire outliers decision parameter would change and the adaptive attacker will not able to establish a fixed decision boundary line for adversarial and nonadversarial input data as adaptive attacker is looking for class classification boundary not adversarial and non-adversarial decision boundary. This update method can

Table 5 detection of adversarial inputs which classified as MNIST class label '0' and with clean input of class label '0'

Attack type	NSA	OCSVM	IF	VAE	SOGAL	MOGAL
FGSM	0.93	0.99	0.93	0.65	0.5	0.5
BIM	0.90	0.98	0.91	0.66	0.5	0.5
PGD	0.95	0.99	0.92	0.50	0.5	0.5
MBIM	0.91	0.98	0.94	0.46	0.5	0.5
HSJ	0.88	0.55	0.41	0.65		
JSMA	0.9	0.56	0.8	0.83		
CW	0.96	0.42	0.66	0.52		
DF	0.91	0.45	0.76	0.55		

also be vulnerable to adaptive attacks which aim to bias the method accuracy, we added a adaptive attack detection module before update/retrain our adversarial detection techniques.

In the adaptive attack detection module, we will analyze distributions of last certain number of inputs are align with total distributions of inputs. We will use the Kolmogorov–Smirnov goodness of fit test (K–S test) compares data with a known distribution and lets us know if they have the same distribution. This test is nonparametric as it doesn't assume any particular underlying distribution [11]. The Kolmogorov–Smirnov test determines a null hypothesis, H_0, that the two samples originate from the same distribution. Then we explore for evidence that this hypothesis should be rejected and formulate this as probability ρ. If the prospect of the samples being from different distributions tops a confidence level we reject the original hypothesis and accept hypothesis H_1, which stated that the two samples are from different distributions. Based on the KS distribution table, if $\rho < \frac{1.22}{\sqrt{n}}$ (where n = number of stored input) than the stored input has inputs from adaptive attack. We disregarded those samples as these may create data bias in our defense learning system.

In summary, our adaptive defense mechanism consists of the following properties

- Dynamic selection of filter set sequence which will make it harder to formulate adaptive attack based on known filter knowledge.
- Dynamic selection of outlier detection method, it will make the adaptive attack to consider all outlier detection method when developing attack input that will make generating input computationally expensive.
- Defense is always learning which will continue changing the filter sequences and decision boundary of outlier detection models. It will make an adaptive attack difficult to search decision boundary.
- To protect against continuous query-based attacks, we will monitor and analyze input trends using the K-S test. The number of inputs considered for the K-S test will be dynamic. Formulate a query-based attack on the defense system will be hard due to the randomness of the K-S test sample number. Our input trend detection system can effectively monitor adaptive attacks and able to take countermeasure.

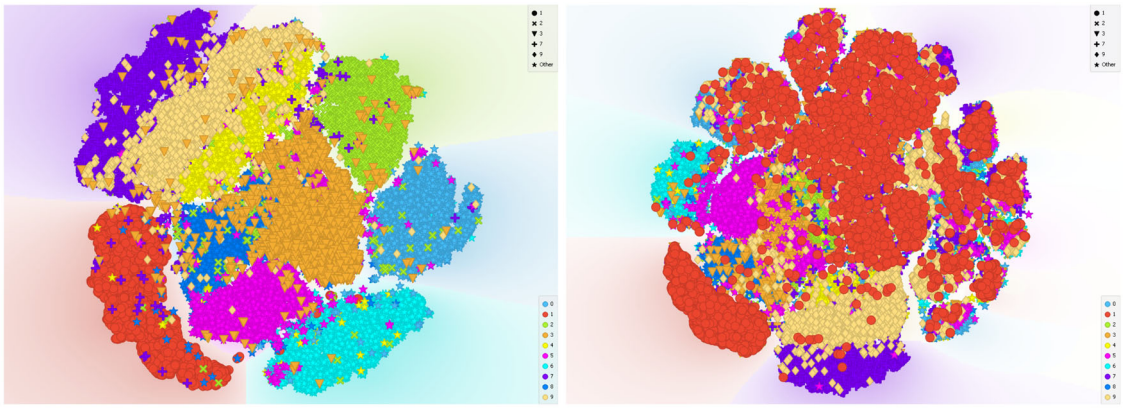
Our defense properties will make the state of the art adaptive attack ineffective and it will make computationally harder to formulate new adaptive attacks.

Experiments

Dataset generation

We did a comprehensive experiment with MNIST and CIFAR-10. We did extensive testing with the MNIST dataset for all the 10 classes and with the full dataset. We did CIFAR testing with two classes. After that, we evaluated our method using EMNIST, Fashion-MNIST, and IMAGENET data-set which re-validated our methodology. We generated FGSM, JSMA, and CW samples to test the results. We generated 100,000 FGSM samples using LENET-5. LeNet-5 LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers. For JSMA we generated 100,000 JSMA samples using a CNN. CNN architecture is made up of 5 layers. The layer composition consists of 3 convolutional layers, 1 flatten and 1 dense layers. All of the activation functions are using RELU. and last we generated 100,000 CW samples using VGG-16 neural net.

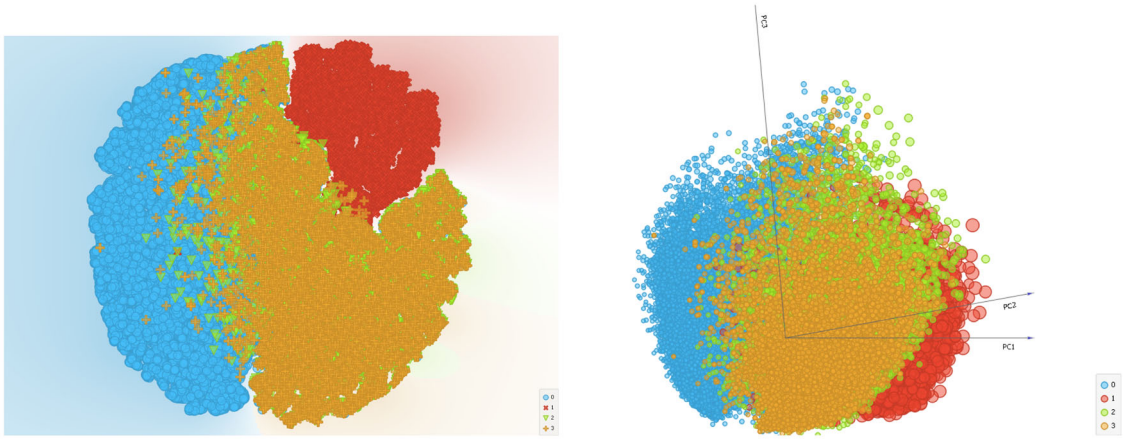
To establish the ground truth for our research we used 30,000 clean image samples, 10,000 FGSM, 10,000 JSMA and 10,000 CW attack samples on MNIST dataset. For filtering operation we picked 14 filters using python opencv library. They are medianblur, GaussianBlur, AverageBlur, Bilateral blur, AdditivePoissonNoise, AdditiveGaussian Noise, Erode TopHat ,Blackhat, Morphology gradient, Opening, Closing, Dialte filter. We apply the filer in the image and than extracted difference between original image and the filtered image. After that we measure the average and



(a) All clean data projected using t-sne visualization with class labels.

(b) All clean and adversarial data projected t-sne visualization with class labels.

Fig. 7 Experimental data representation space for each class of MNIST digits



(a) All clean and adversarial data projected using PCA reduction and t-sne conversion.

(b) All clean and adversarial data projected using 3 PCA components.

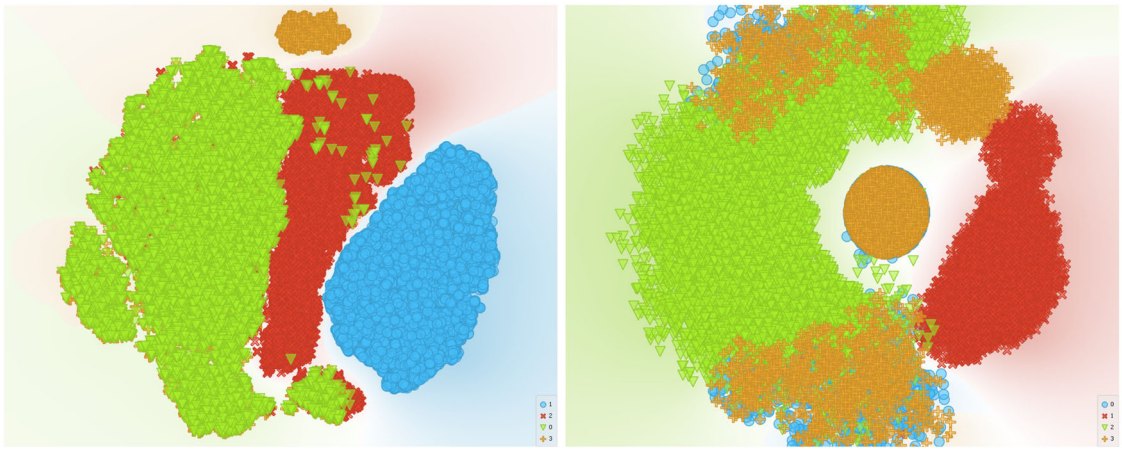
Fig. 8 Experimental data representation space (here clean is green, red is FGSM, blue is JSMA and yellow is CW)

standard deviation of white color histogram for the extracted image and horizontal and vertical signal to noise ration for the extracted image.

In the Fig. 7 'b' we visualized how adversarial (FGSM + JSMA + CW) inputs of one class label overlap with other class label compared with 'a' where only clean inputs where presented. This shows that adversarial samples are hard to distinguish between class labels. In the Fig. 8, we represented all inputs with their adversarial attack type along side the clean input. Here blues are the clean one. We can see here the FGSM which is visualized with red are not overlapping with clean one or other attack type much. But JSMA and CW are highly overlapping with each one and also partially with clean one. CW inputs are more overlapping with clean samples.

In the Fig. 9, we represented the adversarial and clean samples after applying 14 filters. In the Fig. 'a' of 9, we represented the SNR values of the images and it showed FGSM (blue) samples are very easily separable but JSMA (red) and CW(yellow) are hard to separate using SNR values only. However, JSMA are more separable but CW and clean samples are completely overlapping. In the Fig. 'b' of 9, we illustrated using histogram value and it made CW more separable than the clean ones. In the Fig. 10, we applied both SNR and histogram metrics together, and it visible that adversarial and clean samples are now more easily separable. In the Fig. 'b' of 10 blues are the clean samples and re are the adversarial samples. We can see some clean samples are overlapping with adversarial samples but other way is rare. In the figure 'a' we presented the adversarial attack type two and we see some CW samples are also overlapping with clean samples but it is negligible.

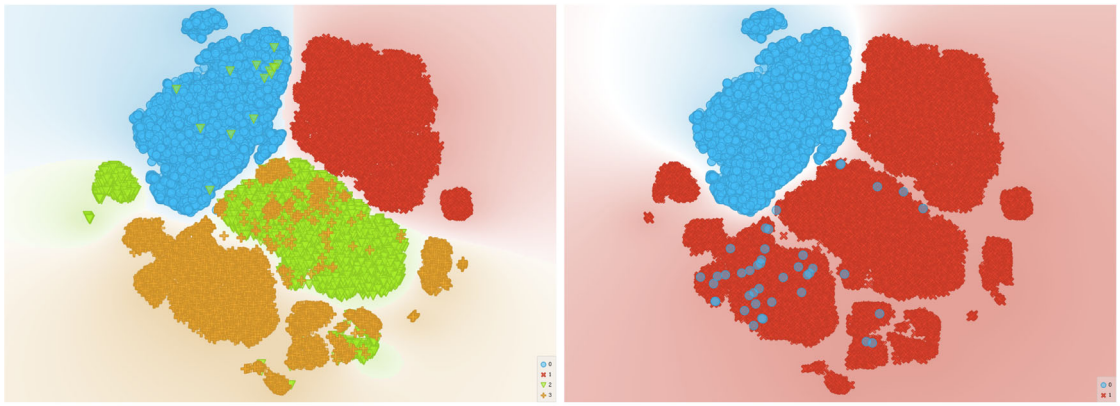
In the Fig. 11, we presented each class label adversarial and clean data both without adversarial classification



(a) All clean and adversarial data projected after applied 14 filters SNR values.

(b) All clean and adversarial data projected after applied 14 filters average histogram values.

Fig. 9 Experimental data representation space after filter applied with one metrics (here clean is green, red is FGSM, blue is JSMA and yellow is CW)



(a) All clean and adversarial data projected after applied 14 filters SNR and Histogram average values (Here clean is blue, red is FGSM, green is JSMA and yellow is CW).

(b) All clean and adversarial data projected after applied 14 filters SNR and Histogram average values (here blue is clean and red is fgsm,jsma and cw.

Fig. 10 Experimental data representation space after filter applied with two metrics

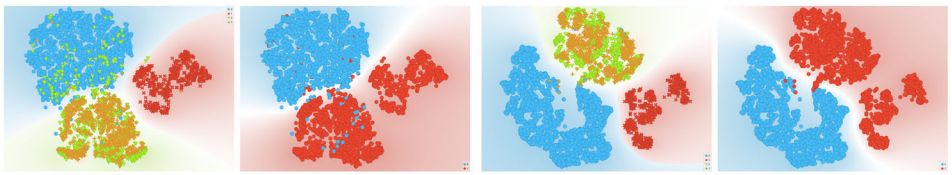
and with adversarial classifications. This visual presentation shows that, when we each class as inlier and all other as outlier, than adversarial samples were more easily detectable.

In the Table 6, we used six different learning method to differentiate between clean and adversarial attack type using image pixel information, it is evident that random forest performs better than others and in the Table 7, we converted in as binary problem where only clean and adversarial input was classified. SVM method performed very poorly as the representation spaces was not linear. But random forest performs well that other methods. But when we applied SNR and Histogram feature based classification all other method except SVM started to performs well and neural network started to outperforms other methods as presented in Tables 8 and 9.

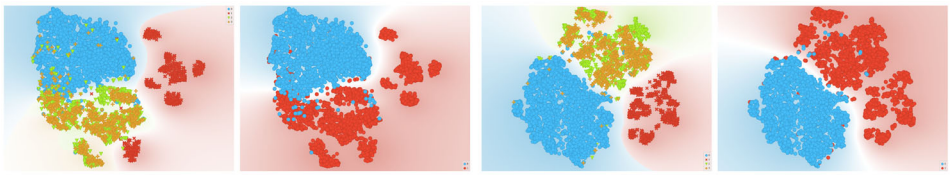
In the Table 10, we presented the identification of different class labels correctly using SNR and histogram value-based checking. We used Random-forest learning. It is seen that some classes are harder to identify than other class labels. As an example, class 2 and 3 is harder than identify adversarial class for input label 9.

Experiment with CIFAR and IMAGENET

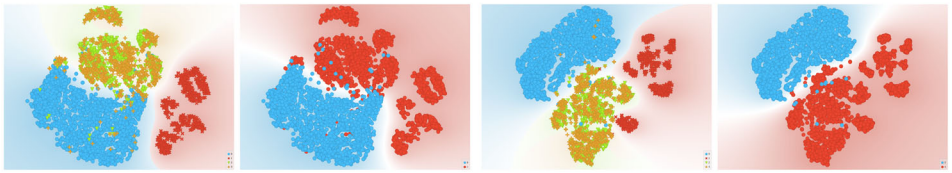
In the Table 11, we compared v-detector performance on MNIST digits (0–9) as illustrated in Fig. 11 and 4 class's of CIFAR-10 dataset. Our result shows that v-detector outperforms other out-lire detector consistently for all attack type and dataset.



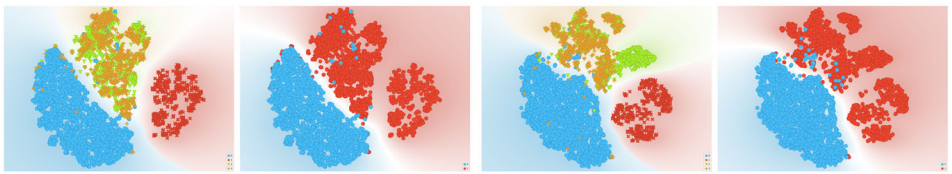
(a) MNIST '0' with FGSM, JSMA and CW separately (b) MNIST '0' with FGSM JSMA and CW together (c) MNIST '1' with FGSM, JSMA and CW separately (d) MNIST '1' with FGSM JSMA and CW together



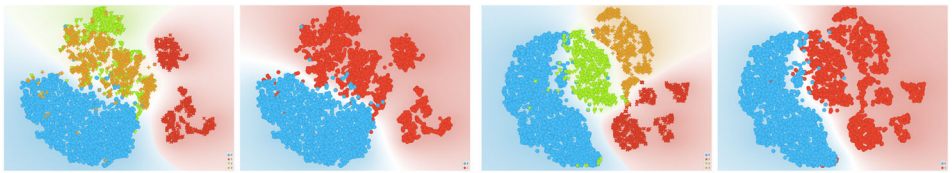
(e) MNIST '2' with FGSM, JSMA and CW separately (f) MNIST '2' with FGSM JSMA and CW together (g) MNIST '3' with FGSM, JSMA and CW separately (h) MNIST '3' with FGSM JSMA and CW together



(i) MNIST '4' with FGSM, JSMA and CW separately (j) MNIST '4' with FGSM JSMA and CW together (k) MNIST '5' with FGSM, JSMA and CW separately (l) MNIST '5' with FGSM JSMA and CW together



(m) MNIST '6' with FGSM, JSMA and CW separately (n) MNIST '6' with FGSM JSMA and CW together (o) MNIST '7' with FGSM, JSMA and CW separately (p) MNIST '7' with FGSM JSMA and CW together



(q) MNIST '8' with FGSM, JSMA and CW separately (r) MNIST '8' with FGSM JSMA and CW together (s) MNIST '9' with FGSM, JSMA and CW separately (t) MNIST '9' with FGSM JSMA and CW together

Fig. 11 Experimental data representation space for each class of MNIST digits with adversarial attack (here clean is blue, red is fgsma, green is JSMA and yellow is CW)

Table 6 Adversarial type classification for MNIST dataset for Clean, FGSM, JSMA, and CW

Model	AUC	CA	F1	Precision	Recall	LogLoss	Specificity
Random Forest	0.973	0.845	0.844	0.844	0.845	0.412	0.926
kNN	0.870	0.643	0.624	0.626	0.643	0.753	0.810
Naive Bayes	0.794	0.562	0.444	0.367	0.562	0.947	0.691
Neural network	0.815	0.573	0.501	0.629	0.573	0.919	0.763
SVM	0.527	0.523	0.399	0.434	0.523	2.073	0.606
Logistic regression	0.813	0.566	0.489	0.442	0.566	0.952	0.724

Table 7 Binary classification for MNIST dataset for clean and adversarial (FGSM, JSMA, and CW)

Model	AUC	CA	F1	Precision	Recall	LogLoss	Specificity
Random Forest	0.998	0.970	0.970	0.970	0.970	0.154	0.985
kNN	0.966	0.844	0.840	0.837	0.844	0.332	0.928
Naive Bayes	0.914	0.737	0.740	0.749	0.737	0.613	0.916
Neural network	0.951	0.816	0.810	0.807	0.816	0.420	0.919
SVM	0.860	0.302	0.208	0.681	0.302	1.598	0.853
Logistic regression	0.937	0.790	0.783	0.778	0.790	0.473	0.910

Table 8 Adversarial type classification for MNIST dataset for Clean, FGSM, JSMA, and CW after applied histogram and SNR based features

Model	AUC	CA	F1	Precision	Recall	LogLoss	Specificity
Random Forest	1.000	0.999	0.999	0.999	0.999	0.007	1.000
kNN	0.999	0.984	0.984	0.984	0.984	0.038	0.995
Naive Bayes	0.999	0.983	0.983	0.984	0.983	0.203	0.994
Neural network	1.000	0.998	0.998	0.998	0.998	0.008	0.999
SVM	0.896	0.590	0.531	0.815	0.590	1.317	0.864
Logistic regression	0.999	0.983	0.983	0.983	0.983	0.056	0.994

Table 9 Binary classification for MNIST dataset for clean and adversarial (FGSM, JSMA, and CW) after apply SNR and histogram features

Model	AUC	CA	F1	Precision	Recall	LogLoss	Specificity
Random Forest	1.000	0.999	0.999	0.999	0.999	0.002	0.998
kNN	1.000	0.998	0.998	0.998	0.998	0.005	0.995
Naive Bayes	0.998	0.999	0.999	0.999	0.999	0.042	0.997
Neural network	1.000	0.999	0.999	0.999	0.999	0.005	0.997
SVM	0.996	0.753	0.652	0.814	0.753	0.563	0.265
Logistic regression	0.999	0.997	0.997	0.997	0.997	0.017	0.992

Table 10 Confusion matrix of MNIST adversarial input detections using SNR and histogram value

Predicted												
		0 (%)	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	7 (%)	8 (%)	9 (%)	∑
Actual	0	***96.7***	0.2	0.4	0.2	0.5	0.2	0.3	0.7	0.6	0.2	3259
	1	0.2	***97.5***	0.3	0.3	0.3	0.4	0.2	0.4	0.4	0.0	3313
	2	0.4	0.3	***95.4***	0.6	0.8	0.8	0.7	0.4	0.5	0.1	3254
	3	0.6	0.3	0.5	***95.4***	0.7	0.6	0.4	0.5	0.7	0.2	3272
	4	0.5	0.5	0.5	0.5	***96.4***	0.3	0.5	0.1	0.5	0.3	3262
	5	0.6	0.5	0.7	0.6	0.7	95.1	0.6	0.6	0.6	0.2	3265
	6	0.5	0.4	0.5	0.3	0.5	0.5	***96.5***	0.2	0.4	0.1	3263
	7	0.3	0.5	0.3	0.3	0.6	0.4	0.3	***96.8***	0.3	0.2	3292
	8	0.5	0.3	0.4	0.1	0.3	0.3	0.6	0.5	***96.5***	0.4	3309
	9	0.1	0.2	0.2	0.3	0.2	0.1	0.2	0.2	0.2	***98.5***	3266
	∑	3275	3334	3224	3225	3295	3227	3271	3307	3331	3266	32755

In the Table 12, we presented results using similar experiments we used for ground truth experiments. Our performance of CIFAR and IMAGENET is very good compare to the state-of-the-art attack. Also, a good portion of false positives was failed adversarial examples due to perturbation loss while converting physical form. This result verifies that the same filters and histogram, SNR-based methods are applicable for all datasets of the same domain. We also tried to formulate BPDA attack against our defense but failed to formulate the attack.

When we were evaluating our defense against an advanced attack (with very low noises/perturbs) we observed that as all adversarial attack types aim to reduce the perturbation in advanced attack types, the magnitude of perturbation gets so

Table 11 Comparison of results with different outlier detection models to compare V-detector NSA performance with other OCC methods

Models used	MNIST			CIFAR		
	FGSM	JSMA	CW	FGSM	JSMA	CW
MCD	0.9846	0.99	0.9101	0.8616	0.864	0.7871
OCSVM	0.6851	0.697	0.5421	0.8731	0.535	0.5417
LMDD	0.6673	0.601	0.553	0.5752	0.561	0.5965
LOF	0.997	0.912	0.93	0.8963	0.832	0.8096
COF	0.3991	0.37	0.3568			
CBLOF	0.9866	0.959	0.9			
HBOS	0.9865	0.915	0.9	0.8354	0.859	0.0016
KNN	0.9993	0.909	0.9628	0.9957	0.925	0.0682
SOD	0.3842	0.461	0.3831			
ABOD	0.9994	0.999	0.9776	0.9982	0.922	0.8881
COPD	0.9273	0.996	0.8105	0.8255	0.803	0.7099
SOS	0.4551	0.37				
FB	0.9942	0.99	0.9692	0.8863	0.839	0.7716
IF	0.9933	0.97	0.89	0.8444	0.834	0.6339
LSCP	0.9992	0.9	0.9832	0.8982	0.827	0.78
XGBOD	0.5			0.5	0.59	
LODA	0.9703	0.99	0.91	0.7766	0.661	0.6286
AE	0.6738	0.73	0.62			
VAE	0.8833	0.78	0.7			
SOGAL	0.4	0.3	0.3			
MOGAL	0.2	0.374	0.34			
V-Detector	0.98	0.99	0.94	0.99	0.86	0.78

Table 12 adversarial attacks on CIFAR and Imagenet detection rate (each class has 200 positive and 200 adversarial samples which classifies as that class by a Alexnet for imagenet and VGG-16 for CIFAR)

Attack	CIFAR 'CAT'	CIFAR 'Truck'	CIFAR 'DOG'	CIFAR 'Ship'	Imagenet 'gorilla'	Imagenet 'hyena'
FGSM	0.93	0.92	0.93	0.92	0.68	0.87
BIM	0.90	0.90	0.91	0.71	0.83	0.82
PGD	0.95	0.92	0.92	0.90	0.73	0.72
MBIM	0.91	0.90	0.94	0.96	0.73	0.72
HSJ	0.84	0.65	0.80	0.65		
JSMA	0.7	0.76	0.7	0.73	0.63	0.62
CW	0.76	0.67	0.66	0.62		

small that they get vanished in rounded values when converting to visual form. Kurkin and Yan Goodfellow in their paper describe this phenomenon of destruction rate by the below equation [47]. Our results in imagenet dataset also effected by this phenomenon.

Comparison with other methods

There are two primary kinds of the way when making defense against adversarial samples, one is Proactive, and another is Reactive. Reactive is detecting the adversarial example before it enters in ML models. An alternative approach is making the ML model better to identify the right class of the adversarial example from the targeted class [29,84]. Defense techniques against adversarial methods can be summarized in three types:

- Denoising strategy or gradient masking: Try to remove the distortions of the image.
- Basic adversarial training: Train the neural network with adversarial example
- Ensemble methods: Add multiple neural network with transformed dataset to combine a majority result

Table 13 Here, we provided a comparison with other adversarial input detection techniques based on accuracy

AML detection method	MNIST				CIFAR				Avg
	FGSM	JSMA	HSJ	CW	FGSM	JSMA	HSJ	CW	
RF [38]	0.96	0.84	0.98	0.66	0.64	0.63	0.60	0.72	0.77
KNN [38]	0.98	0.80	0.98	0.6	0.56	0.52	0.52	0.69	0.73
SVM [38]	0.98	0.89	0.98	–	0.69	0.69	0.64	0.77	0.81
Feature Squeezing [95]	**1.00**	**1.00**		–	0.20	0.88	0.77	–	0.77
Ensemble [10]	0.99	–	0.45	–	0.99	–	0.42	–	0.71
Decision mismatch [59]	0.93	0.93	0.91	–	0.93	**0.97**	0.91	–	0.93
Image quality features [5]	**1.00**	0.90	**1.00**	–	0.72	0.70	0.68	–	0.83
(Our framework)	1.00	1.00	1.00	**1.00**	0.98	0.98	**0.99**	**0.94**	**0.99**

On average, we outperforms other methods. As examples, our methods work with 99% accuracy in the CIFAR data-set where the feature squeezing technique has 0.88% accuracy
Bold indicates best accuracy

In some adversarial defense techniques, well-known robust recognition models are trained on adversarial inputs proactively, performing defensive distillation and training the network with enhanced training data all to create a protection against adversarial example [34,57,66]. For detecting adversarial input, histogram-based methods are also used [68]. In 2017, [19] tested ten defense techniques; by detailed evaluation, they showed that pre-processing techniques could be easily bypassed. In Table 13, we compared our results with other techniques; it is exhibited that our defense's performance is similar to other defense techniques, but our defense technique has some advantages over those like our model does not modify the ML model, it is impossible to have an adaptive attack on our defense. ML model efficiency does not reduce; instead, results get re-verified thus improve trustworthiness. However, the efficiency of our approach largely depends on the individual accuracy of outlier detection methods and noise detection filter sequences.

Adversarial training diminishes the ML model's accuracy and can make the ML model more exposed to generalization [69]. Another disadvantage of Adversarial training based defense techniques is that we need to retrain the model whenever some new attack samples are discovered. It will be hard to update all deployed ML models. Our strategy does not require any dataset not it changes ML anyway, thus no effect on ML model performance. Most preprocessing techniques reduce the adversarial effect before sending it to the ML model. The major drawback of these techniques is that their processing techniques are static; they do not evolve alongside the attack. Our strategy updates itself, it is not vulnerable to this type of adaptive attack. We also have a detection technique module which can detect adaptive attack query. Distillation techniques work by combining the double model, and the second model uses the first model knowledge to improve accuracy. The black-box attack's recent improvement makes this out-of-date defense [22]. The strong transfer-potential of adversarial samples across neural network models [66] is the main reason for this method's collapse. It is not robust as simplistic variation in a neural network can make the system exposed to attacks [18]. The advantage of our approach over defense distillation is we do not need to modify the neural network. Our proposed approach does not need to know or change any ML model layer. So, our model remains the same for both black box and white box attack methods. [39] concluded that combining/ensemble weak defenses does not automatically improve a system's robustness. Also, the ensemble technique remains static and vulnerable to a new attack. Our proposed solution selects defense technique (filer method and outlier detection method) dynamically, thus it is robust and auto-updating decision boundaries also defend against query-based attacks. Feature squeezing [95] method reduces the data, and it reduces the accuracy of the ML model. There is no such reduction in actual model accuracy in our proposed solution. [72] proposed a mechanism to leverage the power of Generative Adversarial Networks to decrease adversarial perturbations' efficiency. The GAN efficiency depends on the GAN training, which is computationally complicated and needs proper datasets, whereas our system does not need a complicated training method. In summary, any commercial product that is using advanced machine/deep/reinforcement learning can benefit from our innovative DF technique.

- Use of commutative dual filtering technique in any AI/ML–based utility applications.
- Use of negative filtering will prevent Trojan AI to change decision resulting in robust AI/ML systems.
- Easy to incorporate in existing and future ML systems will increase adoption and deploy ability.
- Enhanced performance/accuracy and robustness of ML products and online services will increase in diverse applications.
- Improved security will result in quality of experience of users.

Conclusions

We have designed a dual-filtering strategy that does not require any modification to the ML model or information inside the ML model. Our strategy can implement in any ML-based system without costly pre-training. It is to be noted that current adaptive attacks are ineffective in our DF defense. Since our strategy verifies the inputs of the ML model and its output with non-obvious diverse inspection and secondary (outlier) detection. Empirical results exhibited that it could increase the trustworthiness of the ML-based applications. Our experiments were primarily on the computer vision domain, but our DF technique is also suitable for other domains (audio, text, time series). Future work will expand our experiments in different domains and enrich our filter ensemble for better performance. We plan to release this filter collection as a library with our DF framework so that secure learning systems can be developed and deployed. Our technique can be suitably tuned for speed and accuracy; also, as it is independent of the ML, making the DF framework suitable for privacy-preserving applications.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Appendix A: Adversarial attack types

A.1 Gradient-based attacks

This method computes an adversarial image by adding a pixel-wide perturbation of magnitude in the direction of the gradient. This perturbation is computed with a single step, thus is very efficient in terms of computation time [34]. A simple formulation:

$$x' = x + \epsilon \times \text{sign}(\Delta_x J(x, y)) \tag{11}$$

here, x' is the adversarial example that should look similar to x when ϵ is small, and y is the models output. ϵ is a small constant that controls the magnitude of the perturbation, and J denotes the loss function of the model.

Optimization-based attack

The Carlini and Wagner method is a bit different from the above gradient-based methods in that it is an optimization-based attack that constructs adversarial examples by approximately solving the minimization problem [95]. This formulation of the loss function in CW attack can be stated as

$$f(x') = \max(\max\{Z(x_i') : i \neq t\} - Z(x_t'), -k) \tag{12}$$

Here, $Z(x')$ denotes the logits (the outputs of a neural network before the softmax layer) when passing adversarial input (x') and t represents the target misclassification label (the label that we want the adversary to be misclassified as), while k is a constant that controls the desired confidence score .

A.2 Score-based attack

Other usual adversarial images are constructed by perturbing all pixels with an overall constraint on the strength of accumulated modification which they tried to make smaller as possible. But in one pixel or few pixel attack attacker tried to change as much as possible to convert the images to an adversarial image [77]. Here differential evolution (DE) is used, which is a population based optimization algorithm for solving complex multi-modal optimization problems.

A.3 Decision-based attack

Decision-based AAs basic algorithm is it initialized from a point that is already adversarial and then performs a random walk between the adversarial and non-adversarial region in a way that it fill up below criteria,

- (1) It stays in the adversarial region.
- (1) Distance between two image is reduced [13].

Appendix B: Adversarial defense

Angelova and Abu-Mostafam [6] used pruning training sets for learning of object categories they applied to bootstrap and Naïve Bayes algorithm. Brückner and Scheffer use of game theory in 2011 also shows a diverse approach in developing input filters [15]. Goodfellow et al. [34] tried to training on adversarial inputs pro-actively, Papernot et al. performed

defensive distillation [66] and Miyato et al. training the network with enhanced training data all to create a protection against adversarial example [57].

Grosse et al. [35] did statistical tests using a complementary approach to identify specific inputs, that are adversarial. Wong et al. showed convex outer adversarial polytope can be a proven defense [92]. Lu et al. [52] checked whether the depth map is consistent or not (only for image) to detect adversarial examples. Metzen et al. implemented deep neural networks with a small "detector" sub-network were trained on the binary classification task of distinguishing factual data from data containing adversarial perturbations [56]. The same year, Madry et al. [55] published a paper on adversarial robustness of neural networks through the lens of robust optimization. Chen et al. tried to devise adversarial examples with another guardian neural net distillation as a defense from AAs [24]. Wu et al. [93] developed highly confident near neighbor (HCNN), a framework that combines confidence information and nearest neighbor search, to reinforce adversarial robustness of a base model. Also Paudice et al. [67] applied anamoly detection and Zhang et al. detected adversarial examples by identifying significant pixels for prediction which only work for images [98]. Other researchers such as Wang et al. tried with mutation testing [89] and Zhao et al. developed key-based network, a new detection-based defense mechanism to distinguish adversarial examples from normal ones based on error correcting output codes, using the binary code vectors produced by multiple binary classifiers applied to randomly chosen label-sets as signatures to match standard images and reject adversarial examples [99]. Later that year Liu et al. tried to use steganalysis [50] and Katzir et al. implemented a filter by constructing euclidean spaces out of the activation values of each of the deep neural network layers with k-nearest neighbor classifiers (k-NN) [44]. A different notable strategy was taken by researchers Pang et al. They used thresholding approach as the detector to filter out adversarial examples for reliable predictions [63]. For an image classification problem, Tian et al. did image transformation operations such as rotation and shifting to detect adversarial examples [82] and Xu et al. [95] simply reduced the feature space to protect against adversary. Monteiro et al [59] developed inputfiler which is based on bi-model decision mismatch of image. Sumanth Dathathri showed whether prediction behavior is consistent with a set of fingerprints (a data set of NN) named NFP method [31]. Same year, Crecchi et al. used non-linear dimensionality reduction and density estimation techniques [27] and Aigrain et al. tried to use confidence value in CNN [3]. Some other notable works in that year were meta-learning based robust detection method to detect new AAs with limited examples developed by Ma et al. [54]. Another important and effective work was done by Chen et al., where they tried to keep the records of query and used KNN to co-relate that with adversarial examples [25].

References

1. Adam GA, Smirnov P, Goldenberg A, Duvenaud D, Haibe-Kains B (2018) Stochastic combinatorial ensembles for defending against adversarial examples, pp 1–15. arXiv preprint arXiv:1808.06645
2. Aggarwal CC (2015) Outlier analysis. Aggarwal CC (ed) Data mining. Springer , Cham, pp 237–263
3. Aigrain J, Detyniecki M (2019) Detecting adversarial examples and other misclassifications in neural networks by introspection. arXiv preprint arXiv:1905.09186
4. Akhtar Z, Monteiro J, Falk TH (2018) Adversarial examples detection using no-reference image quality features. In: 2018 international Carnahan conference on security technology (ICCST), pp 1–5
5. Akhtar Z, Monteiro J, Falk TH (2018) Adversarial examples detection using no-reference image quality features. In: 2018 international Carnahan conference on security technology (ICCST). IEEE, pp 1–5
6. Angelova A, Abu-Mostafam Y, Perona P (2005) Pruning training sets for learning of object categories. In: 2005 IEEE Computer Society conference on computer vision and pattern recognition (CVPR'05), vol 1. IEEE, pp 494–501
7. Arning A, Agrawal R, Raghavan P (1996) A linear method for deviation detection in large databases. KDD 1141:972–981
8. Athalye A, Carlini N (2018) On the robustness of the cvpr 2018 white-box adversarial example defenses. arXiv preprint arXiv:1804.03286
9. Athalye A, Carlini N, Wagner D (2018) Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv preprint arXiv:1802.00420
10. Bagnall A, Bunescu R, Stewart G (2017) Training ensembles to detect adversarial examples. arXiv preprint arXiv:1712.04006
11. Barr DR, Davidson T (1973) A Kolmogorov–Smirnov test for censored samples. Technometrics 15(4):739–757
12. Bradshaw J, Matthews AGdG, Ghahramani Z (2017) Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks, pp 1–33. arXiv preprint arXiv:1707.02476
13. Brendel W, Rauber J, Bethge M (2017) Decision-based adversarial attacks: reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248
14. Breunig MM, Kriegel HP, Ng RT, Sander J (2000) Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp 93–104
15. Brückner M, Scheffer T (2011) Stackelberg games for adversarial prediction problems. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 547–555
16. Carlini N (2019) Lessons learned from evaluating the robustness of defenses to adversarial examples. USENIX Association, Santa Clara
17. Carlini N, Athalye A, Papernot N, Brendel W, Rauber J, Tsipras D, Goodfellow I, Madry A, Kurakin A (2019) On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705
18. Carlini N, Wagner D (2016) Defensive distillation is not robust to adversarial examples. arXiv preprint arXiv:1607.04311
19. Carlini N, Wagner D (2017) Adversarial examples are not easily detected: bypassing ten detection methods. In: Proceedings of the 10th ACM workshop on artificial intelligence and security, pp 3–14
20. Carlini N, Wagner D (2017) Magnet and efficient defenses against adversarial attacks" are not robust to adversarial examples. arXiv preprint arXiv:1711.08478

21. Carlini N, Wagner D (2018) Audio adversarial examples: targeted attacks on speech-to-text. In: 2018 IEEE security and privacy workshops (SPW). IEEE, pp 1–7
22. Chakraborty A, Alam M, Dey V, Chattopadhyay A, Mukhopadhyay D (2018) Adversarial attacks and defences: a survey. arXiv preprint arXiv:1810.00069
23. Chen J, Jordan MI, Wainwright MJ (2019) Hopskipjumpattack: a query- decision-based attack. arXiv preprint arXiv:1904.02144 **3**
24. Chen J, Meng Z, Sun C, Tang W, Zhu Y (2017) Reabsnet: detecting and revising adversarial examples. arXiv preprint arXiv:1712.08250
25. Chen S, Carlini N, Wagner D (2019) Stateful detection of black-box adversarial attacks. arXiv preprint arXiv:1907.05587
26. Chen Y, Zhou XS, Huang TS (2001) One-class svm for learning in image retrieval. In: Proceedings 2001 international conference on image processing (Cat. No. 01CH37205), vol 1. IEEE, pp 34–37
27. Crecchi F, Bacciu D, Biggio B (2019) Detecting adversarial examples through nonlinear dimensionality reduction. arXiv preprint arXiv:1904.13094
28. Dasgupta D (1994) Handling deceptive problems using a different genetic search. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE World congress on computational intelligence. IEEE, pp 807–811
29. Dasgupta D, Akhtar Z, Sajib S (2020) Machine learning in cybersecurity: a comprehensive survey. J Defense Model Simul Appl Methodol Technol. https://doi.org/10.1177/1548512920951275
30. Dasgupta D, KrishnaKumar K, Wong D, Berry M (2004) Negative selection algorithm for aircraft fault detection. In: International conference on artificial immune systems. Springer, pp 1–13
31. Dathathri S, Zheng S, Murray RM, Yue Y (2018) Detecting adversarial examples via neural fingerprinting. arXiv preprint arXiv:1803.03870
32. Goldstein M, Dengel A (2012) Histogram-based outlier score (hbos): a fast unsupervised anomaly detection algorithm. KI-2012: poster and demo track, pp 59–63
33. Gong Z (2018) Adversarial algorithms in tensorflow, v0.2.0. Zenodo. https://doi.org/10.5281/zenodo.1154272
34. Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples, pp 1–11. arXiv preprint arXiv:1412.6572
35. Grosse K, Manoharan P, Papernot N, Backes M, McDaniel P (2017) On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280
36. Gupta KD, Dasgupta D, Akhtar Z (2020) Applicability issues of evasion-based adversarial attacks and mitigation techniques. In: 2020 IEEE symposium series on computational intelligence (SSCI). IEEE, pp 1506–1515
37. Gupta KD, Dasgupta D, Akhtar Z (2020) Determining sequence of image processing technique (ipt) to detect adversarial attacks. arXiv preprint arXiv:2007.00337
38. Hayes J, Danezis G (2017) Machine learning as an adversarial service: learning black-box adversarial examples. arXiv preprint arXiv:1708.05207 **2**
39. He W, Wei J, Chen X, Carlini N, Song D (2017) Adversarial example defense: ensembles of weak defenses are not strong. In: 11th USENIX workshop on offensive technologies (WOOT 17)
40. He Z, Xu X, Deng S (2003) Discovering cluster-based local outliers. Pattern Recogn Lett 24(9–10):1641–1650
41. Ilyas A, Santurkar S, Tsipras D, Engstrom L, Tran B, Madry A (2019) Adversarial examples are not bugs, they are features. In: Advances in neural information processing systems, pp 125–136
42. Janssens J, Huszár F, Postma E, van den Herik H (2012) Stochastic outlier selection. Technical report
43. Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ (2017) Reluplex: an efficient smt solver for verifying deep neural networks. In: Computer aided verification, pp 97–117
44. Katzir Z, Elovici Y (2018) Detecting adversarial perturbations through spatial behavior in activation spaces. arXiv preprint arXiv:1811.09043
45. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114
46. Kriegel HP, Schubert M, Zimek A (2008) Angle-based outlier detection in high-dimensional data. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 444–452
47. Kurakin A, Goodfellow I, Bengio S (2016) Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533
48. Lazarevic A, Kumar V (2005) Feature bagging for outlier detection. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pp 157–166
49. Li Z, Zhao Y, Botta N, Ionescu C, Hu X (2020) Copod: copula-based outlier detection. arXiv preprint arXiv:2009.09463
50. Liu J, Zhang W, Zhang Y, Hou D, Liu Y, Zha H, Yu N (2018) Detection based defense against adversarial examples from the steganalysis point of view. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4825–4834
51. Liu Y, Li Z, Zhou C, Jiang Y, Sun J, Wang M, He X (2019) Generative adversarial active learning for unsupervised outlier detection. IEEE Trans Knowl Data Eng 32:1517–1528
52. Lu J, Issaranon T, Forsyth D (2017) Safetynet: detecting and rejecting adversarial examples robustly. In: Proceedings of the IEEE international conference on computer vision, pp 446–454
53. Lu J, Sibai H, Fabry E, Forsyth D (2017) No need to worry about adversarial examples in object detection in autonomous vehicles. arXiv preprint arXiv:1707.03501
54. Ma C, Zhao C, Shi H, Chen L, Yong J, Zeng D (2019) Metaadvdet: towards robust detection of evolving adversarial attacks. arXiv preprint arXiv:1908.02199
55. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2017) Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083
56. Metzen JH, Genewein T, Fischer V, Bischoff B (2017) On detecting adversarial perturbations. arXiv preprint arXiv:1702.04267
57. Miyato T, Dai AM, Goodfellow I (2016) Adversarial training methods for semi-supervised text classification. arXiv preprint arXiv:1605.07725
58. Miyato T, Maeda S, Koyama M, Ishii S (2019) Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE Trans Pattern Anal Mach Intell 41(8):1979–1993
59. Monteiro J, Akhtar Z, Falk TH (2018) Generalizable adversarial examples detection based on bi-model decision mismatch. arXiv preprint arXiv:1802.07770
60. Narodytska N, Kasiviswanathan SP (2016) Simple black-box adversarial perturbations for deep networks. arXiv preprint arXiv:1612.06299
61. Nguyen HH, Kuribayashi M, Yamagishi J, Echizen I (2019) Detecting and correcting adversarial images using image processing operations and convolutional neural networks. arXiv preprint arXiv:1912.05391
62. Nicolae MI, Sinn M, Tran MN, Buesser B, Rawat A, Wistuba M, Zantedeschi V, Baracaldo N, Chen B, Ludwig H, Molloy I, Edwards B (2018) Adversarial robustness toolbox v1.1.1. CoRR arXiv:1807.01069
63. Pang T, Du C, Dong Y, Zhu J (2018) Towards robust detection of adversarial examples. In: Advances in neural information processing systems, pp 4579–4589
64. Papernot N, Faghri F, Carlini N, Goodfellow I, Feinman R, Kurakin A, Xie C, Sharma Y, Brown T, Roy A, Matyasko A, Behzadan V, Hambardzumyan K, Zhang Z, Juang YL, Li Z, Sheatsley R, Garg A, Uesato J, Gierke W, Dong Y, Berthelot D, Hendricks P, Rauber J, Long R (2018) Technical report on

the cleverhans v2.1.0 adversarial examples library. arXiv preprint arXiv:1610.00768
65. Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks. IEEE symposium on security and privacy (SP), pp 582–597
66. Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE symposium on security and privacy (SP). IEEE, pp 582–597
67. Paudice A, Muñoz-González L, Gyorgy A, Lupu EC (2018) Detection of adversarial training examples in poisoning attacks through anomaly detection. arXiv preprint arXiv:1802.03041
68. Prakash A, Moran N, Garber S, DiLillo A, Storer J (2018) Deflecting adversarial attacks with pixel deflection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8571–8580
69. Raghunathan A, Xie SM, Yang F, Duchi JC, Liang P (2019) Adversarial training can hurt generalization. arXiv preprint arXiv:1906.06032
70. Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp 427–438
71. Ruff L, Vandermeulen R, Goernitz N, Deecke L, Siddiqui SA, Binder A, Müller E, Kloft M (2018) Deep one-class classification. In: International conference on machine learning, pp 4393–4402
72. Samangouei P, Kabkab M, Chellappa R (2018) Defense-gan: protecting classifiers against adversarial attacks using generative models. arXiv preprint arXiv:1805.06605
73. Settles B (2009) Active learning literature survey. CS Technical Reports, Department of Computer Sciences, University of Wisconsin-Madison
74. Shaham U, Garritano J, Yamada Y, Weinberger E, Cloninger A, Cheng X, Stanton K, Kluger Y (2018) Defending against adversarial images using basis functions transformations, pp 1–12. arXiv preprint arXiv:1803.10840
75. Soll M, Hinz T, Magg S, Wermter S (2019) Evaluating defensive distillation for defending text processing neural networks against adversarial examples. In: International conference on artificial neural networks. Springer, pp 685–696
76. Strauss T, Hanselmann M, Junginger A, Ulmer H (2017) Ensemble methods as a defense to adversarial perturbations against deep neural networks, pp 1–10. arXiv preprint arXiv:1709.03423
77. Su J, Vargas DV, Sakurai K (2019) One pixel attack for fooling deep neural networks. IEEE Trans Evol Comput 23:828–841
78. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. CoRR. arXiv:1312.6199
79. Tabassi E, Burns K, Hadjimichael M, Molina-Markham A, Sexton J (2019) A taxonomy and terminology of adversarial machine learning. https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8269-draft.pdf
80. Tanay T, Griffin L (2016) A boundary tilting perspective on the phenomenon of adversarial examples. arXiv preprint arXiv:1608.07690
81. Tang J, Chen Z, Fu AWC, Cheung DW (2002) Enhancing effectiveness of outlier detections for low density patterns. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, pp 535–548
82. Tian S, Yang G, Cai Y (2018) Detecting adversarial examples through image transformation. In: Thirty-second AAAI conference on artificial intelligence
83. Togbe MU, Barry M, Boly A, Chabchoub Y, Chiky R, Montiel J, Tran VT (2020) Anomaly detection for data streams based on isolation forest using scikit-multiflow. In: International conference on computational science and its applications. Springer, pp 15–30
84. Tramèr F, Boneh D (2019) Adversarial training and robustness for multiple perturbations. In: Advances in neural information processing systems, pp 5858–5868
85. Tramer F, Carlini N, Brendel W, Madry A (2020) On adaptive attacks to adversarial example defenses. arXiv preprint arXiv:2002.08347
86. Tramèr F, Kurakin A, Papernot N, Goodfellow I, Boneh D, McDaniel P (2017) Ensemble adversarial training: attacks and defenses, pp 1–20. arXiv preprint arXiv:1705.07204
87. Uesato J, O'Donoghue B, Oord Avd, Kohli P (2018) Adversarial risk and the dangers of evaluating against weak attacks. arXiv preprint arXiv:1802.05666
88. Umbarkar AJ, Sheth PD (2015) Crossover operators in genetic algorithms: a review. ICTACT J Soft Comput 6(1):1083–1092
89. Wang J, Sun J, Zhang P, Wang X (2018) Detecting adversarial samples for deep neural networks through mutation testing. arXiv preprint arXiv:1805.05010
90. Wang X, Jin H, He K (2019) Natural language adversarial attacks and defenses in word level. arXiv preprint arXiv:1909.06723 pp. 1–15
91. Wiyatno R, Xu A (2018) Maximal jacobian-based saliency map attack. arXiv preprint arXiv:1808.07945
92. Wong E, Kolter JZ (2017) Provable defenses against adversarial examples via the convex outer adversarial polytope. arXiv preprint arXiv:1711.00851
93. Wu X, Jang U, Chen J, Chen L, Jha S (2018) Reinforcing adversarial robustness using model confidence induced by adversarial training. In: International conference on machine learning, pp 5330–5338
94. Xie C, Wang J, Zhang Z, Ren Z, Yuille A (2017) Mitigating adversarial effects through randomization. arXiv preprint arXiv:1711.01991 pp. 1–16
95. Xu W, Evans D, Qi Y (2017) Feature squeezing: detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155
96. Yuan X, He P, Zhu Q, Li X (2019) Adversarial examples: attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst 30(9):2805–2824
97. Zeng Q, Su J, Fu C, Kayas G, Luo L, Du X, Tan CC, Wu J (2019) A multiversion programming inspired approach to detecting audio adversarial examples. In: 2019 49th annual IEEE/IFIP international conference on dependable systems and networks (DSN), pp 39–51
98. Zhang C, Ye Z, Wang Y, Yang Z (2018) Detecting adversarial perturbations with saliency. In: 2018 IEEE 3rd international conference on signal and image processing (ICSIP). IEEE, pp 271–275
99. Zhao P, Fu Z, Hu Q, Wang J et al (2018) Detecting adversarial examples via key-based network. arXiv preprint arXiv:1806.00580
100. Zhao Y, Hryniewicki MK (2018) Xgbod: improving supervised outlier detection with unsupervised representation learning. In: 2018 international joint conference on neural networks (IJCNN). IEEE, pp 1–8
101. Zhao Y, Nasrullah Z, Li Z (2019) Pyod: a python toolbox for scalable outlier detection. J Mach Learn Res **20**(96), 1–7. http://www.jmlr.org/papers/v20/19-011.html
102. Zhou G, Lu J, Wan CY, Yarvis MD, Stankovic JA (2008) Body sensor networks. MIT Press, Cambridge