FlowPure: Continuous Normalizing Flows for Adversarial Purification

Elias Collaert* **Abel Rodríguez*** **Sander Joos*** **Lieven Desmet** **Vera Rimmer**
DistriNet, KU Leuven
{elias.collaert, abel.rodriguezromero, sander.joos,
lieven.desmet, vera.rimmer}@kuleuven.be

Abstract

Despite significant advancements in the area, adversarial robustness remains a critical challenge in systems employing machine learning models. The removal of adversarial perturbations at inference time, known as *adversarial purification*, has emerged as a promising defense strategy. To achieve this, state-of-the-art methods leverage diffusion models that inject Gaussian noise during a forward process to dilute adversarial perturbations, followed by a denoising step to restore clean samples before classification. In this work, we propose **FlowPure**, a novel purification method based on Continuous Normalizing Flows (CNFs) trained with Conditional Flow Matching (CFM) to learn mappings from adversarial examples to their clean counterparts. Unlike prior diffusion-based approaches that rely on fixed noise processes, FlowPure can leverage specific attack knowledge to improve robustness under known threats, while also supporting a more general stochastic variant trained on Gaussian perturbations for settings where such knowledge is unavailable. Experiments on CIFAR-10 and CIFAR-100 demonstrate that our method outperforms state-of-the-art purification-based defenses in preprocessor-blind and white-box scenarios, and can do so while fully preserving benign accuracy in the former. Moreover, our results show that not only is FlowPure a highly effective purifier but it also holds a strong potential for adversarial detection, identifying preprocessor-blind PGD samples with near-perfect accuracy. Our code is publicly available at https://github.com/DistriNet/FlowPure.

1 Introduction

Deep neural networks continue to achieve remarkable success in various domains, yet remain vulnerable to adversarial attacks [30, 10]: carefully crafted perturbations to inputs that lead to incorrect model predictions. Recently, adversarial purification (AP) [28, 25, 36] has gained increasing attention as a promising strategy to enhance adversarial robustness of models. These approaches aim to remove adversarial perturbations during inference, eliminating the need for classifier retraining. Generative models are commonly employed for this purpose, with various techniques, such as GANs [27] and density models [28], being explored in prior work. However, the remarkable success of diffusion models [13] has placed them at the forefront of adversarial purification [36, 25, 19].

The core goal of diffusion models is to transform a pure Gaussian into a complex data distribution through two processes: (i) a forward stochastic process that iteratively adds noise to a sample until it becomes pure noise and (ii) a reverse process that gradually denoises samples towards the original data distribution. As such, diffusion-based purification relies on the injected noise diluting adversarial perturbations during the forward process. The reverse process can then be used to reconstruct clean samples by removing the resulting combination of injected Gaussian and adversarial noise.

*Equal contribution.

Preprint. Under review.

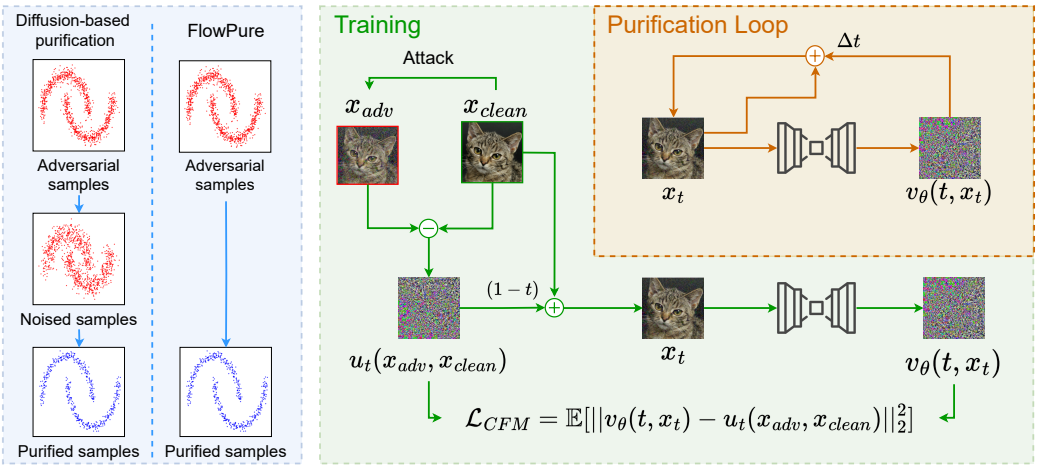


Figure 1: **Left:** Opposed to traditional diffusion-based purification methods that require adding Gaussian noise to samples, FlowPure can learn a direct transformation between arbitrary distributions. **Right:** Training and purification pipelines of FlowPure. The model $v_\theta(t, x_t)$ is trained to approximate the velocity field $u_t(x_{adv}, x_{clean})$ using the loss $\mathcal{L}_{CFM}$. The purification loop illustrates the update rule, which should be applied iteratively, moving the sample towards the clean distribution.

Recent advances in generative modeling have explored alternatives to diffusion models for learning transformations over complex data distributions. Flow Matching (FM) [20], an improved way of training Continuous Normalizing Flows (CNFs) [6], has emerged as a powerful framework, providing a scalable and stable approach to generative modeling. FM learns a vector field defining a continuous transformation between two arbitrary distributions. However, unlike traditional CNFs, which require solving ordinary differential equations (ODEs) during training [6], Flow Matching directly learns a time-dependent velocity field through supervised learning, rendering it more computationally efficient. Moreover, while diffusion models rely on a stochastic process that gradually perturbs the data to a Gaussian distribution before learning a denoising process, Flow Matching avoids the need for explicit noise injection and learns a *direct mapping between general distributions*.

In this work, we introduce *FlowPure*, a novel adversarial purification method that leverages Flow Matching to directly transform adversarial examples into their clean counterparts (see Fig. 1). To the best of our knowledge, FlowPure is the first method to apply Continuous Normalizing Flows to adversarial purification. In contrast to typical purification-based techniques that are exclusively designed to be attack-agnostic, FlowPure can *incorporate knowledge of the attack by sampling adversarial examples during training*. Although this may limit generalizability, our results indicate that it significantly improves performance under known threat models, where robustness is often most critical. To address scenarios involving unknown or adaptive attacks, we introduce *a variant of FlowPure trained using Gaussian noise* instead of specific adversarial perturbations. This attack-agnostic variant outperforms existing state-of-the-art diffusion-based purification methods, demonstrating that FlowPure can remain effective even without explicit access to the attack distribution.

We consider two distinct adversarial threat models, with a primary focus on a preprocessor-blind setting. In this scenario, the attacker has full white-box access to the classifier, but is unaware of the existence of the purification mechanism, serving effectively as an upper bound on attack capabilities on black-box preprocessor-blind settings. Given the increasing prevalence of open-weight models in real-world deployments, we deem this threat model to be especially important, as it closely aligns with how attacks would be typically executed in practice. In such cases, interaction is often minimal and limited to model probing, or completely absent, with attackers focusing on the classifier while remaining unaware of upstream defense mechanisms [11]. Our empirical analysis shows that FlowPure achieves remarkable robustness under these conditions, with no impact on benign performance in contrast to state-of-the-art purifiers. FlowPure, in its deterministic variants, significantly outperforms existing state-of-the-art purifiers under this threat model. Moreover, we show that FlowPure, although primarily trained for purification, holds strong potential as an adversarial detector. Our preliminary investigation reveals that it achieves near-perfect detection accuracy against oblivious PGD attacks across multiple perturbation levels on CIFAR-10 and CIFAR-100.

To better understand the limitations of adversarial purification, we also evaluate FlowPure in a fully adaptive white-box setting, where the attacker is assumed to have full gradient access to both the classifier and the defense mechanism. This represents a worst-case scenario, and in practice, it renders most defenses ineffective [33]. Recent work, such as DiffHammer [35], has demonstrated that the robustness of diffusion-based purification methods has been significantly overestimated. In line with these findings, we observe that all purification-based methods, including FlowPure, struggle under this setting. Nevertheless, our Gaussian-trained variant shows improved performance compared to existing approaches, outperforming state-of-the-art purifiers, even when all methods experience a sharp decrease in robustness. Although white-box evaluations remain paramount for estimating worst-case robustness [4], our findings reinforce the need for strong defenses against black-box threats, which are more practical and aligned with real-world adversarial threats.

We summarize our contributions as follows:

- We introduce *FlowPure*, a novel adversarial purification method that leverages Continuous Normalizing Flows to purify adversarial examples. FlowPure can directly map between arbitrary distributions, enabling the use of task-specific perturbations during training.
- We propose two variants of FlowPure: 1) a *deterministic* variant that can rely on existing knowledge of attacks for effective purification of known threats, and 2) a *stochastic* variant that relies on Gaussian noise offering broad-spectrum robustness.
- We evaluate FlowPure across several attack strategies, parametrizations and types of noise. We demonstrate that FlowPure trained on adversarial noise improves state-of-the-art in preprocessor-blind settings with no degradation in standard accuracy. We also evaluate FlowPure under worst-case, fully adaptive white-box attacks, and show that our Gaussian-trained variant outperforms existing purification methods.
- Finally, we demonstrate that FlowPure can serve as a highly effective adversarial detector, achieving near-perfect detection accuracy against oblivious PGD attacks, even at low perturbation levels.

2 Preliminary & Related Work

This section provides background on diffusion models and prior state-of-the-art diffusion purification techniques. We then introduce Continuous Normalizing Flows, which form the foundation of our proposed method. Lastly, we briefly review relevant work on adversarial detection.

2.1 Diffusion models

Diffusion models [13] learn a transformation from a Gaussian to a complex data distribution. During training, they simulate a forward process that incrementally corrupts real data by injecting noise until only pure Gaussian noise remains. This forward process is modeled as a Markov Chain $\{x_i\}$, with transitions defined by $q(x_{i+1}|x_i)$. The original and most common forward process is the *Variance Preserving* forward process: $q(x_{i+1}|x_i) = \mathcal{N}(x_i; \sqrt{1-\beta_i}x_{i-1}, \beta_i I)$. Here, $\{\beta_i\}$ represents a predefined noise schedule [13]. The diffusion model is trained to estimate the reverse transitions $q(x_i|x_{i+1})$, enabling the reversal of the forward process. This enables the generation of complex data samples by incrementally transforming Gaussian noise through the learned reverse process.

2.2 Diffusion Purification

Adversarial purification aims to remove perturbations introduced by an attacker, often by leveraging generative models to project adversarial examples onto the learned clean data distribution. Recently, diffusion models have emerged as a powerful alternative for adversarial purification. We categorize purification algorithms leveraging diffusion into several types, relative to their working principles:

Denoising purification. DiffPure [25] and ADP [36] inject noise into adversarial samples and then use a diffusion model to remove it. The rationale behind these approaches is that, as more noise is introduced to samples, the distribution of diffused adversarial examples increasingly resembles that of diffused clean data and, as such, denoising is likely to recover a clean data sample. However, noise injection often disrupts semantically meaningful information, resulting in a drop in standard accuracy.

Guided purification. With the objective of addressing the limitations of standard denoising, GDMP [34] incorporates guidance on adversarial examples, ensuring that purified samples remain close to the originals, while still mitigating adversarial perturbations.

Likelihood maximization. Likelihood maximization techniques, such as ScoreOpt [38] and LM [5], leverage the fact that diffusion models approximate the score of the data distribution. These methods purify adversarial examples by maximizing their likelihood under the learned data distribution.

Adversarial fine-tuning. Identifying sub-optimal design choices in the design of traditional denoising purification methods, Li et al. recently introduced ADBM [19]. ADBM aims to enhance the white-box robustness of diffusion-based purification by incorporating adversarial noise during the training of the diffusion model. They achieve this by fine-tuning a standard diffusion model with adversarial noise and then applying DiffPure [25] with their fine-tuned model during inference. It should be noted that, while both ADBM and FlowPure can incorporate adversarial noise during training, the former remains a diffusion model relying on a Gaussian forward process, while FlowPure can directly map from arbitrary adversarial distributions.

Despite the apparent effectiveness of diffusion-based purification methods, recent studies have demonstrated that their robustness is frequently overestimated [18, 14, 35, 22]. Gao et al. [9] argue that stochastic defenses do not inherently confer robustness; rather, they induce shifts in the input distribution that lead to prediction errors, which may be misinterpreted as improved robustness. Furthermore, these defenses exhibit an inherent trade-off between adversarial robustness and the model's invariance to stochastic transformations. As models are optimized to preserve performance under random perturbations, the effectiveness of the defense mechanism correspondingly decreases.

Regarding white-box scenarios, Lee and Kim [18] propose using the full gradients of a surrogate process to approximate the gradients of the complete purification process. This surrogate process removes the same amount of noise in fewer steps, mitigating vanishing or exploding gradients. Additionally, they apply Expectation Over Transformation (EOT) to account for stochasticity in purification. However, Wang et al. [35] argue that EOT-based attacks face a gradient dilemma when applied to stochastic purification methods. The EOT approach assumes that most purification runs of an input share a common vulnerability (they are susceptible to a similar adversarial perturbation). However, diffusion-based defenses introduce high stochasticity, breaking this assumption. Some purification runs may have unshared vulnerabilities, i.e., they respond to different adversarial perturbations. This inconsistency leads to the gradient dilemma, where averaging conflicting gradients across purification runs reduces the attack's effectiveness. To address this, the authors introduce DiffHammer, a selective attack that prioritizes gradients of shared vulnerabilities, outperforming previous benchmarks such as AutoAttack [7], BPDA [2] and EOT+PGD [18].

Research gap. To summarize, while diffusion-based purification methods have shown promise, their robustness is often overestimated due to limitations in evaluation protocols and the inherent trade-offs introduced by stochasticity. Existing approaches typically rely on fixed noise schedules, which can degrade clean accuracy and limit flexibility. In this paper, we aim to improve adversarial robustness through FlowPure, a novel purification framework based on Continuous Normalizing Flows (CNFs). Moreover, the use of CNFs enables training with arbitrary perturbation distributions, allowing the purification process to be tailored to specific threat models. To the best of our knowledge, this is the first approach to apply CNFs to adversarial purification.

2.3 Flow Matching

Continuous Normalizing Flows (CNF) [6] are generative models designed to map samples from a source distribution $q(x_0)$ to a target distribution $q(x_1)$ through a transformation governed by a neural ordinary differential equation (ODE): $dx = u_t(x)dt$. To generate samples from $q(x_1)$, one first samples $x_0 \sim q(x_0)$ and then solves the ODE from time $t = 0$ to $t = 1$ starting at x_0. Unlike diffusion models, which rely on Gaussian priors, CNFs can use any source distribution and are not limited to Gaussian sources. The model approximates the velocity field $u_t(x)$ with a neural network $v_\theta(t, x)$. Traditionally, CNFs were trained using maximum likelihood, which involves solving the ODE during training. This approach is computationally costly and does not scale efficiently [6].

Flow Matching (FM) [20, 32, 21], introduced as a more scalable and tractable alternative that circumvents the need for ODE simulation during training, optimizes the model via the objective:

$$\mathcal{L}_{FM} = \mathbb{E}_{t\sim\mathcal{U}[0,1],x\sim p_t(x)}[||v_\theta(t, x) - u_t(x)||_2^2], \tag{1}$$

where $p_t(x)$ represents the density of samples transported along $u_t(x)$ from $q(x_0)$ at time 0 to time t. However, FM requires the marginal vector field $u_t(x)$ to be tractable, which is not the case for general source and target distributions [32].

To address this limitation, Conditional Flow Matching (CFM) [20, 21, 32] was introduced, which conditions the probability flow on a latent variable with density $q(z)$. This defines conditional probability paths $p_t(x|z)$ and a corresponding conditional vector field $u_t(x|z)$. The CFM training objective is then formulated as follows, allowing CNFs to learn flexible transformations between arbitrary distributions:

$$\mathcal{L}_{CFM} = \mathbb{E}_{t\sim\mathcal{U}[0,1],z\sim q(z),x\sim p_t(x|z)}[||v_\theta(t,x) - u_t(x|z)||_2^2]. \tag{2}$$

Since infinitely many conditional probability paths exist between any two distributions, CFM requires specific design choices [32]. A natural choice is to use optimal transport (OT) paths, also known as rectified flows [21]. These paths condition on samples from both distributions, expressed as $p(z) = p(x_1)p(x_2)$, and define the probability paths as low-variance Gaussians moving linearly between the two conditioning samples. Formally, this is given by:

$$p_t(x|x_0, x_1) = \mathcal{N}(tx_1 + (1-t)x_0, \sigma), \tag{3}$$

$$u_t(x|x_0, x_1) = x_1 - x_0. \tag{4}$$

For rectified flows, the Gaussians are replaced with Dirac deltas since the variance is chosen to be very small. Prior work suggests that OT paths are computationally more efficient than diffusion-based paths, offering faster training and improved generalization [20, 32].

2.4 Detection

While adversarial purification methods aim to recover clean inputs from perturbed ones, an orthogonal line of defense focuses on detecting adversarial examples before they are processed by the classifier. The goal of adversarial detection is to identify inputs that have been maliciously crafted, enabling a system to either reject or flag them for further inspection. Most detection methods leverage intrinsic properties of the inputs that distinguish adversarial examples from clean data [23, 17, 31, 1, 12].

Among these, BEYOND [12] introduces a self-supervised approach that compares an input to its augmented neighbors based on two criteria: *label consistency* and *representation similarity*. Clean samples tend to maintain consistent predictions and exhibit high feature similarity with their neighbors, whereas adversarial examples exhibit bigger deviations.

3 Methodology: FlowPure

In this section, we present FlowPure, our CNF-based approach for adversarial purification. We describe the core design choices, the sampling of adversarial distributions, a stochastic variant for enhanced white-box robustness, and adversarial detection based on the learned velocity field.

Design choices. Adversarial purification aims to map adversarial samples back into the clean data distribution. Since CNFs are capable of mapping between arbitrary distributions, FlowPure achieves this by training a CNF to convert adversarial examples into clean samples. Figure 1 illustrates this process. Given the infinite possible probability paths between distributions, we outline our key design choices: the latent distribution $p(z)$, conditional probability path $p_t(x|z)$ and conditional vector field $u_t(x|z)$. We follow the design principles of *Rectified Flows* by Liu et al. [21], as described in Section 2.3, applied to an adversarial and clean sample pair as latent variable. Specifically these are:

$$p(z) = p(x_{clean})p(x_{adv}|x_{clean}), \tag{5}$$

$$p_t(x|x_{clean}, x_{adv}) = \delta(x - (t-1)x_{adv} - tx_{clean}), \tag{6}$$

$$u_t(x|x_{adv}, x_{clean}) = x_{adv} - x_{clean}. \tag{7}$$

Here, $p(x_{clean})$ denotes the clean data distribution and $p(x_{adv}|x_{clean})$ denotes the adversarial distribution conditioned on a given clean data sample.

Adversarial Distributions. To sample from the adversarial distribution $p(x_{adv}|x_{clean})$, we generate adversarial examples from clean samples using attack methods such as PGD [24] and CW [3]. Since the attacker's exact parameters are unknown, we randomly vary attack parameters during training, ensuring robustness across different attack settings. Specifically, we model $p(x_{adv}|x_{clean}) = p(\xi)p(x_{adv}|x_{clean}, \xi)$, where ξ represents the sampled attack parameters.

An especially important attack parameter to vary during training is the perturbation budget ϵ. Introducing very small perturbations ensures that the model learns a velocity field that remains accurate even for samples near the data manifold. This behavior helps preserve the integrity of clean inputs during inference, thereby maintaining high standard accuracy.

Gaussian FlowPure. While the discussed FlowPure formulation learns to map preprocessor-blind adversarial examples to their clean counterparts, we also propose a variant, *Gaussian FlowPure*, specifically designed to enhance robustness in the white-box setting. In this variant, we train the CNF to transport samples from a Gaussian-corrupted data distribution to the clean data distribution. That is, instead of sampling from the adversarial distribution, we sample $x_{\text{noisy}} \sim \mathcal{N}(x_{\text{clean}}, \sigma^2_{\max})$, where $\sigma_{\max}$ is a fixed, predefined standard deviation. The CNF is trained to model OT paths from x_{noisy} to x_{clean}, enabling efficient purification via ODE integration starting from Gaussian-corrupted inputs.

It is important to note that the Gaussian corruption applied during inference can have any standard deviation $\sigma \leq \sigma_{\max}$. To correctly purify such an input, the ODE must be integrated starting at time $t = 1 - \frac{\sigma}{\sigma_{\max}}$, since the OT path at that point in time corresponds to data corrupted with Gaussian noise of standard deviation σ.

This approach resembles DiffPure in its reliance on denoising Gaussian-corrupted inputs. However, the main distinction lies in the nature of the purification paths: DiffPure follows curved diffusion trajectories, whereas Gaussian FlowPure follows OT trajectories. OT-based probability paths yield straighter flows, allowing more accurate ODE integration in fewer steps [20, 32]. We hypothesize that this increased stability results in greater robustness to adversarial perturbations, while also preserving benign accuracy more effectively. Our empirical results support this conjecture.

Detection. We further explore the use of FlowPure as an adversarial detector. Specifically, we leverage the magnitude of the learned velocity field $||v_\theta(t, \mathbf{x})||_2^2$ at time $t = 0$, as a detection score. The underlying intuition is that adversarial examples, residing outside of the data manifold, exhibit a higher initial velocity compared to clean samples. This enables us to flag samples as adversarial if their initial velocity is above a threshold.

4 Evaluation

This section reports on a comprehensive evaluation of FlowPure [2] under various threat models. We assess purification performance in both realistic (preprocessor-blind) and worst-case (fully adaptive white-box) settings. We also report preliminary results of adversarial detection with FlowPure.

4.1 Experimental setup

Baselines. We evaluate our approach on two benchmark datasets: CIFAR-10 and CIFAR-100 [16]. The victim classifier (referred to as *Victim* in our experiments) for both datasets and for each purifier is the widely used WideResNet-28 [37]. For comparison, we benchmark against multiple diffusion-based purification methods: DiffPure [25], GDMP [34], LM [5] and ADBM [19]. These baselines cover the major design paradigms explored in the literature and allow us to assess the relative performance of our method across a diverse set of approaches. All purification models, including our own, have the DDPM++ [29] architecture. To evaluate preprocessor-blind scenarios, we consider Projected Gradient Descent (PGD) [24] and Carlini & Wagner (CW) [3] attacks, which operate under the L_∞ and L_2 norms, respectively. We also assess white-box robustness using the state-of-the-art DiffHammer [35] attack. It must be noted that DiffHammer is the same as a standard PGD attack using gradients of the surrogate process, when FlowPure is deterministic. For detection, we report preliminary results in which we compare against BEYOND [12]. We consider BEYOND the most relevant baseline detector for our preliminary detection experiments, as it consistently outperforms prior methods. For both attacks and defenses, we use standard configurations as reported in the original works, unless specified otherwise (the details are outlined in Appendix A).

Metrics. We use two metrics to evaluate all purification approaches: *standard accuracy* and *robust accuracy*. Standard accuracy measures the accuracy of the defended model on clean unperturbed data. Robust accuracy measures the accuracy of the defended model on adversarial examples generated by adversarial attacks. For robust accuracy, we employ the framework of resubmission attacks described

[2] Our code is available at https://github.com/DistriNet/FlowPure (License: MIT)

in DiffHammer [35] to study stochastic defenses. As such, we report *average robust accuracy (avg)*, defined as the mean accuracy after N resubmissions, and *worst-case robust accuracy (worst)*, in which an attack is considered successful if the same adversarial example is misclassified once in N resubmissions. For all experiments, we consider $N = 10$ as the number of resubmissions. Note that for deterministic defenses this has no impact. Moreover, and consistent with previous work, we evaluate the complete test set of each dataset for preprocessor-blind attacks, and a subset of 512 images sampled at random for white-box attacks. Due to the small subset, we report the average and standard deviation of 3 runs in the white-box setting. To evaluate detection, we use the area under the curve (AUC) of the receiver operating characteristic (ROC). The ROC illustrates how the detection threshold affects the true positive rate (TPR) and false positive rate (FPR), while the AUC provides a single-value summary of the ROC. Additionally we report the TPR at FPRs of 5% , 3% and 1%.

FlowPure. We train three CNF models for each dataset. The first model, which we denote $FlowPure^{PGD}$, is trained on PGD samples. The second model, $FlowPure^{CW}$, is trained on CW samples. The third, $FlowPure^{Gauss}$, corresponds to the Gaussian FlowPure variant, trained on inputs corrupted with Gaussian noise of fixed standard deviation $\sigma = 0.3$. During inference, Gaussian FlowPure is annotated with the level of injected noise as a subscript (e.g., $FlowPure^{Gauss}_{\sigma=0.15}$). The results for several values of σ can be found in Appendix B.1. In all cases, ODE integration is performed using Euler's method. Further implementation details, including the distributions used for attack parameters, are provided in Appendix A.3. This appendix also outlines the technical considerations involved when the noise level used during inference differs from the training noise distribution in Gaussian FlowPure.

4.2 Preprocessor-blind evaluation

We first evaluate our method in the preprocessor-blind setting, which we deem a more realistic threat to ML systems in practice. Table 1 displays the results for this scenario in CIFAR-10 and CIFAR-100. We summarize our findings as follows:

Benign performance. FlowPure (in its deterministic variants $FlowPure^{PGD}$ and $FlowPure^{CW}$), presents no degradation in the accuracy of clean samples, as opposed to baselines. Furthermore, there is even a slight improvement in the metric. We theorize that this could be due to FlowPure learning a more robust mapping of clean data, as a form of implicit regularization, or through mitigation of non-adversarial noise already present on unperturbed samples. This interpretation is supported by visual exploration of purified samples (Appendix B.3), where FlowPure reconstructions appear cleaner and more faithful to the original inputs, often removing spurious noise while preserving semantic content.

Robustness. FlowPure consistently exhibits superior robustness in all *attack-aware scenarios* (when the evaluated attack matches the one used in training) but one (our Gaussian variant achieves a higher average robust accuracy in CIFAR-100 for PGD), suggesting that the method can effectively learn to reverse adversarial perturbations in a targeted manner.

Transferability. In *transfer scenarios* (the attack does not match the one used during training), $FlowPure^{PGD}$ demonstrates strong transferability to CW attacks. $FlowPure^{CW}$, however, struggles to generalize to PGD, displaying a notable drop in robustness. This asymmetry likely arises from the structural differences between PGD and CW perturbations, as PGD perturbations tend to be larger in magnitude and more distributed, yielding a more generalizable mapping. In contrast, CW perturbations, while smaller in magnitude, are highly specialized, leading to increased complexity for $FlowPure^{CW}$ in effectively counteracting broader perturbations introduced by PGD attacks.

Detection. We additionally explore the use of FlowPure for detection of adversarial examples in preprocessor-blind scenarios, with the results of our preliminary experiments detailed in Appendix B.2. Compared to BEYOND [12], a state-of-the-art SSL-based detector, FlowPure achieves near-perfect detection accuracy on PGD attacks across CIFAR-10 and CIFAR-100, consistently outperforming BEYOND across a range of perturbation budgets. Notably, FlowPure's detection performance improves with increasing perturbation levels, making it particularly effective against high-magnitude PGD attacks. However, its effectiveness diminishes on CW attacks, where the perturbations are more subtle and harder to distinguish.

Table 1: Purification on PGD and CW attacks in the preprocessor-blind setting, and under DH [35] in the fully-adaptive white-box setting. Metrics report average accuracy (avg) and worst-case accuracy ($worst$) after 10 resubmissions (for $FlowPure^{PGD}$, $FlowPure^{CW}$ and *Victim* the two metrics coincide, as they are deterministic). The top-performing method is shown in bold, the second-best is underlined. DH does not apply to the victim in isolation.

	Method	Standard Accuracy		Robust Accuracy					
				Preprocessor-blind				White-box	
		avg	$worst$	PGD_{avg}	$\text{PGD}_{\text{worst}}$	CW_{avg}	CW_{worst}	DH_{avg}	DH_{worst}
CIFAR-10	*Victim*	95.81	95.81	0.04	0.04	0.00	0.00	–	–
	Baselines:								
	DiffPure [25]	88.70	70.55	85.69	63.82	86.31	64.94	$\underline{38.88 \pm 0.74}$	15.56 ± 1.77
	GDMP [34]	91.69	78.16	$\underline{89.23}$	72.51	89.89	74.05	32.65 ± 1.17	16.79 ± 1.7
	LM [5]	82.94	65.03	67.61	44.86	78.70	56.20	12.55 ± 0.39	4.82 ± 0.23
	ADBM [19]	91.93	78.77	89.08	72.59	90.25	74.86	32.35 ± 1.42	13.15 ± 0.62
	Ours:								
	$\text{FlowPure}^{\text{PGD}}$	**96.00**	**96.00**	**92.23**	**92.23**	$\underline{91.45}$	$\underline{91.45}$	1.17 ± 0.2	1.17 ± 0.2
	$\text{FlowPure}^{\text{CW}}$	$\underline{95.96}$	$\underline{95.96}$	87.33	$\underline{87.33}$	**94.36**	**94.36**	1.11 ± 0.49	1.11 ± 0.49
	$\text{FlowPure}^{\text{Gauss}}_{\sigma=0.15}$	91.61	78.18	88.21	70.51	89.66	73.31	36.39 ± 1.16	$\underline{17.57 \pm 1.36}$
	$\text{FlowPure}^{\text{Gauss}}_{\sigma=0.2}$	89.25	72.01	86.06	65.06	86.92	66.66	$\mathbf{42.27 \pm 1.09}$	$\mathbf{21.28 \pm 1.08}$
CIFAR-100	*Victim*	80.73	80.73	0.01	0.01	0.00	0.00	–	–
	Baselines:								
	DiffPure [25]	52.90	24.86	46.85	19.05	47.69	20.11	8.86 ± 0.77	2.02 ± 0.68
	GDMP [34]	56.89	31.63	51.19	25.33	52.38	26.53	6.06 ± 0.22	2.01 ± 0.69
	LM [5]	45.56	23.23	26.89	10.98	39.98	17.99	3.35 ± 0.98	0.46 ± 0.30
	ADBM [19]	64.76	39.07	57.43	29.99	60.24	32.72	7.13 ± 0.45	1.49 ± 0.22
	Ours:								
	$\text{FlowPure}^{\text{PGD}}$	$\underline{80.75}$	$\underline{80.75}$	52.90	**52.90**	$\underline{66.67}$	$\underline{66.67}$	0.38 ± 0.20	0.38 ± 0.20
	$\text{FlowPure}^{\text{CW}}$	**80.84**	**80.84**	28.67	28.67	**75.51**	**75.51**	0.26 ± 0.11	0.26 ± 0.11
	$\text{FlowPure}^{\text{Gauss}}_{\sigma=0.15}$	68.23	44.07	**61.17**	$\underline{35.15}$	64.06	38.03	$\underline{11.31 \pm 1.19}$	$\underline{3.84 \pm 0.78}$
	$\text{FlowPure}^{\text{Gauss}}_{\sigma=0.2}$	63.72	37.89	$\underline{57.61}$	30.61	59.17	32.32	$\mathbf{14.38 \pm 1.62}$	$\mathbf{4.55 \pm 1.00}$

4.3 Fully adaptive white-box evaluation

We further evaluate our approach in a fully adaptive white-box setting where the attacker is assumed to have gradient access to both the classifier and the purification mechanism, using DiffHammer [35]. The results of this evaluation, also detailed in Table 1, reveal a substantial degradation in robust accuracy compared to the preprocessor-blind scenario, given all methods considered suffer significant decreases in robust accuracy. This trend is most apparent in the case of $FlowPure^{PGD}$ and $FlowPure^{CW}$, as the deterministic mapping through CNFs may be more susceptible to gradient-based exploitation than stochastic methods, which inherently introduce noise that can disrupt precise adversarial gradients. Notably, our Gaussian-trained variant consistently achieves the best performance across both datasets, with the gap being particularly significant on CIFAR-100.

Although $FlowPure^{Gauss}$ exhibits the strongest performance under fully adaptive white-box attacks, it is important to note that none of the evaluated methods demonstrate practical viability in this setting. With robust accuracy metrics falling below 45% in CIFAR-10 and below 15% on CIFAR-100, across all approaches, no defense proves effective against adversaries with full gradient access, reinforcing the broader challenge of securing models under such extreme threat conditions.

Ablation study. To better understand the performance of Gaussian FlowPure, we conduct a comprehensive analysis evaluating its performance against DH [35] across varying levels of noise injection. Moreover, we compare it to DiffPure, the most comparable diffusion-based purification method. This setup allows us to isolate the impact of using CNFs for purification. Our results, presented in Figure 2, show that Gaussian FlowPure consistently achieves higher robust accuracy than DiffPure across a range of noise levels, while incurring similar or lower degradation in benign accuracy. The complete results are available in Appendix B.1, including our experiments in the preprocessor-blind scenario.

5 Discussion and Limitations

Our experimental results in the preprocessor-blind setting demonstrated the effectiveness of FlowPure against strong L_∞ (PGD) and L_2 (CW) attacks. Moreover, the deterministic variants of FlowPure

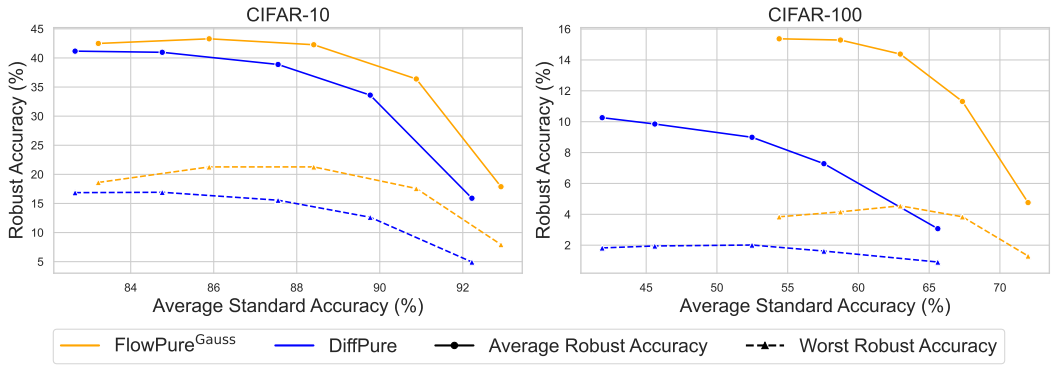


Figure 2: Standard/Robust accuracy trade-off under DH [35] (10 resubmissions), of $FlowPure^{Gauss}$ and DiffPure [25], for different levels of noise injection (noise ranges are $\sigma \in [0.15, 0.35]$ for the former, and $t^* \in [50, 180]$ for the latter). Higher values on both axes are better, with increases in standard accuracy correlated with lower noise levels. Note that the y-axes differ between the plots.

showed no degradation in benign performance, while providing significant robustness against PGD and CW attacks, highlighting their potential as a practical defense mechanism in scenarios where attackers are unaware of the purification process. Although incorporating attack-specific knowledge during training leads to strong empirical performance, this reliance can limit applicability when threat models are unknown or evolve over time, as our evaluations in the fully adaptive white-box setting demonstrated. Despite this, our experiments in transfer scenarios indicated encouraging generalization properties, suggesting that these methods can provide robustness beyond their specific training distribution.

Our evaluation under fully adaptive white-box attacks using DiffHammer [35] shows that robust accuracy drops sharply for all purification-based defenses, reaffirming the challenge of defending against adversaries with full gradient access. This degradation is most pronounced in the case of our deterministic variants, which are more easily exploited with gradient-based attacks. In contrast, incorporating stochasticity during training and inference can indeed improve robustness in this setting, albeit at the cost of reduced benign accuracy, highlighting the inherent trade-off in stochastic preprocessing defenses [9]. Among all the purification-based methods evaluated, Gaussian FlowPure achieves the strongest performance in this scenario, with a particularly significant gap on CIFAR-100.

While white-box evaluations are crucial for understanding the theoretical limits of robustness, their practical utility as the primary benchmark for adversarial defense warrants reconsideration. Although FlowPure outperforms all adversarial purification baselines under fully adaptive white-box attacks, all studied defenses experience a consistent collapse under such conditions. The substantial degradation in robustness in this setting indicates that none of the current methods offer meaningful protection against adversaries with full gradient access. With this work, we underline the importance of redirecting efforts toward more realistic threat models, such as preprocessor-blind settings, that better reflect constraints faced by real-world adversaries.

6 Conclusion

In this work, we introduced FlowPure, a novel adversarial purification method leveraging Continuous Normalizing Flows trained with Conditional Flow Matching. FlowPure improves on state-of-the-art in multiple scenarios: for preprocessor-blind attacks, our approach directly learns the mapping between adversarial and clean data distributions, effectively removing adversarial perturbations. This key difference allows FlowPure to fully preserve benign accuracy on CIFAR-10 and CIFAR-100 datasets during preprocessor-blind attacks. For adaptive white-box attacks we found that our Gaussian variant of FlowPure outperforms state-of-the-art during purification. Overall, FlowPure offers a realistic and deployable purification approach where benign performance is kept high while providing strong robustness. Finally, we showcased the dual capability of FlowPure as a highly effective adversarial detector, as it achieves near-perfect detection accuracy against oblivious PGD attacks.

Acknowledgments

The authors would like to thank Ilia Shumailov for providing valuable feedback on the manuscript. This research is partially funded by the Research Fund KU Leuven, Internal Funds KU Leuven, and the Cybersecurity Research Program Flanders.

References

[1] A. Abusnaina, Y. Wu, S. Arora, Y. Wang, F. Wang, H. Yang, and D. Mohaisen. Adversarial example detection using latent neighborhood graph. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7687–7696, 2021.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018. URL https://arxiv.org/abs/1802.00420.

[3] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, Los Alamitos, CA, USA, May 2017. IEEE Computer Society. doi: 10.1109/SP.2017.49. URL https://doi.ieeecomputersociety.org/10.1109/SP.2017.49.

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

[5] H. Chen, Y. Dong, Z. Wang, X. Yang, C. Duan, H. Su, and J. Zhu. Robust classification via a single diffusion model, 2024. URL https://openreview.net/forum?id=I5lcjmFmlc.

[6] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

[7] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. RobustBench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://openreview.net/forum?id=SSKZPJCt7B.

[8] V. G. T. da Costa, E. Fini, M. Nabi, N. Sebe, and E. Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022. URL http://jmlr.org/papers/v23/21-1155.html.

[9] Y. Gao, I. Shumailov, K. Fawaz, and N. Papernot. On the limitations of stochastic pre-processing defenses. *Advances in Neural Information Processing Systems*, 35:24280–24294, 2022.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

[11] K. Grosse, L. Bieringer, T. R. Besold, and A. M. Alahi. Towards more practical threat models in artificial intelligence security. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4891–4908, 2024.

[12] Z. He, Y. Yang, P.-Y. Chen, Q. Xu, and T.-Y. Ho. Be your own neighborhood: Detecting adversarial examples by the neighborhood relations built on self-supervised learning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=S4LqI6CcJ3.

[13] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[14] M. Kang, D. Song, and B. Li. DiffAttack: Evasion attacks against diffusion-based adversarial purification. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] H. Kim. Torchattacks: A PyTorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

[16] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, CIFAR, 2009.

[17] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[18] M. J. Lee and D. Kim. Robust evaluation of diffusion-based adversarial purification. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 134–144, 2023. URL https://api.semanticscholar.org/CorpusID:257557136.

[19] X. Li, W. Sun, H. Chen, Q. Li, Y. He, J. Shi, and X. Hu. Adversarial diffusion bridge model for reliable adversarial purification. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=g0rnZeBguq.

[20] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[21] X. Liu, C. Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

[22] K. Lucas, M. Jagielski, F. Tramèr, L. Bauer, and N. Carlini. Randomness in ML defenses helps persistent attackers and hinders evaluators. *arXiv preprint arXiv:2302.13464*, 2023.

[23] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[25] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar. Diffusion models for adversarial purification. In *International Conference on Machine Learning (ICML)*, 2022.

[26] J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. URL http://arxiv.org/abs/1707.04131.

[27] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkJ3ibb0-.

[28] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. PixelDefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJUYGxbCW.

[29] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

[31] J. Tian, J. Zhou, Y. Li, and J. Duan. Detecting adversarial examples from sensitivity inconsistency of spatial-transform domain. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9877–9885, 2021.

[32] A. Tong, K. FATRAS, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

[33] F. Tramer, N. Carlini, W. Brendel, and A. Madry. On adaptive attacks to adversarial example defenses. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1633–1645. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/11f38f8ecd71867b42433548d1078e38-Paper.pdf.

[34] J. Wang, Z. Lyu, D. Lin, B. Dai, and H. Fu. Guided diffusion model for adversarial purification. *arXiv preprint arXiv:2205.14969*, 2022.

[35] K. Wang, X. Fu, Y. Han, and Y. Xiang. DiffHammer: Rethinking the robustness of diffusion-based adversarial purification. *Advances in Neural Information Processing Systems*, 37, 2024.

[36] J. Yoon, S. J. Hwang, and J. Lee. Adversarial purification with score-based generative models. In *Proceedings of The 38th International Conference on Machine Learning (ICML 2021)*, 2021.

[37] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

[38] B. Zhang, W. Luo, and Z. Zhang. Enhancing adversarial robustness via score-based optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=M0AHXRzHhm.

A Additional experimental settings

A.1 Baseline defense configurations

The baseline defense configurations used in our experiments largely follow those proposed in the original papers. We used implementations for each baseline provided by the authors of DiffHammer [35][3].

DiffPure [25] DiffPure defends against adversarial attacks by injecting Gaussian noise into input samples via the forward diffusion process until a predefined t^*:

$$x_{t^*} = \sqrt{\bar{\alpha}_{t^*}} x_0 + \sqrt{1 - \bar{\alpha}_{t^*}} \epsilon, \tag{8}$$

where α_t follows a predefined schedule and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Here, $\epsilon \sim \mathcal{N}(0, 1)$ is standard Gaussian noise.

The DDPM reverse process progressively removes noise:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left[x_t - \frac{1 - \alpha_t}{1 - \bar{\alpha}_t} \epsilon_\theta(x_t, t) \right] + \sigma_t z, \tag{9}$$

where $\epsilon_\theta(x_t, t)$ is a parameterized denoising network, $z \sim \mathcal{N}(0, 1)$, and

$$\sigma_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \alpha_t} (1 - \bar{\alpha}_t).$$

We use $t^* = 0.1$. Following [18], we adopt a surrogate process with differing time intervals for adaptive attacks. The time intervals are set to 0.01 for attackers, while defenders use intervals set to 0.005.

GDMP [34] GDMP extends DiffPure by incorporating a similarity-guided term $D(x_t, x_{t'})$ into the reverse process:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left[x_t - \frac{1 - \alpha_t}{1 - \bar{\alpha}_t} \left(\epsilon_\theta(x_t, t) + s \sigma_t \nabla_{x_t} D(x_t, x_{t'}) \right) \right] + \sigma_t z, \tag{10}$$

where s is a guidance scale and

$$x_{t'} = \sqrt{\bar{\alpha}_{t'}} x_0 + \sqrt{1 - \bar{\alpha}_{t'}} \epsilon.$$

We use $t^* = 0.036$, with 4 successive purification rounds. The time intervals are set to 0.06 for attackers, while defenders use intervals set to 0.003.

LM [5] LM is the purification step in the Robust Diffusion Classifier (RDC), optimizing a lower bound on log-likelihood:

$$\min_{\hat{x}} \mathbb{E}_{\epsilon,t} \left\| \epsilon - \epsilon_\theta(\hat{x}_t, t) \right\|_2^2, \quad \text{s.t. } \|\hat{x} - x_0\|_\infty \leq \eta, \tag{11}$$

where

$$\hat{x}_t = \sqrt{\bar{\alpha}_{t'}} \hat{x} + \sqrt{1 - \bar{\alpha}_{t'}} \epsilon,$$

and η is a predefined threshold.

This optimization is performed via projected gradient descent with a step size of 0.1 over 5 iterations. The timestep t is sampled uniformly from $[0.4, 0.6]$. Notably, RDC comprises both the LM and a diffusion classifier (DC). This paper is concerned with adversarial purification, so we only compare against the purification step.

[3] https://github.com/Ka1b0/DiffHammer (License: MIT)

ADBM [19] ADBM improves upon DiffPure by learning a dedicated reverse bridge from the diffused adversarial distribution back to the clean distribution. Given x_0, an adversarial example is constructed as $x_0^a = x_0 + \epsilon_a$, which is then diffused to obtain x_t^a. The model learns to remove both Gaussian and adversarial noise through a modified forward process:

$$x_t^d = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon + \frac{\bar{\alpha}_T(1 - \bar{\alpha}_t)}{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_T)}\epsilon_a,$$

and is trained using the following objective:

$$\mathcal{L}_b = \mathbb{E}_{\epsilon,t}\left\| \frac{\bar{\alpha}_T\sqrt{1 - \bar{\alpha}_t}}{(1 - \bar{\alpha}_T)\sqrt{\bar{\alpha}_t}}\epsilon_a + \epsilon - \epsilon_\theta(x_t^d, t)\right\|^2. \tag{12}$$

The adversarial noise ϵ_a is generated by running PGD on the purification pipeline with a frozen classifier to maximize the classification loss after purification. To reduce training cost, the timestep t and noise ϵ are fixed within each optimization step. This loss and adversarial noise generation is then used to fine-tune a pretrained Diffusion Model.

We fine-tune a pre-trained diffusion model using the configuration and implementation from the original paper: 30K training iterations, a maximum timestep of 0.2, and PGD with 3 steps and a step size of $8/255$ to compute ϵ_a. During inference, ADBM is equivalent to DiffPure but uses the learned model, with $t^* = 0.1$ and the DDIM reverse sampler given by:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1 - \bar{\alpha}_{t-1}} \cdot \epsilon_\theta(x_t, t). \tag{13}$$

Both attacker and defender use a fixed time interval of 0.02.

BEYOND [12] BEYOND detects adversarial examples by leveraging self-supervised learning (SSL) to assess the consistency between an input image and its augmented variants (neighbors). It relies on two detection signals: *label consistency* and *representation similarity*. Given an SSL encoder f, a classification head g, and a projection head h, BEYOND first produces k augmentations $\{\hat{x}_i\}_{i=1}^k$ of input x. An input is flagged as adversarial if the number of neighbors sharing the same label with x falls below a threshold T_{label}, or if the cosine similarity between $r(x)$ and $r(\hat{x}_i)$ is below $T_{\cos}$ for too many neighbors. We use $k = 50$ neighbors, generated using augmentations (e.g., cropping, flipping, color jitter). We use the same SSL models trained with SimSiam from Solo-learn [8], but implement the detection procedure ourselves based on the pseudocode provided in the original paper.

A.2 Attack configurations

PGD [24] For PGD, we use $\epsilon = 8/255$, $\alpha = 2/255$ and $n = 10$, unless otherwise specified.

CW [3] For CW, we use the implementation of FoolBox [26]. We perform 9 binary search steps for c starting at 0.001. We use 50 steps of size 0.01. Confidence is set to 0.

DH [35] For DH, we use the default settings. These are $\epsilon = 8/255$, $\alpha = 0.007$ and $n = 50$. We use 5 EM samples. DH is allowed up to 3 restarts per input, restarting for samples with a lower attack success rate than 50% across 10 purification runs.

A.3 FlowPure

PGD $FlowPure^{PGD}$ is trained on PGD samples. Here, we sample the following distribution for the perturbation budget: $\epsilon \sim \mathcal{U}(0; 0.05)$, while fixing the other attack parameters to the standard $n = 10$ and $\alpha = 2/255$.

CW $FlowPure^{CW}$ is trained on CW samples. In this case we sample the following distributions for each attack parameter: $n \sim \mathcal{U}(1; 50)$, $c \sim \mathcal{U}(0; 2)$, $\kappa \sim \mathcal{U}(0; 1)$, and $lr \sim \mathcal{U}(0; 0.2)$. In the interest of computational efficiency and increased diversity, we make the following adjustments to CW during training. Firstly, we do not apply a hyper-parameter search, as this is too computationally expensive. Secondly, we do not stop optimizing when an adversarial example is found, as this will

Algorithm 1 FlowPure Training

Require: Dataset $p(x_{clean})$, adversarial attack $atk(x_{clean}, \xi)$, distribution for attack parameters $p(\xi)$ and model $v_\theta(t, x)$

while Training **do**
  $x_{clean} \sim p(x_{clean}); \quad \xi \sim p(\xi); \quad t \sim \mathcal{U}(0, 1)$
  $x_{adv} \leftarrow atk(x_{clean}, \xi)$
  $x \leftarrow t x_{clean} + (1 - t) x_{adv}$
  $u_t(x|x_{clean}, x_{adv}) \leftarrow x_{adv} - x_{clean}$
  $\mathcal{L}_{CFM} \leftarrow ||v_\theta(t, x) - u_t(x|x_{clean}, x_{adv})||_2^2$
  $\theta \leftarrow Update(\theta, \mathcal{L}_{CFM})$
end while
return $v_\theta(t, x)$

increase diversity of noise patterns. Note that these modifications are applied only during training, as during inference, we evaluated using the genuine, unmodified CW attack, reflecting reliable robustness assessments.

Gaussian For $FlowPure^{Gauss}$, the CNF is trained to remove Gaussian noise with fixed standard deviation $\sigma_{\max} = 0.3$.

Inference During Inference, FlowPure uses Euler's method to integrate the learned velocity field form $t = 0$ to $t = 1$ in 10 steps. The attacker uses a surrogate process with 5 steps. Euler's method uses the following update scheme: $x_{t+\Delta t} \leftarrow x_t + v_\theta(t, x_t)\Delta t$

When FlowPure is trained on Gaussian noise, we first inject noise and depending on the level of the noise injection, we integrate from $t = 1 - \frac{\sigma}{\sigma_{\max}}$ to $t = 1$. In this case, we still use Euler's method with 10 steps.

B Additional experimental results

B.1 Ablation study

Table 2 shows the performance of Gaussian FlowPure with several levels of σ values. To assess the effectiveness of the CNF paradigm when compared to traditional diffusion-based purification methods, we also evaluate DiffPure with several values of t. Both FlowPure and DiffPure use 10 iterations to remove the injected noise. In the white-box setting, the attacker uses a surrogate process with 5 iterations. FlowPure consistently exhibits superior performance (with the exception of PGD attacks in CIFAR-10), with the gap being especially noticeable in the white-box setting in CIFAR-100.

B.2 Detection

Detection scores. Figure 3 displays the distribution of velocities used as detection scores for clean and adversarial samples. We consider an attack-aware scenario (detector is trained and evaluated in the same attack), and a transfer scenario (detector is trained and evaluated on different attacks).

In the case of the attack-aware PGD detector, both distributions exhibit completely non-overlapping detection scores on both CIFAR-10 and CIFAR-100, enabling near-perfect detection of attacks. Moreover, this trend can be also observed with CW-trained detectors, as those seem to be efficient at detection of PGD adversarial samples. However, CW adversarial examples, both in the attack-aware and in the transfer scenario, demonstrate greater overlap in detection scores with clean samples, likely due to smaller perturbations of CW with less pronounced initial velocity field magnitudes, making them more challenging to distinguish from legitimate samples.

Comparison to BEYOND [12]. We additionally compare FlowPure to BEYOND [12], a recent state-of-the-art detection method based on Self-Supervised Learning (SSL), in preprocessor-blind scenarios. BEYOND detects adversarial examples by examining the label consistency and representation similarity between an input and its augmented neighbors. The results are summarized in Figure 4. For

Table 2: Robustness of Gaussian FlowPure and DiffPure[25] against DH[35] on CIFAR-10 and CIFAR-100, for various noise injection levels. For FlowPure, σ denotes standard deviation of Gaussian noise injected before purification. For DiffPure, t denotes how far along the forward process an image is diffused before purification.

	Method	Standard Accuracy		Robust Accuracy					
				Preprocessor-blind				White-box	
		avg	$worst$	PGD_{avg}	$\text{PGD}_{\text{worst}}$	CW_{avg}	CW_{worst}	DH_{avg}	DH_{worst}
CIFAR-10	*Victim*	95.81	95.81	0.04	0.04	0.00	0.00	–	–
	DiffPure:								
	$t = 50$	93.04	81.72	**89.47**	**73.38**	91.26	76.94	15.90 ± 0.50	4.94 ± 0.68
	$t = 80$	90.72	75.06	87.79	68.34	88.54	69.91	33.61 ± 0.91	12.63 ± 0.45
	$t = 100$	88.77	70.50	85.84	64.15	86.34	65.34	38.88 ± 0.74	15.56 ± 1.77
	$t = 130$	85.79	64.10	82.31	56.83	82.67	57.73	40.97 ± 1.03	16.92 ± 1.92
	$t = 150$	83.41	59.00	79.63	51.87	79.91	52.63	41.17 ± 1.03	16.86 ± 2.08
	FlowPure$^{\text{Gauss}}$:								
	$\sigma = 0.1$	**93.40**	**83.11**	88.42	72.52	**91.87**	**79.03**	17.88 ± 0.91	7.94 ± 0.92
	$\sigma = 0.15$	91.59	78.47	88.22	70.75	89.51	73.17	36.39 ± 1.16	17.57 ± 1.36
	$\sigma = 0.2$	89.22	72.20	85.99	65.34	86.78	66.44	42.27 ± 1.09	$\mathbf{21.28 \pm 1.08}$
	$\sigma = 0.25$	86.60	66.03	83.19	58.85	83.67	59.76	$\mathbf{43.29 \pm 1.03}$	$\mathbf{21.28 \pm 0.39}$
	$\sigma = 0.3$	83.80	59.52	79.86	52.27	80.22	52.70	42.49 ± 1.10	18.61 ± 1.89
CIFAR-100	*Victim*	80.73	80.73	0.01	0.01	0.00	0.00	–	–
	DiffPure:								
	$t = 50$	66.35	41.36	58.07	31.55	62.01	35.75	3.07 ± 0.41	0.91 ± 0.11
	$t = 80$	57.68	30.15	51.58	23.49	52.97	24.95	7.28 ± 0.36	1.62 ± 0.40
	$t = 100$	52.80	24.56	46.84	19.00	47.93	20.07	8.99 ± 0.66	2.01 ± 0.73
	$t = 130$	46.33	19.00	40.66	14.62	41.37	15.12	9.85 ± 0.86	1.95 ± 0.70
	$t = 150$	42.61	16.21	37.17	12.52	37.63	12.93	10.26 ± 0.67	1.82 ± 0.59
	FlowPure$^{\text{Gauss}}$:								
	$\sigma = 0.1$	**72.80**	**51.34**	**61.55**	**37.84**	**68.51**	**45.46**	4.76 ± 0.73	1.30 ± 0.29
	$\sigma = 0.15$	68.13	43.83	60.58	35.41	63.29	38.20	11.31 ± 1.19	3.84 ± 0.78
	$\sigma = 0.2$	63.62	37.81	56.94	30.86	58.46	32.41	14.38 ± 1.62	$\mathbf{4.55 \pm 1.00}$
	$\sigma = 0.25$	59.28	32.41	52.88	25.97	53.86	26.96	15.29 ± 1.61	4.16 ± 0.88
	$\sigma = 0.3$	55.17	27.58	48.75	21.51	49.51	22.05	$\mathbf{15.37 \pm 1.49}$	3.84 ± 0.96

both PGD and CW attacks, we evaluate the area under the ROC curve (AUC), and the true positive rate (TPR) at several false positive rate (FPR) thresholds.

When considering PGD attacks, our results consistently show that FlowPure outperforms BEYOND on both CIFAR-10 and CIFAR-100, with FlowPure achieving near-perfect AUC scores and substantially higher TPR at low FPRs across perturbation levels. Moreover, FlowPure becomes increasingly effective in distinguishing more strongly perturbed PGD samples, whereas BEYOND's detection capabilities appear to plateau or even slightly decrease in relative terms under larger perturbations. Conversely, when dealing with the smaller perturbations of CW attacks, BEYOND is generally more effective, except at very small FPRs on CIFAR-100.

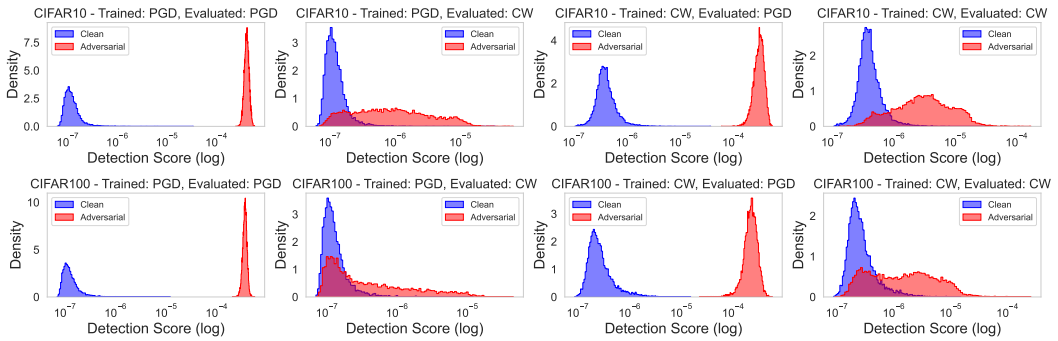
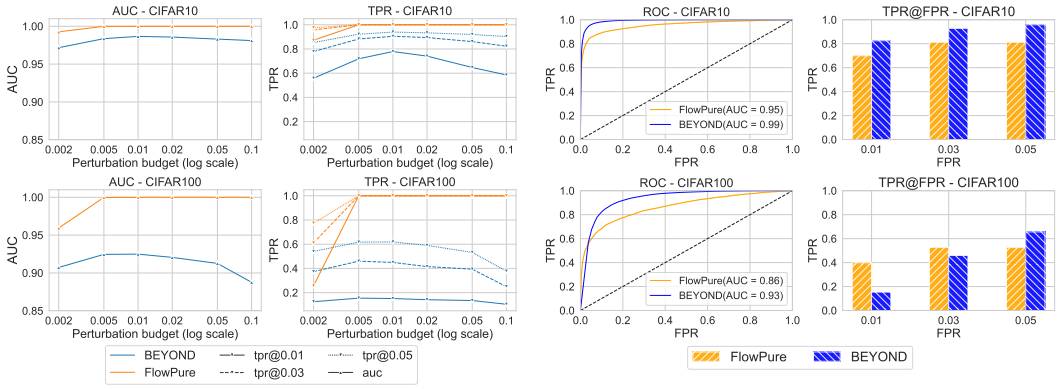


Figure 3: Distribution of detection scores for PGD and CW attacks. PGD attacks can be detected with near-perfect accuracy by FlowPure in both CIFAR-10 and CIFAR-100.



(a) Detection PGD attacks. AUC and TPR@FPR are shown for different perturbation levels.

(b) Detection of CW attacks. ROC curve and TPR@FPR are shown.

Figure 4: Detection performance of FlowPure and BEYOND in the preprocessor-blind setting.

B.3 Visualization

Figure 5 presents a qualitative comparison of reconstruction-based adversarial defenses applied to PGD adversarial examples generated on CIFAR-10 with an ℓ_∞ perturbation budget of 8/255. The grid shows 16 randomly selected images, comparing clean inputs, PGD attacks, and the outputs of several defense methods. Notably, $FlowPure^{PGD}$ consistently produces the most faithful reconstructions, often removing the adversarial noise while also denoising minor artifacts present in the original image. This ability may contribute to its strong preservation of benign classification accuracy. In contrast, DiffPure tends to hallucinate structural or semantic changes in the images, which can negatively affect downstream performance.

C Reproducibility

All experiments were performed on PyTorch version 2.31, using a NVIDIA GeForce RTX 3090 with 24 GB of memory. The WideResNet classifiers are trained using this implementation [4]. To train our models, we used the DDPM++ [29] architecture with the original implementation [5]. To apply CFM we use the following library [6] and to generate adversarial examples, we base our implementation on *torchattacks* [15]. Training on each dataset for each adversarial attack is the same. Training is done on batches of size 64 for 300000 iterations, with the Adam optimizer with a learning rate of 0.0002.

C.1 Computational resources

Table 3: Left: Time to process a batch (64 samples) during training for PGD and CW. The time to train a CNF on a Gaussian distribution is included for reference. Right: Combined time to purify and classify a batch.

	PGD (s)	CW (s)	Gaussian (s)
Total	$0.91 \pm 3.1e{-3}$	$5.52 \pm 6.4e{-2}$	$0.31 \pm 3.2e{-3}$
Gradient Descent	$0.31 \pm 8.7e{-4}$	$0.32 \pm 1.3e{-3}$	$0.31 \pm 3.2e{-3}$
Noise Generation	$0.60 \pm 2.7e{-3}$	$5.20 \pm 6.4e{-2}$	$2.3e{-3} \pm 7.9e{-5}$

	Inference Time (s)
DiffPure	$1.33 \pm 3.5e{-3}$
GDMP	$3.18 \pm 1.2e{-2}$
LM	$1.74 \pm 6.0e{-3}$
ADBM	$0.67 \pm 2.4e{-3}$
FlowPure	$1.20 \pm 4.0e{-3}$

Table 3 contains an indication of the training and inference time of our method.

Training. The total training time of FlowPure is dominated by the computation of adversarial attacks, as we chose a naive implementation for simplicity, in which adversarial examples are computed inside

[4] https://github.com/bmsookim/wide-resnet.pytorch (License: MIT)
[5] https://github.com/yang-song/score_sde_pytorch (License: Apache)
[6] https://github.com/atong01/conditional-flow-matching (License: MIT)

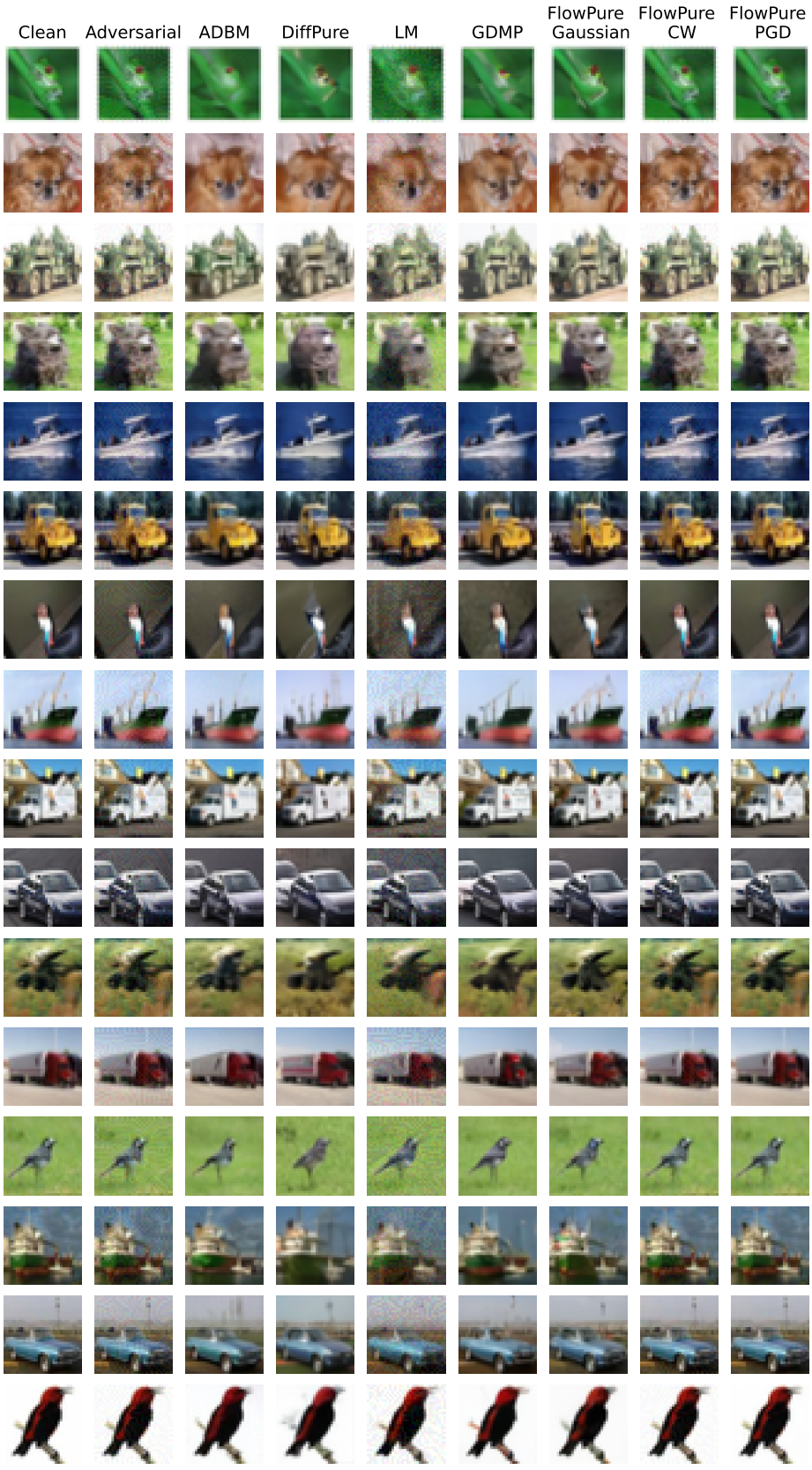


Figure 5: PGD adversarial examples in CIFAR-10 purified with several methods. $FlowPure^{PGD}$ produces the clearest reconstructions, often removing minor artifacts present in the original image.

the training loop. However, the adversarial examples used depend exclusively on the classifier, and as such, can be generated prior to training the CNF. Additionally, we noticed transferability properties (our model trained on PGD also performed well on purifying CW attacks), so it is worth exploring to use computationally simpler attacks during training.

Inference. During inference, computation time of each purification method is dominated by the number of times the diffusion or flow model is evaluated. Since every method uses the same architecture, this is equal among all methods. The step size can be tuned to make trade-offs between computation time and robustness. As we use the default configurations of each method, there is a difference in the time each method takes. The only exception to this would be LM [5], which requires computing gradients with respect to the diffusion model.