Image Features Reordering and Adjustment to Mitigate the Transferability Effect of Adversarial Attacks for Autonomous Driving Cars

Ahmad Fakhr Aldeen Sattout, Ali Chehab
Department of Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon,
Email:{aas136@aub.edu.lb, chehab@aub.edu.lb}

***Abstract*—In this paper, we introduce a novel adversarial detection framework designed to enhance the security and robustness of DNNs, with a particular focus on applications in autonomous driving systems. Our approach leverages segmentation masks and innovative techniques for image feature reordering and adjustment to detect adversarial attacks effectively. We comprehensively evaluate the proposed framework under a white-box attack scenario, comparing its performance against the state-of-the-art Feature Squeezing method across eleven adversarial attack techniques and three heterogeneous benchmark datasets. The results demonstrate that our framework achieves detection rates exceeding 96% across all datasets, establishing its efficacy and robustness against diverse attack strategies.**

***Index Terms*—Adversarial Attack Algorithms, Autonomous Driving Cars, Computer Vision, Cybersecurity, Deep Learning, Machine Learning, Segmentation mask**

I. Introduction

Autonomous driving systems have become heavily reliant on artificial intelligence, particularly Deep Neural Networks (DNNs), to perform critical tasks such as processing 3D sensor data from LiDARs and depth cameras [1], recognizing traffic signs [2], and controlling essential vehicle functions like steering [3]. These advancements have positioned AI as a cornerstone of modern autonomous systems, enabling unprecedented levels of automation and precision in real-world applications.

However, despite their success, DNNs exhibit significant vulnerabilities to adversarial attacks [4] in which attackers carefully craft perturbations to input data that can lead to erroneous model predictions aligned with an attacker's intent. Such adversarial manipulations pose serious risks in safety-critical domains like autonomous driving, where even minor misclassifications can result in catastrophic consequences.

The challenge of securing these models is exacerbated by several factors, including the limited generalizability of existing defense mechanisms, their computational overhead, and their reduced efficacy against a diverse range of adversarial strategies. Furthermore, research has demonstrated the real-world transferability of adversarial examples [1], [5]–[7] , further highlighting the gravity of this threat. Addressing these issues is crucial for the widespread adoption of DNN-based autonomous driving systems, which play a pivotal role in enabling Vehicle-to-Everything (V2X) communication and applications like cooperative perception and path planning [8], [9]. Solving these challenges will be vital to ensuring the safety and reliability of autonomous systems in real-world environments.

We introduce a novel method for detecting adversarial attacks for autonomous driving systems. The proposed approach employs a U-Net model to extract segmentation masks from input images, which are subsequently analyzed using a one-class Support Vector Machine (SVM). This initial verification step evaluates whether the extracted segmentation patterns align with the expected dataset characteristics, serving as the first line of defense. Next, the features of the masked image are reordered as well as filtered and then the contrast is adjusted, using our novel IFRA algorithm. This algorithm produces two versions of the input: a split-swapped image and its mirrored version. Finally, a verifier DNN model classifies these images and their predictions are compared to the main model prediction for final decision.

This paper presents three key contributions:

1) Segmentation-Based Detection: Instead of analyzing the input image directly, adversarial noise is detected by verifying its segmentation mask. This is achieved using a U-Net model to generate segmentation masks, followed by a One-Class Support Vector Machine (SVM) to identify abnormal patterns.
2) Novel Feature Processing Algorithm: A new algorithm is introduced to reorder and adjust input features, producing two alternative versions of the input for enhanced analysis.
3) Verifier Model Training: The verifier model is trained using these generated input versions to validate the predictions of the target model, adding an additional layer of defense.

The structure of this paper is as follows: Section II reviews relevant literature on adversarial attacks targeting autonomous driving cars. Section III details the proposed detection framework designed to safeguard DNN models against such adversarial threats. Section IV describes the experimental setup and tools used for implementing and validating the method. Section V presents the experimental results, highlighting the effectiveness of the proposed defense mechanism. Lastly,

Section VI presents a summary highlighting the key findings of this work.

II. Literature Review

This section focuses on adversarial attacks that target autonomous driving cars, highlighting the potential hazards of such adversarial attacks on real-world autonomous systems.

1) Adversarial Traffic Signs: Morgulis et al. [6] showcased the real-world applicability of adversarial examples in deceiving commercial Traffic Sign Recognition (TSR) systems. Building on the methodology of Sitawarin et al. [5], they developed an enhanced pipeline to generate adversarial traffic signs. Key improvements included applying a mask to isolate the sign, resizing images to align with the TSR classifier's input requirements, and refining perturbations with penalty constraints on grayscale levels to minimize visual detectability. The physical experiments revealed that 40% of the crafted adversarial signs successfully misled the TSR system, with some attacks causing the system to freeze, mimicking a denial-of-service (DoS) attack.

2) GAN for Generating Adversarial Signs: Kong et al. [10] introduced an approach to generate adversarial images aimed at disrupting a Deep Learning-Based Steering Wheel Control System in autonomous vehicles. This technique simulated real-world conditions by processing multiple frames of the same traffic sign, accounting for dynamic changes in the sign's dimensions as the vehicle approached. The method was evaluated on three datasets Udacity [11], DAVE-2 [12], and KITTI [13] to test its robustness across diverse scenarios. The experimental results revealed that adversarial signs induced significant steering deviations in the vehicle's trajectory, with average deviations ranging between 17 and 19 degrees.

3) Attacking End-to-End Autonomous driving cars: Wu et al. [7] investigated two white-box adversarial attack methods targeting NVIDIA's autonomous driving model [14]. The first approach utilized the Fast Gradient Sign Method (FGSM) to create image-specific perturbations, while the second employed universal perturbations for an image-agnostic attack strategy. Both methods were tested in three distinct environments within the Robot Operating System (ROS) framework [15], encompassing both scenarios, simulated and real world. The experiments demonstrated that these attacks effectively caused vehicles to veer off their lanes.

4) Adversarial Attacks against LIDAR System: Cao et al. [16] revealed a critical vulnerability: LiDAR systems can be manipulated using physical adversarial objects. To exploit this weakness, LiDAR-adv is introduced, the first comprehensive framework for generating physical adversarial objects. In real-world testing, these objects successfully bypassed LiDAR detection systems, posing significant safety concerns. Additionally, prior research [17], [18] highlighted the susceptibility of LiDAR systems to laser spoofing attacks. These involve injecting false laser signals into the system, allowing attackers to manipulate object detection and localization, potentially causing dangerous driving errors.

III. Methodology

Detecting imperceptible and human-recognizable adversarial inputs is crucial, particularly in unsupervised AI systems like autonomous vehicles. Our method addresses these needs by targeting both types of input. The following subsections outline the key elements of the proposed pipeline.

A. U-Net Model for Mask Segmentation

The U-Net model is a CNN architecture introduced by Ronneberger et al. [19] for image segmentation, especially in the biomedical field. The advantage of using the U-Net model for image segmentation comes from its effective learning even if it is trained on limited training data or small objects.

B. One Class Support Vector Machine

The One-Class Support Vector Machine (OCSVM) [20], learns the boundary around normal data to separate them from potential anomalies without needing labeled samples of these anomalies. The OCSVM is used for anomaly detection tasks where only normal data is available, such as fraud detection and network security [21]. Based on the generated segmentation masks of the U-net, OCSVM can detect adversarial input associated with large perturbations that form our first line of defense.

Algorithm 1 Image Features Reordering and Adjustment (IFRA)

Input: Masked Image $\boldsymbol{x}$
Output: Split-Swapped and Split-Mirrored images (x_{ss}, x_{sm})

$w_x \leftarrow \textbf{\textit{width}}(x)$
$h_x \leftarrow \textbf{\textit{height}}(x)$
$h_{crop} \leftarrow h_x/2$, $w_{crop} \leftarrow w_x/2$
$s_1 \leftarrow \textbf{\textit{empty\_matrix}}(\textbf{shape}(x))$
$s_2 \leftarrow \textbf{\textit{empty\_matrix}}(\textbf{shape}(x))$
$s_1[0:h_x, 0:w_{crop}] \leftarrow x[0:h_x, w_{crop}:w_{end}]$
$s_1[0:h_x, w_{crop}:w_{end}] \leftarrow x[0:h_x, 0:w_{crop}]$
$f \leftarrow \textbf{\textit{horizontal\_flip}}(x)$
$s_2[0:h_x, 0:w_{crop}] \leftarrow x[0:h_x, w_{crop}:w_{end}]$
$s_2[0:h_x, w_{crop}:w_{end}] \leftarrow x[0:h_x, 0:w_{crop}]$
$\text{mask}_{\text{med}} \leftarrow \textbf{\textit{create\_Median\_Filter}}(\text{filter size} = (3 \times 3))$
$x_{\text{ss}_{med}} \leftarrow \textbf{\textit{Median\_filtering}}(s_1, \text{mask}_{\text{med}})$
$x_{\text{sm}_{med}} \leftarrow \textbf{\textit{Median\_filtering}}(s_2, \text{mask}_{\text{med}})$
$\text{clahe}_{\text{prop}} \leftarrow \textbf{\textit{create\_CLAHE}}(\text{grid size} = (3 \times 3), \text{clip limit} = 5)$
$x_{\text{ss}} \leftarrow \textbf{\textit{CLAHE}}(x_{\text{ss}_{med}}, \text{clahe}_{\text{prop}})$
$x_{\text{sm}} \leftarrow \textbf{\textit{CLAHE}}(x_{\text{sm}_{med}}, \text{clahe}_{\text{prop}})$
return (x_{ss}, x_{sm})
end

C. Image Features Reordering and Adjustment

Images are high-dimensional and unstructured data [22]. The attackers exploit these natural properties to introduce subtle, often imperceptible perturbations in the image space, making it adversarial. We propose the Image Features Reordering and Adjustment (IFRA) algorithm 1 as a mitigation

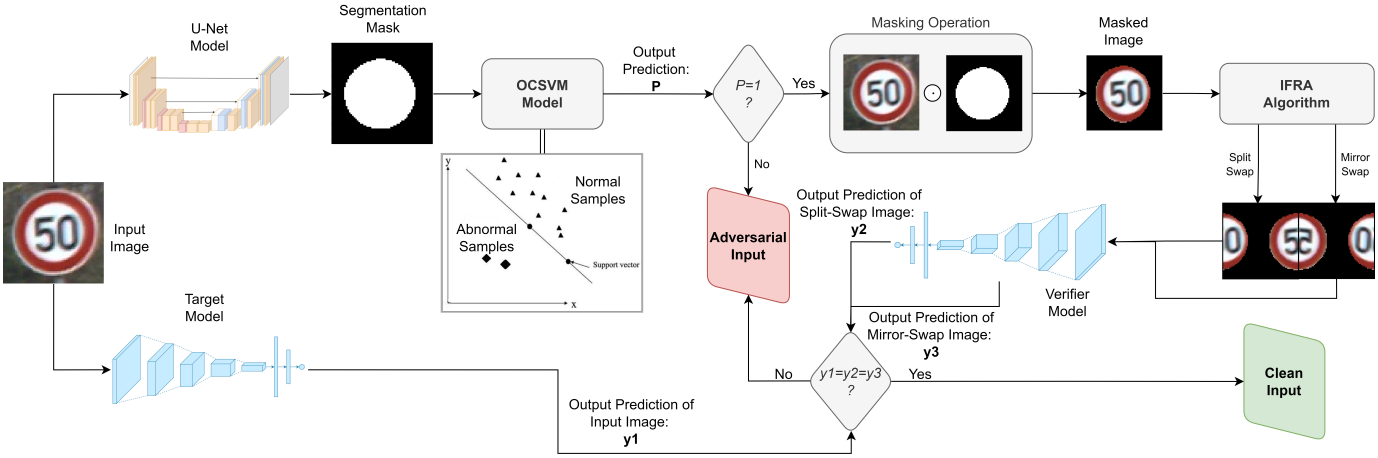

Fig. 1: The proposed adversarial detection framework.

strategy to reduce the transferability effect [23] of adversarial attacks between AI models. The IFRA algorithm reorders the image features by splitting the input image into two halves and then swapping them changing the position of the features. Another version of the input image is created by mirroring the input first, then splitting and swapping. In addition, we adjust the image features by applying median filtering and adaptive histogram equalization operation [24], [25].

D. Adversarial Detection Mechanism

Our method aims to mitigate the transferability effect of adversarial images in which it can detect them. Therefore, we train a second DNN model, called a verifier model, to validate the prediction of the main model. The generated images of IFRA algorithm form the training set of the verifier model. Thus, for every input image, there will be three predictions, the main model's prediction, and two predictions of the verifier model. If the three predicted classes are the same, the input image is considered clean, otherwise deemed adversarial.

IV. Experimental Setup

We compared our proposed pipeline performance depicted in Figure 1 against Feature Squeezing, a leading adversarial detection approach. We tested under a white-box attack scenario, following the implementation of Feature squeezing [26]. Additionally, we evaluated the robustness of our method against varying levels of adversarial noise generated by the FGSM attack algorithm.

A. Datasets and Preprocessing

MNIST Dataset [27]: This 28×28 grayscale dataset contains $60,000$ training and $10,000$ test images of handwritten digits (0-9). Digits are centered with high grayscale values, and backgrounds are mostly zero as shown in Figure 2. Segmentation masks were generated by marking non-zero pixels as foreground.

CIFAR-10 Dataset [28]: Consisting of 32×32 RGB images across 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) as shown in Figure 2, this dataset includes $50,000$ training and $10,000$ test images. As segmentation masks were unavailable, we manually labeled 200 training and 25 test images per class. Data augmentation expanded this set to $1,000$ images per class, which trained a YOLO-V8-Seg model [29] to generate $5,000$ masks per class. Additional images were sourced for the Cat and Deer classes [30], [31] to enhance mask quality generated by the YOLO-V8-Seg model, and most masks were manually verified and corrected where needed.

GTSRB-8 Dataset (Speed-Limit Subset) [32]: This subset of the German Traffic Sign Recognition Benchmark (43 classes) focuses on eight classes of speed-limit shown in Figure 2, with $8,140$ training and $3,142$ test images. We created 200 training and 20 test masks manually, excluding any noisy or indiscernible images as shown in Figure 3 for dataset quality.

We use the online CVAT tool [33] to manually generate the segmentation masks for the experimented datasets.

B. Target Models

Model Performance: Table I summarizes the accuracy and mean confidence of target models tested on MNIST, CIFAR-10, and GTSRB-8 datasets. For MNIST, the pre-trained Carlini model [34] achieved 99.43% accuracy, while a pre-trained DenseNet model [35] reached 94.84% accuracy on CIFAR-10 and 96.8% accuracy on GTSRB-8. Using the source code of the Feature Squeezing [26], adversarial samples were created. For MNIST and CIFAR-10, 100 adversarial images 10 per class were generated per attack; for GTSRB-8, 96 adversarial images 12 per class were generated per attack.

TABLE I: Target models accuracy and confidence rates.

Dataset	Model	Test Accuracy	Mean Confidence	Selected Samples	Samples Accuracy	Mean Confidence
MNIST	Carlini	99.43%	99.39%	100	100%	100%
CIFAR-10	Densenet	94.84%	92.15%	100	100%	95.55%
GTSRB-8	Densenet	96.8%	91.8%	96	100%	91.3%

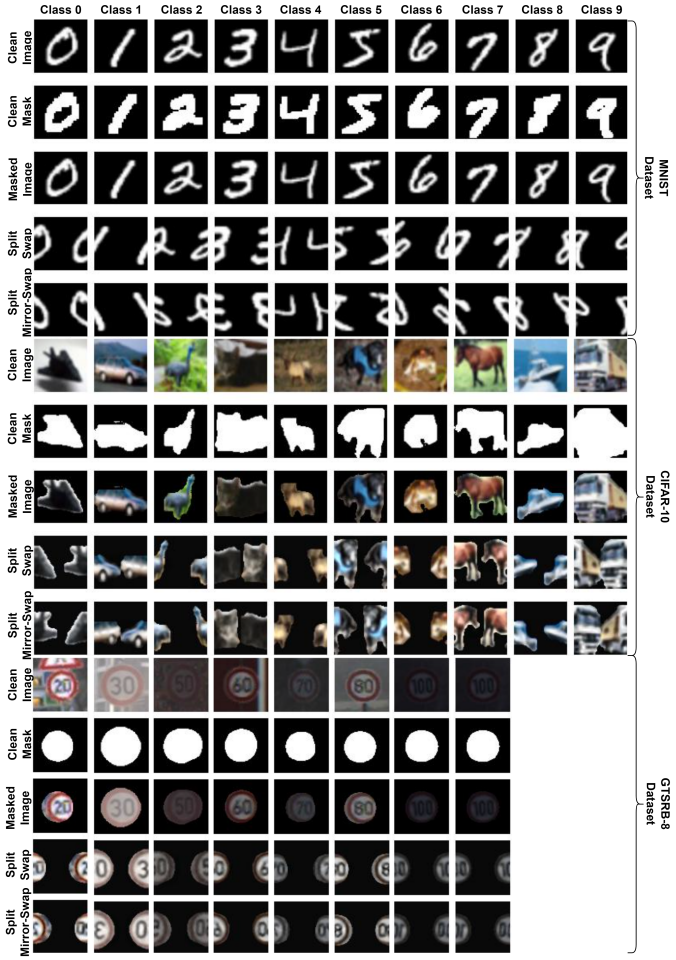

Fig. 2: The flow of the input image in the proposed framework.

Fig. 3: Distorted images from the GTSRB-8 dataset.

TABLE II: Evaluation metrics of the U-Net models.

Dataset Name	Model Name	Test IOU	Test Dice
MNIST	U-Net	97.17%	98.39%
CIFAR-10	U-Net	73.85%	84.10%
GTSRB-8	U-Net	83.9%	91.1%

TABLE III: OCSVM models parameters and accuracy.

Dataset Name	Model Name	γ	μ	Test Accuracy
MNIST	OCSVM	auto-selected	0.01	99.17%
CIFAR-10	OCSVM	auto-selected	0.01	99.25%
GTSRB-8	OCSVM	auto-selected	0.01	97.7%

TABLE IV: Verifier models accuracy.

Dataset Name	Model Name	Test Accuracy Split-Swapped	Test Accuracy Mirrored-Swapped	Test Accuracy All
MNIST	Carlini [34]	98.19%	98.48%	98.33%
CIFAR-10	Densenet [35]	86.43%	86.48%	86.45%
GTSRB-8	Densenet [35]	98.56%	97.64%	98.10%

C. U-Net Model Setup

The U-Net model was trained to generate segmentation masks for each dataset. The model was trained in two phases: an initial 30 epoch on 90% of the data 10% for validation and an additional 20 epoch on the complete dataset, with data augmentation techniques applied to improve the robustness. The U-net performance is measured by the Intersection over Union (IoU) and the Dice metrics in Table II. The U-net model performed well on the MNIST and GTSRB-8 datasets, while it generalized less effectively for the CIFAR-10 dataset.

D. Adversarial Detection Flow

OCSVM Model: It is configured using the default parameters from the Sklearn library as outlined in Table III, and demonstrated high accuracy in all datasets.

IFRA algorithm: IFRA algorithm involves two key processes to generate split-swapped and mirrored-swapped versions of the masked input image, as illustrated in the pseudo-code 1. It includes 3×3 median filtering and 3×3 grid-size with contrast clip-limit $= 5$ for the CLAHE operator applied to the three datasets except for the MNIST dataset where the clip-limit was set to 1.

Verifier Model: The verifier model is the second defense layer, and is trained on the output images of IFRA algorithm. It shares the architecture of the target model as shown in Table I, with the input image resized to 64×64. Table IV summarizes the test accuracy of verifier models. While the verifier models for the MNIST and GTSRB-8 datasets performed similarly to their target models, the CIFAR-10 verifier model showed an 8.39% accuracy drop, achieving a test accuracy of 86.45%.

E. Adversarial Attack Configuration

Table V outlines eleven adversarial attack setups across three modes. The non-targeted mode generates adversarial images that the model misclassifies into any incorrect class. The next-target mode shifts the target class to the next one using $t = L + 1$ mod $No.Classes$, where L is the correct class and t is the target class. The least-likely (LL) mode targets the class with the lowest confidence of the model's output, $t = min(\hat{y})$. All adversarial images were clipped to fit within an 8-bit color range. The attack success rate with the associated confidence rates and distortion metrics are summarized in Table V. Most of the attacks resulted in high-confidence misclassifications, except for the JSMA attacks. The CW_0 attack, although computationally intensive, achieved a 100% success rate for all datasets.

F. The robustness of the proposed pipeline against a wide range of Perturbation Values

We constructed adversarial samples from the GTSRB-8 data set using the fast gradient sign method (FGSM). Fourteen perturbation values ε were tested, ranging from minimal, nearly imperceptible values starting from 0.01 to larger distortions 0.5. The performance of our detection method was compared with various Feature Squeezing techniques, with results presented in Table VII.

TABLE V: Specification and metrics of crafting adversarial images.

	Distance Metric	Attack	Mode	Cost (s)	Success Rate	Prediction Confidence	Distortion		
							L_∞	L_2	L_0
MNIST	L_∞	FGSM $\varepsilon = 0.3$	Untargeted	0.001	46%	94.76%	0.302	5.916	0.561
		BIM $\varepsilon = 0.3$	Untargeted	0.005	92%	99.82%	0.302	4.820	0.523
	L_∞	CW_∞	Next	64.887	100%	99.99%	0.261	4.277	0.496
			LL	64.635	100%	99.98%	0.279	4.666	0.510
	L_2	DeepFool	Untargeted	0.077	100%	89.16%	2.174	0.532	0.732
		CW_2	Next	0.310	99%	99.99%	0.689	2.878	0.458
			LL	0.376	100%	99.99%	0.733	3.209	0.458
	L_0	CW_0	Next	84.928	100%	99.99%	0.995	4.737	0.051
			LL	84.874	100%	99.99%	0.997	5.172	0.060
		JSMA	Next	0.544	63%	64.92%	1.000	4.814	0.049
			LL	0.677	45%	64.54%	1.000	5.619	0.064
CIFAR-10	L_∞	FGSM $\varepsilon = 0.0156$	Untargeted	0.013	86%	96.93%	0.016	0.864	0.998
		BIM $\varepsilon = 0.3$	Untargeted	0.083	92%	98.75%	0.008	0.367	0.994
	L_∞	CW_∞	Next	327.833	100%	98.82%	0.012	0.443	0.988
			LL	311.504	100%	97.38%	0.014	0.521	0.994
	L_2	DeepFool	Untargeted	0.254	97%	83.79%	0.028	0.235	0.993
		CW_2	Next	7.907	100%	97.90%	0.033	0.284	0.753
			LL	9.070	100%	97.21%	0.040	0.352	0.847
	L_0	CW_0	Next	498.414	100%	98.21%	0.640	2.041	0.018
			LL	449.072	100%	97.60%	0.693	2.479	0.024
		JSMA	Next	8.219	100%	42.61%	0.897	4.992	0.078
			LL	10.641	98%	39.17%	0.896	5.599	0.098
GTSRB-8	L_∞	FGSM $\varepsilon = 0.0156$	Untargeted	0.031	89.58%	95.66%	0.016	1.730	1.000
		BIM $\varepsilon = 0.008$	Untargeted	0.128	86.46%	96.34%	0.008	0.733	0.999
	L_∞	CW_∞	Next	297.163	100%	97.44%	0.012	0.892	0.974
			LL	330.601	97.92%	97.54%	0.015	1.183	0.995
	L_2	DeepFool	Untargeted	0.158	97.92%	90.93%	0.054	0.591	0.999
		CW_2	Next	23.537	100%	97.32%	0.048	0.544	0.574
			LL	23.854	100%	96.95%	0.059	0.760	0.727
	L_0	CW_0	Next	457.162	100%	97.86%	0.682	2.788	0.012
			LL	574.574	100%	97.56%	0.770	3.970	0.025
		JSMA	Next	135.021	93.75%	33.45%	0.891	8.820	0.040
			LL	166.015	95.83%	29.54%	10.300	0.904	0.050

All experiments were carried out on NVIDIA 4060 GPU laptop (8GB), 64GB RAM, and Windows 11. To promote reproducibility, the implementation of our adversarial detection framework for all experiments can be accessed at https://github.com/AhmadFASattout/Image-Features-Reordering-and-Adjustment-to-Mitigate-the-Transferability-Effect-of-Adversar-Attacks-.git

V. Results

Table VI presents the detection rates of our proposed pipeline compared to the best combination of feature squeezers for adversarial images generated using eleven attacks across three datasets. Our method achieved over 96% detection rates for successful adversarial attacks on all datasets. Notably, detection rates for FGSM and BIM attacks surpassed the best-combination Feature Squeezing detector. This improvement stems from fundamental differences in approach: Feature Squeezing relies on bit-depth reduction that uniformly normalizes image features and adversarial noise using one squeezer and applies denoising filters using another two squeezers. In contrast, our method leverages the IFRA algorithm, which reorders pixel positions and enhances the contrast, mitigating the smoothing effects of median filtering. The overall detection rates of the proposed detection method for CIFAR-10 and GTSRB-8 datasets outperform the best Feature Squeezing detector by more than 10%. However, for the MNIST dataset, our detection rates score slightly lower than Feature Squeezing by only 0.9%.

The false positive rates of our method are similar to those of Feature Squeezing for the MNIST and GTSRB-8 datasets. However, for CIFAR-10, the higher false positive rate stems from the verifier model's reduced classification accuracy, as shown in Table IV. This reduction is primarily due to differences in the training data. Unlike the main model, which is trained on the full dataset, the verifier model uses the masked version of the data. To confirm this, we trained the same main dense-net model using the masked CIFAR-10. The resulting accuracy was 88.17%, which is 6.67% lower than the target model's accuracy. This indicates that masking the background in the dataset negatively impacts the accuracy. Furthermore, the precision of the U-Net model in generating accurate segmentation masks plays a critical role in the verifier model's performance. The same reasoning applies to the OCSVM model.

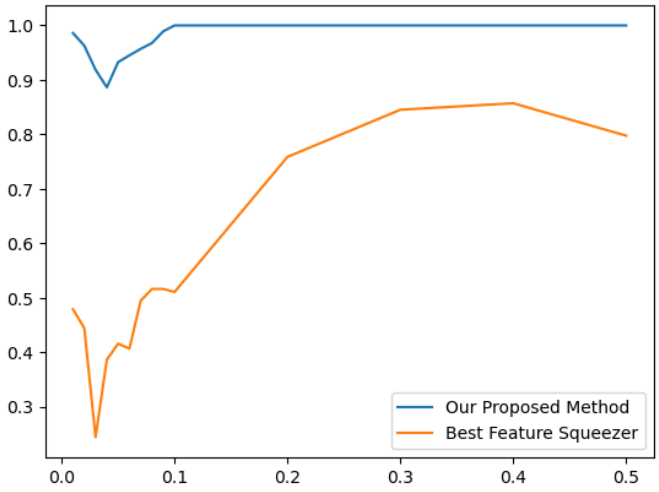

Fig. 4: Detection rates of our proposed method vs best feature squeezer for a wide range of adversarial perturbations under FGSM attack on the GTSRB-8 dataset.

The results presented in Table VII demonstrate that our detection method delivers state-of-the-art performance across all tested perturbation values, achieving an overall detection rate of 96.71%. Remarkably, the method consistently outperformed the most effective Feature Squeezing techniques for every tested ε value of the FGSM attack, showcasing its robustness and effectiveness against a wide range of adversarial inputs. Figure 4 reflects these findings in a graphical representation. The figure underscores the resilience and reliability of our detection approach, consistently maintaining a high detection rate exceeding 88%, irrespective of the perturbation level. The FGSM attack at $\varepsilon = 0.04$ proved to be the most effective, yet our method successfully achieved a detection rate of 88.63%. By comparison, the best-performing Feature Squeezing technique managed only a 38.63% detection rate at the same perturbation level, highlighting the superior robustness of our approach against adversarial attacks.

Finally, to evaluate the computational efficiency of the proposed method in comparison to the Feature Squeezing approach within the context of autonomous driving systems, we measured the detection time of the proposed method and Feature Squeezing for the GTSRB-8 dataset. Specifically, the comparison was conducted against the best-performing Feature

TABLE VI: Adversarial detection rates of the proposed method compared to Feature Squeezing with the required configuration.

	Configuration			L_∞ Attacks				L_2 Attacks			L_0 Attacks				Overall Detection Rate	False Positive Rate
	Detector	Parameters	Threshold	FGSM	BIM	CW_∞		Deep Fool	CW_2		CW_0		JSMA			
						Next	LL		Next	LL	Next	LL	Next	LL		
MNIST	FeatureSqueezing	1-bit	0.00025	97.8%	98.9%	100%	100%	100%	100%	100%	71%	64%	100%	100%	92.9%	4.1 %
	FeatureSqueezing	2-bit	0.00008	73.9%	8.6%	94%	98%	100%	95.9%	95%	63%	76%	100%	100%	81.5%	3.9 %
	FeatureSqueezing Best Combination	Median-Filter 2×2 1-bit	0.00275	97.8%	97.8%	100%	100%	100%	100%	100%	93%	95%	100%	100%	98.4%	4 %
	Our Method	Split-Algorithm Median-Filter 3×3 CLAHE 3×3 Clip-Limit = 1	–	**100%**	**100%**	95%	100%	100%	94.9%	99%	95%	95%	98.4%	97.7%	97.5%	4.9 %
CIFAR-10	FeatureSqueezing	5-bit	0.2997	4.6%	14.1%	38%	67%	56.7%	79%	92%	2%	5%	6%	8.1%	34.3%	4.9%
	FeatureSqueezing	Median-Filter 2×2	1.1683	25.5%	48.9%	95%	100%	71.1%	98%	100%	99%	100%	77%	86.7%	82.9%	4.9%
	FeatureSqueezing	Non-local Mean $13 - 3 - 2$	0.3588	17.4%	30.4%	85%	95%	74.2%	91%	95%	4%	6%	24%	21.4%	49.9%	4.7%
	FeatureSqueezing Best Combination	Median-Filter 2×2 5-bit Non-local Mean $13 - 3 - 2$	1.1683	27.9%	50%	97%	100%	76.2%	99%	100%	99%	100%	77%	86.7%	83.9%	5.1%
	Our Method	Split-Algorithm Median-Filter 3×3 CLAHE 3×3 Clip-Limit = 5	–	**83.7%**	**90.2 %**	**98 %**	100%	**94.8%**	99%	100%	99%	99%	**97%**	**98.9%**	**96.5%**	16.8%
GTSRB-8	FeatureSqueezing	5-bit	0.1324	31%	19.2%	37.5%	55.2%	33.3%	37.5%	59.3%	6.2%	10.4%	11.4%	5.3%	27.9%	4.3%
	FeatureSqueezing	Median-Filter 3×3	0.2988	39.1%	37.3%	93.7%	100%	56.2%	92.7%	98.9%	100%	100%	98.9%	100%	84.8%	4.0%
	FeatureSqueezing	Non-local Mean $11 - 3 - 4$	0.1774	60.8%	69.8%	100%	100%	85.4%	98.9%	100%	5.2%	12.5%	23.9%	4.3%	60.1%	4.3%
	FeatureSqueezing Best Combination	Median-Filter 3×3 5-bit Non-local Mean $11 - 3 - 4$	0.3741	44.5%	57.8%	97.9%	100%	66.6%	98.9%	100%	98.9%	100%	95.8%	98.9%	88.5%	3.7%
	Our Method	Split-Algorithm Median-Filter 3×3 CLAHE 3×3 Clip-Limit = 5	–	**95.9%**	**97.5%**	**100%**	100%	**98.9%**	98.9%	100%	**100%**	100%	**98.9%**	**100%**	**99.2%**	4.9%

TABLE VII: Adversarial detection rates of the proposed method compared to Feature Squeezing under FGSM Attack with varying ε values and their configuration for the GTSRB-8 dataset.

	Configuration			FGSM attack with different ε values														Overall Detection Rate
	Detector	Parameters	Threshold	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.2	0.3	0.4	0.5	
GTSRB-8	FeatureSqueezing	5-bit	0.12085	28.16%	24.69%	11.62%	19.31%	10.11%	4.39%	2.15%	1.07%	0%	2.12%	0%	0%	0%	0%	6.97%
	FeatureSqueezing	Median-Filter 3×3	0.26721	47.88%	44.44%	24.41%	38.63%	41.57%	40.65%	49.46%	51.61%	51.61%	51.06%	75.86%	84.52%	85.71%	79.76%	54.59%
	FeatureSqueezing	Non-local Mean $13 - 3 - 4$	0.17390	69.01%	53.08%	32.55%	25%	17.97%	12.08%	15.05%	7.52%	4.30%	1.06%	0%	0%	0%	0%	16%
	FeatureSqueezing Best Combination	Median-Filter 3×3 5-bit Non-local Mean $13 - 3 - 4$	0.33005	57.74%	44.44%	27.90%	32.95%	35.95%	35.16%	44.08%	47.31%	44.08%	44.68%	65.51%	82.14%	76.19%	76.19%	50.57%
	Our Method	Split-Algorithm Median-Filter 3×3 CLAHE 3×3 Clip-Limit = 5	–	**98.59%**	**96.92%**	**91.86%**	**88.63%**	**93.25%**	**94.50%**	**95.69%**	**96.77%**	**98.92%**	**100%**	**100%**	**100%**	**100%**	**100%**	**96.71%**

Squeezing detector, as presented in Table VI. Experimental results revealed that for detecting a single input, the Feature Squeezing method required approximately 0.01751 seconds per image, whereas our proposed detection framework demonstrated a faster detection time of around 0.01526 seconds per image. It is important to highlight that this comparison was carried out under equivalent conditions, ensuring that both Feature Squeezing pipeline and our method processed input data of identical dimensions namely, 64×64 RGB images.

VI. Conclusion

In summary, we presented an innovative adversarial detection framework employing a two-layer defense strategy. The method is able to mitigate the transferability effect of adversarial attacks and detect them. Our framework achieved state-of-the-art detection rates exceeding 96% across eleven adversarial attack methods on three diverse datasets. This methodology is particularly effective in safeguarding autonomous driving systems, achieving a detection rate over 99% against adversarial attacks targeting speed-limit signs. Experimental results reveal that the framework significantly outperforms Feature Squeezing techniques on GTSRB-8 and CIFAR-10 datasets while maintaining comparable results on MNIST. Specifically, it achieves average detection rates of 93.2% and 95.9% for FGSM and BIM attacks, respectively, across all datasets, compared to 56.73% and 68.53% achieved by Feature Squeezing method. In addition, our detection method is faster than the best-performing feature squeezer in the context of autonomous driving cars on the GTSRB-8 dataset. Overall, this robust and versatile framework demonstrates high detection rates across grayscale and RGB datasets, underscoring its potential for generalizability and practical application.

References

[1] C. Xiang, C. R. Qi, and B. Li, "Generating 3d adversarial point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9136–9144, 2019.

[2] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *The 2011 international joint conference on neural networks*, pp. 2809–2813, IEEE, 2011.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58443–58469, 2020.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[5] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "Darts: Deceiving autonomous cars with toxic signs," *arXiv preprint arXiv:1802.06430*, 2018.

[6] N. Morgulis, A. Kreines, S. Mendelowitz, and Y. Weisglass, "Fooling a real car with adversarial traffic signs," *arXiv preprint arXiv:1907.00374*, 2019.

[7] H. Wu, S. Yunas, S. Rowlands, W. Ruan, and J. Wahlström, "Adversarial driving: Attacking end-to-end autonomous driving," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1–7, IEEE, 2023.

[8] L. Zeng, K. Zhang, Q. Han, S. Chen, L. Ye, R. Wang, J. Lei, and Q. Xie, "Research of path planning model based on hotspots evaluation," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2429–2434, IEEE, 2019.

[9] S.-W. Kim, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus, "The impact of cooperative perception on decision making and planning of autonomous vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 3, pp. 39–50, 2015.

[10] Z. Kong, J. Guo, A. Li, and C. Liu, "Physgan: Generating physical-world-resilient adversarial examples for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14254–14263, 2020.

[11] Udacity, "Udacity Steering Wheel Dataset," 2020.

[12] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *arXiv preprint arXiv:1704.03952*, 2017.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[14] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[15] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[16] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li, "Adversarial objects against lidar-based autonomous driving systems," *arXiv preprint arXiv:1907.05418*, 2019.

[17] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, no. 2015, p. 995, 2015.

[18] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pp. 445–467, Springer, 2017.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.

[20] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 395–408, 2005.

[21] C. Wang, Y. Sun, S. Lv, C. Wang, H. Liu, and B. Wang, "Intrusion detection system based on one-class support vector machine and gaussian mixture model," *Electronics*, vol. 12, no. 4, p. 930, 2023.

[22] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[23] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[24] G. Yadav, S. Maheshwari, and A. Agarwal, "Contrast limited adaptive histogram equalization based enhancement for real time video system," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2392–2397, 2014.

[25] A. F. A. Sattout and S. K. Bisoy, "Frontal facial expression recognition based on convolutional neural network and haar-cascade descriptors," in *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pp. 280–284, 2021.

[26] W. Xu, D. Evans, and Y. Qi, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," in *Proceedings of the 2018 Network and Distributed Systems Security Symposium (NDSS)*, 2018.

[27] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[28] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8." https://github.com/ultralytics/ultralytics, 2023.

[30] segmentation, "cat2 dataset." https://universe.roboflow.com/segmentation-w6eax/cat2-6nv4t, mar 2023.

[31] R. of sheep, "Recognition of sheep dataset." https://universe.roboflow.com/recognition-of-sheep/recognition-of-sheep, jul 2023.

[32] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: a multi-class classification competition," in *The 2011 international joint conference on neural networks*, pp. 1453–1460, IEEE, 2011.

[33] CVAT, "Online annotation tool." https://www.cvat.ai/. 2017.

[34] N. Carlini, "Nicholas carlini. robust evasion attacks against neural network to find adversarial examples." https://https://github.com/carlini/nn_robust_attacks. visited on 2024-08-16.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.