FAdeML: Understanding the Impact of Pre-Processing Noise Filtering on Adversarial Machine Learning

Faiq Khalid*, Muhammad Abdullah Hanif*, Semeen Rehman*, Junaid Qadir†, Muhammad Shafique*
*Vienna University of Technology, Vienna, Austria
†Information Technology University, Lahore, Pakistan
Email: {faiq.khalid,muhammad.hanif,semeen.rehman, muhammad.shafique}@tuwien.ac.at, junaid.qadir@itu.edu.pk

***Abstract*—Deep neural networks (DNN)-based machine learning (ML) algorithms have recently emerged as the leading ML paradigm particularly for the task of classification due to their superior capability of learning efficiently from large datasets. The discovery of a number of well-known attacks such as dataset poisoning, adversarial examples, and network manipulation (through the addition of malicious nodes) has, however, put the spotlight squarely on the lack of security in DNN-based ML systems. In particular, malicious actors can use these well-known attacks to cause random/targeted misclassification, or cause a change in the prediction confidence, by only slightly but systematically manipulating the environmental parameters, inference data, or the data acquisition block. Most of the prior adversarial attacks have, however, not accounted for the pre-processing noise filters commonly integrated with the ML-inference module. Our contribution in this work is to show that this is a major omission since these noise filters can render ineffective the majority of the existing attacks, which rely essentially on introducing adversarial noise. Apart from this, we also extend the state of the art by proposing a novel pre-processing noise *F*ilter-aware *Ad*versarial *ML* attack called *FAdeML*. To demonstrate the effectiveness of the proposed methodology, we generate an adversarial attack image by exploiting the "VGGNet" DNN trained for the "German Traffic Sign Recognition Benchmarks (GTSRB)" dataset, which despite having no visual noise, can cause a classifier to misclassify even in the presence of pre-processing noise filters.**

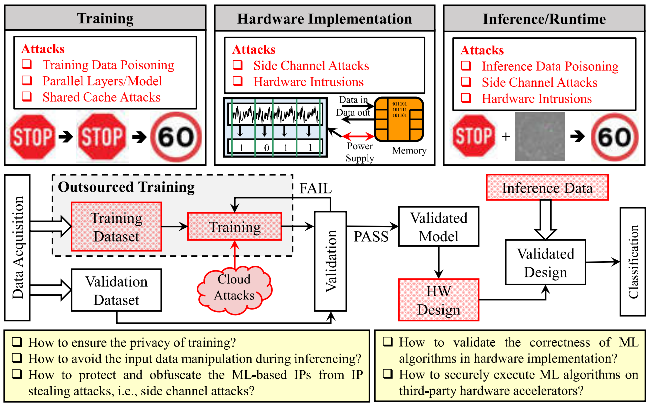


Fig. 1: An overview of security threats/attacks and their respective payloads for ML algorithms during training and inference [7]

I. Introduction

Machine learning (ML) has been the great success story of the last decade. In particular, deep neural networks (DNN), which can be efficiently trained to eke out the maximum information from "big data", is the standout ML framework that has revolutionized diverse fields such as object recognition, information retrieval, signal processing (including video, image, and speech processing), and autonomous systems (with self-driving cars a prominent example). The impressive performance of DNN-based ML algorithms can be gauged from the fact that DNNs now regularly outperform human beings in a rapidly increasing number of domains that were historically considered amenable only for human analysis and outside the reach of machine algorithms—e.g., NLP, emotion recognition, and games such as Go. *But despite their great prowess and success, DNN-based ML algorithms suffer from a critical problem:* these algorithms, as they are currently designed, are particularly vulnerable to security attacks from malicious adversaries [1].

A major reason behind this security vulnerability of ML algorithms stems from their implicit assumption that the testing or inference data will be similar in distribution to the training data, and that the model output would be sought in good faith by a trusted benign interacting party. Unfortunately this assumption flies in the face of any adversarial attempts to compromise the ML system, where a mismatch between the distributions on which the model is being trained and test is purposefully sought. We can safely anticipate that incorporation of ML models in critical settings (such as transportation, power grids) will make them an immediate target for malevolent adversaries who will be motivated to compromise these ML models and inflict massive damage. Addressing the lack of security of ML algorithms, therefore, assumes paramount importance and requires immediate attention from the community.

DNNs in particular have been shown to be very vulnerable to adversarial attacks. A striking example of DNN's vulnerability to *adversarial examples*[1] was first brought to prominence in 2013/14 through the work of Szegedy et al. [2] who showed that deep networks used for object recognition can be easily fooled through input images that have been perturbed in an imperceptible manner.[2] *Adversarial ML* is now a burgeoning field attracting significant attention [3]. An elaborate coverage of the history of adversarial machine learning and the issues involved can be seen in recent book-level treatments of adversarial ML [5], [6].

A. Types of Security Attacks on Machine Learning

With reference to Fig. 1, the major types of ML security attacks are:

Firstly, **training attacks** or **poisoning attacks**, an attacker can adversarially perturb (or 'poison') the training dataset, or otherwise manipulate the learning model/architecture or tool, with the goal of maximizing the classification error and denying service (e.g., by classifying a benign user as an intruder) through the injection of samples into the training data. For example, an attack can modify the underlying network structure through addition or deletion of parallel layers or neurons. In addition, when the training is being outsourced to some remote provider, other attacks (such as remote side-channel) can be used to steal the intellectual property (IP).

Secondly, **inference attacks** or **evasion attacks**, the attacker can attempt to steal the IP through a side channel or through successive polling. The attacker can also perturb the inference data to create "adversarial examples". The goal in such attacks is evasion at the test time or the inference time through the manipulation of test samples [8],

[1] "Adversarial examples" are minor perturbations of the input (so minor that the changes could be visually imperceptible) especially crafted by the adversary purposefully to maximize the prediction error [2], [3].

[2] Although the adversarial vulnerability of DNNs was pointed out in 2013/14, the broader work on adversarial pattern recognition goes back further, at least till 2004 as outlined by Roli et al. in [4]

978-3-9819263-2-3/DATE19/©2019 EDAA

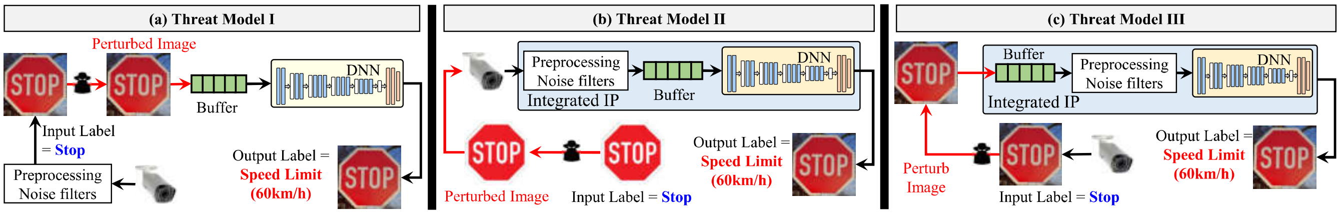


Fig. 2: Threat Models: different attack scenarios based on the attack's methodology and access to the different blocks during the ML inference

[9], [7]. Other possible attackers can be indirect beneficiaries who can either manipulate the inference data or intrude the hardware. Moreover, attackers can also perform side-channel attacks for IP stealing (see Fig. 1). In addition, the use of adversarial examples is an important case of an inference stage attack [10], [11], [12].

Lastly, **hardware implementation attacks**, the attacker seeks to exploit the hardware security threats, such as the manipulation of the hardware implementation of the trained ML model (hardware Trojans, [13]) and IP stealing (e.g., through side-channel or remote cyber-attacks) [14], [15]. Although a hardware intrusion can be used to launch several attack payloads, it requires complete access to the hardware blocks, without which these attacks are difficult or impossible to perform. Moreover, several sophisticated defense mechanism against the hardware attacks are becoming available.

B. Challenges in Resisting ML Security Attacks

Unlike the traditional systems, the security of ML-based systems is data dependent, which makes the job of ensuring security more challenging. One of the most common attacks is to exploit the data-dependency by manipulating/intruding the training dataset [16], [17], [18] or the corresponding labels [19]. Similarly, the baseline ML algorithm and training tools can also be attacked by adding new layers/nodes or manipulating the hyper-parameters [13], [20], [21], [22], as illustrated in Fig. 1. Moreover, in the case of the outsourced training, remote side-channel or cyber-attacks can be used to steal the IPs. Although data poisoning and ML algorithms/tools/architectures manipulation of attacks are quite effective, their effectiveness is limited by their substantial assumption of having complete access to information about the underlying ML architectures/baseline-model and the training dataset.

C. Motivating Pre-processing Noise Filter-Aware Adversarial ML

Although most current adversarial ML security attacks incorporate pre-processing elements (such as shuffling, gray scaling, local histogram utilization and normalization) [23] in their design and assume that an attacker can access the output of the pre-processing noise filtering, getting this access requires hardware manipulations and is practically difficult. If an attacker, on the other hand, does not have hardware access to the pre-processing filters, it becomes very challenging to incorporate the effect of pre-processing along with noise filtering, which raises the following key research questions:

1) How can we analyze the impact of pre-processing noise filtering on adversarial examples?
2) How can we incorporate the effects of the pre-processing noise filtering effects in the design of an improved adversarial ML attack?

D. Novel Contributions

The major contributions of this paper are:

Firstly, we provide an **elaborate analysis** on the impact of the pre-processing noise filtering on existing adversarial ML attack strategies (Section III). We demonstrate that most state-of-the-art adversarial ML attacks on classification can be neutralized using pre-processing noise filters. Based on this analysis, and our anticipation that future attackers will evolve new strategies to defeat or even leverage pre-processing noise filters, we have made our next contribution.

Secondly, we propose a **new attack methodology**, called pre-processing noise-*F*ilter-aware *Adv*ersarial ML or simply *FAdeML*, which is able to exploit pre-processing noise filtering as part of its attack strategy (Section IV). *As far as we know, this is the first work that has explicitly exploited noise filtering as part of an attack strategy on the ML security*. We demonstrate the effectiveness of the proposed FAdeML attack methodology by analyzying state-of-the-art adversarial attacks on the "VGGNet" [24] DNN trained for the "German Traffic Sign Recognition Benchmarks (GTSRB)" dataset [25] and show that FAdeML can force a classifier to misclassify even in the presence of pre-processing noise filters without any perceptible visual noise or change in the overall accuracy of the DNN.

II. Background: Threat Models and Adversarial Attacks

In this section, we will introduce the various attack/threat models and then some common state-of-the-art adversarial ML attacks.

A. Attack Threat Models

In order to systematically reason about security, we need to articulate the threat model we are assuming: *who are the possible attackers? what is the intention of the attack? what are the potential attack mechanisms? is the attack targeted or indiscriminate?* [26]. It is usually a good practice to adopt a conservative security attack model [6] and access for the "worst-case scenarios, but the best strategy is to assess the system security using different assumptions about the level of the adversarys capability. In this paper, we assume the following three threat models:

1) **Threat Model I:** An attacker *has access to the output of the pre-processing noise filter* and can perturb the image before storing it into the input buffer of the ML modules; see Fig. 2(a).
2) **Threat Model II:** An attacker *does not have access to the output of the pre-processing noise filter or input buffers of the ML module* but it can manipulate the data before acquisition; see Fig. 2(b).
3) **Threat Model III:** An attacker *does not have access to the pre-processing noise filter but it can directly perturb the acquired data* before storing it into the input buffer of the ML modules; see Fig. 2(c).

B. State-of-the-Art Adversarial ML Attacks

An adversarial ML attack typically attempts to either reduce prediction confidence or to cause a (random or targeted) misclassification by adding an imperceptible purposefully crafted noise into the data. It is assumed that an adversary's knowledge may encompass all, or part, of the following: details of the learning algorithm; details of the parameters (e.g., feature weights); and feedback on decisions. Analysis assuming perfect knowledge provides an upper bound on the performance degradation that can occur under attack. It is however usually fair to assume that the adversary is not totally unconstrained—in particular, it is a common assumption that an attacker can only control a given small fraction of the training samples; and in the case of test-time attacks can only enforce up to a certain maximum number of modifications.

Based on the adversary's knowledge about the targeted ML model, we can classify adversarial attacks into two categories: (1) a *white-box attack*, and (2) a *black-box attack*. In white-box attacks, it is assumed that the adversary has complete knowledge of the ML model including its architecture, training data, and the hyper-parameters. In black-box attacks, the ML model is an opaque black box for the adversary and no information (in terms of the technique being used or the hyper-parameters) is assumed to be available. It is however granted in black-box attacks that the adversary can operate as a standard user who can query the system with examples and note the model response. These query/response pairs can then be used by the attacker to infer the "ML black-box" and adversarial examples can be accordingly crafted.

Most of these adversarial ML attacks operate according to the following two-step methodology. Firstly, an attacker chooses the target image/images or target output class/classes (in the case of targeted misclassification) and defines the optimization goals, i.e., correlation coefficients, accuracy or other parameters to analyze imperceptibility. Secondly, a random noise is introduced into the target image to compute the imperceptibility based on the defined optimization goals. If the optimum imperceptibility is achieved, the intruded image is considered as an adversarial image; otherwise, the noise is updated based on imperceptibility parameters and a new image is generated.

Many adversarial attacks have been proposed in the literature such as the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method [2], [27]; the fast gradient sign method (FGSM) [28], [29], [30] method; the basic iterative method (BIM); the Jacobian-based Saliency Map Attack (JSMA) [1]; the one-pixel attack [31]; the DeepFool attack [32]; the Zeroth Order Optimization (ZOO) attack [33]; the CPPN EA Fool [34]; and the C&W's attack [35]. However, to study the effects of pre-processing noise filters on adversarial examples, in this paper, we limit ourselves to the most commonly used attacks, i.e., the L-BFGS, FGSM, BIM attack methods, which we briefly discuss next.

1) The L-BFGS Attack: This method is proposed by Szegedy et al. to generate an adversarial example in DNNs [2], [27]. The basic principle of the L-BFGS method is to achieve the optimization goal as defined in Equation 1, where noise represents the perturbations and minimizing it represents its imperception. The main *limitation of the L-BFGS method* is that it utilizes a basic linear search algorithm to update the noise for optimization which increases its converging time.

$$\min ||noise||_2 \implies f(x + noise) \neq f(x) \tag{1}$$

2) The FGSM Attack: To address the computational cost issue, Goodfellow et al. proposed the FGSM algorithm for generating adversarial examples with fewer computations by performing a one-step gradient update along the direction of the sign of gradient at each pixel [28], [29] Their proposed imperceptive noise can be defined as $\eta = \epsilon \bigtriangledown_x J(\theta, x, f)$, where, ϵ and η are the magnitude of the perturbation and the imperceptible noise, respectively. J is the cost minimizing function (based on original image x, classification function f and cost with respect to target class θ) obtained through stochastic gradient descent. So, the generated adversarial example can be computed by adding η into the targeted image. The main *limitation of the FGSM method* is that these attacks are robust for white box attacks rather than for the black box attack. Several variants of FGSM were proposed to handle the white box assumption but increases it converging time which limits its applicability in real-world applications [36].

3) The BIM Attack: Previous methods assume adversarial data can be directly fed into the DNNs. However, in many applications, people can only pass data through devices (e.g., cameras, sensors). Kurakin et al. applied adversarial examples to the physical world [30] by extending the FGSM algorithm by running a finer optimization (smaller change) for multiple iterations. In each iteration, the authors proposed to clip the pixel values to avoid large changes on each pixel. The main *limitation of the BIM method* is that it ignores the pre-processing stages after acquiring the data.

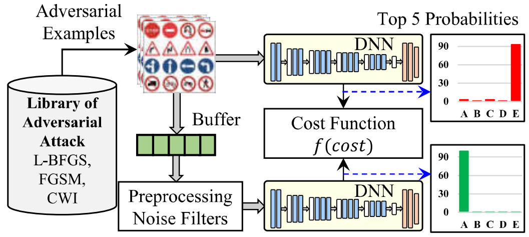


Fig. 3: Proposed analysis methodology to analyze the impact of the pre-processing noise filtering on Adversarial ML methods.

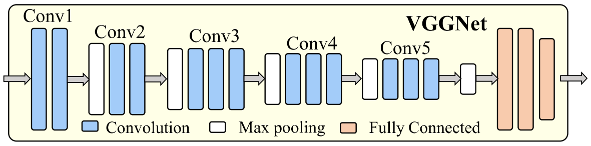


Fig. 4: VGGNet: Conv1 (64 output filters), Conv2 (128 output filters), Conv3 (256 output filters), Conv4 (512 output filters) and Conv5 (512 output filters)

III. Effect of Pre-processing Noise Filtering on Adversarial ML

In this section, we propose our analysis methodology for studying the effects of the pre-processing noise filter on adversarial security attacks during the ML inference. Our analysis methodology comprises the following steps (also illustrated in Fig. 3):

1) We initially choose an attack method from the adversarial attack library to generate adversarial examples according to their corresponding optimization functions and scaling methods.
2) Inference is then performed on a trained DNN model to compute the classification probabilities for the generated adversarial examples *assuming Threat Model I* (which assumes that the attacker can access the pre-processing noise filter's output).
3) Inference is also performed on the same trained DNN model using the same adversarial examples, but now assuming *Threat Model II or III*, and classification probabilities are computed.
4) Finally, we compare the difference between the classification probabilities computed under the assumptions of Threat Models I and II/III, using the following cost function:

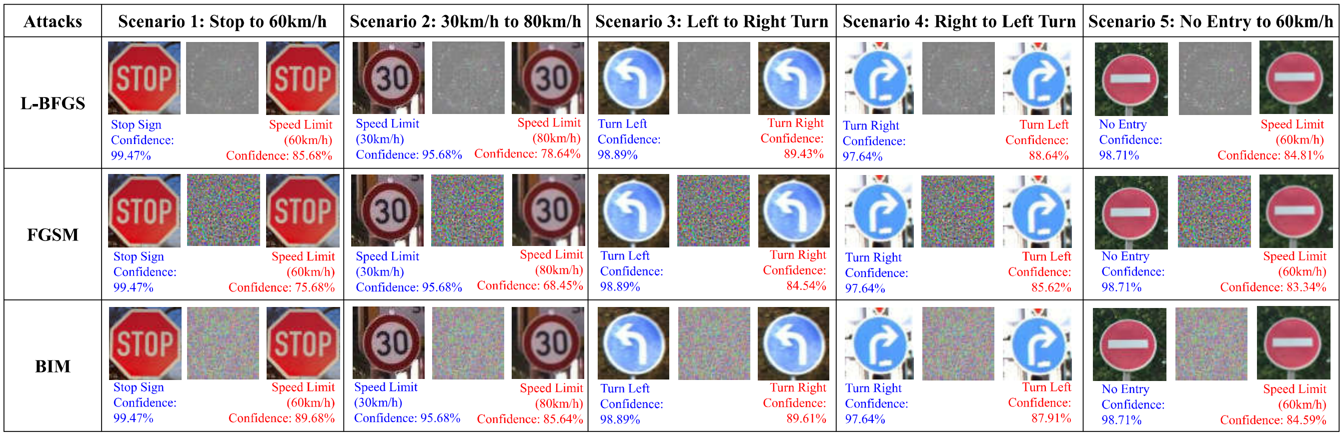
$$f(cost) = \sum_{n=1}^{5} P(C_n) - P(C_n^*) \tag{2}$$

where, C_n and $C_n^* \in \{C_1, C_2, ..., C_5\}$, $P(C_n)$ and $P(C_n^*)$ represent the top five predictions/classes for particular input/adversarial example, under the assumptions of the Threat Models I and II/III and their respective classification probabilities.

A. Experimental Setup

To demonstrate the effectiveness of our analysis methodology, and to analyze how state-of-the-art adversarial attacks perform in the presence of noise filters, the following experimental setup is used:

1) **DNN:** we use VGGNet model, which is composed of five convolutional layers and one fully connected layer (Fig. 4).
2) **Pre-processing Noise Filter:** We implement the filters *local average with neighborhood pixels* (LAP) and *local average with radius* (LAR). For comprehensive analysis, we use five distinct configurations of LAP each for different number of neighboring pixels $np = 4, 8, 16, 32, 64$ and radius $r = 1, 2, 3, 4, 5$.



Attacks	Scenario 1: Stop to 60km/h	Scenario 2: 30km/h to 80km/h	Scenario 3: Left to Right Turn	Scenario 4: Right to Left Turn	Scenario 5: No Entry to 60km/h
L-BFGS	Stop Sign Confidence: 99.47% Speed Limit (60km/h) Confidence: 85.68%	Speed Limit (30km/h) Confidence: 95.68% Speed Limit (80km/h) Confidence: 78.64%	Turn Left Confidence: 98.89% Turn Right Confidence: 89.43%	Turn Right Confidence: 97.64% Turn Left Confidence: 88.64%	No Entry Confidence: 98.71% Speed Limit (60km/h) Confidence: 84.81%
FGSM	Stop Sign Confidence: 99.47% Speed Limit (60km/h) Confidence: 75.68%	Speed Limit (30km/h) Confidence: 95.68% Speed Limit (80km/h) Confidence: 68.45%	Turn Left Confidence: 98.89% Turn Right Confidence: 84.54%	Turn Right Confidence: 97.64% Turn Left Confidence: 85.62%	No Entry Confidence: 98.71% Speed Limit (60km/h) Confidence: 83.34%
BIM	Stop Sign Confidence: 99.47% Speed Limit (60km/h) Confidence: 89.68%	Speed Limit (30km/h) Confidence: 95.68% Speed Limit (80km/h) Confidence: 85.64%	Turn Left Confidence: 98.89% Turn Right Confidence: 89.61%	Turn Right Confidence: 97.64% Turn Left Confidence: 87.91%	No Entry Confidence: 98.71% Speed Limit (60km/h) Confidence: 84.59%

Fig. 5: The impact of implemented adversarial attacks, i.e., L-BFG, FGSM and BIM, for misclassification ***under the assumptions of Threat Models I,*** *on the top 5 accuracy of VGGNet trained on GTSRB.*

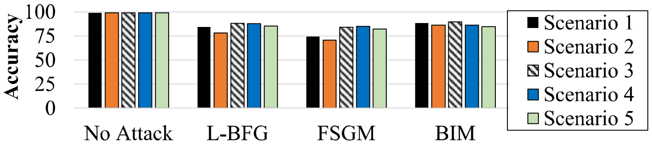


Fig. 6: Top 5 accuracy of overall VGGNet, a DNN trained on GTRSB dataset with 43 classes, for different adversarial attacks, i.e., L-BFG, FGSM , BIM

3) **Dataset:** we use the German Traffic Sign Recognition Benchmarks (GTSRB) dataset
4) **Adversarial Attacks and Threat Models:** we use the L-BFGS, FGSM, and BIM attacks and assume Threat Models I and II/III.
5) **Payload:** We perform five targeted misclassification scenarios, i.e., (i) stop sign to 60km/h; (ii) 30km/h to 80km/h; (iii) left turn to right turn; (iv) right turn to left turn; and (v) no entry to 60km/h.

B. Experimental Analysis

In this section, we discuss the two analyses that we performed based on the aforementioned experimental setup under the assumptions of Threat Model I (Fig. 5) and II/III (Fig. 7), for the selected adversarial attacks. The analysis in Fig. 5, *which assumes the Threat Model I*, shows that the implemented adversarial attacks successfully performed misclassfication for all the attacking scenarios. Though, adversarial noise is invisible but still the adversarial examples have an (up to 10%) on the overall top 5 accuracy of the VGGNets for complete GTSRB dataset. The analysis in Fig. 7, *which assumes the Threat Models II/III*, however shows that the smoothing filters (LAP and LAR) can nullify the impact of the adversarial examples on classification, but the attack is shown to affect the top 5 accuracy of overall DNN. The detailed discussion and insights of this is given below:

1) **L-BFGS and FGSM:** The LAP filters nullify the effects of L-BFGS filters but the top 5 accuracy is also reduced by 10%, as shown in Fig. 5. However, if we increase the number of "np" then the accuracy improves but it decreases after $np = 32$. In the case of LAR filters, the top 5 accuracy starts decreasing after $r = 4$. The impact of LAP and LAR filters on FGSM attacks follows similar trends. This impact is more pronounced on L-BFGS because its optimization function is highly dependent on the sharp edges or sudden changes in data samples, which are removed by smoothing filters. Moreover, both of these attacks do not consider the effect of pre-processing while designing adversarial examples. Therefore, in the presence of smoothing filters, they still reduce confidence of the network by decreasing the top 5 accuracy.
2) **BIM:** This analysis shows the smoothing filters also nullify the effects of this attack. Since, this attack considers the feedback from the pre-processing, therefore, the effects of this attack is nullified when smoothing strength is relatively high. This has a significant impact on the confidence even after filtering effects, as shown by the confidence values of all scenario in Fig. 7, similarly, the top 5 accuracy of the network increases with increase in number of neighboring pixel (till $np = 32$) or radius ($r = 3$) for local average filters. However, further increase in the neighboring pixel (till $np > 32$) or radius ($r > 3$) decreases the top 5 accuracy because it compromises some of the distinguishing features as well, as shown in the top 5 accuracy analysis of Fig. 7.

C. Key Insights

1) The adversarial noise from the adversarial examples that are gradient descent based can be removed by applying the smoothing filters, i.e., LAR or LAP. However, the attacks do still reduce the confidence of the overall classification.
2) The selection of smoothing parameters in LAR or LAP has a direct impact on the top 5 accuracy of the DNN. Top 5 accuracy increases with increase in smoothing parameters till a certain value, e.g., for LAP and LAR values are $np = 32$ and $r = 4$. When going beyond this threshold value, the top 5 accuracy starts to decrease because of the degradation of input image/sample quality.
3) To perform a successful adversarial attack, the effects of all the pre-processing stages should be considered while optimizing the adversarial noise.

IV. Pre-processing Noise Filtering-aware Adversarial ML

The methodology of our proposed pre-processing noise filter aware FAdeML attack comprises the following steps, shown in Fig. 8:

1) We initially choose a reference sample "x" to be perturbed and a sample of the targeted class "y" for the misclassification such that $prediction(x) \neq prediction(y)$, and choose an attack from the adversarial attack library.
2) We compute the prediction/classification probabilities for sample "x" and "y", *assuming Threat Model I*, to identify the difference between their respective prediction/classification probabilities by computing the following cost function: $f(cost) = \sum_{n=1}^{5} P_x(C_n) - P_y(C_n^*)$
3) We compute and add the adversarial noise "n" to "x" to generate the adversarial example "x^*" (i.e., $x^* = \eta \times n + x$), where, η is the noise scaling factor to make it imperceptible.

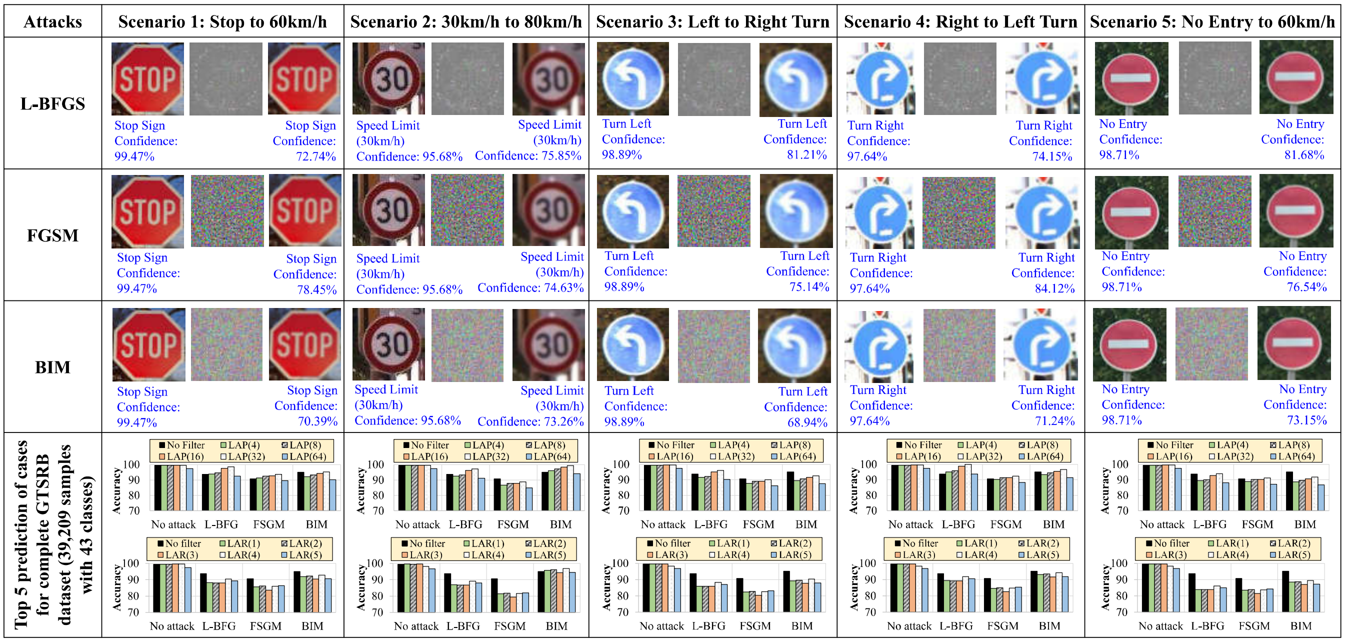


*Fig. 7: The traditional adversarial attacks, i.e., L-BFG, FGSM and BIM, are neutralized by pre-processing low pass filters, i.e., LAP and LAR, (**under the assumptions of Threat Models II and III**), in the expense of confidence reduction.*

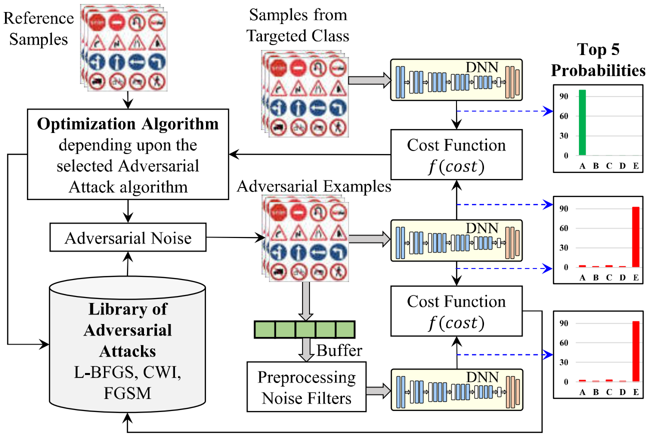

Reference Samples
Samples from Targeted Class
Top 5 Probabilities
Optimization Algorithm depending upon the selected Adversarial Attack algorithm
DNN
Cost Function $f(cost)$
Adversarial Examples
Adversarial Noise
Library of Adversarial Attacks L-BFGS, CWI, FGSM
Buffer
Preprocessing Noise Filters
90
60
30
0
A B C D E

Fig. 8: The proposed methodology to design FAdeML Attacks to incorporate the Impact of Pre-processing Noise Filtering on Adversarial ML

4) We then compute the prediction/classification probabilities for "x^*", *assuming Threat Model II or III*, to identify the impact of pre-processing noise filtering on generated adversarial example "x^*".
5) We then analyze the difference between the prediction/classification probabilities of adversarial example "x^*" with "x''", assuming Threat Models I and II/III, based on the cost function of Equation 2.
6) Finally, we repeat the optimization depending upon the selected algorithm to incorporate the impact of pre-processing noise filtering.

$$x^* = \eta \times (n + \frac{\delta n}{\delta f(cost)}) + x \qquad (3)$$

A. Experimental Analysis

To illustrate the effectiveness of the proposed FAdeML attack methodology, we utilize the same experimental setup (discussed in Section III-A). We performed experiments to those shown in Fig. 9 for the proposed FAdeML attacks, and identified the following key observations:

1) FAdeML attack performs better even in the presence of pre-processing filters, because these attacks incorporates the smoothing effects of low pass pre-processing noise filters (i.e., LAP or LAR).
2) Another key observation is that the attack confidence reduces slightly because the smoothing effect also compromises the sudden changes, which are then exploited by most of the adversarial attacks.
3) Similar observation can be depicted from the analysis presented in Fig. 9. It shows that top 5 accuracy of the network increases with increase in number of neighboring pixel (till np = 32) or radius (r = 3) for local average filters. However,further increase neighboring pixel (till np > 32) or radius (r > 3) decreases the top 5 accuracy because it compromises some of the distinguishing features as well.

B. Key Insights

1) The effectiveness of the adversarial noise can be increased significantly by optimizing it with respect to pre-processing stages, especially noise filtering.
2) The effects of gradient-descent function optimized adversarial noise can be nullified because they rely heavily on the sudden changes in the input images/samples.

V. Conclusion

In this paper, we proposed an analysis methodology to understand the impact of pre-processing filter on adversarial attacks and an attack methodology to incorporates these effects into the adversarial examples. We have demonstrated the effectiveness of the proposed analysis and attack methodology using three state-of-the-art attacks (L-BFG, FGSM and BIM) on the "VGGNet" Deep Neural Network (DNN) trained on the "German Traffic Sign Recognition Benchmarks (GTSRB)" dataset. The experimental results shows that the effects of existing adversarial attacks on classification can be nullified by applying the smoothing filters, LAP and LAR, on the input samples before sending it the DNNs, even though the confidence classification is still affected. Based on this analysis, we developed a new pre-processing noise-*F*ilter-aware *Ad*versarial ML (FAdeML) attack and showed that FAdeML is able to force a misclassification even after the application of the smoothing

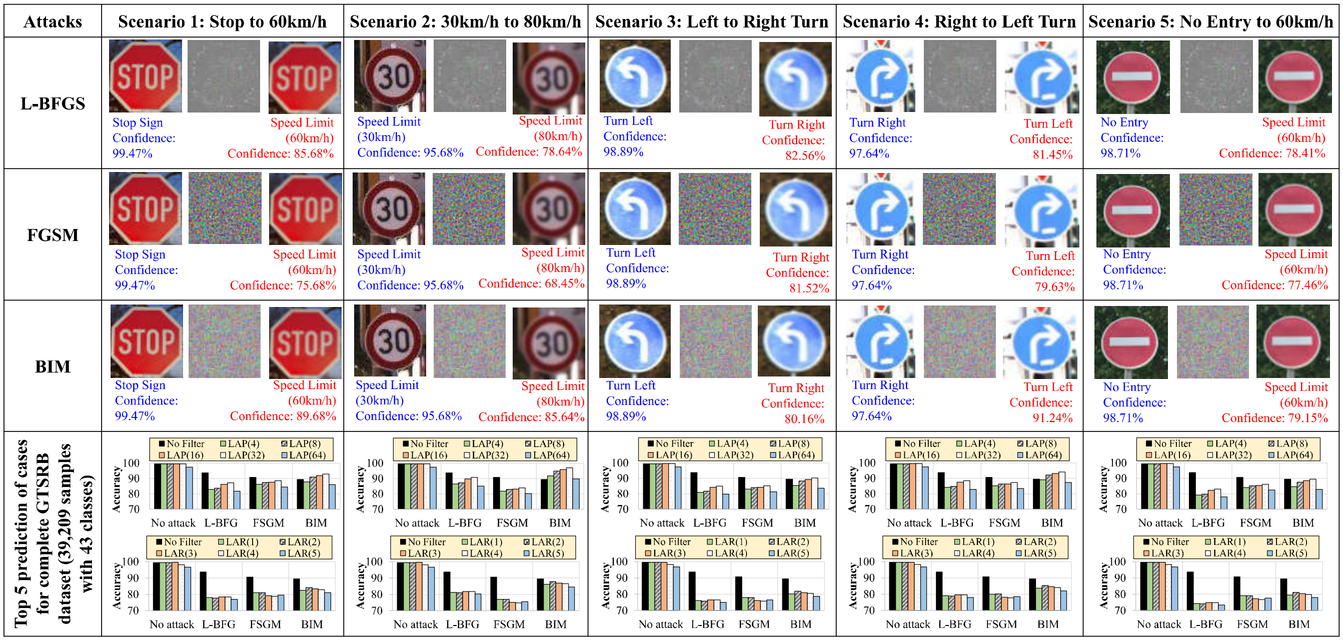


*Fig. 9: Unlike the the traditional adversarial attacks, **FAdeML attacks** are not neutralized by pre-processing low pass filters but their impact on top 5 accuracy of overall neural network is relatively higher.*

filters. Although previous work in the related literature has explored using pre-processing for defense purposes, we are the first to explicitly exploit noise filtering to improve an adversarial ML attack. Our overall goal through this work is to communicate to the community the potential vulnerabilities of current ML systems and thereby inspire researchers to develop ML architectures that are effective yet can resist adversarial examples.

References

[1] N. Papernot et al., "The limitations of deep learning in adversarial settings," in ***EuroS&P***. IEEE, 2016, pp. 372–387.

[2] C. Szegedy et al., "Intriguing properties of neural networks," *arXiv:1312.6199*, 2013.

[3] I. Goodfellow et al., "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[4] B. Biggio et al., "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[5] Y. Vorobeychik et al., "Adversarial machine learning," *Synthesis Lectures on AI and ML*, vol. 12, no. 3, pp. 1–169, 2018.

[6] A. D. J. et al., *Adversarial Machine Learning*. Cambridge University Press, 2018.

[7] M. Shafique et al., "An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the IoT era," in ***DATE***. IEEE, 2018, pp. 827–832.

[8] B. Biggio et al., "Evasion attacks against machine learning at test time," in ***ECML PKDD***. Springer, 2013, pp. 387–402.

[9] N. Papernot et al., "SoK: Security and privacy in machine learning," in ***EuroS&P***. IEEE, 2018, pp. 399–414.

[10] W. Xu et al., "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv:1704.01155*, 2017.

[11] N. Papernot et al., "CleverHans v2. 0.0: an adversarial machine learning library," *arXiv:1610.00768*, 2016.

[12] W. Brendel et al., "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv:1712.04248*, 2017.

[13] M. Zou et al., "PoTrojan: powerful neural-level Trojan designs in deep learning models," *arXiv:1802.03043*, 2018.

[14] N. Papernot et al., "Practical black-box attacks against machine learning," in ***AsiaCCS***. ACM, 2017, pp. 506–519.

[15] R. Shokri et al., "Membership inference attacks against machine learning models," in ***S&P***. IEEE, 2017, pp. 3–18.

[16] M. Zhao et al., "Data poisoning attacks on multi-task relationship learning," in ***AAAI***, 2018, pp. 2628–2635.

[17] Y. Wang et al., "Data poisoning attacks against online learning," *arXiv:1808.08994*, 2018.

[18] M. Jagielski et al., "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," *arXiv:1804.00308*, 2018.

[19] A. Shafahi et al., "Poison frogs! targeted clean-label poisoning attacks on neural networks," *arXiv:1804.00792*, 2018.

[20] W. Li et al., "Hu-fu: Hardware and software collaborative attack framework against neural networks," *arXiv:1805.05098*, 2018.

[21] T. Liu et al., "SIN 2: Stealth infection on neural network—a low-cost agile neural trojan attack methodology," in ***HOST***. IEEE, 2018, pp. 227–230.

[22] C. Liao et al., "Server-based manipulation attacks against machine learning models," in ***CODASPY***. ACM, 2018, pp. 24–34.

[23] A. S. Rakin et al., "Robust pre-processing: A robust defense method against adversary attack," *arXiv:1802.01549*, 2018.

[24] L. Wang et al., "Places205-VGGNet Models for scene recognition," *arXiv:1508.01667*, 2015.

[25] J. Stallkamp et al., "The German traffic sign recognition benchmark: a multi-class classification competition," in ***IJCNN***. IEEE, 2011, pp. 1453–1460.

[26] M. Barreno et al., "Can machine learning be secure?" in ***AsiaCCS***. ACM, 2006, pp. 16–25.

[27] P. Tabacof et al., "Exploring the space of adversarial images," in ***IJCNN***. IEEE, 2016, pp. 426–433.

[28] A. Rozsa et al., "Adversarial diversity and hard positive generation," in ***CVPR Workshop***. IEEE, 2016, pp. 25–32.

[29] I. Goodfellow et al., "Explaining and harnessing adversarial examples," *stat*, vol. 1050, p. 20, 2015.

[30] A. Kurakin et al., "Adversarial examples in the physical world," *arXiv:1607.02533*, 2016.

[31] J. Su et al., "One pixel attack for fooling deep neural networks," *arXiv:1710.08864*, 2017.

[32] S. Moosavi-Dezfooli et al., "Deepfool: a simple and accurate method to fool deep neural networks," in ***CVPR***. IEEE, 2016, pp. 2574–2582.

[33] P. Chen et al., "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in ***AISec***. ACM, 2017, pp. 15–26.

[34] A. Nguyen et al., "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in ***CVPR***. IEEE, 2015, pp. 427–436.

[35] N. Carlini et al., "Towards evaluating the robustness of neural networks," in ***S&P***. IEEE, 2017, pp. 39–57.

[36] F. Tramèr et al., "Ensemble adversarial training: Attacks and defenses," *arXiv:1705.07204*, 2017.