Adversarial Input Detection Using Image Processing Techniques (IPT)

Kishor Datta Gupta
Computer Science
University of Memphis
Memphis, TN, USA
kgupta1@memphis.edu

Dipankar Dasgupta
Computer Science
The University of Memphis
Memphis, TN, USA
ddgupta@memphis.edu

Zahid Akhtar
Computer Science Department
State University of New York Polytechnic Institute
Newyork, USA
akhtarz@sunypoly.edu

***Abstract*—Modern deep learning models for the computer vision domain are vulnerable against adversarial attacks. Image prepossessing technique based defense against malicious input is currently considered obsolete as this defense is not effective against all types of attacks. The advanced adaptive attack can easily defeat pre-processing based defenses. In this paper, we proposed a framework that will generate a set of image processing sequences (several image processing techniques in a series). We randomly select a set of Image processing technique sequences (IPTS) dynamically to answer the obscurity question in testing time. This paper outlines methodology utilizing varied datasets examined with various adversarial data manipulations. For specific attack types and dataset, it produces unique IPTS. The outcome of our empirical experiments shows that the method can efficiently employ as processing for any machine learning models. The research also showed that our process works against adaptive attacks as we are using a non-deterministic set of IPTS for each adversarial input.**

***Index Terms*—Adversarial machine learning, Neural network, Image processing, AI security**

I. Introduction

Acronym	FullMeaning	Acronym	Full Meaning
BS	Bilateral smoothing	BPDA	BackPass Differential Approximation
AS	Adaptive Smoothing	PGD	Projected Gradient disent
AN	Additive Noise	BIM	Basic Iterative method
FGSM	First Gradient Sign Method	MBIM	Momentum Basic Iterative Method
JSMA	Jacob Saliency Map Method	SIPTS	Set of Image Processing Technique Sequence
DF	Deep Fool method	DI	Pixel Difference between before and after IPTS applied
HSJ	HopSkipJump	ED	Euclidean distance
MLM	Machine Learning Model	LBP	Local binary pattern
IPT	Image Processing Techniques	PDE	Probability Density Equation
IPTS	Image Processing Technique Sequence	TN	Thining
PX	Pixellate	GS	Grey-scaled
SIPTS	Set of Image Processing Sequence	AML	Adversarial machine learning

TABLE I
ACRONYM USED IN THIS PAPER

Generally speaking, adversarial examples are input data that get misclassified by an ML/AI system, but not by a human subject. Several defense methods against adversarial attacks, such as adversarial training, defensive distillation, etc., have been proposed [1]. These methods have some drawbacks (Example: reduce model accuracy, enormous computation cost, impracticability, etc. [2], which our proposed method can mitigate.

In this study, we explore several metrics such as histogram of the difference image (DI), Euclidean distances (ED), cross-entropy (CE) value, loss function, and probability density value to measure the effectiveness of IPTS. Specifically, DI is the pixel-wise difference image of the original input image and the processed input image. A histogram is obtained by plotting the color range on the x-axis and the number of pixels with a specific color on the y-axis. The average Histogram (Havg) is the average value of the y-axis. Our proposed solution is to obtain a GA that will determine a set of IPTS sequences IPTS and Havg range. Based on a Havg value, we can classify as adversarial input within an estimated range. The process is presented in figure 2. We found that different sequences of IPTS are effective against different attack types, as shown in table II. Determining these sequences is hard as only with 4 IPT (any IPT can be repeated), then the total possible sequences (considering sequence's max length 8) would be $1^4 + 2^4 + 3^4 + 4^4 + \ldots + 8^8 = 24,684,612$. Thus, finding the right sequence from that many possible sequences is a massive task. We have attempted to solve using a GA because it has less computation time to do an exhaustive search and better result than any random search.

The main research contributions of this paper are

- We proved that 'difference metrics' of the processed and original images could detect the adversarial attack.
- We observed that different image processing technique sequences are effective against various types of adversarial attacks.
- We demonstrate that a GA can select a set of suitable image processing technique sequences to counter an adversarial attack.
- We provided a dynamic selection of IPTS to counter adaptive attack.

In section II, the adversarial attacks, the existing adversarial defense, IPTs, and the basics of GA are described. The proposed method with the threat model and its implementation details are presented in section III. The experimental results

Adversarial Attack	Sequence (MNIST)	Sequence(CIFAR)	Reason
FSGM	BS+BS	Greysale+AN+AN	Perturbation which added in image is not an edge, edge-preserving algorithm remove those, so differences of before after adversarial image inputs are higher than non-adverse image
JSMA	TN+BS	PX+greyscale+BS	JSMA extend some edge which reduced when we do thining
CW	TN+TN+TN or GS+GS+GS+TBS	GS+greyscale +GS+TBS+TN+TN	In CW object edge getting thining and blur,so applying more blur and thining algorithm diffence will apply
DeepFool	GS+AN+GS+AN	GreySale+AN+AN+AN	Deepfool create few pixel around border to effect the model, using additive noise boost this effect and we can have a difference

TABLE II
DIFFERENT IMAGE PROCESSING TECHNIQUE SEQUENCE FOR DIFFERENT ADVERSARIAL ATTACK TYPES AND THE REASON EXPLAINED

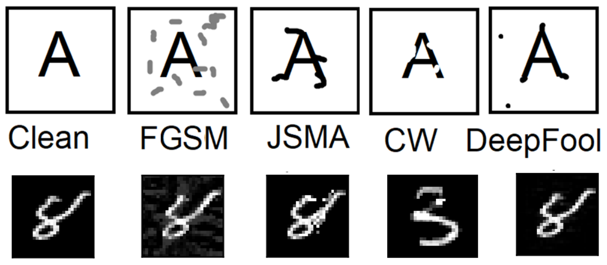

Fig. 1. The first rows show how different types of attack change a clean image (for visual purpose, the effects are exaggerated. This row is created from observation and not real sample). The second rows are actual corresponding real attacks on the MNIST dataset. Where 8 recognized as a different label.

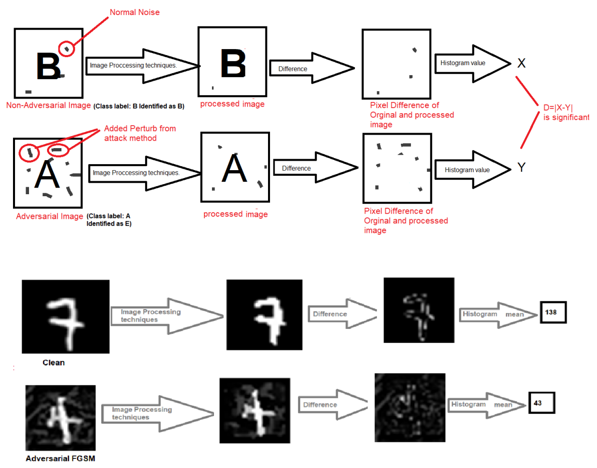
Normal Noise
Image Proccessing techniques.
Difference
Histogram value
X
Non-Adversarial Image (Class label: B Identified as B)
processed image
Pixel Difference of Orginal and processed image
D=|X-Y| is significant
Added Perturb from attack method
Adversarial Image (Class label: A Identified as E)
Y
Image Processing techniques
Histogram mean
138
Clean
43
Adversarial FGSM

Fig. 2. On the top side, clean (class A) vs. adversarial (B, which classified as other class) images differences after IPTS applied histogram calculation on the difference. At the bottom, MNIST with the FGSM example has shown.

and discussions of the study are given in section IV and in section VI, and conclusions are drawn with our limitations and future work plan.

II. BACKGROUND

This section highlighted related works for the adversarial input detections and the background of adversarial attacks.

A. Related work

Defenses against adversarial attacks can be classified in two ways: attack detection and robust recognition model. Detection is a binary classification problem, where input is classified as an adversarial or not. In robust recognition methods, the correct class of an adversarial image is recognized where the attack classed are known prior. This work is focused on the detection technique. Well-known robust recognition models are training on adversarial inputs proactively [3], performing defensive distillation [4], training the network with enhanced training data all to create a protection against adversarial example [5]. Image histogram-based [6] methods are also used to detect adversarial inputs. In [7], authors proposed an adversarial attack detection scheme based on image quality related features to detect various adversarial attacks. Carlini et. al. [8] tested ten defense techniques, by detailed evaluation they showed that pre-processing techniques can be easily bypassed. Also, Akhtar et. al. [9] provided a brief summary of most attacks and described all defense techniques.

B. Adversarial attack

Rauber et. al. [10] mentioned three types of adversarial image manipulation attacks, i.e., gradient, score, and decision based. In particular, the 'First Gradient Sign Method' (FGSM), is a gradient based strategies, which was proposed by Ian Goodfellow in 2014. Carlini and Wagner (CW) [11] uses an iterative attack that constructs adversarial examples by approximately solving the minimization problem. The Jacobian-based saliency map attack [4] is another type of the adversarial attack for deceiving classification models. A saliency map is an image that shows each pixel's unique quality. The purpose of a saliency map is to simplify and improve the representation of a picture into something more significant and better suitable to analyze. For instance, if a pixel has a high grey level or other unique color quality in an image, that pixel's class will show in the saliency map and in an obvious way. DeepFool [12](DF) attacks find the closest distance from the original input to the decision boundary.

III. PROPOSED FRAMEWORK

In this section, we considered the threat model for our defense, basic hypothesis of our techniques and different steps of our proposed framework.

A. Threat model

Yuan et al. . [13] suggested that making threat models consist of Adversarial Falsification (False negative, False Positive), white-box, BlackBox, targeted, non- targeted, onetime and iterative attacks. Whereas, Carlini et al. [14], proposed that adversarial attack and defense models need to be tested against

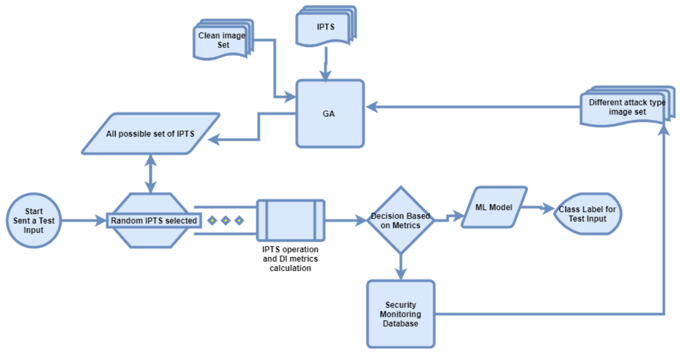
Clean image Set
IPTS
GA
Different attack type image set
All possible set of IPTS
Start Sent a Test Input
Random IPTS selected
IPTS operation and DI metrics calculation
Decision Based on Metrics
ML Model
Class Label for Test Input
Security Monitoring Database

Fig. 3. Illustrates the overall flow diagram of our adversarial input detection technique. We used a genetic search to find IPTS, which can detect manipulation attacks on image datasets. During testing, a randomly picked SIPTS will be selected for pre-processing the image. If the Image is not Adversarial, it will go to the ML system; otherwise, its information will be stored in a Security monitoring tool. **We are sending the original input to the ML model, not the filtered/processed input.**

a diverse set of attacks. Also, they need to be evaluated against adaptive attacks. Moreover, Tramer et al. [15] suggested different themes to evaluate a defense model. Keeping these guidelines in mind, we developed our threat model inclusive of basic(high noise), advanced attack(low noise), and adaptive attack(against our proposed defense) types. To compare our defense work against suggested attack types, we tested with basic iterative methods (BIM) [16] and their variations (e.g., MBIM [17] and PGD). We selected CW2 and DF attacks as advanced attacks. Carlini et al. [8] recommended using at least one gradient-free and one hard-label attack. To address that concern, we also evaluated our proposed method with gradient-free attacks such as local search attack [18] and hop-skip-jump attack [19]. For testing against an adaptive attack, we used BPDA (Backward Pass Differential Approximation) [20]. We used our defense technique that works against FGSM and BIM attacks to generate BPDA attacks. As other researchers [14] pointed out, 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 [15] recommend. Thus, our threat model combines gradient-based, gradient-free, and adaptive evasion based adversarial attacks on multiple datasets.

B. Our Hypothesis

Adversarial examples are input data that get misclassified by a ML/AI system but not by a human subject. For a ML model M, if A is Non adversarial input added noise is ϵ, adversarial example is $A_x = A + \epsilon$.

Adversarial samples are distorted versions of non-adversarial samples. This distortion can be measured in euclidean distances or pixel difference distance. However, we assume that some image processing algorithms can signify the misuse of an adversarial example so that significance can be used to filter out adversarial examples. For example, if an adversarial image has added perturb pixel all around its leading edges, an edge-preserving technique algorithm can remove these perturbs. So, differences after-before image have a significant value, which makes it distinguishable from a non-adversarial picture.

Let, adversarial image sets are $A_s + \epsilon_s$, and clean image set is A_s, and IPTS denoted by F_{ipts}, ϵ_s is total added perturb of all adversarial images So, after applied IPTS on adversarial and clean image set we will get,
$F_{ipts}(A_s + \epsilon_s) \approx A_s + \epsilon_s + K_a$
$F_{ipts}(A_s) \approx A_s + K$
where K, K_a is the approximate effect of IPTS in clean and adversarial image set So our difference DI is
$DI \approx |(A_s + \epsilon_s) - (A_s + \epsilon_s + K_a)) - (A_s - A_s - K)|$
$DI \approx A_s + \epsilon_s - A_s - \epsilon_s - K_a - A_s + A_s + K|$
$DI \approx |K - K_a|$ We can see that in our DI equation no image (A or A_s) is present. our aim is analyze the effects not the core image/image content but pre-processing effects. Note: K, K_a is the generalized approximate effect of IPTS in clean and adversarial image set,

In testing time, we will calculate the k_i value for the testimage i by applying F_{ipts}.Here k_i is the generalize effects from applied IPTS. Now i will be an adversarial image if $|K - k_i| > |k_a - k_i|$.

C. Our Methodology

In this work, we developed a denoising approach to detect adversarial inputs using a sequence of image filters with the inspiration of Prakas et al. . [6]. But the key differences are that our process is automated. Our method does not need to consider if the attacks are Whitebox, Blackbox, targeted, non-targeted, or how the attack samples have been generated. Therefore, our approach is independent of the attack type, frequency, or models. Our method determines a suitable IPT sequence, which can detect adversarial image different attack types. From a cyber-security point of view, it is not enough to see an attack. It is also necessary to know the attack type. This knowledge could help later on to assess the attacker's resources and help further to optimize defense. Adversarial samples are distorted versions of non-adversarial samples. Such distortions can be measure in Euclidean distances or pixel difference distance. We assume that some IPT can signify the distortions of an adversarial example so that significance can be used to filter out adversarial examples. For instance, if an adversarial image has added perturbed pixels all around its leading edges, an edge-preserving technique can remove these perturbations. The differences before and after processed images can be measured by different metrics (e.g., Histogram difference). These metrics will have significantly different values from the clean dataset's same metrics. We can differentiate between non-adversarial and adversarial images using these metrics. It could be possible that one unique sequence of edge-preserving and other IPT could make it more distinguishable than different sequences, thereby providing metrics of such differences to be used as a threshold value to classify adversarial and non-adversarial images. As many IPTS exist, we use GA to find out one of the most appropriate ones. After that, we will develop

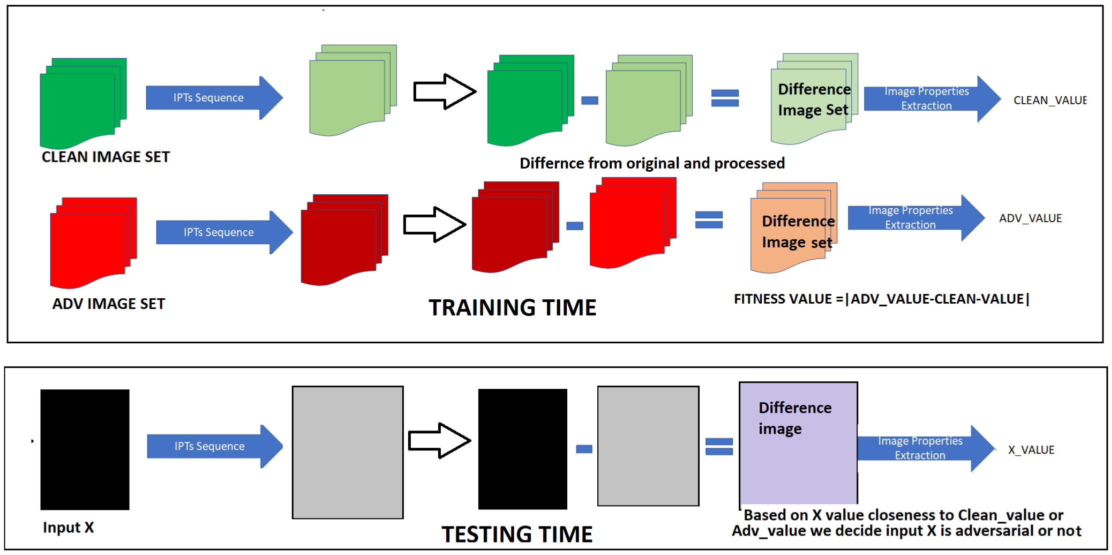
IPTs Sequence
CLEAN IMAGE SET
Difference Image Set
Image Properties Extraction
CLEAN_VALUE
Differnce from original and processed
ADV IMAGE SET
ADV_VALUE
TRAINING TIME
FITNESS VALUE =|ADV_VALUE-CLEAN-VALUE|
Input X
Difference image
X_VALUE
TESTING TIME
Based on X value closeness to Clean_value or Adv_value we decide input X is adversarial or not

Fig. 4. Sequence generation in training and testing time. A set of adversarial and clean processed using the same IPT sequence and provide different value, the distance of these values used in testing time. We used GA to determine the best possible IPTS. So these values are also used as fitness values in GA.

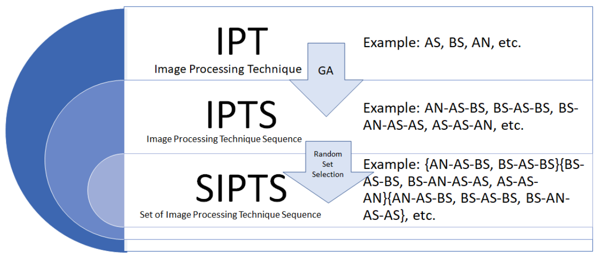
IPT
Image Processing Technique
GA
Example: AS, BS, AN, etc.
IPTS
Image Processing Technique Sequence
Example: AN-AS-BS, BS-AS-BS, BS-AN-AS-AS, AS-AS-AN, etc.
Random Set Selection
SIPTS
Set of Image Processing Technique Sequence
Example: {AN-AS-BS, BS-AS-BS}{BS-AS-BS, BS-AN-AS-AS, AS-AS-AN}{AN-AS-BS, BS-AS-BS, BS-AN-AS-AS}, etc.

Fig. 5. Illustrates the steps to generate SIPTS from IPT. At first, using GA, we determine appropriate IPTS. As IPTS and attack types have many to many relations, Randomly choose one of the sets which are effective against all attack types.

a set of IPTS (SIPTS) which cover all attack types and select them dynamically in testing time for detection. In diagram 3, we provided a diagram of the proposed methodology. The diagram shows how GA is generating sequence and fitness values. In figure 4, we presented how we are generating the range or fitness value from the different image sets for different IPTS.

D. Problem Formulation

We need to determine a set of IPT $f(..)$, which can generate an optimal value of $\bar{\delta_h}$ and σ_{δ_h} for a set of adversarial images $A_{x_{(i=0...n)}}$ and a non-adversarial set of image $A_{y_{(i=0...n)}}$ that we can use them to distinguish from non-adversarial image. Let IPT be $f_1, f_2 ... f_n$. Same algorithm can be applied multiple times. The total number of algorithms applied for a set F be T, where $T = count(f_1) + count(f_2) ... + count(f_n)$. For example, three IPTs are $f1, f2, f3$ and an optimal set for $F(f2, f3, f1, f1, f2, f1)$, here T is 6. So, our input is adversarial A_x and non-adversarial A_y set, and our output will be $f(..)$. and threshold value of different difference metrics T_h (e.g., histogram average). When a test input will come, we will run $f(..)$ and determine the T_{h_t}. If $T_h > T_{h_t}$ then it is an adversarial, otherwise it is a clean example.

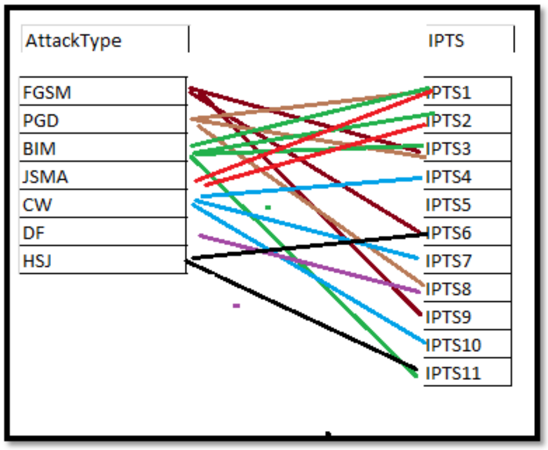

Fig. 6. Many to Many relations between attack types and IPTS which works against these attacks.

E. Selection of IPT and IPTS

Numerous IPT can perform different operations in images. So, for a specific type of attack, a specific type of IPT sequence will work better. From our empirical observation, we selected a diverse set of unique IPT's such as BS, thinning [21], AN, blur [22], sharpen [23], thickening [21], and AS. These techniques and their combinations are best suited to maximize enhancement between the difference image (DI)'s of adversarial and clean input. It can be observed in Figure 1. Here, for the MNIST dataset, the FGSM method tends to pixelate noise around the object in the image, while JSMA seems to add along with the object borders, and CW tends to lose some erosion DF adds a minimal amount of perturbing. Therefore, we assumed that sharpening could help highlight JSMA attacks, performing blur will have more effect in CW, and AS will have a good effect on FGSM attacks. We can see different IPT can work better for different attack types. So, we found a combination of these IPTS that performs better. If we consider different datasets (e.g., ImageNet, CFIAR), we have to consider different IPTS. We select a variety of sets of IPTS, which can perform required all basic operations. GA helps us to find suitable SIPTS. If we do not encode an IPT in the GA population, it will not be presented in any possible solution of the IPT sequence. Hence, the selection of IPT to generate the initial population is very important. Next step

IPT	2bit	3bit	IPT	2bit	3bit
Adaptive Smooth	01	001	Thining	-	100
Bilateral Smooth	10	010	Pixellete	-	101
Additive noise	11	011	Blur	-	110
Do-Nothing	00	000	Sharpen	-	111

TABLE III
GA ENCODING: 2BIT FOR MNIST, 3 BIT FOR CIFAR & IMAGENET.

is selection of Image Processing Technique sequence(IPTS) .First, we will create our individuals from the set of filters. We will denote every IPT with binary digits, and these will work as a chromosome. If the encoding of IPTs are $F1, F2, ..., F3$ then encoding can be seen as F1=000, F2=001, F3=010, F4=011. In practices, we encoded the filters as presented in table III. Now, we will randomly generate the populations from the combinations of the IPTs. Here $I1, I2, ..., In$ are individuals with total population N. Lets encoded them as I1= F2F1F2F4F2..Fi = 01010010101..1, I2= F1F2F2F4F3..Fi = 11010010101..1, I3= F2F2F1F4F3..Fi = 01011010101..1, I4= F1F2F2F4F4..Fi = 11010110101..1. We will run I1, I2,. . . , In of IPT on Ax and Ay and we will get DI_x and DI_y. From DI_x and DI_y we will get histogram average of $\bar{\delta_h}(x)$, $\sigma_{\delta_h}(x)$, $\bar{\delta_h}(y)$, $\sigma_{\delta_h}(y)$ as per as formulation 1 for each individual. Fitness value for histogram F_H can be calculate as:

$$F_{fx} = |(\bar{\delta_h}(x_{I_i}) + \sigma_{\delta_h}(x_{I_i})) - (\sigma_{\delta_h}(x_{I_i}) + \bar{\delta_h}(x_{I_i}))| \tag{1}$$

$$F_{fy} = |(\bar{\delta_h}(y_{I_i}) - \sigma_{\delta_h}(y_{I_i})) - (\sigma_{\delta_h}(y_{I_i}) - \bar{\delta_h}(y_{I_i}))| \tag{2}$$

$$F_h = |F_{fx} - F_{fy}| \tag{3}$$

The following are the other four measures we implemented to measure the distances between adversarial and clean DI's histograms. We measure cross-entropy as another fitness value F_{CE}

$$F_{CE} = \sum_x DI_{x_{hist}}(i) \log\left(\frac{1}{DI_{y_{hist}}(i)}\right) \tag{4}$$

We also measure probability density (PD) cross entropy of DI histograms as another fitness value $F_{CE_{PDE}}$.

$$PD(DI_{x_{hist}})(i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{i-\mu}{\sigma}\right)^2\right] \tag{5}$$

$$F_{CE_{PD}} = \sum_x PD(DI_{x_{hist}})(i) \log\left(\frac{1}{PD(DI_{y_{hist}})(i)}\right) \tag{6}$$

We also calculate loss function to get another fitness value F_L.

$$F_L = \sum_{i=0}^{n} -(Y \log(DI_{x_{hist}}(i)) + (1-Y) log(1 - DI_{x_{hist}}(i))) \tag{7}$$

here Y (= 0 and 1), respectively, denotes adversarial or clean sample. Using equation 4, 5, 6, 7 and 8, we get our fitness value. We normalize these values between 0 to 1. We weighted them based on their distance value compared to each other.

$$W_1 = \frac{F_H}{F_{CE} + F_{ED} + F_{CE_{PD}} + \frac{1}{F_L}} \tag{8}$$

$$W_2 = \frac{F_{CE}}{F_H + F_{ED} + F_{CE_{PD}} + \frac{1}{F_L}} \tag{9}$$

$$W_3 = \frac{F_{ED}}{F_{CE} + F_H + F_{CE_{PD}} + \frac{1}{F_L}} \tag{10}$$

$$W_4 = \frac{F_{CE_{PD}}}{F_{CE} + F_{ED} + F_H + \frac{1}{F_L}} \tag{11}$$

$$W_5 = \frac{\frac{1}{F_L}}{F_{CE} + F_{ED} + F_{CE_{PD}} + F_H} \tag{12}$$

then we calculate our final fitness value F_V.

$$F_V = W_1 \times F_h + W_2 \times F_{CE} + W_3 \times F_{ED} + W_4 \times F_{CE_{PD}} + W_5 \times \frac{1}{F_L} \tag{13}$$

We calculated W values based on accuracy on detection from single fitness tests. We will now sort the population based on the value obtained from each individual's fitness function. As it is a steady GA, we will do two-point crossover [24] and replace the individuals with new off-springs. We will do elitism [25] and keep some of the best individuals in our population. To avoid local optima, we will do mutation after selection. After the fitness values of the population converge, we will select the best individual for the IPTS for A_x attack type. In the testing phase, we will check for the range of histogram, Euclidean distances, cross-entropy similarity, etc. to measure the input sample's likelihood to be an adversarial attack or not.

Dataset	Attack Type	IPTS	IPTS(decoded)	F1	GA %
MNIST	PGD [16]	111100	GS+2X Additive noise	0.94	0.7
CIFAR	PGD [16]	111111101001	2X Sharp+Blur+AS	0.92	0.5
ImageNet	FGSM [3]	111111101001	2X Sharpen+Blur+AS	0.86	0.3
MNIST	JSMA [4]	111111	GS+3X AN	0.99	0.8
CIFAR	JSMA [4]	111001111000	GS+sharpen+AS+sharpen	0.95	0.3
ImageNet	JSMA [4]	111001111000	2X sharpen+AS+sharpen	0.79	0.6
MNIST	CW [11]	011011	AS+BS+AN	0.92	0.4
CIFAR	CW [11]	111111011101	GS+2x Sharp+AN+blur	0.94	0.3
MNIST	DF [12]	111101	2XAN +AS	0.96	0.3
CIFAR	DF [12]	101101111011	GS+2blur+sharpen+AN	0.95	0.2
MNIST	HopSkipJump [19]	101101	BS+AN +AS	0.97	0.6
CIFAR	Localsearch [18]	111101111111	PX+ blur+PX+PX	0.75	0.2
MNIST	BPDA(PGD) [14]	N/A	blur + PX +AS	0.91	N/A
CIFAR	BPDA(PGD) [14]	N/A	blur + PX +AS	0.89	N/A

TABLE IV
DIFFERENT SEQUENCE GENERATIONS FOR DIFFERENT ATTACK TYPES AND DATASETS WITH THEIR F1 SCORE. GA % IS THE SUCCESS RATE OF THE CORRESPONDING SEQUENCE SELECTED AS A SOLUTION IN GA. FOR BPDA WE TESTED AGAINST "BLUR+PX+AS" IN PGD ATTACK. THIS SEQUENCE WORKS WITH 0.93 AND 0.91 F1 SCORES AGAINST THE PGD ATTACK. WITH BPDA IT HAS LOWER ACCURACY AS SHOWN IN THE TABLE. NOTE:GREY-SCALE IS NOT ENCODED, IT WAS PREDEFINED TO SPEEDUP THE PROCESS.

F. Selection of SIPTS

The process of selection was showed in a flow diagram with an example in figure 5. From each attack type, we selected 2-3 IPTS from the local optima of GA. From 6, we observed (a colored line from one attack type to multiple IPTS) that some IPTS also works for multiple attacks. There is multiple possible Set of IPTS (SIPTS) exist which works against all attack types. For each test time, we will select different SIPTS for the test image. As example, we have evaluated 20 attack types and found 40 IPTS where 20 IPTS can resolve at least 2 attack types. We can have reasonably above 50k possible SIPTS. If we randomly pick one for each test time, the chance of attacker to guess that one will be 1/50k which is close to 0. So an attacker who has complete knowledge of our system and developed an adversarial sample that can bypass our IPTS (such as BPDA attack does) is very unlikely due to low probability.

Attack Method	**F1 (MNIST)**	**F1 (CIFAR)**
PGD	0.91	089
BIM	0.90	0.81
MBIM	0.91	0.88
FGSM	0.98	0.97
JSMA	0.88	0.76
Hop-skip-Jump	0.78	NP

TABLE V
EXPERIMENTAL RESULTS USING ADAPTIVE ATTACK BPDA [20] (WITH DEFENSE SEQUENCE BLUR+AS+PIXELLETE).

Detection Method	**MNIST**				**CIFAR**				**Avg**
	FGSM	**JSMA**	**DF**	**CW**	**FGSM**	**JSMA**	**DF**	**CW**	
RF [26]	0.96	0.84	0.98	0.66	0.64	0.63	0.60	0.72	0.77
KNN [26]	0.98	0.80	0.98	0.6	0.56	0.52	0.52	0.69	0.73
SVM [26]	0.98	0.89	0.98	-	0.69	0.69	0.64	0.77	0.81
Feature Squeezing [11]	**1.00**	**1.00**		-	0.20	0.88	0.77	-	0.77
Ensemble [27]	0.99	-	0.45	-	0.99	-	0.42	-	0.71
Decision Mismatch [28]	0.93	0.93	0.91	-	0.93	**0.97**	0.91	-	0.93
Image quality features [7]	**1.00**	0.90	**1.00**	-	0.72	0.70	0.68	-	0.83
Our Proposed Method	0.99	0.99	0.96	**0.92**	0.94	0.95	**0.96**	**0.94**	**0.96**

TABLE VI
COMPARISON WITH OTHER ADVERSARIAL INPUT DETECTION TECHNIQUE BASED ON F1-SCORE.

IV. EXPERIMENTS

For our initial testing, we applied AS, AN, BS, and thinning algorithms for MNIST. We added thinning, pixellete, blur and sharpen for CIFAR dataset. Our attack samples are from PGD, BIM, MBIM, FGSM, JSMA, DF, HopSkipJump, Localsearch, and CW methods. We generated minimum 250 adversarial samples from each attack type to ran our experiments. We find out a sequence of blur+AS+pixellete works for PGD, BIM, FGSM, MBIM. which we used as a defended model in BPDA adaptive attack and generated new attack samples. We generated FGSM, JSMA, CW, and DF using Pytorch [29]. We used BIM, MBIM, PGD, local Search attack and HopSkipJump using IBM-ART-Toolbox [30] and Cleverhans adversarial library [31]. We used modified advertorch [32] to generate BPDA attack samples. We noticed that the destruction rate (i.e., the rate of failure of adversarial attack when it is converted to visual form) [33] is higher in advanced attack types. We disregarded those images from attack samples. Also due to our restriction of ϵ = 8/255 for CIFAR-10 as maximum noise value, we had to discard some samples from our dataset. For our GA, we used 75% of random adversarial examples from our generated attack samples and the same number of random non-adversarial images from a clean data set. The remaining 25% of adversarial images, combined with the same number of random non-adversarial images from the clean dataset is used for testing. Using the GA, we obtained an IPTS and difference metrics threshold values. When we applied that IPTS on the test set, we compare test data's difference metrics values with the threshold values. Based on that, we determine which images in the test set are adversarial and which ones not. We calculated true positive, false positive, true negative data and reported using F1 score. We ran our experiment with different configurations to evaluate the impacts. In MNIST, we selected configuration as described in Table III for 2bit. Our initial individuals after 1st iteration are (every value multiplied with 100) 110110 fitness value 61.65, 000100 fitness value 51.11,...,110110 fitness value 00.92. After 20 iterations, we obtained population as 111100 fitness value 89.46, 111100 fitness value 89.46, ...,110100 fitness value 88.46. The best sequence was 111100 in this run, which decoded as two steps of additive noise. One of the distance metrics average histogram value has upper and lower range between 114-100. Using this range and other distance measures, we ran IPTS on 1000 image (500 are random FGSM samples and other 500 are random clean samples), and able to detect adversarial samples with F1 score: 0.98.

V. RESULT ANALYSIS AND COMPARISON

In the table, IV, V,and VI, we presented our results in the F1 score for MNIST and CIFAR-10 dataset mainly. We observe that against common attacks such as PGD, JSMA, CW, we are achieving over 90% F1 score(Table IV). It is also noted that the same IPTS also works for multiple methods. Here in the last column, we showed the percentage of time (after 100 iterations) the IPTS occurred. There is a probability that a better IPTS exists if we iterate more. We did this experiment with three datasets. In MNIST, we got the best result. For CFIAR, we first convert an image to a GS image before applying the IPTS. In distance metrics calculation, we used the GS image as the original one. In IMAGENET, we also converted to GS and then reduce the image size to reduce computation time; it may limit the reliability of our result for the IMAGENET dataset. When we evaluated our defense against an advanced attack (with very low noises/perturbs), we found that all adversarial attack types aim to reduce the perturbation in advanced attack types. The magnitude of perturbation gets so small that they vanished in rounded values when converting to visual form [33]. Advanced attacks such as Deepfool and Local Search show a very high destruction rate after reducing our sample size. Our test result is limited in the remaining samples, but if we combine the destruction rate with our detection rate, the F1 score is above 90%. As for more advanced attack methods, adversarial examples generally can't have visualized form. As our method only works with adversarial inputs with visualized form, we could disregard those attacks. However, with 90% accuracy, we can state that our defense is sufficient for low noise advanced attacks. We compared the proposed framework with existing adversarial attack detection methods, and results are reported in Table VI. We notice that, on average, our approach has better accuracy comparing with Random forest(RF), Support Vector machine learning, KNN learning, and other techniques. In terms of practicability, our defense technique is more practical as it is done in pre-process steps and requires no knowledge of the ML model than other technologies such as distillation methods or adversarial training. In the adversarial training method, the ML model's performance got reduced. Our proposed method doesn't reduce the performance of the ML. The difference with other image pre-processing based techniques is our method provides dynamic filter sequences that are effective against adaptive attacks.

VI. Conclusion

We could not provide complete results from all other attack types and their adaptive-ness with the BPDA attack because of space limitations. The experimental results on three datasets showed that the presented method could detect various attacks, including adaptive attacks. Since this method does not need any knowledge about the Deep learning model whose input it is detecting, it can work as an independent input data filter for deep learning models. It can also classify what type of attacks are coming and when such attacks are happening, which is valuable from a security perspective. In the future, we will further explore parameter tuning and variable-length representation, ordered tuples, specialized operations, add more IPTS and other difference measures, etc. We aim to implement this GA in cloud computing (Hadoop/Pyspark) to speed up the appropriate sequence estimation.

References

[1] D. Dasgupta, Z. Akhtar, and S. Sen, "Machine learning in cybersecurity: a comprehensive survey," *The Journal of Defense Modeling and Simulation*, vol. 0, no. 0, p. 1548512920951275, 0.

[2] K. D. Gupta, D. Dasgupta, and Z. Akhtar, "Applicability issues of evasion-based adversarial attacks and mitigation techniques," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

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

[5] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *arXiv preprint arXiv:1605.07725*, 2016.

[6] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8571–8580.

[7] Z. Akhtar, J. Monteiro, and T. H. Falk, "Adversarial examples detection using no-reference image quality features," in *2018 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2018, pp. 1–5.

[8] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 3–14.

[9] Z. Akhtar and D. Dasgupta, "A brief survey of adversarial machine learning and defense strategies," 2019.

[10] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.

[11] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[13] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[14] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[15] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *arXiv preprint arXiv:2002.08347*, 2020.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[17] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[18] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," *arXiv preprint arXiv:1612.06299*, 2016.

[19] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," *arXiv preprint arXiv:1904.02144*, vol. 3, 2019.

[20] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[21] M. Goyal, "Morphological image processing," *IJCST*, vol. 2, no. 4, 2011.

[22] F. M. Waltz and J. W. Miller, "Efficient algorithm for gaussian blur using finite-state machines," in *Machine Vision Systems for Inspection and Metrology VII*, vol. 3521. International Society for Optics and Photonics, 1998, pp. 334–341.

[23] H. N. Gross and J. R. Schott, "Application of spectral mixture analysis and image fusion techniques for image sharpening," *Remote Sensing of Environment*, vol. 63, no. 2, pp. 85–94, 1998.

[24] G. Lin and X. Yao, "Analysing crossover operators by search step size," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*. IEEE, 1997, pp. 107–110.

[25] J. Romero-Hdz, S. Aranda, G. Toledo-Ramirez, J. Segura, and B. Saha, "An elitism based genetic algorithm for welding sequence optimization to reduce deformation." *Research in Computing Science*, vol. 121, pp. 17–36, 2016.

[26] J. Hayes and G. Danezis, "Machine learning as an adversarial service: Learning black-box adversarial examples," *arXiv preprint arXiv:1708.05207*, vol. 2, 2017.

[27] A. Bagnall, R. Bunescu, and G. Stewart, "Training ensembles to detect adversarial examples," *arXiv preprint arXiv:1712.04006*, 2017.

[28] J. Monteiro, Z. Akhtar, and T. H. Falk, "Generalizable adversarial examples detection based on bi-model decision mismatch," *arXiv preprint arXiv:1802.07770*, 2018.

[29] Gongzhitaao, "gongzhitaao/tensorflow-adversarial," Jan 2018. [Online]. Available: https://github.com/gongzhitaao/tensorflow-adversarial/tree/v0.2.0

[30] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.1.1," *CoRR*, vol. 1807.01069, 2018. [Online]. Available: https://arxiv.org/pdf/1807.01069

[31] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.

[32] G. W. Ding, L. Wang, and X. Jin, "AdverTorch v0.1: An adversarial robustness toolbox based on pytorch," *arXiv preprint arXiv:1902.07623*, 2019.

[33] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.