LightPure: Realtime Adversarial Image Purification for Mobile Devices Using Diffusion Models

Hossein Khalili
University of California, Los Angeles (UCLA)
Los Angeles, CA, USA

Seongbin Park
University of California, Los Angeles (UCLA)
Los Angeles, CA, USA

Vincent Li
University of California, Los Angeles (UCLA)
Los Angeles, CA, USA

Brandan Bright
University of California, Los Angeles (UCLA)
Los Angeles, CA, USA

Ali Payani
Cisco Systems
San Jose, CA, USA

Ramana Rao Kompella
Cisco Systems
San Jose, CA, USA

Nader Sehatbakhsh
University of California, Los Angeles (UCLA)
Los Angeles, CA, USA

ABSTRACT

Autonomous mobile systems increasingly rely on deep neural networks for perception and decision-making. While effective, these systems are vulnerable to adversarial machine learning attacks where small perturbations in the input could significantly impact the outcome of the system. Common countermeasures include leveraging adversarial training and/or data or network transformation. Although widely used, the main drawback of these countermeasures is that they require full and invasive access to the classifiers, which are typically proprietary. Additionally, the cost of training or retraining is often prohibitively expensive for large models. To tackle this, purification models have recently been proposed. The aim is to incorporate a "purification" layer before classification, thereby eliminating the necessity to modify the classifier. Despite their effectiveness, state-of-the-art purification methods are compute-intensive, rendering them unsuitable for mobile systems where resources are constrained and large latency is not desired.

This paper presents a new approach, LightPure, that enhances the purification of adversarial images. It improves the accuracy of the current leading purification methods while also providing notable enhancements in speed and computational efficiency, making it suitable for mobile devices with limited resources. Our approach uses a two-step diffusion and one-shot Generative Adversarial Network (GAN) framework for purification, prioritizing latency without compromising robustness. We propose several new techniques in designing our model to achieve a reasonable balance between classification accuracy and adversarial robustness while maintaining a desired latency. We design and implement a proof-of-concept on a Jetson Nano board and evaluate our method using several attack scenarios and datasets. Our results show that LightPure can outperform existing purification methods by up to 10x in terms of latency while achieving higher accuracy and robustness for various black-, gray-, and white-box attack scenarios. The fusion of speed and robust defense mechanisms positions our method as a significant advancement in the field of adversarial image purification, offering a scalable and effective solution for real-world mobile systems.

CCS CONCEPTS

• **Security and privacy → Embedded systems security**; • **Computing methodologies → Neural networks**; • **Computer systems organization → Embedded and cyber-physical systems**.

KEYWORDS

Autonomous mobile system, adversarial machine learning, diffusion models.

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/11
https://doi.org/10.1145/3636534.3690684

ACM Reference Format:
Hossein Khalili, Seongbin Park, Vincent Li, Brandan Bright, Ali Payani, Ramana Rao Kompella, and Nader Sehatbakhsh. 2024. LightPure: Realtime Adversarial Image Purification for Mobile Devices Using Diffusion Models. In *The 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3636534.3690684

1 INTRODUCTION

Deep learning models have been increasingly integrated into various aspects of decision-making in embedded and mobile devices. For example, deep learning-based computer vision techniques are now commonly used in commercial off-the-shelf advanced autonomous mobile systems (e.g., cars, robots, drones) [15, 33, 39, 48, 61, 65, 66].

Despite their exceptional performance in various machine learning tasks, deep neural networks are vulnerable to adversarial attacks [7, 17, 54] where small perturbations in the input could greatly alter the output of the classifier. In the context of real autonomous mobile systems, such errors could lead to catastrophic failures and critical damages [5, 6, 8, 23, 34].

Developing new attack and defense mechanisms for adversarial machine learning attacks has been an active area of research in the past few years [2] and many different attack and defense strategies have been proposed. On the defense side, the main solutions can be categorized into *adversarial training* [13, 24], *data transformation* [3], *gradient masking* [42], and *moving target defenses* [51]. Despite their many advantages, these methods often have poor generalization [13, 28, 41]. Additionally, adversarial training tends to be more computationally complex than standard training [57]. This complexity increases further when continuous retraining of the classifier is necessary [51]. Moreover, these methods require full access to the classifier, which is often proprietary. This makes them impractical for real-world autonomous systems due to the large models, limited computational power on the mobile device, and the need for retraining the classifier. For instance, the state-of-the-art moving target defense (MTD) technique [51] demands continuous training and additional models, resulting in significant overheads such as area, latency, and power consumption.

To address these issues, new methods based on *adversarial purification* have been proposed [41]. The key idea is to develop a machine learning block to purify adversarially perturbed images *before* classification. In comparison to adversarial training methods, purification can defend against unseen threats in a cascadable plug-and-play manner without retraining the classifiers. In the context of autonomous mobile systems, such a defense strategy is much more desirable because retraining the classifier is either not feasible (proprietary) and/or extremely time-consuming. Furthermore, purification methods do not make assumptions on the form of attack and the classification model and thus can defend different pre-existing classifiers, making them more applicable in diverse scenarios (e.g., different models of autonomous systems, applicability in various geographical locations, etc.).

Early attempts at purification primarily concentrated on encoder-based and GAN-based techniques for data transformation [38, 47]. However, they proved inadequate in ensuring robustness against powerful and adaptable attacks, such as white-box projected gradient descent [41, 60]. Recently, more potent purification models have emerged, utilizing ***diffusion models*** [49]. These models involve initially mixing the input with a small amount of noise through a forward diffusion process, followed by the restoration of the clean image via an iterative reverse generative process.

While more effective than existing invasive adversarial training models and earlier purification ones, the current state-of-the-art diffusion purification methods face ***significant latency issues*** because both the forward and backward processes *may need thousands of steps* to achieve the desired quality for the purified image. As we will show in our experiments, such a purification process *takes hundreds of milliseconds* to process a single image on a standard mobile device. This makes it unsuitable for autonomous mobile systems that need low-latency processing and real-time decision-making.

To address the need for a *cascadable*, *generic*, *non-invasive* but *fast* purification model, we develop **LightPure**. Our model leverages a lightweight diffusion strategy to achieve purification where the noise is added in one step and the recovery is also done in one shot. Our key idea is to develop a carefully designed GAN model instead of using an expensive iterative diffusion process. As we will explain in this paper, a naive design of the GAN, proposed in prior work [47], will result in either low robustness against adversarial and adaptive attacks, low accuracy due to poor noise recovery, and/or high latency. Instead, our design balances the robustness, accuracy, and latency by introducing several new contributions including a novel GAN structure and employing multiple new techniques during its training that combine diffusion models, GANs, and similarity-based loss estimation.

A high-level overview of our method is shown in Figure 1. LightPure can be applied as a plug-and-play component as it does not require any prior knowledge about the classifier. Details of our design are provided in Section 3. It is also important to mention that purification is an orthogonal defense method to adversarial training, hence purification can be ***combined*** with existing defense methods including MTD and adversarial training (i.e., feeding the purified images from our method to the adversarially trained classifiers).

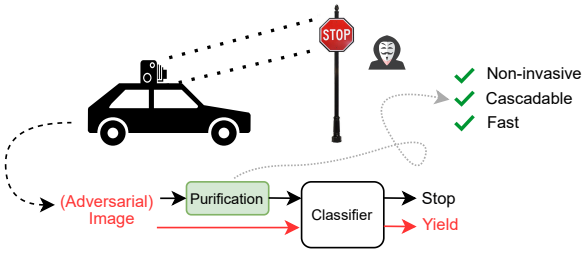


Figure 1: To protect the system from adversarial attacks, we develop a method to purify the image before classification. Our method does not require access to the classifier and is significantly faster than the state-of-the-art while maintaining robustness and accuracy.

We develop a ***proof-of-concept*** for LightPure using an Nvidia Jetson Orin Nano board. Using standard datasets like CIFAR-10 and GTSRB, we show that our system can achieve strong resistance against different types of black-, gray-, and even white-box attacks, similar to leading defense methods such as adversarial training and purification. Additionally, we analyze the speed and accuracy of our model, demonstrating a significant improvement in latency compared to the best purification models for mobile systems. Our results show that LightPure ***outperforms existing GAN-based and diffusion-based methods*** by 2x and 10x in terms of latency, respectively, while also achieving slightly *better* robustness and accuracy. Furthermore, compared to state-of-the-art non-purification methods (MTD [51]), LightPure improves the latency by more than 2x while also outperforming them in terms of accuracy and robustness.

In short, the contributions of this paper are as follows:

- A new latency-aware diffusion model based on a GAN architecture and a novel training scheme to achieve robust and low latency purification.
- Improving the accuracy of the diffusion model by introducing a feedback-based accuracy-aware algorithm.
- Proof-of-concept implementation of our system on a Jetson Nano embedded system.
- Evaluating LightPure's robustness, accuracy, and latency using standard benchmarks. Comparing it with state-of-the-art adversarial defense models [51].
- Our design and evaluations are publicly available and open-sourced at https://github.com/ssysarch/LightPure.

2 BACKGROUND

Adversarial Attacks. Machine learning models are vulnerable to a range of attacks targeting their availability, integrity, and privacy. This paper focuses on *evasion* attacks where the goal is to craft an input during inference time to cause misclassification by the victim model. More formally, the objective is to find an adversarial input, $\tilde{x}$, such that for a function, F, $F(\tilde{x}) \neq F(x)$, where x is the original input.

Several methods exist for crafting adversarial samples. For example, the Fast Gradient Sign Method (FGSM) was designed to attack classification models that utilize Stochastic Gradient Descent (SGD) [17]. FGSM calculates the gradients of the model's loss function with respect to each pixel, then perturbs the input data to maximize the loss function:

$$\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon \operatorname{sign}\left(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)\right). \tag{1}$$

Given that FGSM relies on the model's parameters, it is generally considered a *white-box attack*, indicating the attacker has full access to the model's architecture and parameters.

The Projected Gradient Descent (PGD) is a more powerful multi-step variant of FGSM [37]. PGD operates on a schedule of iterative perturbations where the noise is clipped by a maximum allowed perturbation ϵ on each iteration t. More formally:

$$\mathbf{x}^{(t+1)} = \operatorname{clip}_{\mathbf{x},\epsilon}\left(\mathbf{x}^{(t)} + \alpha \cdot \operatorname{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}^{(t)}, y))\right). \tag{2}$$

The PGD process is solely guided by maximizing the model's loss without accounting for the confidence of the model's prediction. Similar to the FGSM, the PGD and its variants are also *white-box attacks*. However, these attacks can still be utilized in *black-box* and *gray-box* scenarios. In *black-box attacks*, the internal details of the target model are completely abstracted or hidden from the attacker. Resilience to *black-box attacks* is fundamental to the robustness of a model. By training an FGSM or PGD attack on a *white-box* SGD model, the attack may be effective against a *black-box* target SGD model. A *gray-box attack* lies between both *white and black-box* scenarios where the attacker is given a portion of the model, while the rest is hidden. In systems with a purifier and a classifier, a possible *gray-box attack* could consist of the attacker having access to the classifier but not the purifier and/or the adversary knowing the architecture of the purifier and classifier but not the internal gradients. In this context, defining an attacker's access to the model means the attacker knows the parameters of the model, not that they have physical access to the model. Further details about our threat model, assumption, and attack strategies are discussed in Sections 3 and 4.

Defense Mechanisms. Several techniques exist to defend against adversarial evasion attacks. These include adversarial training, data transformation, gradient masking, and purification.

Adversarial training involves adding perturbed images into the training data for the classifier [11]. The idea is that the classifier would be able to learn what perturbations adversaries are using in attacks to improve model robustness.

However, this method suffers drawbacks since generating such perturbed images for training is expensive, and this training method can decrease the classifier's accuracy on clean images.

Data transformation involves applying linear transformations such as Principal Component Analysis (PCA) on the input data before classification [3]. The idea behind these types of defenses is that the transformations will remove the adversarial perturbations of the input before classification to improve robustness.

Gradient masking is designed to increase the difficulty of generating adversarial samples given a classifier [1]. By purposely introducing random noise into the input of the classifier, it is harder for an adversary to determine how to perturb the input in order to degrade the performance of the classifier. This method also suffers the drawback of decreased accuracy on clean images, as purposely introducing noise can interfere with the classifier's performance.

To address this, a *moving target* defense (MTD) has recently been proposed [51, 52]. The key idea is to leverage multiple independently trained models during inferences to increase robustness against adversarial inputs. Furthermore, MTD dynamically retrains the networks to further improve robustness. The main drawbacks of MTD are: ***(i)*** it requires multiple models (up to 20) during inferences to achieve robustness. This in turn significantly increases the latency, storage, and energy overheads. ***(ii)*** MTD requires model retraining. While for small models this is possible, for large models this is not feasible.

More recently, methods based on *purification* have been proposed [47, 59]. Purification involves using a purifying model to remove the adversarial perturbations on input images before classification The purified input is passed to the classifier for classification. The main advantage of purification is that it is independent from the classification, hence retraining the classifier is not needed. Furthermore, it is an orthogonal solution, hence it can be combined with MTD and/or adversarial training.

Adversarial Purification. Methods based on purification are first proposed in MagNet [38] and DefenseGAN [47]. Specifically, MagNet leverages an auto-encoder to move adversarial examples closer to the manifold of legitimate examples. The robustness is further improved by utilizing a collection of auto-encoders during runtime and having a mechanism to randomly pick one of them.

DefenseGAN [47] further improves this by introducing a GAN structure. Despite these advantages, the performance usually falls behind current adversarial training methods [11, 24, 44], particularly against adaptive attacks where the attacker has the full knowledge of the defense method [1, 55]. This is usually attributed to the shortcomings of current generative models used as a purification model, such as mode collapse in GANs [16] and the lack of randomness [41, 43].

Recent advancements in purification research have been driven by the use of *diffusion models*. This idea was first introduced in Diffpure [41]. It includes two steps: *(i)* adding noise to the input in the forward process with small but repetitive diffusion timesteps, *(ii)* removing the noise through a generative (backward) process by solving a reverse stochastic differential equation (SDE). Several follow-up works improve the speed and robustness of this process by introducing new techniques for the forward and/or backward processes [40, 58, 60].

More formally, given an adversarial sample $\mathbf{x}_0 \sim q(x_0)$, the first step in diffusion-based adversarial purification is adding noise to smooth the adversarial perturbations. In diffusion models [22, 49], this is done via the forward diffusion process of T steps and variance schedule $\{\beta_t\}_{t\in[T]}$ modeled by a conditional probability distribution:

$$\begin{aligned} q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) &= \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \\ q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) &= \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}). \end{aligned} \tag{3}$$

The total amount of noise inserted should be significant enough to remove the perturbations while preserving the global semantics of the image for accurate classification. Adversarial perturbations are typically small and therefore can be eliminated using fewer timesteps than generative tasks. Empirically, it has been shown that it is only necessary to inject noise for 7.5% to 15% of the total time steps required for the full diffusion process, depending on the dataset [41].

The next step is to denoise $\mathbf{x}_T$ to obtain the purified image $\hat{\mathbf{x}}_0$, which is typically defined in diffusion models with the distribution

$$\begin{aligned} p_\theta(\mathbf{x}_{0:T}) &= p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \\ p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) &= \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)\right). \end{aligned} \tag{4}$$

However, this process is extremely slow because T must be large enough to ensure that the step size β_t is sufficiently small for the Gaussian assumption on the denoising distribution to hold [14]. This is typically in the order of hundreds.

Our approach builds on established purification insights but introduces a new architecture tailored to address the practical drawbacks seen in current methods, especially in mobile systems. Details of our design are presented in the next section.

3 DESIGNING LIGHTPURE

The main objective of our model, LightPure, is to design a fast and efficient purification model that is suitable for

resource-constrained mobile systems. In the following, we first explain our threat model and assumptions and then describe the model in detail.

3.1 Threat Model and Assumptions

We are concentrating on mobile autonomous systems that use cameras and deep neural networks for sensing and perception. Examples include autonomous cars, drones, and robots. Compared to high-end servers, these mobile devices have often limited resources including CPU and GPU computational power, memory storage, and energy. Yet, they require real-time processing given the time-sensitive cyber-physical nature of their operations (e.g., path planning, driving, etc.).

We focus on evasion attacks during inference time, specifically image classification using a deep neural network. This is a fundamental basic block for perception in many mobile autonomous systems. We assume that the model is trained correctly and that even standard defense mechanisms such as certified robustness and adversarial training were applied during the training.

An adversary can control the inputs and can generate adversarial samples using state-of-the-art techniques. Particularly, we utilize standard and widely used AutoAttack benchmarks [10, 11] for crafting adversarial samples. We consider both targeted and non-targeted attacks. Further, we assume that the adversary has either *(i)* full knowledge about the internals of the system, including the structures and the internal values (gradients). This is referred to as a *white-box* attack; or *(ii)* partial knowledge where the attacker only has access to the gradient of the classifier and does not know anything about the purifier. We refer to this as a *gray-box* attack; or *(iii)* no knowledge about the system, considering the model as a *black-box*.

The assumption of white-box attacks is particularly strong in the context of mobile systems. Previous studies in this field operated under the assumption that attackers do not possess physical access to internal system values, such as gradients. Consequently, they considered white-box attacks to be outside the scope of their research [51, 52]. The rationale is that if attackers did have such access, they could potentially carry out more direct and powerful attacks without requiring adversarial inputs. This paper, however, examines all three types of attacks and presents findings for each. Nonetheless, it is highlighted that gray-box attack scenarios are the most realistic in contemporary autonomous mobile systems.

3.2 LightPure Overview

As discussed in Section 2, purification models offer various advantages such as flexibility and suitability for autonomous mobile systems. However, the main issue is that existing purification models are slow. Our main contribution is creating a new purification model that is accurate, fast, and robust. The key insight here is that to speed up the purification process, we should ***(i)*** use much larger time steps (meaning fewer iterations), and ***(ii)*** make denoising (both forward and backward) less complex. Consequently, if we increase the step size, β_t in Equation 3, to decrease the number of steps, the true denoising distribution becomes more complex and multimodal. Our main observation is that a GAN model can be used since they have been proven effective in modeling such distributions in the image domain [30, 58]. Furthermore, evaluating GANs is much more lightweight compared to SDEs which were used in prior methods, further improving the speed.

Designing a GAN-based purification model, however, is ***not trivial***. The important consideration is that in an autonomous mobile system, three important metrics should be jointly optimized: *accuracy*, *robustness*, and *latency*. Naive design decisions could lead to poor designs where some or all metrics are sacrificed in favor of others (e.g., poor accuracy but good robustness and latency, large latency with good accuracy, etc.). To optimize this, we design LightPure. Our design has two important contributions: ***(i)*** **Latency-aware robust diffusion model** and ***(ii)*** **Accuracy-aware training scheme**. Note that our method is fundamentally different than GAN-only methods for purification [38, 47]. These methods do not employ diffusion and as a result, are significantly less robust against adaptive attacks [41]. As we will show, combining GANs in a diffusion model, however, is non-trivial and requires a careful set of considerations.

Figures 2 and 3 present the main steps in our design. In the following, we explain each component in detail.

3.3 Latency-Aware Diffusion Model

To explain our design, we first provide its high-level intuition and then present the formal details. Recall that our goal is to design a purifier that has a low number of denoising steps and each step is preferably simple. As a result, initially, we aim for a method that has only one forward step (diffusion) and one backward step (denoising). Internally, the design includes a generator that learns how to generate (denoise) an input using a discriminator that can distinguish between noisy (fake) and clean (real) images.

Our initial analysis revealed that this model lacks generalization and robustness primarily because it predominantly learns to replicate the input. While this is desirable for standard image generation tasks, it proves detrimental for adversarial purification. In such cases, closely mimicking the input is insufficiently robust, given that the input image may contain adversarial elements (recall that the objective is to *eliminate* adversarial noise). The only benefit of mimicking the input is that the 'clean' data accuracy remains unchanged

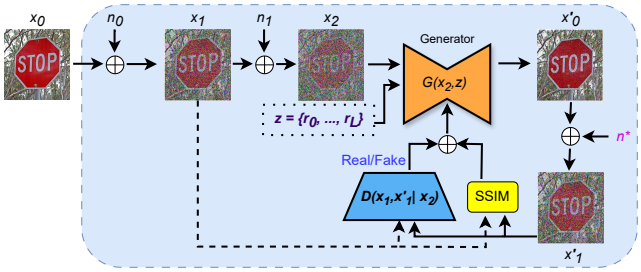


Figure 2: Training the purifier involves multiple steps. The original (clean) image is first diffused in two steps. The perturbed image is then fed into a generator. The generator is trained using a loss function that is a combination of a conditional discriminator and similarity (SSIM) losses.

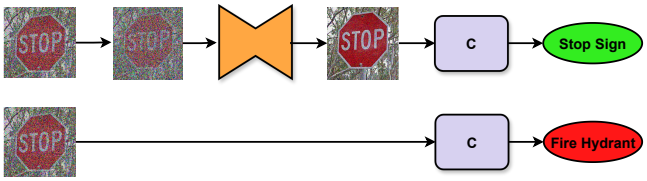


Figure 3: The trained generator is used during the inference to purify images dynamically.

as the purification does not significantly change the input (note that this is desired for image generation scenarios).

To address this, we've made a new observation that a two-step diffusion (forward) process could potentially solve the issue. Essentially, our model first generates a noisy image (depicted as x_1 in Figure 2) from the original clean image. Instead of using this for the discriminator and generator, we create *another* noisy image by adding more noise to x_1 (and not the clean image). The goal is to then use these two noisy samples (x_1, x_2) to train the generator and discriminator ($G()$ and $D()$ in Figure 2). The idea is for the model to learn to denoise x_2 given a noisy input x_1. In contrast, originally (with just one step), the model learned how to denoise x_1 given a clean input x_0. The key difference lies in the fact that during inference, inputs are more similar to x_1 than x_0, hence the purification will be more robust. Note that here we don't make any assumption about the noise and/or distribution. Instead, the observation is that the model should learn to purify a noisy image rather than mimicking a clean one. In Section 6, we will show that our model remains robust under different noise distributions, further confirming this claim.

The steps during inference are also shown in Figure 3. Here, we only add one more step (given that the input is already noisy) and the goal is to generate a clean image from these two samples.

To enhance robustness further, we observe that the discriminator could also benefit from a two-step approach. Instead of using x_2 and $G(x_2)$ for the discriminator (to distinguish between noisy and clean images), we compare x_1 and its noisy version, x'_1, which is created by adding noise to the purified image (shown as x'_0 in Figure 2). The idea is for the generator function to learn how to *completely* purify the image (i.e., one-shot) using x_2 without recreating x_1 (which remains noisy). In simpler terms, since the objective is to purify adversarial samples, the generator should be able to produce a clean image (similar to x_0) from a noisy sample, while the discriminator should be able to distinguish between an original noisy image and its fake version (purified and then perturbed again). Additionally, guiding this process using x_2 as a condition further improves the discriminator's accuracy (more details later).

More formally, to train our GAN for denoising with larger step sizes, we reduce the number of timesteps without having to increase β_t by setting $T = 2$ and train a time-independent discriminator and a one-shot generator. Specifically, from the raw image $\mathbf{x}_0$, we obtain $\mathbf{x}_1$ and $\mathbf{x}_2$ by diffusing for one and two timesteps, respectively:

$$\mathbf{x}_1 = \sqrt{1-\beta_1}x_0 + \sqrt{\beta_1}\epsilon. \tag{5}$$

$$\mathbf{x}_2 = \sqrt{1-\beta_2}x_1 + \sqrt{\beta_2}\epsilon. \tag{6}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and β_1, β_2 is the variance schedule. Our generator, a one-shot model denoted as $G_\theta(\mathbf{x}, \mathbf{z}) : \mathbb{R}^N \times \mathbb{R}^L \rightarrow \mathbb{R}^N$, models the denoising distribution $p_\theta(\mathbf{x}_1 \mid \mathbf{x}_2)$. In other words, it predicts $\mathbf{x}'_0$ from $\mathbf{x}_2$ and an L-dimensional latent variable $\mathbf{z} \sim p(\mathbf{z}) := \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$:

$$\mathbf{x}'_0 = G_\theta(\mathbf{x}_2, \mathbf{z}). \tag{7}$$

$\mathbf{x}'_1$ is sampled using the posterior distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ given $\mathbf{x}_2$ and $\mathbf{x}'_0$ (similar approach has been proposed in prior work by Xiao et al. [58]). The noise is denoted by n^* in Figure 2.

The discriminator with parameters ϕ, denoted as $D_\phi(\mathbf{x}, \mathbf{x}_2) : \mathbb{R}^N \times \mathbb{R}^N \rightarrow [0, 1]$, takes two N-dimensional inputs, $\mathbf{x}$ and $\mathbf{x}_2$, and determines whether $\mathbf{x}$ is an output from the generator ($\mathbf{x}'_1$) or from a real sample ($\mathbf{x}_1$). We discriminate on a diffused input to prevent overfitting of the discriminator, since the diffusion process smooths the data distribution [36]. The loss function of the discriminator quantifies how well the discriminator can differentiate between real and denoised samples by comparing its predictions to the actual labels of the data. It is formalized as follows:

$$\begin{aligned}\min_\phi \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_1|\mathbf{x}_0)q(\mathbf{x}_2|\mathbf{x}_1)}\left[-\log\left(D_\phi(\mathbf{x}_1, \mathbf{x}_2)\right)\right] \\ + \mathbb{E}_{q(\mathbf{x}_2)p_\theta(\mathbf{x}_1|\mathbf{x}_2)}\left[-\log\left(1 - D_\phi(\mathbf{x}'_1, \mathbf{x}_2)\right)\right].\end{aligned} \tag{8}$$

For the generator's architecture, we follow Xiao et al. and use StyleGAN, a non-saturating GAN variant [25, 26, 58].

While we considered other architectures, we found that StyleGAN achieves the right balance between computational complexity and accuracy.

Since the goal of the generator is to generate samples that the discriminator classifies as real, it is penalized by the loss function when the discriminator correctly identifies its output as fake. The loss function is formulated as follows:

$$\min_{\theta} \mathbb{E}_{q(\mathbf{x}_2)} \mathbb{E}_{p_\theta(\mathbf{x}_1|\mathbf{x}_2)} \left[-\log \left(D_\phi \left(\mathbf{x}'_1, \mathbf{x}_2 \right) \right) \right]. \tag{9}$$

Note that, we did not further try to optimize the GAN structure as our main novelty relies on the design algorithm (two-step forward and one-shot reverse training) rather than the GAN architecture itself. We left further optimization of the internal architecture to future work. Details of the network are provided in Section 4, and are shown in Figure **??** (see Appendix).

3.4 Accuracy-Aware Denoising Model

The objective in LightPure is to optimize robustness, latency, and accuracy simultaneously. The design outlined above enhances both robustness and latency by employing a one-shot generation during inference, as depicted in Figure 3. This approach is fast and utilizes a generalized model that is robust. However, our analysis revealed that the purification process significantly affects downstream task accuracy due to its invasive nature. To address this issue, we enhance LightPure by introducing a new denoising step that considers accuracy as a factor.

Inspired by Cycle-Consistent Adversarial Networks [63], to improve the quality of denoised images, we incorporate a metric comparing the similarity between a denoised image and the original clean image into the generator's loss function [63]. Specifically, we use the Structural Similarity Index Measure (SSIM), which is based on the computation of three factors: luminance, contrast, and structure [56, 62]. The SSIM calculates the similarity between x_1 and x'_1, adding it as an extra component to the generator's loss function. The updated loss function is

$$\min_{\theta} \mathbb{E}_{q(\mathbf{x}_2)} \mathbb{E}_{p_\theta(\mathbf{x}_1|\mathbf{x}_2)} \left[-\log \left(D_\phi \left(\mathbf{x}'_1, \mathbf{x}_2 \right) \right) \right] + \lambda (1 - \text{SSIM}(x, y)). \tag{10}$$

Here, λ is a regularization factor, balancing the importance of accuracy vs. robustness. Details are also shown in Figure 2. In Section 6, we will show that using SSIM as part of the loss function has a significant impact on improving the overall accuracy and robustness metrics across different datasets.

4 PROOF-OF-CONCEPT IMPLEMENTATION

Device. We utilize an Nvidia Jetson Orin Nano board to develop the proof of concept for LightPure. This board is commonly used for running deep neural networks in various applications like automotive, manufacturing, retail, and more. It features a six-core ARM CPU, a 1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores, and 8GB LPDDR4X memory. Despite being smaller and more resource-constrained compared to other Jetson family versions, our results in Section 6 demonstrate that LightPure can achieve real-time latency. Furthermore, the implementation can easily apply to other versions with minimal effort. Our board uses Linux (Ubuntu 22.04) and Nvidia-provided SDK (JetPack 6.0). Pytorch 2.1.0 is used to run the codes and models with CUDA toolkit version 11.5.

Generating Adversarial Samples Since the computing power of the Jetson is limited, we generate the adversarial images beforehand on a separate server with access to more computing power. For generating samples and training the model, we rely on a server equipped with two Nvidia A6000 GPUs featuring NVLink, 96 GB of memory, and a 10-core Intel 11th-gen processor. We use various datasets and attack strategies to create the baseline and adversarial samples. Details are explained in Section 5.

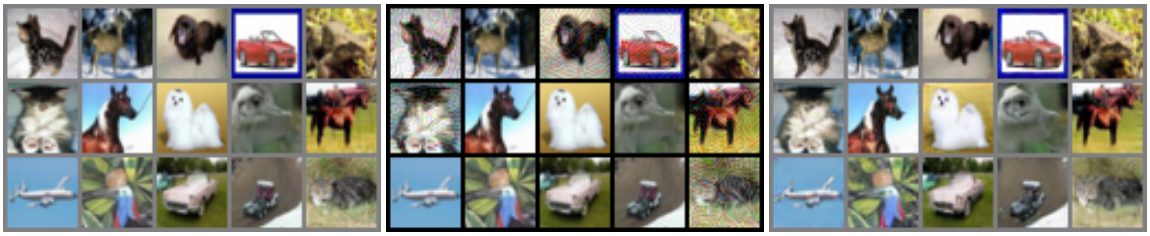
LightPure Architecture. The LightPure purifier is developed and trained on the server (details above). Once trained, the model weights are exported to the Jetson for testing. Our LightPure purifier has 45,610,598 parameters which take up 174MB of memory. The structure of our purifier (the generator) is shown in the Appendix (Figure **??**).

Our generator structure largely follows the U-net structure [46] which consists of multiple ResNet and Attention blocks. As mentioned in Section 3, the key novelty in our design is the usage of a two-step forward with a one-shot reverse mechanism that was further enhanced by incorporating SSIM (similarity) loss.

We use $\lambda = 3$ for the loss function (see Equation 10). Also, we select 256 for the latent dimension and 512 for the latent embedding dimension (z). For the diffusion process, we choose $\beta_1 = 0.0167$ and $\beta_2 = 0.0331$ for the noise levels, and 10^{-4} for the learning rate.

Once trained, the purifier is then paired with different classifiers (details in Section 5) during inference. *Note that, LightPure does not need to know the details of the classifier nor does it require retraining the classifier.* Readers can refer to Figures 2 and 3 for the design and steps during the training and inference. Our code and data will be ***open-source*** and publicly available.

The examples of clean, perturbed (using forward diffusion), and purified (using the generator) are shown in Figure 4. The examples show that LightPure can maintain the main visual aspects of the original image while being capable of removing the perturbations. The detailed results for accuracy and robustness are presented in Section 6.



(a) Clean. (b) (Adversarial) Perturbed. (c) Purified.

Figure 4: Examples of clean images, perturbed adversarial images, and purified images on the CIFAR-10 dataset.

Measuring Latency. Images are classified individually, and the time it takes to classify each image is measured. To obtain the duration for each model to return a result, we utilize Python's built-in time package's `perf_counter()` function. We record the time just before classification starts and immediately after classification produces a result. By calculating the difference, we determine the time it took to classify one image. The latency is the averaged number over 1000 instances. This approach is employed to emphasize the actual latency, rather than measuring the end-to-end (sensor-computation-actuation) latency, which could be affected by sensor/actuator and/or network delays. Additionally, we did not take batching into account, as the assumption is that the model must process images individually in a real-world system, such as autonomous driving.

5 EVALUATION SETUP

Metrics. We use two main metrics, clean accuracy and robust accuracy. The 'clean accuracy' or 'standard accuracy' refers to the performance of the model on untampered data. When employing a purifier before the classifier, there might be a slight decrease in clean accuracy due to perturbations. Note that this is common even for other defense mechanisms such as adversarial training. The reduction in accuracy is primarily because training on adversarial examples can inadvertently influence the model's learning in a way that slightly compromises its performance on clean data. Therefore, it is ideal to have zero or minimal impact on clean accuracy when adding purification.

We use three standard configurations for 'robust accuracy'.

1-Black-Box Attack: In this setup, the adversary has no knowledge about either the classifier or the purifier. Essentially, the adversary is operating without understanding the internal mechanisms or configurations of either component.

2-Gray-Box Attack: The adversary is knowledgeable about both the classifier and a purifier (their architecture), but the classifier used for generating the attack is not the one targeted in the evaluation. The attack is crafted based on this external purifier and the classifier, and the robustness is tested against the intended (targeted) purifier and classifier.

3-Complete White-Box Attack: This setup allows full access to both the classifier and the purifier including the architecture, hyperparameters, and the ability to calculate the full gradients of both the classifier and purifier. Note that in real mobile systems, this is less realistic as it requires full physical access to the device and its internal computations. Given such a strong capability, the adversary could possibly launch a more direct and more powerful attack instead of launching an ML-based evasion attack.

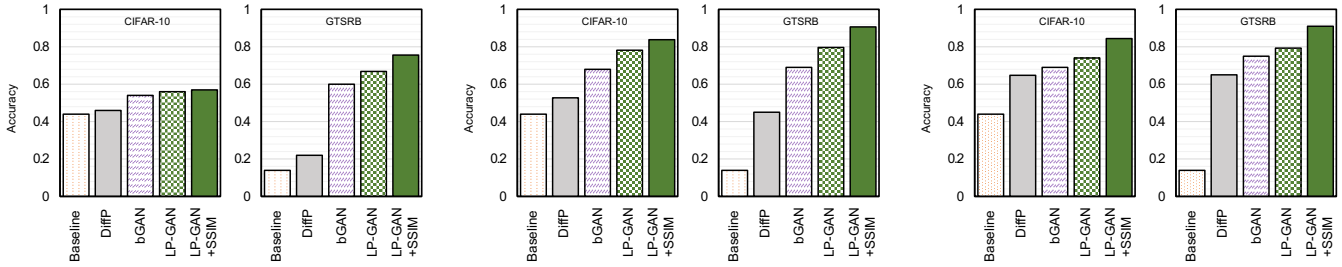
Attack Methods. To assess the robustness of our model against adversarial attacks, we employ the RobustBench benchmark, which uses AutoAttack, an ensemble of white-box and black-box attacks [10, 11]. Autoattack utilizes Auto-PGD, a more powerful variant of PGD that automatically adapts the step size. This can be applied to various loss functions, including cross-entropy loss (APGD-CE) and difference of logits ratio loss (APGD-DLR):

$$\mathrm{CE}(x, y) = -\log p_y = -z_y + \log\left(\sum_{j=1}^{K} e^{z_j}\right) \tag{11}$$

$$\mathrm{DLR}(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}} \tag{12}$$

Due to the stochastic nature of our model, we utilize the AutoAttack version against randomized adversarial defenses (APGD-rand), which is an ensemble of APGD-CE and APGD-DLR. To counter randomness, APGD-rand applies Expectation over Transformation (EOT), which averages 20 computations of the gradient at the same point, to get the direction for the update step [1]. Additionally, we run an experiment with APGD-CE without EOT to evaluate our model against faster but less powerful attacks. Similar to prior work, we choose to use the L-infinity norm, we set the number of restarts to 1, set the number of target classes to 9, and train with a learning rate of 8/255.

To avoid out-of-memory issues, prior purification methods such as Diffpure calculate the gradients using an adjoint



(a) White-Box. **(b) Gray-Box.** **(c) Black-Box.**

Figure 5: Robustness for three different configurations using CIFAR-10 and GTSRB datasets. The results are for the baseline (no protection), Diffpure (DiffP), a baseline GAN (bGAN), and *our method* (LightPure) which includes the latency-aware diffusion model (LP-GAN) and the diffusion model with the accuracy-aware model (LP-GAN+SSIM).

method, which has been reported to be sensitive to numerical errors [41, 67]. When compared to an attack using full gradients obtained from direct backpropagation, the adjoint method returns a higher robust accuracy, leading to an overestimation of the robustness of the model [31]. Therefore, in this paper, we utilize full gradients for all attacks.

Datasets. We use *three* standard datasets. ***CIFAR-10*** is a 10-class dataset comprising 50,000 training samples and 10,000 test samples. Each sample is a 32×32 RGB color image. The 10 classes include airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. We use only the Test dataset to evaluate our accuracy and robustness, and to train our purifier we use only the Train dataset, but we do not use any label to train our purifier.

GTSRB (German Traffic Sign Recognition Benchmark) is a 43-class dataset with over 50,000 images ranging in size from 15×15 to 250×250 pixels. We divide them into training, and test datasets with 45000 and 5000 samples, respectively. Also, we resize the pictures to 32x32 for training and evaluation.

Tiny ImageNet is a 200-class dataset derived from ImageNet [12], containing 100,000 training images and 20,000 validation/test images, each resized to 64×64 pixels. We used 5,000 images from the validation set for our evaluation [29].

Classifiers. For CIFAR-10, we use Resnet-56 [9], a publicly available state-of-the-art classifier with a top-1 accuracy of 94.37 percent on clean images. Resnet-56 has 855,770 parameters which takes up 3.3MB of memory in our embedded board (Jetson Orin Nano). For GTSRB, we train a Resnet-20. It achieves 97.26 percent accuracy on clean images. It has 272,474 parameters and takes up 1.04MB. For Tiny ImageNet, we train a ResNet-18 model. It achieves 45% accuracy on clean images. The model has approximately 11.2 million parameters and takes up 44.6MB.

For black-box attacks, we use Resnet-50 as the *shadow* model which will be used by the adversary to create the adversarial samples.

State-of-the-Art Models. In addition to LightPure, we also implement other models on Jetson and reported latency, accuracy, and robustness results. Specifically, we implement Diffpure [41]. We test performance with the purifying diffusion model using 10, 20, and 30 timesteps. The diffusion model has 35,746,307 parameters which uses 136.4MB of memory. Additionally, we compare LightPure with a GAN-only design (i.e., when no diffusion is used) based on the method proposed in Defense-GAN [47]. Lastly, we compare our method with a recently proposed technique, *Moving Target Defense* (MTD) [51]. We assume that MTD has at least 10 base models (for voting) with early stopping. Depending on the classifier, the MTD's storage overhead changes. On average, for CIFAR-10, it takes about 35MB of Jetson's memory. For GTSRB, the storage is about 12MB. Detailed results for different metrics and models are provided in Section 6.

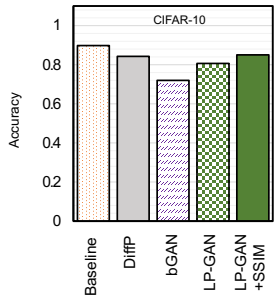
6 RESULTS

6.1 Latency, Accuracy, and Robustness

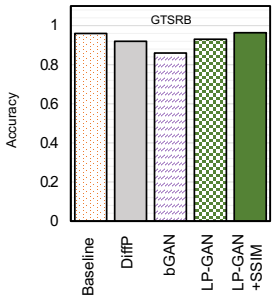
Robustness. We report the robustness of our method against the attack model and datasets described in Section 5. We report the results for the baseline (no protection), Diffpure (DiffP) [41], a GAN-only solution (i.e., without diffusion) based on Defense-GAN [47], which we refer to as baseline GAN (bGAN), and our method (LightPure). For our model, we report two configurations: the latency-aware diffusion model (LP-GAN) and the diffusion model with the accuracy-aware model (LP-GAN+SSIM). Results are reported for two primary datasets (CIFAR-10 and GTSRB), with additional comparisons made on Tiny ImageNet.

Results are reported for black-, gray-, and white-box adversary models (see Section 5), as shown in Figure 5.

Our evaluations demonstrate that *LightPure consistently achieves the highest robustness across different attack configurations.* While black-box configuration is evidently the most

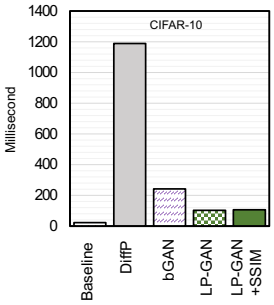


(a)

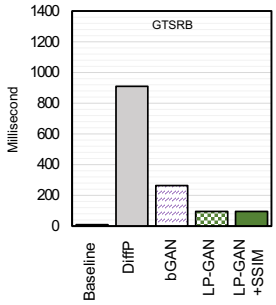


(b)

Figure 6: Clean accuracy (higher is better) for the downstream classification task using two different datasets.



(a)



(b)

Figure 7: Latency for purification+classification for different methods (lower is better). y-axis shows the latency in milliseconds.

secure, LightPure still manages to achieve notable robustness under white-box and gray-box attacks (especially for the GTSRB dataset). Comparing the two datasets, it is much harder to defend against white-box attacks (even with Diffpure) on CIFAR-10 than on GTSRB, indicating that internal information is more beneficial for fine-tuning APGD attacks on CIFAR-10. Furthermore, it should be noted that white-box attacks are considered less feasible in mobile systems due to the need for direct physical access, which is often not practical. In our view, gray-box attacks represent the most realistic scenario for an attack in a mobile system.

Looking more closely at grey-box results, LightPure achieves >80% robustness, on average, across datasets. Comparing LP-GAN and LP-GAN with SSIM, the results are improved by 5% for CIFAR-10 and by 10% for GTSRB, highlighting the usefulness of utilizing a similarity metric to further improve the generation accuracy.

Accuracy. Results for (clean) accuracy are reported in Figure 6. Recall that we use two different classifiers (details in Section 5) for the downstream classification task. It is important to analyze the impact of adversarial defense (e.g., purification, adversarial training, etc.) on the accuracy as retraining and/or adding and removing noise could potentially impact the clean accuracy.

As shown in Figure 6, LightPure incurs a minimal accuracy drop compared to the baseline. Specifically, for CIFAR-10 the drop for LightPure is about 4% when using the GAN+SSIM model. For GTSRB, the drop is almost zero (<.01%). Compared to other methods, the usage of an accuracy-aware training model using SSIM loss and a generalized (two-step) discriminator/generator training method (see Section 3).

Tiny ImageNet Results. While CIFAR-10 and GTSRB provide a solid basis for evaluating our method's robustness, we also tested our approach on the Tiny ImageNet dataset, a more challenging benchmark due to its numerous classes and reduced image size. Results are shown in Figure 8. Our model outperforms both DiffPure and the Baseline GAN in gray-box and white-box scenarios. Also, note that the baseline accuracy in all cases is much lower than that of other datasets due to the complexity of the classification task. These results further highlight the robustness of our method, even in more complex datasets like Tiny ImageNet.

Latency. We compare the inference latency (purification and classification combined) for different methods (refer to Section 4 for details of the device and measurement methodology). The results are displayed in Figure 7 in milliseconds.

Compared to the baseline (classification only), purification methods add between >50x to 4x. The reason for added latency is the extra processing needed during purification. Among different methods, LightPure achieves the lowest. We improve DiffPure's latency by an order of magnitude since our method does not need multiple steps and each step is much simpler. Furthermore, compared to the baseline GAN method [47], LightPure is more than twice faster mainly because the generation is one-shot as opposed to the multi-step generation needed in state-of-the-art GAN methods.

Also, note that the latency for LP-GAN and LP-GAN+SSIM is the same since the generator architecture and steps stay similar for both methods. The difference is during the training phase which does not impact the inference latency.

6.2 Comparison with Moving Target Defense (MTD) Method

We further compare our results with the *state-of-the-art non purification methods*. Specifically, we compare LightPure (LP+SSIM) with MTD [51] in terms of accuracy, latency, and robustness. Instead of using purification, MTD leverages several base models (10-20 models) and develops a majority voting and detection method to eliminate adversarial attacks.

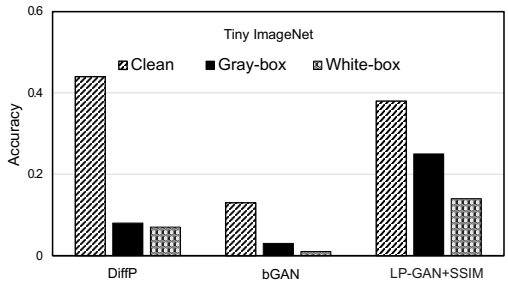
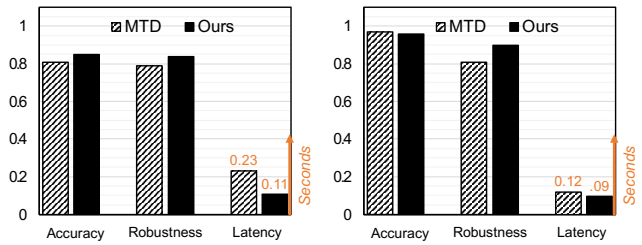


Figure 8: Comparison of accuracies across different models and attack scenarios on Tiny ImageNet.



(a) CIFAR-10 **(b) GTSRB**

Figure 9: Comparison between our method and state-of-the-art non-purification method, MTD [51]. Accuracy and robustness are normalized percentages (higher is better) while latency is in seconds (lower is better).

For accuracy and robustness, we use the results reported by the authors [51]. We use the gray-box attack configuration[1]. For latency, we assume that MTD queries at least 10 base models (this is a favorable assumption) using the early stopping scheme (with $w = 0.2$) [51].

Results are shown in Figure 9. For CIFAR-10, LightPure achieves higher accuracy and robustness. The results are fairly comparable in general. The clear advantage of LightPure, however, is in latency as it has more than 2x lower latency for the same classifier. The main reason for this is that LightPure leverages a one-shot purifier while MTD needs several serialized queries from the classifier. The problem is exacerbated when using a more sophisticated classifier.

Looking at the results for GTSRB in Figure 9 (b), similar trends can be observed. Compared to the CIFAR-10 dataset, LightPure achieves better robustness but similar accuracy (both around 96%). Similar to CIFAR-10, LightPure outperforms MTD in terms of latency. The difference is smaller since the classifier is faster (smaller).

[1]MTD's threat model is neither black-box nor gray-box. To be fair, we assume the gray box in our setting which has lower robustness.

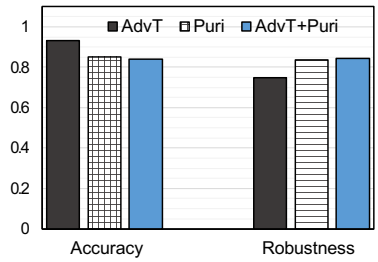


Figure 10: Accuracy and robustness results for combining adversarial training (AdvT) with our purification (Puri) model.

The storage overhead can be also compared. Based on the implementation results in Sections 4 and 5, LightPure has about 174MB storage overhead. For MTD, the overhead depends on the classifier. For a smaller Resnet-18 classifier, this is about 12MB. For a larger Resnet-56 classifier, the storage overhead is about 35MB. Please note that while the storage overhead of LightPure is higher, the overall storage requirement in LightPure is still only 2% of the total memory. In both cases, the storage requirement is negligible.

Overall, the key takeaway is that LightPure is more beneficial for larger classifiers when latency overhead is large. Furthermore, LightPure is beneficial in scenarios where retraining is prohibitive or infeasible. Given the trend of using larger and more sophisticated models in autonomous mobile systems, we believe LightPure provides an effective solution for current and future models. Additionally, in the following section, we'll demonstrate how the method can be further combined to improve the system's overall performance.

6.3 LightPure with Adversarial Training/MTD

Since our proposed purification model is an orthogonal defense method to adversarial training and/or moving target defense, we can also combine our method with them by feeding the purified images from LightPure (LP+SSIM) to the adversarially trained and/or MTD classifiers.

To experimentally evaluate this, Figure 10 shows that this combination ("AdvT + Puri") can improve the robust accuracies against different attack models. The figure shows the accuracy and robustness under three different setups: *1)* Training a robust classifier using adversarial training [11]. *2)* Using LightPure to purifier the image but using a regular classifier (not adversarially trained); and *3)* Combining the robust classifier in (1) with our purifier (2).

For robustness, we report the gray-box accuracy for all three cases. The results in Figure 10 show that the overall robustness slightly improves when combining adversarial

training (AdvT) with adversarial purification (Puri). This, however, comes at the cost of a slight reduction in accuracy as both AT and Puri cause degradation in image quality and overall classification accuracy due to adding perturbations to the model and/or input. Although not shown in this section, a similar approach could be used to combine Puri with MTD. Even further, Puri, AdvT, and MTD can be combined if desired.

The main takeaway from this experiment is that LightPure proposes an orthogonal solution to existing non-purification solutions. When combined, the results could improve. However, the combination introduces new tradeoffs between accuracy, robustness, and training time. The designers can ultimately choose the right balance to achieve their goals.

7 FURTHER ANALYSIS AND DISCUSSIONS

7.1 Ablation Study

As explained in Section 3, we utilize a two-step noise addition method during training. To highlight the importance of such a method, we initially experimented with a single-step noise addition process. However, we observed that this approach led to significant overfitting, particularly when using SSIM loss, as the generated image was directly compared to the original, uncorrupted image. *Our results show more than 50% reduction in robustness.* To address this issue, we introduced a two-step diffusion process. First, we generate a noisy image x_1 from the original clean image. Rather than using x_1 for direct training, we add an additional layer of noise to create a second noisy image x_2. The model is then trained to denoise x_2. After denoising x_2, we calculate the SSIM loss between the original x_1 and the x_1 generated from the denoised version of x_2. This step helps to increase generalization by ensuring that the model effectively handles various levels of noise, rather than merely replicating the clean images. This approach leads to better generalization and improved performance across different noise conditions.

7.2 Comparison to Latent Diffusion Models

In the context of adversarial image purification, we chose GAN-based networks over Latent Diffusion Models (LDMs) due to their superior effectiveness in preserving image accuracy. While LDMs are known for their speed, with a latency of around 300 ms- significantly faster than traditional Denoising Diffusion Probabilistic Models (DDPMs) [22, 45]- this speed does not necessarily translate to better performance in purification tasks. Our GAN-based model, with a latency of 200 ms, not only offers comparable speed but also ensures higher purification effectiveness. The key advantage of GANs lies in their ability to operate directly in the pixel space, where adversarial noise is introduced, enabling more precise noise removal without the complications introduced by transforming images into a latent space, as LDMs do.

Transforming images into a latent space, as required by LDMs, can obscure the details necessary for effective noise removal and potentially shift decision boundaries, complicating the purification process and reducing robustness. In contrast, our GAN-based approach maintains the image in its original domain, ensuring that the purification process directly addresses the adversarial noise at the pixel level. This approach results in better preservation of the image's accuracy and robustness, making GAN networks a more suitable choice for adversarial image purification, where maintaining the integrity of the original image is crucial.

7.3 Our Model Limitations

Our model's limitations include the necessity to train our GAN separately for each dataset and using different noise levels to determine the optimal level for purification. Additionally, determining the appropriate ratio of SSIM loss to GAN loss presents another challenge. In contrast, DiffPure does not require retraining the diffusion model to identify the required noise level for purifying adversarial samples [41]. Consequently, finding these hyperparameters makes training our model more challenging compared to DiffPure, which utilizes a pre-trained diffusion model.

7.4 Exploring Potential Architectural Optimizations in StyleGAN Models

Various architectural optimizations can be employed to enhance the efficiency and performance of our model. One area of interest is **pruning**, where reducing the model's redundancy by incrementally removing less impactful weights might decrease memory usage and improve computation speed. Other techniques such as structured and iterative pruning can be used to improve the latency without significantly impacting output quality [19]. Additionally, **knowledge distillation** could be examined, where a smaller "student" model would be trained to replicate the "teacher" model. This approach would focus on matching both the output distribution and intermediate feature maps to potentially reduce the model size and accelerate inference times [21].

8 RELATED WORK

Adversarial Defenses. First introduced by Madry *et al.*, *adversarial training* incorporates adversarial samples during training. Many improvements have been made to this technique, including the work done by He *et al.*, which injects trainable Gaussian noise at each layer of the model to introduce randomness [20]. Gowal *et al.* utilized generative models for data augmentation, with diffusion models resulting in the best robustness [18]. A major drawback of

adversarial training, however, is the need for retraining the classifier.

Adversarial purification, on the other hand, involves training a separate purification model that removes adversarial perturbations on input images before the downstream task. MagNet leverages auto-encoders to move adversarial examples closer to the manifold of legitimate examples [38], and Samangouei *et al.* proposes using GANs as the purification model and uses GD minimization to yield higher robustness [47]. More recently, Song *et al.* proposed PixelDefend, which relies on PixelCNN, an autoregressive generative model [53], and Yoon *et al.* used score-based generative models to purify adversarial samples [59]. Diffpure was the first attempt to use a diffusion process followed by a backward denoising process for purification [41].

As extensively discussed in this paper, LightPure improves the prior work in two major directions. Compared to GAN-only based methods, LightPure significantly improves robustness and accuracy while also slightly improving latency as it uses one-step forward and backward processes. Compared to diffusion-based models, LightPure significantly improves latency while also achieving better robustness.

Attacks to Physical Autonomous Mobile Systems. Also relevant to this work are methods that target physical sensors (e.g., camera, LIDAR, mmWave, RF, etc.) to attack autonomous mobile systems. Some attacks target the availability and/or integrity of the sensor by manipulating the environment and/or the operation of the sensors [65].

More relevant to this work are methods that create physical adversarial samples to impact the machine learning-based classification task [27, 35, 50]. These attacks can further be categorized as physical adversarial attacks [4] and physical sensor attacks [66]. In both categories, there is a rich literature for creating adversarial patches [32]. They commonly use similar methods (e.g., FGSM, PGD, etc.) to those considered in this paper to generate adversarial samples. Depending on the physical modalities and threat model, various methods could be used to conduct the attack (e.g., shooting a laser, jamming the signal, causing interference, 3D printing the object, etc [64].)

Compared to these methods, we consider a strong and comprehensive threat model ranging from fully white-box (complete knowledge) to black-box. As mentioned in the paper, we believe that gray-box attacks are the most reasonable assumption in our setting.

Comparison. We conclude this section by summarizing different solutions. In Table 1, we compare adversarial training, defense-GAN, DiffPure, and MTD on the metrics of latency, accuracy, robustness, and retraining.

Table 1: Comparing different methods for visual analysis on latency, accuracy, robustness, and retraining. For each property, a filled circle means better. (AT = Adversarial Training, DG = Defense-GAN, DP = DiffPure, MTD = Moving Target Defense).

	AT [11]	DG [47]	DP [41]	MTD [51]	Ours
Latency	●	◐	○	◐	●
Accuracy	●	◐	●	●	●
Robustness	◐	◐	◐	●	●
Retraining	○	●	●	○	●

9 CONCLUSIONS

This paper addressed the critical challenge of defending autonomous mobile systems against adversarial machine learning attacks, which pose significant threats to their reliability and safety. While previous countermeasures have shown promise, they often require invasive modifications to classifiers or incur high computational costs, making them unsuitable for resource-constrained mobile devices.

To overcome these limitations, we introduced an innovative purification approach that leverages a GAN framework. Our method achieves a remarkable balance between classification accuracy, adversarial robustness, and computational efficiency, making it well-suited for real-world applications where speed and resource constraints are paramount. Our key contribution was the design and implementation of a new one-shot generator that leverages a two-step forward diffusion process during its training. The method was further improved by introducing a similarity-based loss function for training the generator.

Through extensive experimentation and evaluation, we demonstrated the superiority of our approach over existing purification methods, achieving significant improvements in both latency and performance across various attack scenarios. Our method represents a notable advancement in the field of adversarial image purification, offering a scalable and effective solution for safeguarding autonomous mobile systems in practical settings.

ACKNOWLEDGEMENT

We thank our shepherd for their guidance. This work has been supported, in part, by CISCO Systems Inc., NSF grants CNS-2211301, CNS-2303115, and CNS-2312089, and a gift received from Accenture. The views and findings in this paper are those of the authors and do not necessarily reflect the views of Cisco, NSF, and Accenture.

REFERENCES

[1] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to

adversarial examples. In *International conference on machine learning*. PMLR, 274–283.

[2] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. 2021. Recent Advances in Adversarial Training for Adversarial Robustness. *arXiv preprint arXiv:2102.01356* (2021).

[3] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. 2018. Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. 1–5.

[4] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).

[5] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 176–194.

[6] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2267–2281.

[7] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 39–57.

[8] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX security symposium (USENIX Security 11)*.

[9] chenyaofo. 2021. PyTorch CIFAR models. https://github.com/chenyaofo/pytorch-cifar-models.

[10] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. 2020. RobustBench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670* (2020).

[11] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*. PMLR, 2206–2216.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[13] Hadi M Dolatabadi, Sarah Erfani, and Christopher Leckie. 2022. l_∞-robustness and beyond: Unleashing efficient adversarial training. In *European Conference on Computer Vision*. Springer, 467–483.

[14] W Feller. 1949. On the Theory of Stochastic Processes, With Particular Reference to Applications. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*. Univ of California Press, 403.

[15] Dario Floreano and Robert J Wood. 2015. Science, technology and the future of small autonomous drones. *nature* 521, 7553 (2015), 460–466.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[18] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. 2021. Improving robustness using generated data. *Advances in Neural Information Processing Systems* 34 (2021), 4218–4233.

[19] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. 1135–1143.

[20] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. 2019. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 588–597.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[23] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. 2017. Cyber-physical systems security—A survey. *IEEE Internet of Things Journal* 4, 6 (2017), 1802–1831.

[24] Xiaojun Jia, Yong Zhang, Baoyuan Wu, Ke Ma, Jue Wang, and Xiaochun Cao. 2022. LAS-AT: adversarial training with learnable attack strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13398–13408.

[25] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.

[26] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8110–8119.

[27] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. 2020. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14254–14263.

[28] C Laidlaw, S Singla, and S Feizi. 2021. Perceptual Adversarial Robustness: Defense Against Unseen Threat Models. In *International Conference on Learning Representations (ICLR)*.

[29] Y. Le and X. Yang. 2015. Tiny ImageNet Visual Recognition Challenge. CS 231N Course Project Report, Stanford University. http://cs231n.stanford.edu/reports/2015/pdfs/LeXie.pdf

[30] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

[31] Minjong Lee and Dongwoo Kim. 2023. Robust evaluation of diffusion-based adversarial purification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 134–144.

[32] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. 2019. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 1028–1035.

[33] Liangkai Liu, Sidi Lu, Ren Zhong, Baofu Wu, Yongtao Yao, Qingyang Zhang, and Weisong Shi. 2020. Computing systems for autonomous driving: State of the art and challenges. *IEEE Internet of Things Journal* 8, 8 (2020), 6469–6486.

[34] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. 2019. Edge computing for autonomous driving: Opportunities and challenges. *Proc. IEEE* 107, 8 (2019), 1697–1716.

[35] Giulio Lovisotto, Henry Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. 2021. {SLAP}: Improving physical adversarial examples with {Short-Lived} adversarial perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*. 1865–1882.

[36] Siwei Lyu. 2009. Interpretation and Generalization of Score Matching. In *The International Conference on Uncertainty in Artificial Intelligence (UAI)*. Montreal, QC, Canada.

[37] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

[38] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 135–147.

[39] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque. 2020. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems* 22, 7 (2020), 4316–4336.

[40] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International conference on machine learning*. PMLR, 8162–8171.

[41] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Animashree Anandkumar. 2022. Diffusion Models for Adversarial Purification. In *International Conference on Machine Learning*. PMLR, 16805–16827.

[42] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2016. Practical black-box attacks against machine learning, 2017. In *ACM Asia Conference on Computer and Communications Security*.

[43] Rafael Pinot, Raphael Ettedgui, Geovani Rizk, Yann Chevaleyre, and Jamal Atif. 2020. Randomization matters how to defend against strong adversarial attacks. In *International Conference on Machine Learning*. PMLR, 7717–7727.

[44] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. 2021. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems* 34 (2021), 29935–29948.

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv preprint arXiv:2112.10752* (2022).

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. 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*. Springer, 234–241.

[47] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*.

[48] Emerson Sie, Zikun Liu, and Deepak Vasisht. 2023. BatMobility: Towards Flying Without Seeing for Autonomous Drones. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.

[49] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR, 2256–2265.

[50] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. 2018. Physical adversarial examples for object detectors. In *12th USENIX workshop on offensive technologies (WOOT 18)*.

[51] Qun Song, Zhenyu Yan, and Rui Tan. 2019. Moving target defense for embedded deep visual sensing against adversarial examples. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 124–137.

[52] Qun Song, Zhenyu Yan, and Rui Tan. 2021. Deepmtd: Moving target defense for deep visual sensing against adversarial examples. *ACM Transactions on Sensor Networks (TOSN)* 18, 1 (2021), 1–32.

[53] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2018. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*.

[54] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*.

[55] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. *Advances in neural information processing systems* 33 (2020), 1633–1645.

[56] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[57] Eric Wong, Leslie Rice, and J Zico Kolter. 2019. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*.

[58] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. 2022. Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. In *International Conference on Learning Representations (ICLR)*.

[59] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. 2021. Adversarial purification with score-based generative models. In *International Conference on Machine Learning*. PMLR, 12062–12072.

[60] Jiawei Zhang, Zhongzhu Chen, Huan Zhang, Chaowei Xiao, and Bo Li. 2023. {DiffSmooth}: Certifiably robust learning via diffusion models and local smoothing. In *32nd USENIX Security Symposium (USENIX Security 23)*. 4787–4804.

[61] Yan Zhang, Yi Zhu, Zihao Liu, Chenglin Miao, Foad Hajiaghajani, Lu Su, and Chunming Qiao. 2022. Towards Backdoor Attacks against LiDAR Object Detection in Autonomous Driving. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. 533–547.

[62] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. 2015. Loss functions for neural networks for image processing. *arXiv preprint arXiv:1511.08861* (2015).

[63] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

[64] Wenjun Zhu, Xiaoyu Ji, Yushi Cheng, Shibo Zhang, and Wenyuan Xu. 2023. Tpatch: A triggered physical adversarial patch. *arXiv preprint arXiv:2401.00148* (2023).

[65] Yi Zhu, Chenglin Miao, Foad Hajiaghajani, Mengdi Huai, Lu Su, and Chunming Qiao. 2021. Adversarial attacks against lidar semantic segmentation in autonomous driving. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 329–342.

[66] Yi Zhu, Chenglin Miao, Hongfei Xue, Zhengxiong Li, Yunnan Yu, Wenyao Xu, Lu Su, and Chunming Qiao. 2023. TileMask: A Passive-Reflection-based Attack against mmWave Radar Object Detection in Autonomous Driving. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 1317–1331.

[67] Juntang Zhuang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan. 2020. Adaptive checkpoint adjoint method for gradient estimation in neural ode. In *International Conference on Machine Learning*. PMLR, 11639–11649.