TOWARDS LEARNING IMPERCEPTIBLE ADVERSARIAL DISTRIBUTION FOR BLACK-BOX ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

An effective black-box threat model should find a sweet spot that balances well across success rate, perceptual quality, and query efficiency. In this paper, we propose PadvFlow, a black-box attack method that achieves the desirable property. Instead of searching for examples in a conventional ℓ_p space, PadvFlow leverages the use of normalizing flows (NFs) to model the density distribution of natural and indistinguishable adversarial examples in a perceptual space. The expressive NFs can reduce the perceptible noises. Meanwhile, searching for adversarial samples via the perceptual space improves details of generation. Thus, PadvFlow can generate perceptually-natural adversarial examples. Our comprehensive experiments show that PadvFlow not only successfully attacks 6 undefended and 4 defended image classifiers on CIFAR-10 and SVHN, but also can be scaled up to attack ImageNet of pixel size 299×299. The effectiveness of PadvFlow is also validated for a different modality by attacking an automatic speech recognition system.

1 INTRODUCTION

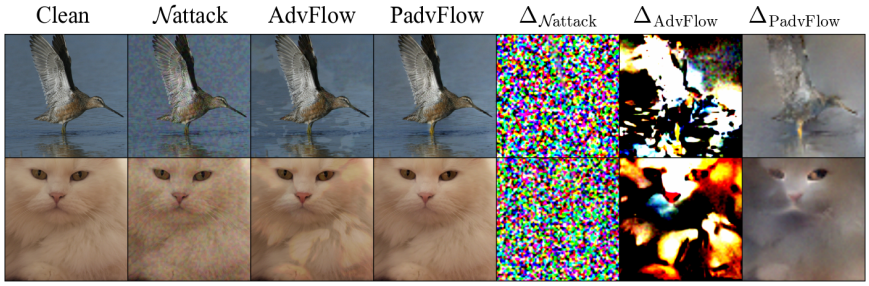


Figure 1: Generated adversarial examples and their individual magnified differences (denoted as Δ_{method}) by PadvFlow (the proposed method), $\mathcal{N}$attack (Li et al., 2019) and AdvFlow (Mohaghegh Dolatabadi et al., 2020). The classifier is InceptionV3 (Szegedy et al., 2016) trained on ImageNet of pixel size 299×299. $\mathcal{N}$attack tends to generate noise-like patterns in images as shown in $\Delta_{\mathcal{N}\text{attack}}$. Moreover, they are independent from the structures of input examples. On the other hand, PadvFlow and AdvFlow craft adversarial examples that correlate with the structure of data. However, we found that AdvFlow sometimes over-amplify certain areas of the images, resulting in artificial and translucent stains. In contrast, PadvFlow produces rather natural examples.

Deep learning (DL) has been proving its success in various fields including applications to daily life, like objection detection/classification in self-driving cars (Huang & Chen, 2020) and automatic speech recognition (ASR) systems (Li et al., 2022) in AI assistants. However, Szegedy et al. (2013); Biggio et al. (2013) discovered that neural networks are vulnerable even to small perturbations of data (known as adversarial examples), which raises a significant concern about their reliability. Such a phenomenon is initially discovered in an image classification problem and has been intensively investigated (Akhtar et al., 2021). In parallel, there have been a few work focusing on adversarial attacks in the audio domain (Zhang et al., 2017; Carlini & Wagner, 2018; Biolková & Nguyen, 2022). For example, adversaries can carefully construct perturbations that cause the original audio signals to be transcribed into malicious phrases or target transcripts (Carlini & Wagner, 2018). Adversarial examples in audio domain are often more perceptible to human ears, which has motivated growing

research interest to improve the imperceptibility mainly based on the psychoacoustic principle (Qin et al., 2019; Schönherr et al., 2018).

Despite modalities, the general objective of adversarial attack is to search for an example close to a given example so that it fools the target model. Typically, the closeness is measured by the ℓ_p-distance, known as ℓ_p-attack. Most existing adversarial attack methods fall into two main categories, including white-box and black-box attacks. Early methods focused on white-box attack (Xu et al., 2020a), where a full access to the target model is given to the adversaries. Under this assumption, the adversaries can leverage the model information such as gradients to solve a continuous optimization problem (Madry et al., 2017). In contrast, Song et al. (2018) search non-ℓ_p-restricted adversarial examples in the AC-GAN (Odena et al., 2017) latent spaces. Laidlaw & Feizi (2019); Xu et al. (2020b) lift the perturbation to features using some transformation, and Laidlaw et al. (2020) design the perceptually adversarial training to increase model robustness (see Appx. B.1.1 for details).

On the contrary, black-box attack is considered to be more practical since the adversaries have only access to the output of the target model through queries. In a realistic scenario, the number of trials of passing input through the target model is usually limited (known as query budgets). Because no internal information about the target model is given to the adversaries (Bhambri et al., 2019), black-box attack methods often require expensive query budgets. In the meanwhile, the generated adversarial samples should remain natural (no noises or corruptions) and imperceptible by human judgement. However, we observed that (as illustrated in Fig. 1) existing black-box methods based on the ℓ_p-norm perturbations sometimes generate examples with strong artifacts (easily perceptible) even they may reach high success rate or low query (see Sec. 3). Therefore, a method which can achieve the sweet spot of success rate, perceptual quality, and query efficiency is necessary.

To mitigate the generation of unnatural adversarial examples and to minimize the trade-off across success rate, perceptual quality and query efficiency, we introduce Perceptually Natural Adversarial Generating Flow (PadvFlow), an effective black-box method for generating imperceptible adversarial examples. PadvFlow aims to learn the distribution of imperceptible adversarial examples around a given data by leveraging the use of pre-trained normalizing flows (NFs) (Rezende & Mohamed, 2015) to transform the adversarial distribution into a normal distribution. Due to the expressivity of NFs, it can reduce noise of synthetic samples. To achieve imperceptibility and enhance the fidelity of details, the distance for perturbations is defined in the feature space (given by a perceptual map, which is a neural network-based feature extractor) instead of raw data spaces. By considering examples in the "imperceptible region" (Zhang et al., 2018; Laidlaw et al., 2020), we show that PadvFlow can generate imperceptible adversarial examples close to human judgments. Our methodology is testified to be effective across two modalities: image classifications and ASR systems. Indeed, designing realistic adversarial examples plays a crucial role in understanding the limitations of current DL models in many applications Gleave et al. (2019); Takahashi et al. (2021). Our study may raise awareness of imperceptible adversarial examples in a black-box manner, and stimulate the improvement of the robustness against such a serious threat to existing DL systems.

Related work. We review the current state of the art in black-box attacks as it is the main focus of this work. Due to the absence of internal information on the target model, one can perform gradient estimation by querying the target model. For instance, Chen et al. (2017); Tu et al. (2019) proposed to approximate the gradients via difference quotient (Lax & Terrell, 2020). To further reduce the number of queries, Ilyas et al. (2018) employed the bandit optimization (Hazan et al., 2016) to extract the prior information from previous iteration steps or data. Alternatively, HSJA (Chen et al., 2020) estimated the gradients via binary information on the decision boundary.

Another plausible approach to estimate the gradients is based on Natural Evolution Strategy (NES) (Wierstra et al., 2014). Rather than optimizing an objective function directly, NES searches for a parametrized distribution that maximizes the objective function in expectation (see Sec. 2.3 and Appx. A.2). Recently, this technique has been extended to solve the black-box optimization problem in adversarial attacks. The idea is to learn a probability distribution of examples around a given example such that a sample drawn from this distribution is likely an adversarial example. However, the adversarial distribution can be very complex and intractable for both sampling and density evaluation. To simplify this problem, Li et al. (2019) developed $\mathcal{N}$attack that used the $\tanh$ function to map the adversarial distribution into a normal distribution. Mohaghegh Dolatabadi et al. (2020) further improved $\mathcal{N}$attack by replacing the $\tanh$ with a pre-trained normalizing flow. Despite their high success rates, the last two methods still constrain adversarial examples being within the

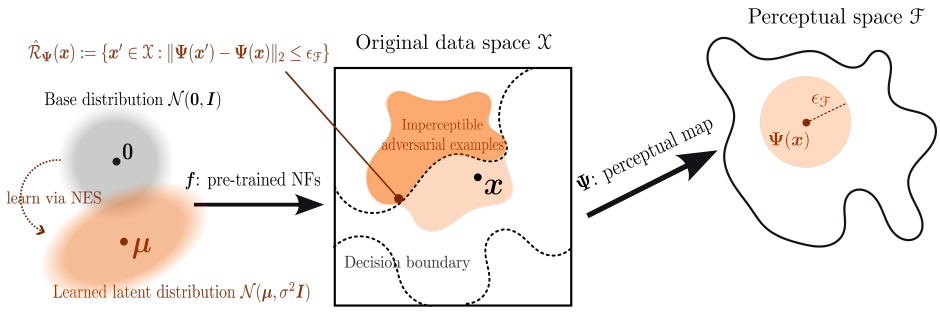


Figure 2: Overview of PadvFlow. We leverage pre-trained NFs to model the distribution of adversarial examples. Instead of using a conventional ℓ_p-norm, we employ the search on a perceptual space induced by a perceptual map Ψ to improve the imperceptibility.

ℓ_p-ball. Consequently, it may lead to perceptible perturbations. PadvFlow also belongs to the NES family but generally has better query-efficiency and sample quality than alternative methods.

Contributions. We observe that the common ℓ_p-norm used in adversarial attacks leads to easily-perceptible perturbations by humans (as shown in Fig. 1). A more practical attack method should deceive human perceptibility. Generating such deceptive examples can help not only to better understand deep neural networks, but also to improve their robustness. Our work was mainly motivated by this observation. Our contributions can be summarized as follows. (i) We introduce PadvFlow, a black-box attack method which generates adversarial examples within an approximated imperceptible region with high imperceptibility. The effectiveness of the approximated imperceptible region can be explained by analyzing the properties of perceptual maps and NFs (see Sec. 5). (ii) We comprehensively validate the effectiveness of PadvFlow against ten image classifiers and show its ability to scale-up to images with pixel size of 299×299 (see Sec. 3.1). Finally, we extend PadvFlow to a different modality and empirically show its high success rate in attacking an ASR system (see Sec. 3.2). We found out that PadvFlow can generate high fidelity adversarial samples and can generally balance well across success rate, perceptual quality and query efficiency.

Outline of this work. In Sec. 2, we formulate the problem and introduce PadvFlow in details. In Sec. 3, we testify PadvFlow for attacking image classifiers and ASR systems. In Sec. 4, we provide additional evidence to support the effectiveness of PadvFlow in terms of success rate and perceptual quality. In Sec. 5, we explain the mechanism of PadvFlow. At last, we conclude this work in Sec. 6.

2 The Proposed Method: PadvFlow

In Sec. 2.1, we establish our problem setup by formulating an untargeted adversarial attack on an imperceptible region. In Sec. 2.2, we introduce PadvFlow with the modeling of imperceptible region. We describe its optimization, based on NES in Sec. 2.3. Fig. 2 shows the outline of PadvFlow.

2.1 Problem Formulation

Let $\mathscr{X}$ be a data space, $\mathscr{Y}$ be a label space and $\mathscr{P}$ be the space indicating the probability distribution on $\mathscr{Y}$. $h\colon \mathscr{X} \to \mathscr{P}$ is a black-box model that predicts the label corresponding to an input $\boldsymbol{x} \in \mathscr{X}$ with a probability $\boldsymbol{z} \in \mathscr{P}$. Let $\mathcal{T}_h\colon \mathscr{X} \times \mathscr{Y} \to [0, 1]$ denote an operator that extracts the probability of a certain label predicted by h for a given input-label pair $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathscr{X}, \mathscr{Y})$, i.e., $P(Y = \boldsymbol{y} \mid X = \boldsymbol{x}; h) = \mathcal{T}_h(\boldsymbol{x}, \boldsymbol{y})$. Then, we denote an imperceptible region of $\boldsymbol{x}$ as $\mathcal{R}_{\boldsymbol{x}} \subset \mathscr{X}$, where the difference between $\boldsymbol{x}$ and the samples in $\mathcal{R}_{\boldsymbol{x}}$ are imperceptible. The untargeted adversarial attack associated with $\mathcal{R}_{\boldsymbol{x}}$ can be described as the procedure of searching an adversarial example $\boldsymbol{x}_{\text{adv}} \in \mathcal{R}_{\boldsymbol{x}}$ such that it deceives the model h, i.e., $\arg\max_{\boldsymbol{y}' \in \mathscr{Y}} \mathcal{T}_h(\boldsymbol{x}_{\text{adv}}, \boldsymbol{y}') \neq \boldsymbol{y}$. This is usually depicted via the Carlini and Wagner (C&W) loss (Carlini & Wagner, 2017) $\mathcal{L} : \mathscr{X} \times \mathscr{Y} \to \mathbb{R}_+$, which is defined as

$$\mathcal{L}(\boldsymbol{x}', \boldsymbol{y}; \mathcal{T}_h) := \max\left(0, \log \mathcal{T}_h(\boldsymbol{x}', \boldsymbol{y}) - \max_{\boldsymbol{y}' \neq \boldsymbol{y}} \log \mathcal{T}_h(\boldsymbol{x}', \boldsymbol{y}')\right).$$

If $\mathcal{L}(\boldsymbol{x}', \boldsymbol{y}; \mathcal{T}_h) = 0$, it implies that $\boldsymbol{x}'$ deceived h. Finally, the untargeted attack is formulated as:

$$\boldsymbol{x}_{\text{adv}} = \arg\min_{\boldsymbol{x}' \in \mathcal{R}_{\boldsymbol{x}}} \mathcal{L}(\boldsymbol{x}', \boldsymbol{y}; \mathcal{T}_h). \tag{1}$$

Although this work focuses on untargeted attacks, it can be easily adapted for targeted attacks by letting $\boldsymbol{x}_{\text{adv}} = \arg\max_{\boldsymbol{x}' \in \mathcal{R}_{\boldsymbol{x}}} \mathcal{T}_h(\boldsymbol{x}', \boldsymbol{y}_{\text{t}})$, where $\boldsymbol{y}_t \neq \boldsymbol{y}$ denotes the target label. It is noticed that defining the perceptual similarity required for $\mathcal{R}_{\boldsymbol{x}}$ is highly subjective and generally infeasible, we therefore propose and explain an alternative approach of modeling $\mathcal{R}_{\boldsymbol{x}}$ in Sec. 2.2 and Sec. 5. Moreover, the design of operator $\mathcal{T}_h$ is often task-dependent. We demonstrate two $\mathcal{T}_h$ examples for an image classifier and an ASR system in Appx. A.1.

2.2 Approximated Imperceptible Region

As commonly done (Li et al., 2019; Mohaghegh Dolatabadi et al., 2020), adversarial examples are considered within an ℓ_p-ball around the original data with a fixed radius $\epsilon_{\mathscr{X}} > 0$, i.e., $\mathbb{B}_{\mathscr{X},p}(\boldsymbol{x}, \epsilon_{\mathscr{X}}) := \{\boldsymbol{x}' \in \mathscr{X} : \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \epsilon_{\mathscr{X}}\}$. However, Zhang et al. (2018) pointed out that $\mathbb{B}_{\mathscr{X},p}$ is not always a good approximation to the imperceptible region $\mathcal{R}_{\boldsymbol{x}}$ (see Fig. 1) since ℓ_p and other conventional metrics do not agree with human similarity judgments. Consequently, examples within a large ℓ_p-ball might still be imperceptible, while those lie within a small ℓ_p-ball can be easily perceptible.

To mitigate the problem, we follow Zhang et al. (2018) and Laidlaw et al. (2020) to model $\mathcal{R}_{\boldsymbol{x}}$ as an approximated imperceptible region $\hat{\mathcal{R}}_\Psi(\boldsymbol{x}, \epsilon_{\mathscr{F}})$ with a given threshold $\epsilon_{\mathscr{F}}$ by introducing a perceptual space $\mathscr{F}$, which is induced by a feature map $\Psi : \mathscr{X} \to \mathscr{F}$. More precisely, we define

$$\hat{\mathcal{R}}_\Psi(\boldsymbol{x}, \epsilon_{\mathscr{F}}) := \{\boldsymbol{x}' \in \mathscr{X} : d_\Psi(\boldsymbol{x}', \boldsymbol{x}) \leq \epsilon_{\mathscr{F}}\},$$

where $d_\Psi(\boldsymbol{x}', \boldsymbol{x}) := \|\Psi(\boldsymbol{x}') - \Psi(\boldsymbol{x})\|_p$ with $p \geq 1$. We write it as $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ for simplicity. Since some pre-trained DNNs extract features that reflect perceptual information greatly (Zhang et al., 2018), adopting them for Ψ brings us effective perceptual distances. For different modalities, we utilize different d_Ψ's in PadvFlow (details are in Sec. 3 and Appx. B). Thanks to high alignment of such perceptual metrics and human judgement, we may expect that $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ serves as a satisfactory approximation for the imperceptible region $\mathcal{R}_{\boldsymbol{x}}$ ($\hat{\mathcal{R}}_\Psi(\boldsymbol{x}) \approx \mathcal{R}_{\boldsymbol{x}}$).

We now explain the construction of the map $\text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}$. We may just consider the case when $\boldsymbol{x}'$ is out of $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$. The idea is to look for a point $\boldsymbol{x}'_{\text{bdry}}$ that lies on the boundary of $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$. More precisely, we consider a continuous curve $\gamma(t) := d_\Psi(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x}), \boldsymbol{x}) - \epsilon_{\mathscr{F}}$ and aim at solving $t^* \in [0, 1]$ such that $\gamma(t^*) = 0$ to obtain $\boldsymbol{x}'_{\text{bdry}} = \boldsymbol{x} + t^*(\boldsymbol{x}' - \boldsymbol{x})$. Notice that $\gamma(0) < 0$, $\gamma(1) > 0$, the equation is always solvable on $[0, 1]$ (see Appx. A.3 for a guarantee). To avoid the computation of the gradient of Ψ, any "derivative-free" root-solver can be adopted for the equation solving. In our implementation, we utilize the "bisection method" (Corliss, 1977) to solve for t^* (see Appx. A).

2.3 PadvFlow: Learning Imperceptible Adversarial Distributions

In this section, we explain how to learn an approximated adversarial distribution for any given pair $(\boldsymbol{x}, \boldsymbol{y})$ within an approximated imperceptible region $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ by using NES. Instead of solving (1) directly, we smooth the problem by introducing a parametric family of distributions, $\{\pi(\boldsymbol{x}'|\boldsymbol{\theta}_{\boldsymbol{x}}) : \boldsymbol{\theta}_{\boldsymbol{x}}\}$, as in Ilyas et al. (2018). We expect to find a $\pi(\boldsymbol{x}'|\boldsymbol{\theta}_{\boldsymbol{x}})$ that resides in $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ via updating $\boldsymbol{\theta}_{\boldsymbol{x}}$.

$$\min_{\boldsymbol{\theta}_{\boldsymbol{x}}} \mathcal{J}(\boldsymbol{\theta}_{\boldsymbol{x}}) := \mathbb{E}_{\boldsymbol{x}' \sim \pi(\boldsymbol{x}'|\boldsymbol{\theta}_{\boldsymbol{x}})} \left[\mathcal{L}(\boldsymbol{x}', \boldsymbol{y}; \mathcal{T}_h)\right]. \tag{2}$$

Since $\{\pi(\boldsymbol{x}'|\boldsymbol{\theta}_{\boldsymbol{x}})\}_{\boldsymbol{\theta}_{\boldsymbol{x}}}$ is generally intractable, Li et al. (2019); Mohaghegh Dolatabadi et al. (2020) introduced a latent space $\mathscr{Z}$ with a bijective map $\boldsymbol{f} : \mathscr{Z} \to \mathscr{X}$ which transforms variables between the latent and data space via $\boldsymbol{x}' = \text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}\big(\boldsymbol{f}(\boldsymbol{z})\big)$ with $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{x}})$, where $\mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{x}})$ is a normal distribution with learnable parameters $\boldsymbol{\mu}_{\boldsymbol{x}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}}$, and $\text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}$ is a continuous projection onto $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ (details in Sec. 2.2). Empirically, we found that admitting a trainable $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ only yields minor performance improvement at the cost of heavily increased computation. Thus, we fix $\boldsymbol{\Sigma}_{\boldsymbol{x}} \equiv \sigma^2 \boldsymbol{I}$ for a constant $\sigma > 0$. Therefore, problem (2) becomes

$$\min_{\boldsymbol{\mu}_{\boldsymbol{x}}} \mathcal{J}(\boldsymbol{\mu}_{\boldsymbol{x}}) := \mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma^2 \boldsymbol{I})} \left[\mathcal{L}\big(\text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}(\boldsymbol{f}(\boldsymbol{z})), \boldsymbol{y}; \mathcal{T}_h\big)\right].$$

NES solves the optimization problem in a gradient descent manner, where the "log-likelihood trick" (see Appx. A) is applied to compute the gradient

$$\nabla_{\boldsymbol{\mu}_{\boldsymbol{x}}} \mathcal{J}(\boldsymbol{\mu}_{\boldsymbol{x}}) = \mathbb{E}_{\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma^2 \boldsymbol{I})} \left[\mathcal{L}\big(\text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}(\boldsymbol{f}(\boldsymbol{z})), \boldsymbol{y}; \mathcal{T}_h\big) \nabla_{\boldsymbol{\mu}_{\boldsymbol{x}}} \log \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma^2 \boldsymbol{I})\right].$$

Thanks to the closed-form of $\nabla_{\boldsymbol{\mu_x}} \log \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu_x}, \sigma^2 \boldsymbol{I}) = \sigma^{-2}(\boldsymbol{z} - \boldsymbol{\mu_x})$, parameters can be updated via a simple formula with batches of samples $\{\boldsymbol{z}_j := \boldsymbol{\mu_x} + \boldsymbol{\epsilon}_j \sigma\}_{j=1}^M$ where $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

$$\boldsymbol{\mu_x} \leftarrow \boldsymbol{\mu_x} - \frac{\eta}{\sigma M} \sum_{j=1}^{M} \mathcal{L}\big(\text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}(\boldsymbol{f}(\boldsymbol{z}_j)), \boldsymbol{y}; \mathcal{T}_h\big)\boldsymbol{\epsilon}_j,$$

where η is a learning rate and M is the population size. We notice that it enables to bypass the gradient information on $\mathcal{L}\big(\text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}(\boldsymbol{f}(\boldsymbol{z})); \boldsymbol{y}, \mathcal{T}_h\big)$ and hence, also the gradient of $\mathcal{T}_h$ (black-box). In PadvFlow, we take the invertible mapping $\boldsymbol{f}$ as pre-trained NFs with the based distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Moreover, it is worthwhile to mention that the adversarial distribution is learned individually for each $\boldsymbol{x}$. Hence, it depends on $\boldsymbol{x}$. We summarize PadvFlow as Alg. 2 in Appx A.4.

3 Experimental Results

In this section, we examine the effectiveness of PadvFlow regarding success rate[1] and testify with different modalities of $\mathcal{T}_h$, including image classifiers (Sec. 3.1) and ASR systems (Sec. 3.2).

3.1 Image Classifiers

In this section, we focus on the case when $\mathcal{T}_h$ is image classifiers. All benchmarks are ℓ_p-attacks; we compare when $p = \infty$ with a fixed conventional threshold $\epsilon_{\mathscr{X}} = 8/255$ and maximum query budget as $10,000$. Following the conventional setting of Li et al. (2019); Mohaghegh Dolatabadi et al. (2020), we fix maximum iteration as 500, population size $M = 20$, learning rate $\eta = 0.02$ and std. of the base normal distribution as $\sigma = 0.1$. The perceptual map Ψ in PadvFlow is taken as Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) and the threshold is set to be $\epsilon_{\mathscr{F}} = 0.5$ to bound d_Ψ. We utilize RealNVP (Dinh et al., 2016) as $\boldsymbol{f}$ and follow the implementation of (Mohaghegh Dolatabadi et al., 2020) (details in Sec.B.1.4). We test PadvFlow with with different invertible maps $\boldsymbol{f}$ in Sec. 4 and explore its possible extension in Appx. E.

CIFAR-10 and SVHN We compare PadvFlow with five benchmark methods: AdvFlow, $\mathcal{N}$attack, HSJA, Bandits, and AutoAttack. We test with six undefended and four defended classifiers $\mathcal{T}_h$ trained on CIFAR-10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011). We utilize the entire clean test set ($10,000$ instances for CIFAR-10 and $26,032$ for SVHN) to evaluate a threat method. In PadvFlow, the perceptual mapping Ψ is constructed as (Zhang et al., 2018) with a pre-trained AlexNet (Krizhevsky et al., 2009) on CIFAR-10 and SVHN, respectively. We refer to Appx. B for more details about the setup of benchmarks, $\mathcal{T}_h$ and Ψ. We further compare the query efficiency of PadvFlow, AdvFlow, and $\mathcal{N}$attack. Results are reported in Table 1.

ImageNet We extend PadvFlow to higher resolution image, ImageNet (Deng et al., 2009) of pixel size 299×299, and compare with two benchmarks (AdvFlow and $\mathcal{N}$attack). To evaluate, we randomly select 10,000 samples in the test set of ImageNet for undefended classifiers while we randomly select 1,000 samples from the test set of ImageNet-100 (a subset of ImageNet with fewer classes) for the defended classifier due to its eay availability of the pre-trained model. Table 2 records results of success rate, query, and objective perceptual measurement.

We observe that PadvFlow generally outperforms benchmark methods with large margins among all classifiers. In particular, AdvFlow usually has lower success rates on undefended classifiers compared with its ancestor, $\mathcal{N}$attack. However, we found that $\mathcal{N}$attack produces noisy adversarial examples. In contrast, PadvFlow can reach higher success rates across wide categories of defended/undefended classifiers and generate natural adversarial images (see Fig. 6 or perceptual measurements on ImageNet in the last panel of Table 1). On the other hand, we point out that the defended classifier, RN50-PAT, is obtained by perceptual adversarial training (PAT) Laidlaw et al. (2020), which is designed to inherently defense perceptually generated adversarial samples with the access to the gradient of $\mathcal{T}_h$. Surprisingly, PadvFlow can also fool such classifiers with high success rates. Moreover, PadvFlow only requests a moderate query amount across all classifiers, and actually for some defended ones, PadvFlow is the most efficient method.

[1] Success rate is the ratio of the number of successfully attacked samples over correctly classified data.

Table 1: First two tables compare success rate of PadvFlow and benchmark methods across six undefended and four defended classifiers. Last table compares averaged query and median of PadvFlow, AdvFlow and $\mathcal{N}$attack across one undefended and four defended classifiers. Classifiers are trained on CIFAR-10 and SVHN, respectively.

Type	CIFAR-10 classifiers	Clean acc. (%)	Success rate (%)					
			PadvFlow	AdvFlow	$\mathcal{N}$attack	HSJA	Bandits	AutoAttack
undefended	VGG19 (Simonyan & Zisserman, 2014)	93.59	94.98	90.55	**96.41**	71.62	86.38	93.82
	GoogLeNet (Szegedy et al., 2015)	95.51	**97.49**	97.27	97.01	83.08	97.55	90.38
	MobileNetV2 (Sandler et al., 2018)	94.92	93.24	85.12	**99.24**	79.82	93.99	89.52
	DLA (Yu et al., 2018)	94.86	93.44	81.66	**98.83**	78.08	72.29	89.37
	DenseNet121 (Huang et al., 2017)	95.54	**98.13**	93.26	98.03	75.41	82.46	90.45
	WRN34 (Zagoruyko & Komodakis, 2016)	95.42	97.49	98.82	99.90	75.77	93.36	**100.00**
Defended	WRN34-Free (Shafahi et al., 2019)	81.30	86.92	41.06	38.39	81.15	35.56	**89.04**
	WRN34-Fast (Wong et al., 2020)	86.35	**90.85**	40.24	36.41	80.64	33.29	86.99
	WRN34-RotNet (Hendrycks et al., 2019)	57.10	**89.88**	54.74	53.59	60.89	47.86	85.61
	RN50-PAT (Laidlaw et al., 2020)	80.54	**73.77**	39.25	37.92	61.20	36.79	66.57

Type	SVHN classifiers	Clean acc. (%)	Success rate (%)					
			PadvFlow	AdvFlow	$\mathcal{N}$attack	HSJA	Bandits	AutoAttack
Undefended	VGG19 (Simonyan & Zisserman, 2014)	96.23	96.64	86.01	**98.04**	87.32	85.33	88.47
	GoogLeNet (Szegedy et al., 2015)	96.81	**99.19**	87.63	98.51	78.21	88.64	91.68
	MobileNetV2 (Sandler et al., 2018)	96.71	**99.89**	88.69	99.51	72.35	94.05	89.32
	DLA (Yu et al., 2018)	96.88	98.56	84.19	**98.67**	79.52	76.94	90.546
	DenseNet121 (Huang et al., 2017)	96.97	98.18	84.59	**98.49**	71.47	78.58	90.46
	WRN34 (Zagoruyko & Komodakis, 2016)	94.20	91.11	90.17	94.72	83.54	80.58	**96.39**
Defended	WRN34-Free (Shafahi et al., 2019)	63.48	**87.52**	56.23	62.23	38.42	60.34	68.05
	WRN34-Fast (Wong et al., 2020)	87.99	**90.79**	49.81	48.45	50.59	43.77	73.31
	WRN34-RotNet (Hendrycks et al., 2019)	82.81	**89.82**	42.06	41.39	59.81	36.56	68.54
	RN50-PAT (Laidlaw et al., 2020)	67.31	59.69	41.257	39.69	49.64	37.12	**64.63**

Type	Classifiers	Query of CIFAR-10 (average/median)			Query of SVHN (average/median)		
		PadvFlow	AdvFlow	$\mathcal{N}$attack	PadvFlow	AdvFlow	$\mathcal{N}$attack
Undefended	WRN34	629.13/400	966.70/400	**238.92/200**	1031.01/600	1698.21/600	**411.12/200**
Defended	WRN34-Free	662.16/**200**	**514.52/200**	1165.87/600	**680.20/200**	701.93/400	1034.83/400
	WRN34-Fast	**408.88/200**	447.39/**200**	917.76/400	**432.75/200**	548.78/400	1201.16/400
	WRN34-RotNet	478.93 /**200**	**414.78/200**	897.90/400	539.26/200	**476.63/200**	791.11/400
	RN50-PAT	726.22/400	**584.69/200**	1042.67/600	1031.31/400	**617.03/200**	1402.87/600

Table 2: Comparison of PadvFlow with AdvFlow and $\mathcal{N}$attack via success rate, query efficiency and samples quality across three undefended (InceptionV3 (Szegedy et al., 2016), VGG16 (Simonyan & Zisserman, 2014), RN50 (He et al., 2016)) and one defended (RN50-PAT (Laidlaw et al., 2020)) classifiers on ImageNet. In order, they have clean accuracy: 77.17%, 71.59%, 80.28% and 56.37%.

Classifiers	Success rate			Query (average/median)			Perception (SSIM ↑)		
	PadvFlow	AdvFlow	$\mathcal{N}$attack	PadvFlow	AdvFlow	$\mathcal{N}$attack	PadvFlow	AdvFlow	$\mathcal{N}$attack
InceptionV3	**94.42** %	90.83 %	94.27 %	**921.84/600**	1636.10/800	1022.58/**600**	**0.9713**	0.9670	0.9259
VGG16	95.47%	93.81%	**98.80** %	870.87/600	1860.87/1000	**357.80/200**	0.9651	**0.9728**	0.9326
RN50	96.45%	92.79 %	**99.15**%	1049.09/600	2181.73/1400	**419.87/200**	0.9625	**0.9719**	0.9263
RN50-PAT (ImageNet-100)	53.61 %	11.32%	15.56 %	912.23/600	873.01/400	**568.62/200**	**0.9851**	0.9797	0.9581

3.2 Automatic Speech Recognition Systems

In this section, the effectiveness of PadvFlow is validated on speech domain. We benchmark our method on the LibriSpeech dataset (Panayotov et al., 2015), which consists of English speech derived from audiobooks sampled at 16kHz. Our target model is an end-to-end ASR system from SpeechBrain (Ravanelli et al., 2021). This ASR system is trained on LibriSpeech and achieves a Word Error Rate (WER) of 3.09% on the clean test set. In our experiment, we randomly select 1000 audio examples from the clean test set to evaluate an attack method. All examples are correctly transcribed by the target ASR system. An attack is considered as successful when the WER between the original and the adversarial transcripts is greater than 10%. We use a budget of maximum 3000 queries. We slightly modified the invertible neural network WaveGlow (Prenger et al., 2019) for both PadvFlow and AdvFlow. The network is trained on the LibriSpeech train-clean-360 dataset.

Table 3: Comparison of $\mathcal{N}$attack, AdvFlow, and PadvFlow for attacking ASR under different allowed magnitudes of perturbations.

Method	Distortion	Success rate (%)	Query (avg/median)	CDPAM ↓	PESQ ↑
$\mathcal{N}$attack	$\epsilon_{\mathscr{X}} = 0.01$	57.20	1136.45/964	0.3490	3.2002
	$\epsilon_{\mathscr{X}} = 0.05$	98.00	527.71/364	0.5641	2.2532
AdvFlow	$\epsilon_{\mathscr{X}} = 0.01$	16.90	1290.63/1244	**0.0316**	**4.3026**
	$\epsilon_{\mathscr{X}} = 0.05$	54.60	1071.84/924	0.0649	3.9770
PadvFlow	$\epsilon_{\mathscr{F}} = 0.05$	98.80	509.02/404	0.0429	3.6108
	$\epsilon_{\mathscr{F}} = 0.10$	**99.90**	**435.51/364**	0.0639	3.6066

Note that we do not use any information of transcripts to train WaveGlow. The perceptual map Ψ in PadvFlow is considered as Contrastive learning-based multi-dimensional Deep Perceptual Audio similarity Metric (CDPAM) (Manocha et al., 2021).

For evaluation, we report the Perceptual Evaluation of Speech Quality (PESQ) (Rix et al., 2001) and CDPAM as perceptual metrics to measure the speech quality of adversarial examples. We encourage the readers to listen to the adversarial examples (can be found in the supplementary materials[2]) to hear how they sound. Table 3 shows the results under different perturbations. It can be seen that PadvFlow achieves a high success rate compared to AdvFlow and $\mathcal{N}$attack. The average CDPAM distances of both PadvFlow and AdvFlow are much smaller than those of $\mathcal{N}$attack. We observe the same behaviors for PESQ scores. Compared to $\mathcal{N}$attack, adversarial examples generated by PadvFlow are much better in terms of perceptual metrics when both methods have similar success rate. Importantly, PadvFlow is query-efficient, which only requires on average of 500 queries to obtain a high success rate. In contrast, AdvFlow needs a lots more queries to explore the adversarial examples. We visualize the samples via spectrograms and plot in Appx. C.2.

4 Ablation Study and Additional Experiments

We comprehensively validate PadvFlow with additional experiments. In Sec. 4.1, we conduct an ablation study to show each component of PadvFlow is essential. In Sec. 4.2, we show the fairness of $\hat{\mathcal{R}}_{\Psi}(\boldsymbol{x})$ in PadvFlow by comparing quality of adversarial samples, conditioned on a assigned success rate. Unless otherwise stated, we focus on PadvFlow and AdvFlow, and consider defended image classifiers $\mathcal{T}_h$ = WRN34-Free and RN50-PAT trained on CIFAR-10 for a demonstration purpose.

4.1 Ablation Study

In this section, we demonstrate that both expressive invertible function $\boldsymbol{f}$ (= NFs) and perceptual map Ψ (= LPIPS) in PadvFlow are crucial to reach a high success rate and simultaneously maintain high quality of synthesized adversarial examples. We consider three variants of PadvFlow, where PadvFlow-$(\tanh, \Psi)$ and PadvFlow-(id, Ψ) replaces NFs with $\boldsymbol{f} = \tanh$ and identity map, respectively. In addition, PadvFlow-$(\text{NFs}, \Psi_{\text{rand}})$ replaces its LPIPS with Ψ = randomly initialized map (not trained) from $\mathscr{X}$ to $\mathscr{F}$. We utilize the entire test set of CIFAR-10 and locate the samples where all methods are successful for evaluation. We record the success rate and the average of pair-wise (clean data and its adversarial copy) perceptual measurements over those common samples. For a complete comparison, the results of $\mathcal{N}$-attack and AdvFlow are also included. We report results in Table 4. We further visualize some examples generated by those compared methods in Fig. 3.

Surprisingly, all the variants of PadvFlow outperform $\mathcal{N}$-attack and AdvFlow in success rate. However, their generated images may be unnatural (noisy or blurry), which is also reflected by perceptual measurements. In PadvFlow, NFs can help reducing noise. Meanwhile, perceptual map can improve quality of details. Both components of PadvFlow are crucial to the balance between success rate and sample quality. In Appx. E, we discuss a plausible usage of a probability flow ODE (Song et al., 2020) as $\boldsymbol{f}$ attempting to increase perceptual quality of samples.

[2]The source code and some generated samples for ASR attacks are attached in `https://drive.google.com/drive/folders/1VmH4siq4ii40jCkmHeOnRu9nny66jixX?usp=sharing`.

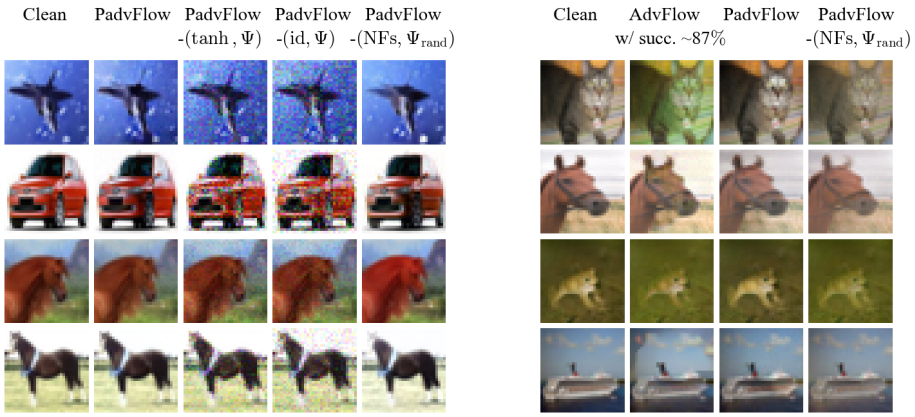
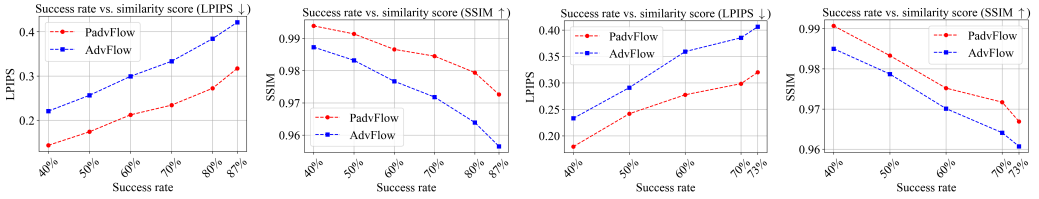


Figure 3: Left panel illustrates PadvFlow and its variants. Right panel demonstrates AdvFlow with adjusted $\epsilon_{\mathcal{X}} \approx 17.85/255$ so that its has success rate closed to PadvFlow ($\approx 87\%$), and compare with PadvFlow and PadvFlow-$(\text{NFs}, \Psi_{\text{rand}})$.

Table 4: Objective perceptual measures of PadvFlow, its ablated variants, $\mathcal{N}$attack, and AdvFlow.

	Methods	$\mathcal{N}$attack	AdvFlow	PadvFlow	PadvFlow-$(\tanh, \Psi)$	PadvFlow-(id, Ψ)	PadvFlow-$(\text{NFs}, \Psi_{\text{rand}})$
	$\boldsymbol{f}$	$\tanh$	NFs	NFs	$\tanh$	identity map	NFs
	$\mathcal{R}_{\boldsymbol{x}}$	ℓ_∞	ℓ_∞	$\hat{\mathcal{R}}_{\text{LPIPS}_{\text{CIFAR}}}(\boldsymbol{x})$	$\hat{\mathcal{R}}_{\text{LPIPS}_{\text{CIFAR}}}(\boldsymbol{x})$	$\hat{\mathcal{R}}_{\text{LPIPS}_{\text{CIFAR}}}(\boldsymbol{x})$	$\hat{\mathcal{R}}_{\Psi_{\text{rand}}}(\boldsymbol{x})$
WRN34-Free	Success rate	38.39%	41.06%	86.92%	82.64%	72.27%	**95.71** %
	$\text{LPIPS}_{\text{ImageNet}} \downarrow$	0.4096	0.2396	**0.2378**	0.3361	0.4101	0.4009
	SSIM ↑	0.9495	0.9856	**0.9867**	0.9546	0.9464	0.9432
RN50-PAT	Success rate	37.92%	39.25%	73.77%	62.59%	51.10%	**95.22** %
	$\text{LPIPS}_{\text{ImageNet}} \downarrow$	0.4270	0.2462	**0.2422**	0.3508	0.4219	0.4063
	SSIM ↑	0.9438	**0.9762**	0.9709	0.9418	0.9366	0.9373

4.2 Success Rate vs. Quality of Generation



(a) WRN34-Free; LPIPS (b) WRN34-Free; SSIM (c) RN50-PAT; LPIPS (d) RN50-PAT; SSIM

Figure 4: "Perceptual measurements versus success rates" comparisons of PadvFlow and AdvFlow. We examine for two defended classifiers (WRN34-Free and RN50-PAT) and quantify by perceptual measurements: $\text{LPIPS}_{\text{ImageNet}}$ (simplified as LPIPS in figures), SSIM.

Although we validated the effectiveness of PadvFlow with success rate in Sec. 3, unlike many ℓ_p-attacks, $\hat{\mathcal{R}}_{\Psi}(\boldsymbol{x})$ is not a typical ℓ_p region in a raw data space. A natural question may arise: are the comparisons fair? We answer the question from a different aspect – – the perceptual quality of adversarial samples, another significant standard to judge a threat model. By adjusting the threshold ($\epsilon_{\mathscr{F}}$ for PadvFlow and $\epsilon_{\mathscr{X}}$ for AdvFlow), each threat model can achieve any given success rate. We then numerically compare the quality of their synthetic adversarial examples with 2 objective metrics: $\text{LPIPS}_{\text{ImageNet}}$ (trained on ImageNet) and SSIM. We randomly sample 1000 instances from the test set of CIFAR-10, find the samples where both the attack methods succeed, and plot the correlation between objective metrics and success rate in Fig. 4. Table 6 in Appx. B summarizes the specific choices of approximated $\epsilon_{\mathscr{F}}$ and $\epsilon_{\mathscr{X}}$ to reach an assigned success rate. We observe that conditioning on the same success rate, PadvFlow can always synthesize higher fidelity adversarial instances than AdvFlow. This implies PadvFlow is not just recklessly extending the searching region

with $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$; instead, PadvFlow utilizes a more appropriate domain to search for imperceptible adversarial candidates. At last, we point that an algorithm (e.g., PadvFlow-(NFs, Ψ_{rand})) might easily reach a high success rate and be treated as an effective method. However, their samples have inferior perceptual quality, which indicates the need of simultaneously taking all three aforementioned aspects into account to evaluate a threat model. Given the empirical results, we can conclude that PadvFlow succeeds in easily attacking target models with high fidelity adversarial copies.

5 Mechanism and Intuition of PadvFlow

We provide a heuristic explanation of the effectiveness of PadvFlow in this section. We start with a summary of geometric properties of $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ and investigate its advantage.

Observations 1. In general, $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ is not necessarily convex;

2. Denote $\|\boldsymbol{A}\|_{\text{op}}$ the operator norm of the matrix $\boldsymbol{A}$. The first-order estimate to $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ is

$$\left\{\boldsymbol{x}' \in \mathscr{X} : \left\|\nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x})\big(\boldsymbol{x}' - \boldsymbol{x}\big)\right\|_p \leq \epsilon_{\mathscr{F}}\right\} \supseteq \left\{\boldsymbol{x}' \in \mathscr{X} : \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \frac{\epsilon_{\mathscr{F}}}{\|\nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x})\|_{\text{op}}}\right\}.$$

First, $\hat{\mathcal{R}}_\Psi(\boldsymbol{x}) = \Psi^{-1}\big(\mathbb{B}_{\mathscr{F},p}(\Psi(\boldsymbol{x}), \epsilon_{\mathscr{F}})\big)$, the pre-image of $\mathbb{B}_{\mathscr{F},p}(\Psi(\boldsymbol{x}), \epsilon_{\mathscr{F}})$ under Ψ. Due to the non-linearity of Ψ, $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ is not necessarily convex, indicating a complicated geometric structure than ℓ_p-ball and may approximate the ground truth imperceptible region better. Hence, we may discover imperceptible adversarial data that the ℓ_p-ball does not contain. Actually, the first-order estimate of $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ can be viewed as weighted ℓ_p-norm by the Jacobian matrix $\nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x})$ that is data-dependent and generally non-isotropic (see Fig. 5 in Appx. C.1 for an illustration). It indicates $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ is adaptive to data, a feature that those attackers (e.g., $\mathcal{N}$attack and AdvFlow) who model $\mathcal{R}_{\boldsymbol{x}}$ as a fixed and simple geometric shape do not enjoy.

Next, we present Prop. 1 which explains the compounding effect of invertible map $\boldsymbol{f}$ and perceptual map Ψ and indicates the importance of both components. The proof is presented in Appx. D.

Proposition 1 *Let $\boldsymbol{f}\colon \mathscr{Z} \to \mathscr{X}$ be a continuous differentiable invertible map whose Jabocian $\nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x})$ is invertible for all $\boldsymbol{x}$, and $\Psi\colon \mathscr{X} \to \mathscr{F}$ be a differentiable perceptual map. For any $\boldsymbol{\delta_z}$ and $\boldsymbol{x}$, write $\boldsymbol{x}' := \boldsymbol{f}\big(\boldsymbol{f}^{-1}(\boldsymbol{x}) + \boldsymbol{\delta_z}\big)$, we have an estimated perturbation in the perceptual space $\Psi(\boldsymbol{x}') - \Psi(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x}) \circ \big(\nabla_{\boldsymbol{x}}\boldsymbol{f}^{-1}(\boldsymbol{x})\big)^{-1}\boldsymbol{\delta_z} + \mathcal{O}(\|\boldsymbol{\delta_z}\|_2^2)$.*

It is noticed that $\boldsymbol{\delta_z}$ and whence $\boldsymbol{x}'$ are implicitly depend on $\nabla_{\boldsymbol{x}}\boldsymbol{f}^{-1}(\boldsymbol{x})\big)^{-1}$ and $\nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x})$. We first explain intuitively the importance of the selection of $\boldsymbol{f}$. Suppose that we have a less expressive $\boldsymbol{f}$, for instance $\tanh$ used in (Li et al., 2019), which Mohaghegh Dolatabadi et al. (2020) show $\tanh$ produces data perturbation only with uncorrelated components. Then the perceptual adversarial example is found with the adjustment of $\nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x})$ based on less informative data perturbation $\big(\nabla_{\boldsymbol{x}}\boldsymbol{f}^{-1}(\boldsymbol{x})\big)^{-1}\boldsymbol{\delta_z}$ (notice the order of composition in Prop. 1). Hence, the resulted adversarial data may loss information of the original data even Ψ may help with perceptual adjustment (see Fig. 3 and Table 4 for empirical supports). In contrast, PadvFlow enjoys an expressive $\boldsymbol{f}$ given by NFs; meanwhile, the search of adversarial examples are leveraged into perceptual space with the adjustment of $\nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x})$, based on informative candidates provided by $\boldsymbol{f}$. Therefore, PadvFlow can find wider range of imperceptible adversarial examples.

6 Conclusion

In this paper, we have proposed PadvFlow, a black-box attack method that balances well across success rate, perceptual quality, and query efficiency. Our method leverages the use of pre-trained flow-based models to learn a distribution of adversarial examples. For imperceptibility, PadvFlow relies on perceptual maps to model the search region. Compared to its counterparts ($\mathcal{N}$attack and AdvFlow), PadvFlow can synthesize more perceptually-natural adversarial examples while maintaining a high success rate. The effectiveness of PadvFlow has been demonstrated in attacking several image classifiers. We also extend PadvFlow to a different modality by showing its high success rate and good perceptual quality in attacking an ASR system.

ETHICS STATEMENT

The proposed PadvFlow is an adversarial attack algorithm, which can be used to mislead classifiers, and ASR systems. PadvFlow might be potentially used in a malicious manner to attack deep learning models. However we claim that PadvFlow has some limitations which may prevent it from being exploited for potential harms. First of all, the effectiveness of attacking commercial products, like AI assistants or automatic driving cars has not been testified. Second, to get a stronger performance of PadvFlow, it relies on some prior information of data (e.g., similar data distribution) to train expressive NFs and perceptual map. As we examine in Sec. 4, missing one of the components of PadvFlow (expressive invertible generative modeling or perceptual map) may degrade its performance. Training NFs and perceptual map are generally expensive and NFs is hard to scale up to higher dimensional space. These requirements are too restrictive and difficult to be fulfilled in real-world scenario.

REPRODUCIBILITY STATEMENT

Code of the proposed method We attach our code as an anonymous supplemental material[3] Alg. 2 in Sec. 2.2 and Appx. A carefully describes the algorithm of PadvFlow. Moreover, Appx. B introduces the construction of invertible generative models and perceptual maps of PadvFlow in details. We also refer to the sources/pre-trained models of each component in PadvFlow.

Details of implementation We carefully explain the implementation details at the first paragraphs of Sec. 3.1 and 3.2. They are supported with explanations of target models and benchmark methods in Appx. B.

Dataset Exploited datasets are all open sources. We introduce them and refer to their sources in Appx. B.1.3.

[3] https://drive.google.com/drive/folders/1VmH4siq4ii40jCkmHeOnRu9nny66jixX?usp=sharing

REFERENCES

Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9:155161–155196, 2021.

Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

Siddhant Bhambri, Sumanyu Muku, Avinash Tulasi, and Arun Balaji Buduru. A survey of black-box adversarial attacks on computer vision models. *arXiv preprint arXiv:1912.01667*, 2019.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Marie Biolková and Bac Nguyen. Neural predictor for black-box adversarial attacks on speech recognition. *arXiv preprint arXiv:2203.09849*, 2022.

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

Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE security and privacy workshops (SPW)*, pp. 1–7. IEEE, 2018.

Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pp. 1277–1294. IEEE, 2020.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26, 2017.

George Corliss. Which root does the bisection algorithm find? *Siam Review*, 19(2):325–327, 1977.

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

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

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

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, 32, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Yu Huang and Yue Chen. Autonomous driving with deep learning: A survey of state-of-art technologies. *arXiv preprint arXiv:2006.06091*, 2020.

Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*, 2018.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. *Advances in neural information processing systems*, 32, 2019.

Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*, 2020.

Peter D Lax and Maria Shea Terrell. *Calculus with applications*. Springer, 2020.

Jinyu Li et al. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *International Conference on Machine Learning*, pp. 3866–3876. PMLR, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Pranay Manocha, Zeyu Jin, Richard Zhang, and Adam Finkelstein. Cdpam: Contrastive learning for perceptual audio similarity. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 196–200. IEEE, 2021.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Hadi Mohaghegh Dolatabadi, Sarah Erfani, and Christopher Leckie. Advflow: Inconspicuous black-box adversarial attacks using normalizing flows. *Advances in Neural Information Processing Systems*, 33:15871–15884, 2020.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Trung Nguyen. The convergence of the regula falsi method. *arXiv preprint arXiv:2109.03523*, 2021.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pp. 2642–2651. PMLR, 2017.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning*, pp. 5231–5240. PMLR, 2019.

Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, et al. Speechbrain: A general-purpose speech toolkit. *arXiv preprint arXiv:2106.04624*, 2021.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, volume 2, pp. 749–752. IEEE, 2001.

Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *arXiv preprint arXiv:1808.05665*, 2018.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. 2018.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *Advances in Neural Information Processing Systems*, 31, 2018.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Naoya Takahashi, Shota Inoue, and Yuki Mitsufuji. Adversarial attacks on audio source separation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 521–525. IEEE, 2021.

Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 742–749, 2019.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020a.

Qiuling Xu, Guanhong Tao, Siyuan Cheng, and Xiangyu Zhang. Towards feature space adversarial attack. *arXiv preprint arXiv:2004.12385*, 2020b.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2403–2412, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 103–117, 2017.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

A SUPPLEMENTAL EXPLANATIONS TO ALGORITHM

In Appx. A.1 we introduce the design of $\mathcal{T}_h$ depending on different modalities. In Appx. A.2 and Appx. A.3, we explain in details about NES for optimization and bisection algorithm for constructing projection to approximated perceptual region, respectively. We introduce two variants algorithms of PadvFlow which extends to higher resolution images in Appx. A.5.

A.1 DETAILS OF CONSTRUCTION OF $\mathcal{T}_h$ FOR TWO DIFFERENT MODALITIES

The design of operator $\mathcal{T}_h$ is task-dependent. We introduce its construction for an image classifier and an ASR system in Appx. A.1.

Image classification In this case, h outputs a vector $h(\boldsymbol{x}) \in \mathbb{R}^k$, which indicates the probabilities of a given image $\boldsymbol{x}$ belonging to each of the k categorical classes. Let $\boldsymbol{y} \in \{0,1\}^k$ be the one-hot vector that representing a specific label, $\mathcal{T}_h$ can be defined as: $\mathcal{T}_h(\boldsymbol{x}, \boldsymbol{y}) = h(\boldsymbol{x})^\top \boldsymbol{y}$.

Automatic speech recognition Given a speech feature sequence $\boldsymbol{x}$ of length T, the black-box ASR system h outputs a matrix $h(\boldsymbol{x}) \in \mathbb{R}^{L\times T}$ of probabilities over an alphabet Γ of L tokens. The i-th column vector $h(\boldsymbol{x})^{(i)}$ denotes the probability distribution of the i-th input over Γ. $\mathcal{T}_h$ is characterized by computing the probability of assigning an input audio $\boldsymbol{x}$ to its correct transcript $\boldsymbol{y}$ using the Connectionist Temporal Classification (CTC) (Graves et al., 2006; Carlini & Wagner, 2018). More precisely, it is defined as: $\mathcal{T}_h(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\forall \boldsymbol{a}\in\mathcal{A}(\boldsymbol{x},\boldsymbol{y})} \prod_i h(\boldsymbol{x})^{(i)}_{a_i}$, where $\boldsymbol{a} \in \Gamma^T$ is an alignment of $\boldsymbol{x}$ with respect to $\boldsymbol{y}$ from a set of valid alignments $\mathcal{A}(\boldsymbol{x}, \boldsymbol{y})$ as in (Graves et al., 2006). $\mathcal{T}_h(\boldsymbol{x}, \boldsymbol{y})$ can be computed efficiently using the forward-backward algorithm (Graves et al., 2006).

A.2 DETAILS OF NES

Here we denote a general objective function as $\mathcal{L}$ and let $\{\pi(\boldsymbol{x}'|\boldsymbol{\theta})\}_{\boldsymbol{\theta}}$ be a variational family with parameters $\boldsymbol{\theta}$. NES looks for $\boldsymbol{\theta}$ to make the objective small in expectation.

$$\min_{\boldsymbol{\theta}_{\boldsymbol{x}}} \mathcal{J}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{x}'\sim\pi(\boldsymbol{x}'|\boldsymbol{\theta})} \left[\mathcal{L}(\boldsymbol{x}')\right]$$

By adjusting $\boldsymbol{\theta}$, the distribution can learn to capture the local information of $\mathcal{L}$. The search of $\boldsymbol{\theta}$ is fulfilled by gradient descent. We can apply the so-called "log-trick" to compute the gradient in $\boldsymbol{\theta}$:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \int \mathcal{L}(\boldsymbol{x}')\pi(\boldsymbol{x}'|\boldsymbol{\theta})d\boldsymbol{x}' \\
&= \int \mathcal{L}(\boldsymbol{x}')\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{x}'|\boldsymbol{\theta})d\boldsymbol{x}' \\
&= \int \mathcal{L}(\boldsymbol{x}')\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{x}'|\boldsymbol{\theta}) \cdot \frac{\pi(\boldsymbol{x}'|\boldsymbol{\theta})}{\pi(\boldsymbol{x}'|\boldsymbol{\theta})}d\boldsymbol{x}' \\
&= \int \Big[\mathcal{L}(\boldsymbol{x}')\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{x}'|\boldsymbol{\theta})\Big] \cdot \pi(\boldsymbol{x}'|\boldsymbol{\theta})d\boldsymbol{x}' \\
&= \mathbb{E}_{\boldsymbol{x}'\sim\pi(\boldsymbol{x}'|\boldsymbol{\theta})} \left[\mathcal{L}(\boldsymbol{x}')\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{x}'|\boldsymbol{\theta})\right].
\end{aligned}$$

When $\{\pi(\boldsymbol{x}'|\boldsymbol{\theta}) : \boldsymbol{\theta}\}$ is a family of normal distribution, that is,

$$\pi(\boldsymbol{x}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}' - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}' - \boldsymbol{\mu})\Big)$$

the gradients $\nabla_{\boldsymbol{\mu}} \log \pi(\boldsymbol{x}'|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\nabla_{\boldsymbol{\Sigma}} \log \pi(\boldsymbol{x}'|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be compute analytically as

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}} \log \pi(\boldsymbol{x}'|\boldsymbol{\mu}) &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}' - \boldsymbol{\mu}) \\
\nabla_{\boldsymbol{\Sigma}} \log \pi(\boldsymbol{x}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \frac{1}{2}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}' - \boldsymbol{\mu})(\boldsymbol{x}' - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} - \frac{1}{2}\boldsymbol{\Sigma}^{-1}.
\end{aligned}$$

A.3 DETAILS OF BISECTION METHOD FOR CONSTRUCTING PROJECTION ONTO $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$

Algorithm 1 Bisection Method

Input: Real-valued function $\gamma(t)$; endpoints $a < b$; tolerance threshold τ; max. iteration T
Ensure: $\gamma(a)\gamma(b) < 0$
Output: A point c close to a root of $\gamma(t) = 0$ within the tolerance threshold τ

$k \leftarrow 0$
while $k < T$ **do**
  $c \leftarrow \frac{a+b}{2}$
  if $\gamma(c) = 0$ or $\frac{b-a}{2} < \tau$ **then**
    Output c
    end procedure
  end if
  $k \leftarrow k + 1$
  if $\text{sign}(\gamma(c)) = \text{sign}(\gamma(a))$ **then**
    $a \leftarrow c$
  else
    $b \leftarrow c$
  end if
end while

The projection of a point $\boldsymbol{x}' \notin \hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ onto $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ is constructed by searching another point $\boldsymbol{x}'_{\text{bdry}}$ that lies on the boundary of $\hat{\mathcal{R}}_\Psi(\boldsymbol{x})$ by solving a root $t^* \in [0, 1]$ for the equation $\gamma(t) = 0$, where $\gamma(t) := d_\Psi\big(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x}), \boldsymbol{x}\big) - \epsilon_{\mathscr{F}}$. To be mentioned, since the gradient of Ψ is often unavailable, a "derivative-free" root solver is preferred. In Alg. 1, we review bisection method, a commonly used "derivative-free" root solver. We remark that other root solver can also be applied, for instance, regula falsi (false position) method (Nguyen, 2021).

The solvability is ensured by Intermediate Value Theorem (Rudin et al., 1976), which we formulate the following lemma. We remark that there might be multiple solutions but we just need to find one.

Lemma 1 *Let $\boldsymbol{x}' \notin \hat{\mathcal{R}}_\Psi(\boldsymbol{x})$, then there is a $t^* \in [0, 1]$ so that $\boldsymbol{x}'_{bdry} := \boldsymbol{x} + t^*(\boldsymbol{x}' - \boldsymbol{x})$ satisfies $\Big\|\Psi(\boldsymbol{x}'_{bdry}) - \Psi(\boldsymbol{x})\Big\|_p = \epsilon_{\mathscr{F}}$.*

Additionally, we provide a more general proposition (Prop. 2) which insures the existence of an intersect $\boldsymbol{x}'_{\text{bdry}}$ for any continuous curve (not necessary γ) connecting points which are respectively residing outside and inside a general region $\mathcal{R}$. The proof is presented in Appx. D. Intuitively, the proposition says no matter what the region $\mathcal{R}$ is, any continuous path connecting two points which are respectively lie inside and outside of $\mathcal{R}$ should intersect at some point on the boundary of $\mathcal{R}$.

Proposition 2 *Let $\mathcal{R} \subset \mathbb{R}^D$ be a nonempty set. Denote the interior, exterior, and boundary of $\mathcal{R}$ as* $\text{int}(\mathcal{R})$, $\text{ext}(\mathcal{R})$, *and $\partial\mathcal{R}$, respectively. If $\boldsymbol{x}_0 \in \text{int}(\mathcal{R})$ and $\boldsymbol{x}_1 \in \text{ext}(\mathcal{R})$, then any continuous curve connecting $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ must intersect with a point on $\partial\mathcal{R}$.*

This proposition can guarantee the existence of projected point on the boundary for a general construction of continuous projection.

A.4 Algorithm of PadvFlow

Alg. 2 presents the algorithm of PadvFlow. We remark that in Line 10, we follow (Li et al., 2019) to standardize the objective functions to stabilize the training.

A.5 Details of Extending PadvFlow to ImageNet

Since the training of NFs is difficult to scale up to dataset in higher dimensional space, we utilize the upsampling and downsampling technique as in (Mohaghegh Dolatabadi et al., 2020) to extend PadvFlow for higher resolution images. The idea is rather having a pre-trained normalizing flow on the original data space $\mathscr{X} \subset \mathbb{R}^D$, we downsample the input $\boldsymbol{x}$ by a transform $\mathcal{M}_{\text{down}}$ into a

Algorithm 2 PadvFlow: learn imperceptible adversarial distributions

Input: $(\boldsymbol{x}, \boldsymbol{y})$ clean data and ground truth label; pre-trained NF $\boldsymbol{f}$ (or invertible map); target model $\mathcal{T}_h$; loss function $\mathcal{L}(\cdot, \boldsymbol{y}; \mathcal{T}_h)$; pre-trained perceptual map Ψ; perceptual radius $\epsilon_{\mathscr{F}}$; population size M; max. iteration; learning rate η; standard deviation (std.) of the base distribution σ

Output: Mean $\boldsymbol{\mu}_{\boldsymbol{x}}$ and an imperceptible adversarial distribution sampled as $\boldsymbol{x}' = \text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}\big(\boldsymbol{f}(\boldsymbol{f}^{-1}(\boldsymbol{x}) + \boldsymbol{z})\big), \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \sigma^2 \boldsymbol{I})$

Initialize $\boldsymbol{\mu}_{\boldsymbol{x}}$
Set $\boldsymbol{z}_{\text{clean}} = \boldsymbol{f}^{-1}(\boldsymbol{x})$.
for steps in the max. iteration range **do**
  for $k = 1, \cdots, M$ **do**
    Sample $\boldsymbol{\delta}_k \leftarrow \boldsymbol{\mu}_{\boldsymbol{x}} + \sigma \boldsymbol{\epsilon}_k$, $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    $\boldsymbol{x}'_k \leftarrow \text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}\Big(f(\boldsymbol{z}_{\text{clean}} + \boldsymbol{\delta}_k)\Big)$
    $\mathcal{L}_k \leftarrow \mathcal{L}(\boldsymbol{x}'_k, \boldsymbol{y}; \mathcal{T}_h)$
  end for
  $m, s \leftarrow \text{mean}(\mathcal{L}_1, \cdots \mathcal{L}_M), \text{std}(\mathcal{L}_1, \cdots, \mathcal{L}_M)$
  Update $\boldsymbol{\mu}_{\boldsymbol{x}} \leftarrow \boldsymbol{\mu}_{\boldsymbol{x}} - \frac{\eta}{\sigma M} \sum_{k=1}^{M} \frac{\mathcal{L}_k - m}{s} \boldsymbol{\epsilon}_k$
end for

lower dimensional space $\boldsymbol{x}_{\text{down}} \in \mathscr{X}_{\text{down}} \subset \mathbb{R}^d$ with $d < D$ and search the parameters for the latent distribution in the lower dimensional latent space $\mathscr{Z}_{\text{down}} \subset \mathbb{R}^d$. The pre-trained NFs $\boldsymbol{f}$ is then to relate $\mathscr{Z}_{\text{down}}$ and $\mathscr{X}_{\text{down}}$, residing in $\mathbb{R}^d$. We can utilize upsampling map $\mathcal{M}_{\text{up}}$ to bring from $\mathscr{X}_{\text{down}}$ back to $\mathscr{X}$.

This general idea can be exploited in two ways, depending on the accessibility of perceptual map on (H1) the downsampled space $\Psi_{\text{down}} \colon \mathscr{X}_{\text{down}} \to \mathscr{F}_{\text{down}}$, or (H2) the original $\Psi \colon \mathscr{X} \to \mathscr{F}$. Let $\boldsymbol{z}_{\text{clean}} = \boldsymbol{f}^{-1}(\boldsymbol{x}_{\text{down}}) \in \mathbb{R}^d$. PadvFlow-(H1) is an extension of Algorithm 3 in AdvFlow, where we search imperceptible adversarial candidate $\boldsymbol{x}'_{\text{down}}$ in the lower dimensional space

$$\boldsymbol{x}'_{\text{down}} := \text{proj}_{\hat{\mathcal{R}}_{\Psi_{\text{down}}}(\boldsymbol{x})}\big(f(\boldsymbol{z}_{\text{clean}} + \boldsymbol{\delta}_{\text{down}})\big),$$

where $\boldsymbol{\delta}_{\text{down}} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_{d \times d})$ and $\boldsymbol{\mu}$ is a learnable parameter. Then we upsample the difference of $\boldsymbol{x}'_{\text{down}}$ and $\boldsymbol{x}_{\text{down}}$ and treat it as an adversarial perturbation in the original space $\mathscr{X}$. Thus, the perceptible candidate $\boldsymbol{x}'$ in $\mathscr{X}$ is considered as

$$\boldsymbol{x}' = \text{proj}_{\mathscr{X}}\Big(\boldsymbol{x} + \mathcal{M}_{\text{up}}\big(\boldsymbol{x}'_{\text{down}} - \boldsymbol{x}_{\text{down}}\big)\Big),$$

where $\text{proj}_{\mathscr{X}}$ indicates the projection to valid image space; namely, $[0, 1]^D$.

On the other hand, we propose PadvFlow-(H2), a variant of PadvFlow-(H1), where adversarial candidates $f(\boldsymbol{z}_{\text{clean}} + \boldsymbol{\delta}_{\text{down}})$ are first promoted in the lower dimensional space $\mathscr{X}_{\text{down}}$. Then we upsample them back to the original input space $\mathscr{X}$ and adjust them with the projection to approximated imperceptible region $\text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}$. That is,

$$\boldsymbol{x}' = \text{proj}_{\hat{\mathcal{R}}_\Psi(\boldsymbol{x})}\Big(\mathcal{M}_{\text{up}}\big(f(\boldsymbol{z}_{\text{clean}} + \boldsymbol{\delta}_{\text{down}})\big)\Big).$$

In our implementation of ImageNet (e.g., Table 2), we utilize PadvFlow-(H2) due to the easy accessibility to the pre-trained perceptual map[4]. We summarize the algorithms of respective cases in Alg. 3 and Alg. 4. The differences in contrast to Alg. 2 are marked in colors of magenta and cyan, respectively.

B Details of Implementation

In Appx. B.1, we explain the details of implementation when it is a image classification problem. It includes the throughout review of defended image classifiers (in Appx. B.1.1), benchmark methods (in Appx. B.1.2, image (in datasets B.1.3), and additional implementation details of PadvFlow (in

[4] https://pytorch.org/hub/pytorch_vision_alexnet/

Algorithm 3 PadvFlow-(H1): PadvFlow for high resolution images. It upsamples perturbation (between candidates and $\boldsymbol{x}_{\text{down}}$) found in $\mathscr{X}_{\text{down}}$

Input: $(\boldsymbol{x}, \boldsymbol{y})$ clean data and its ground truth label; pre-trained NF $\boldsymbol{f}\colon \mathscr{Z}_{\text{down}} \to \mathscr{X}_{\text{down}}$; target model $\mathcal{T}_h$; loss function $\mathcal{L}(\cdot, \boldsymbol{y}; \mathcal{T}_h)$; pre-trained perceptual map $\Psi_{\text{down}}\colon \mathscr{X}_{\text{down}} \to \mathscr{F}_{\text{down}}$ defined on $\mathscr{X}_{\text{down}}$ and it corresponds to a lower dimensional approximated imperceptible region $\hat{\mathcal{R}}_{\Psi_{\text{down}}} \subset \mathbb{R}^d$; perceptual radius $\epsilon_{\mathscr{F}}$; population size M; max. iteration; learning rate η; standard deviation (std.) of the base distribution σ

Output: Learned $\boldsymbol{\mu}_{\text{down}}$ and an imperceptible adversarial distribution sampled as follows. First, sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{down}}, \sigma^2 \boldsymbol{I}_{d\times d})$ and define $\delta_{\text{down}} := \text{proj}_{\hat{\mathcal{R}}_{\Psi_{\text{down}}}(\boldsymbol{x})}\Big(\boldsymbol{f}\big(\boldsymbol{f}^{-1}(\boldsymbol{x}_{\text{down}}) + \boldsymbol{z}\big)\Big) - \boldsymbol{x}_{\text{down}}$. A sample following imperceptible adversarial distribution is given by $\boldsymbol{x}' = \text{proj}_{\mathscr{X}}\big(\boldsymbol{x} + \mathcal{M}_{\text{up}}(\delta_{\text{down}})\big)$

Initialize $\boldsymbol{\mu}_{\text{down}} \in \mathbb{R}^d$
$\boldsymbol{x}_{\text{down}} \leftarrow \mathcal{M}_{\text{down}}(\boldsymbol{x})$
Set $\boldsymbol{z}_{\text{clean}} = \boldsymbol{f}^{-1}(\boldsymbol{x}_{\text{down}}) \in \mathbb{R}^d$
for steps in the max. iteration range **do**
  for $k = 1, \cdots, M$ **do**
    Sample $\boldsymbol{\delta}_{\text{down},k} \leftarrow \boldsymbol{\mu}_{\text{down}} + \sigma\boldsymbol{\epsilon}_k$, $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d\times d})$
    $\boldsymbol{x}'_{\text{down},k} \leftarrow \text{proj}_{\hat{\mathcal{R}}_{\Psi_{\text{down}}}(\boldsymbol{x})}\Big(f(\boldsymbol{z}_{\text{clean}} + \boldsymbol{\delta}_{\text{down},k})\Big)$
    $\boldsymbol{x}'_k \leftarrow \text{proj}_{\mathscr{X}}\Big(\boldsymbol{x} + \mathcal{M}_{\text{up}}\big(\boldsymbol{x}'_{\text{down},k} - \boldsymbol{x}_{\text{down}}\big)\Big)$
    $\mathcal{L}_k \leftarrow \mathcal{L}(\boldsymbol{x}'_k, y; \mathcal{T}_h)$
  end for
  $m, s \leftarrow \text{mean}(\mathcal{L}_1, \cdots \mathcal{L}_M), \text{std}(\mathcal{L}_1, \cdots, \mathcal{L}_M)$
  Update $\boldsymbol{\mu}_{\text{down}} \leftarrow \boldsymbol{\mu}_{\text{down}} - \frac{\eta}{\sigma M}\sum_{k=1}^{M}\frac{\mathcal{L}_k - m}{s}\boldsymbol{\epsilon}_k$
end for

Algorithm 4 PadvFlow-(H2): PadvFlow for high resolution images. It upsamples candidates in $\mathscr{X}_{\text{down}}$

Input: $(\boldsymbol{x}, \boldsymbol{y})$ clean data and its ground truth label; pre-trained NF $\boldsymbol{f}\colon \mathscr{Z}_{\text{down}} \to \mathscr{X}_{\text{down}}$; target model $\mathcal{T}_h$; target model $\mathcal{T}_h$; loss function $\mathcal{L}(\cdot, \boldsymbol{y}; \mathcal{T}_h)$; pre-trained perceptual map $\Psi\colon \mathscr{X} \to \mathscr{F}$ which corresponds to the approximated imperceptible region $\hat{\mathcal{R}}_{\Psi} \subset \mathbb{R}^D$; perceptual radius $\epsilon_{\mathscr{F}}$; population size M; max. iteration; learning rate η; standard deviation (std.) of the base distribution σ

Output: Learned $\boldsymbol{\mu}_{\text{down}}$ and an imperceptible adversarial distribution sampled as $\boldsymbol{x}' = \text{proj}_{\hat{\mathcal{R}}_{\Psi}(\boldsymbol{x})}\Big(\mathcal{M}_{\text{up}}\big(\boldsymbol{f}(\boldsymbol{f}^{-1}(\boldsymbol{x}_{\text{down}}) + \boldsymbol{z})\big)\Big)$, where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{down}}, \sigma^2 \boldsymbol{I}_{d\times d})$.

Initialize $\boldsymbol{\mu}_{\text{down}} \in \mathbb{R}^d$
$\boldsymbol{x}_{\text{down}} \leftarrow \mathcal{M}_{\text{down}}(\boldsymbol{x})$
Set $\boldsymbol{z}_{\text{clean}} = \boldsymbol{f}^{-1}(\boldsymbol{x}_{\text{down}}) \in \mathbb{R}^d$
for steps in the max. iteration range **do**
  for $k = 1, \cdots, M$ **do**
    Sample $\boldsymbol{\delta}_{\text{down},k} \leftarrow \boldsymbol{\mu}_{\text{down}} + \sigma\boldsymbol{\epsilon}_k$, $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d\times d})$
    $\boldsymbol{x}'_k \leftarrow \text{proj}_{\hat{\mathcal{R}}_{\Psi}(\boldsymbol{x})}\Big(\mathcal{M}_{\text{up}}\big(f(\boldsymbol{z}_{\text{clean}} + \boldsymbol{\delta}_{\text{down},k})\big)\Big)$
    $\mathcal{L}_k \leftarrow \mathcal{L}(\boldsymbol{x}'_k, y; \mathcal{T}_h)$
  end for
  $m, s \leftarrow \text{mean}(\mathcal{L}_1, \cdots \mathcal{L}_M), \text{std}(\mathcal{L}_1, \cdots, \mathcal{L}_M)$
  Update $\boldsymbol{\mu}_{\text{down}} \leftarrow \boldsymbol{\mu}_{\text{down}} - \frac{\eta}{\sigma M}\sum_{k=1}^{M}\frac{\mathcal{L}_k - m}{s}\boldsymbol{\epsilon}_k$
end for

Appx. B.1.4). On the other hand, in Appx. B.2, we explain the implementation when the modality is a ASR problem. Last in Appx. B.3, we summarize the selection of thresholds which we use in Sec. 4.2.

B.1 IMPLEMENTATION OF IMAGE CLASSIFICATION

We first review the exploited defended classifiers and we explain the implementation details when $\mathcal{T}_h =$ image classifier.

B.1.1 REVIEW OF DEFENDED CLASSIFIERS

Let $h_{\boldsymbol{\theta}}$ be a target classifier with trainable parameters $\boldsymbol{\theta}$ and ℓ be a loss function for a classification problem, for instance, Cross-entropy loss. A defended classifier is obtained by the adversarial training (Madry et al., 2017; Bai et al., 2021) which learns (outer minimization problem) to be robust to adversarial perturbation (inner maximization problem)

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \max_{\boldsymbol{x}'\in\mathcal{R}_{\boldsymbol{x}}} \ell\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}'), y\big), \tag{3}$$

where $\mathcal{D}$ is data-label distribution. Typically, adversarial training algorithms model $\mathcal{R}_{\boldsymbol{x}}$ as $\{\boldsymbol{x}' \in \mathscr{X} : \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \varepsilon\}$ for a small threshold ε, in which our three exploited defended classifiers Free (Shafahi et al., 2019), Fast (Shafahi et al., 2019), and RotNet (Hendrycks et al., 2019) fall into this category. On the other hand, a current method PAT (Laidlaw et al., 2020) models $\mathcal{R}_{\boldsymbol{x}}$ as

$$\hat{\mathcal{R}}_{\xi_{\boldsymbol{\phi}}}(\boldsymbol{x}) := \left\{\boldsymbol{x}' \in \mathscr{X} : \|\xi_{\boldsymbol{\phi}}(\boldsymbol{x}') - \xi_{\boldsymbol{\phi}}(\boldsymbol{x})\|_2 \leq \varepsilon'\right\}, \tag{4}$$

where $\xi_{\boldsymbol{\phi}}$ is a pre-trained or trainable perceptual map (with parameters $\boldsymbol{\phi}$) induced by a classifier (see Appx. B.1.4 for its construction) and ε' is a threshold (may be different from ε). Below, we are going to review the four defended methods.

Free (Shafahi et al., 2019) The inner maximization is typically solved via multiple update steps of Projected Gradient Descent (PGD) (Madry et al., 2017) to find potential adversarial candidates, then the outer minimization updates the model parameters on those adversarial samples. Both the inner and outer loop require forward-backward passes of the network, which is expensive. Instead, Shafahi et al. (2019) proposes free adversarial training to update both the model parameters and data perturbations using one simultaneous backward pass which can reduce runtimes while remain comparable performance as (Madry et al., 2017).

Fast (Wong et al., 2020) Fast gradient sign method (FGSM) (Goodfellow et al., 2014), which can be viewed as a simplified version PGD with a single update step, is a faster algorithm to solve the inner maximization problem while had been considering to be less effective. (Wong et al., 2020) improve FGSM with a random initialization and accelerate its with standard tricks for efficient training of DL models (e.g., cyclic learning rates (Smith & Topin, 2018), mixed-precision arithmetic (Micikevicius et al., 2017)).

RotNet (Hendrycks et al., 2019) They introduce an auxiliary rotation-based self-supervision (Gidaris et al., 2018) which can learn representations of data to increase the robustness of models. They train classifiers together with a separate auxiliary head to predict the degree of rotation applied to a given input image.

PAT (Laidlaw et al., 2020) They utilize a neural network-based approximation to imperceptible adversarial region as Eq. 4. The classifier induced the perceptual map $\xi_{\boldsymbol{\phi}}$ can be identical to the target model $h_{\boldsymbol{\theta}}$. The corresponding inner maximization problem can be approximately solved via their proposed Projected Perceptual Gradient Descent (PPGD) and Lagrangian Perceptual Attack (LPA). PPGD regularizes with the first-order Taylor estimation of $\hat{\ell}_{\boldsymbol{\theta}}(\boldsymbol{x}) := \ell\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\big)$ to encourage the projection onto $\hat{\mathcal{R}}_{\xi_{\boldsymbol{\phi}}}(\boldsymbol{x})$. Thus, the inner maximization problem becomes

$$\max_{\boldsymbol{\delta}:\|\nabla_{\boldsymbol{x}}\xi_{\boldsymbol{\phi}}(\boldsymbol{x})\cdot\boldsymbol{\delta}\|_2\leq\eta} \Big[\hat{\ell}_{\boldsymbol{\theta}}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}(\hat{\ell}_{\boldsymbol{\theta}})^\top(\boldsymbol{x})\boldsymbol{\delta}\Big],$$

for a small $\eta > 0$. In contrast, LPA utilizes the trick of Lagrange multipliers to relax the expensive computation of Jacobian in PPGD. The inner maximization problem turns into

$$\max_{\boldsymbol{x}'} \Big[\hat{\ell}_{\boldsymbol{\theta}}(\boldsymbol{x}') - \lambda \max\big(0, \|\xi_{\boldsymbol{\phi}}(\boldsymbol{x}') - \xi_{\boldsymbol{\phi}}(\boldsymbol{x})\|_2 - \varepsilon'\big)\Big],$$

where λ is a hyper-parameter. We remark that they both require the gradient information of $\hat{\ell}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $h_{\boldsymbol{\theta}}(\boldsymbol{x})$.

B.1.2 BENCHMARK METHODS

We implement AdvFlow by following the official repository[5] and we modify the invertible generative function $\boldsymbol{f}$ from NFs to $\tanh$ for $\mathcal{N}$attack. For Bandits, we utilize their repository[6] but follow the setup of hyper-parameters described in AdvFlow. We follow the same setup of hyper-parameters of the official repository of AutoAttack[7] with `version = 'standard'`. For HSJA, we consult with their official repository[8] and modify it to PyTorch with a reference[9].

B.1.3 REVIEW OF IMAGE DATASETS

We review and refer to sources of image datasets we test on in this section.

CIFAR-10 [10] contains ten categories of natural images of pixel size 32×32 in RGB scale. We utilize its test set (in total $10,000$ instances) or random subset for evaluation. The source is from PyTorch library.

SVHN [11] contains ten classes of colour digits from house numbers, which is of pixel size 32×32. We utilize its test set (in total $26,032$ instances) or random subset for evaluation. The source is also from PyTorch library.

ImageNet [12] is a large scale dataset contains 1000 classes of natural images in RGB scale. We utilize $10,000$ random samples from its test set for tasks with undefended classifiers. On the other hand, we utilize ImageNet-100, a subset of ImageNet which contains 100 classes for defended classifiers, RN50-PAT (Laidlaw et al., 2020) due to its easy accessibility of pre-trained defended classifier.

B.1.4 IMPLEMENTATION

Invertible mapping For a fair comparison, we utilize the invertible generative model as RealNVP (Dinh et al., 2016) as AdvFlow. The pre-trained models on CIFAR-10 and SVHN can be obtained from the source of AdvFlow repository[13].

Perceptual metric We exploit LPIPS (Zhang et al., 2018) as the perceptual distance Ψ in PadvFlow. Consider a pre-trained classifier $g : \mathscr{X} \to \mathscr{P}$ (refer to notations in Sec. 2.1) which has L feature layers. Let $\boldsymbol{x}'$ and $\boldsymbol{x}$ be a pair of images and g_l be the l-th feature layer of g ($1 \leq l \leq L$). (Zhang et al., 2018; Laidlaw et al., 2020) construct a perceptual feature map as follows. They first extracts features of l-th layer and normalize with ℓ_2 norm with respect to the channel dimension, denoted it as $\hat{g}_l(\boldsymbol{x}')$ and $\hat{g}_l(\boldsymbol{x})$. Then they further normalize each with width w_l and height h_l in l-th layer and stack all layers into a vector to get the perceptual feature map

$$\Psi \colon \boldsymbol{x} \mapsto \big(\frac{\hat{g}_1(\boldsymbol{x})}{\sqrt{w_1 h_1}}, \cdots, \frac{\hat{g}_L(\boldsymbol{x})}{\sqrt{w_L h_L}}\big).$$

The LPIPS of $\boldsymbol{x}'$ and $\boldsymbol{x}$ is then defined as

$$d_\Psi(\boldsymbol{x}', \boldsymbol{x}) := \|\Psi(\boldsymbol{x}') - \Psi(\boldsymbol{x})\|_2 .$$

In PadvFlow for image classification, we take AlexNet (Krizhevsky et al., 2009) as the pre-trained classifier for the feature extraction.

Target classifiers We introduce the target classifiers we test on and summarize their sources in Table 5.

[5] https://github.com/hmdolatabadi/AdvFlow

[6] https://github.com/MadryLab/blackbox-bandits

[7] https://github.com/fra31/auto-attack

[8] https://github.com/Jianbo-Lab/HSJA

[9] https://github.com/I-am-Bot/Black-Box-Attacks

[10] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[11] https://pytorch.org/vision/stable/generated/torchvision.datasets.SVHN.html

[12] https://www.image-net.org/

[13] https://drive.google.com/file/d/18J8eh-KLaPq9vUe_TwhuQMBW4WKBVX0L/view

- CIFAR-10/SVHN: We consider six undefended (VGG19, GoogLeNet, MobileNetV2, DLA, DenseNet121 and WRN34) and four defended models (WRN34-Free, WRN34-Fast, WRN34-RotNet and RN50-PAT). We train the the first five undefended classifiers by following the implementation of the repository[14]. On the other hand, we utilize the pre-trained classifiers or follow the setup to train classifiers provided by AdvFlow[15].
- ImageNet: We consider three undefended models (InceptionV3, VGG16, RN50) that the pre-trained models are publicly available on PyTorch Hub. For the defended model (RN50-PAT), we utilize the model as (Laidlaw et al., 2020).

Table 5: Sources of image classifiers on CIFAR-10, SVHN and ImageNet

Type	CIFAR-10/SVHN classifiers	Source
Undefended	VGG19 (Simonyan & Zisserman, 2014)	https://github.com/kuangliu/pytorch-cifar
	GoogLeNet (Szegedy et al., 2015)	〃
	MobileNetV2 (Sandler et al., 2018)	〃
	DLA (Yu et al., 2018)	〃
	DenseNet121 (Huang et al., 2017)	〃
	WRN34 (Zagoruyko & Komodakis, 2016)	https://github.com/hmdolatabadi/AdvFlow
Defended	WRN34-Free (Shafahi et al., 2019)	https://github.com/mahyarnajibi/FreeAdversarialTraining
	WRN34-Fast (Wong et al., 2020)	https://github.com/locuslab/fast_adversarial
	WRN34-RotNet (Hendrycks et al., 2019)	https://github.com/hendrycks/ss-ood
	RN50-PAT (Laidlaw et al., 2020)	https://github.com/cassidylaidlaw/perceptual-advex

Type	ImageNet classifiers	Source
Undefended	InceptionV3 (Szegedy et al., 2016)	https://pytorch.org/hub/pytorch_vision_inception_v3/
	VGG16 (Simonyan & Zisserman, 2014)	https://pytorch.org/hub/pytorch_vision_vgg/
	RN50 (He et al., 2016)	https://pytorch.org/hub/pytorch_vision_resnet/
Defended	RN50-PAT (Laidlaw et al., 2020)	https://github.com/cassidylaidlaw/perceptual-advex

B.2 Implementation of Automatic Speech Recognition

Invertible mapping Our invertible generative model is WaveGlow (Prenger et al., 2019), a neural vocoder based on a flow neural network[16]. Originally, it was developed for synthesizing speech conditioned on mel-spectrograms. We remove the conditioning blocks to perform unconditioned generation. In particular, the network consists of 12 coupling layers and 12 invertible 1×1 convolutions. Each coupling layer has 8 layers of dilated convolutions with 256 channels. Considering the time complexity, we only train this model for 500 epochs on the LibriSpeech train-clean-360 dataset using the Adam optimizer with a learning rate of 0.0001. All audio recordings were sampled at a rate of 16kHz.

Target ASR model We target an end-to-end ASR system[17] from SpeechBrain (Ravanelli et al., 2021) trained on the full LibriSpeech (960 hours) with the CTC loss (Graves et al., 2006). Internally, the system employs an encode, decoder, and an attention mechanism. The network consists of several convolutional, recurrent, and fully-connected layers. At every frame, the network outputs a probability distribution over all tokens. Byte Pairwise Encoding (Gage, 1994) was used to extract the basic recognition tokens.

Perceptual metric To judge how similar two audio recording are, CDPAM (Manocha et al., 2021) was employed[18]. The model uses a neural network to approximate the human perception of audio quality using contrastive learning combined with multi-dimensional representation learning. Since CDPAM was trained based on human judgments, it correlates quite well with human perception.

[14] https://github.com/kuangliu/pytorch-cifar

[15] https://github.com/hmdolatabadi/AdvFlow

[16] An implementation available at https://github.com/NVIDIA/waveglow

[17] The pre-trained model available at https://huggingface.co/speechbrain/asr-crdnn-rnnlm-librispeech.

[18] An implementation available at https://github.com/pranaymanocha/PerceptualAudio/tree/master/cdpam

B.3 SELECTION OF THRESHOLDS IN SEC. 4

In Table 6, we summarize the approximated thresholds $\epsilon_{\mathscr{F}}$ of PadvFlow and $\epsilon_{\mathscr{X}}$ AdvFlow to achieve target success rates in Sec. 4.2. The estimated thresholds are obtained via binary search.

Table 6: The corresponding approximated thresholds $\epsilon_{\mathscr{F}}$ of PadvFlow and $\epsilon_{\mathscr{X}}$ AdvFlow to achieve an assigned success rates. We consider two defended classifiers, WRN34-Free and WRN50-PAT, trained on CIFAR-10.

Dataset	Assigned success rate	87%	80%	70%	60%	50%	40%
WRN34-Free	Approximated $\epsilon_{\mathscr{F}}$	0.5	0.42	0.36	0.31	0.275	0.23
	Approximated $\epsilon_{\mathscr{X}}$	$\frac{17.85}{255}$	$\frac{16.653}{255}$	$\frac{13.769}{255}$	$\frac{11.6}{255}$	$\frac{9.5}{255}$	$\frac{8}{255}$
Dataset	Assigned success rate	73%	70%	60%	50%	40%	
WRN50-PAT	Approximated $\epsilon_{\mathscr{F}}$	0.5	0.476	0.41	0.36	0.295	
	Approximated $\epsilon_{\mathscr{X}}$	$\frac{15.2}{255}$	$\frac{14.469}{255}$	$\frac{12.5}{255}$	$\frac{10.46}{255}$	$\frac{8}{255}$	

C ADDITIONAL DEMONSTRATION

In Appx. C.1, we illustrate the histograms of singular values of $D\Psi(\boldsymbol{x})$ for two different clean image $\boldsymbol{x}$'s when $\mathcal{T}_h$ is a classifier. In Appx. C.2, we illustrate spectrograms of generated samples and magnified perturbation by PadvFlow and comparison methods, $\mathcal{N}$attack and AdvFlow.

C.1 ILLUSTRATION OF SINGULAR VALUES OF $D\Psi(\boldsymbol{x})$

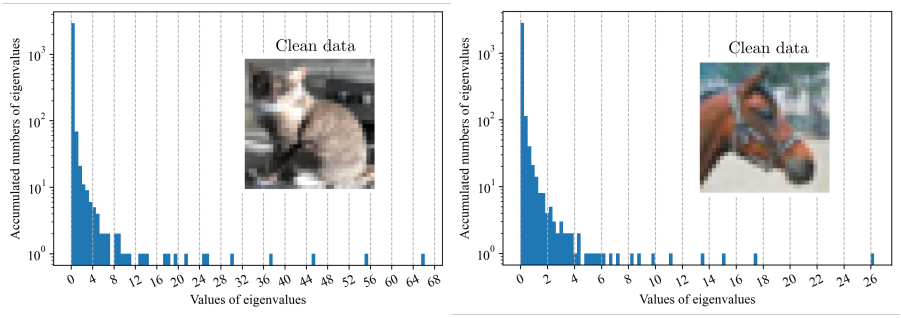


Figure 5: Demonstration of histograms of singular values of $D\Psi(\boldsymbol{x})$ for two different $\boldsymbol{x}$'s of the scale $32 \times 32 \times 3$. $D\Psi(\boldsymbol{x})$ is non-isotropic and data-dependent.

C.2 ILLUSTRATION OF SPECTROGRAM

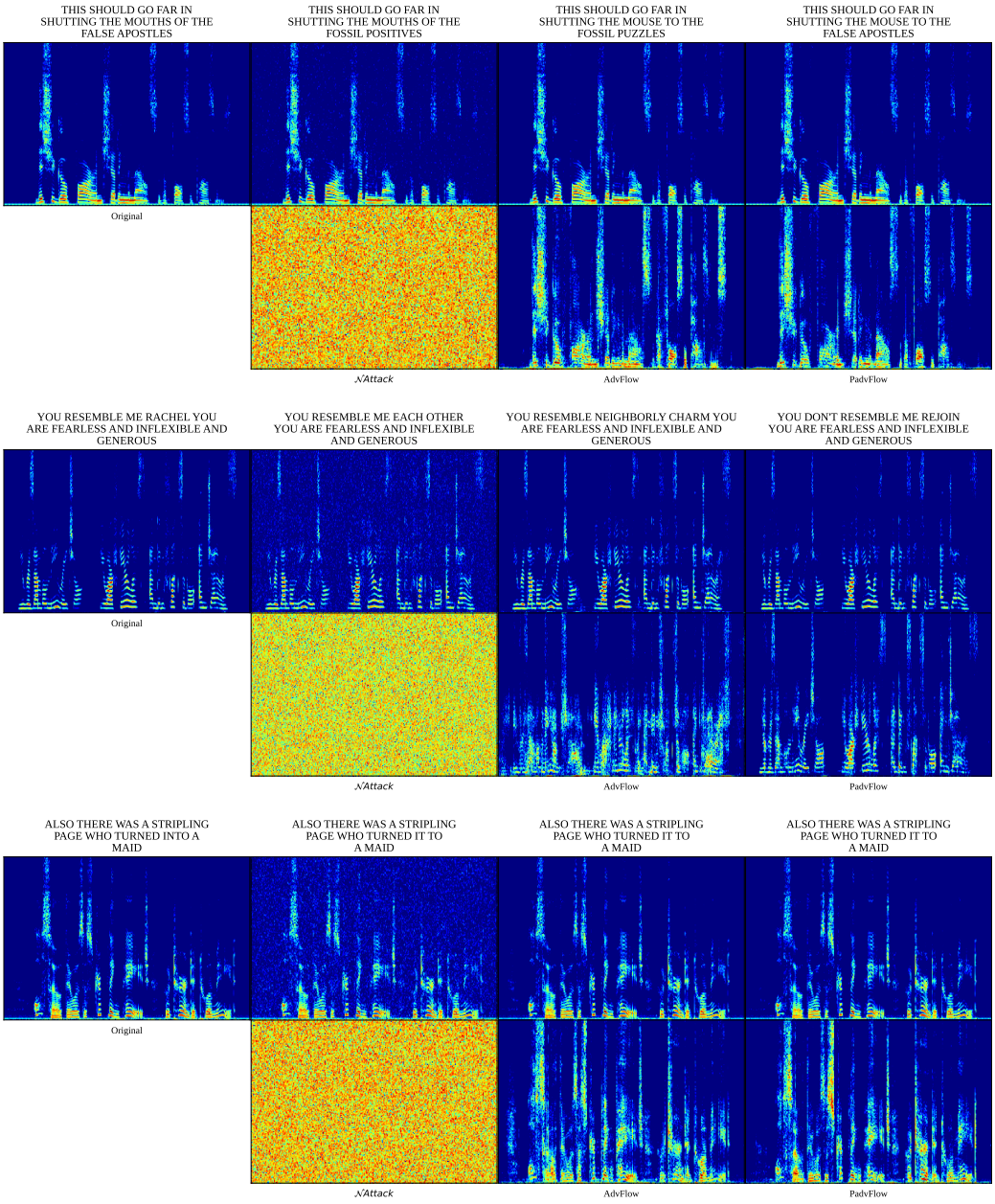


Figure 6: Demonstration of the adversarial examples and their differences synthesized by $\mathcal{N}$attack (Li et al., 2019) and AdvFlow (Mohaghegh Dolatabadi et al., 2020), and the proposed PadvFlow. Transcripts are shown on the top and the residuals between the adversarial examples and the originals are illustrated in the bottom of each figure.

D PROOFS OF PROPOSITIONS

In this section, we provide the proofs of Prop. 1, Lemma 1, and Prop. 2.

D.1 PROOF OF PROPOSITION 1

Proof. The proof is similar to Lemma 3.1 in (Mohaghegh Dolatabadi et al., 2020). Let us denote $\boldsymbol{z} := \boldsymbol{f}^{-1}(\boldsymbol{x})$. Then by first-order Taylor expansion, we have

$$\begin{aligned}\Psi(\boldsymbol{x}') - \Psi(\boldsymbol{x}) &= (\Psi \circ \boldsymbol{f})(\boldsymbol{z} + \boldsymbol{\delta}_{\boldsymbol{z}}) - (\Psi \circ \boldsymbol{f})(\boldsymbol{z}) \\ &= \nabla_{\boldsymbol{z}}(\Psi \circ \boldsymbol{f})(\boldsymbol{z})\boldsymbol{\delta}_{\boldsymbol{z}} + \mathcal{O}(\|\boldsymbol{\delta}_{\boldsymbol{z}}\|_2^2) \\ &= \nabla_{\boldsymbol{x}}\Psi(\boldsymbol{x}) \circ \nabla_{\boldsymbol{z}}\boldsymbol{f}(\boldsymbol{z})\boldsymbol{\delta}_{\boldsymbol{z}} + \mathcal{O}(\|\boldsymbol{\delta}_{\boldsymbol{z}}\|_2^2),\end{aligned}$$

where the last equality comes from the chain rule. Now we apply the inverse function theorem to $\boldsymbol{f}$ and get $\nabla_{\boldsymbol{z}}\boldsymbol{f}(\boldsymbol{z}) = \left(\nabla_{\boldsymbol{x}}\boldsymbol{f}^{-1}(\boldsymbol{x})\right)^{-1}$. Therefore, the proposition is proved.

■

D.2 PROOF OF LEMMA 1

Proof. We recall the definition of the continuous curve connecting $\boldsymbol{x}'$ and $\boldsymbol{x}$ defined on $[0, 1]$

$$\gamma(t) := d_{\Psi}(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x}), \boldsymbol{x}) - \epsilon_{\mathscr{F}}.$$

We notice that $\gamma(0) = d_{\Psi}(\boldsymbol{x}, \boldsymbol{x}) - \epsilon_{\mathscr{F}} = -\epsilon_{\mathscr{F}}$ and that $\gamma(1) = d_{\Psi}(\boldsymbol{x}', \boldsymbol{x}) - \epsilon_{\mathscr{F}} > 0$. By Intermediate Value Theorem, there is a $t^* \in (0, 1)$ so that $\gamma(t^*) = 0$. That is, the point $\boldsymbol{x}'_{\text{bdry}} := \boldsymbol{x} + t^*(\boldsymbol{x}' - \boldsymbol{x})$ satisfies $\left\|\Psi(\boldsymbol{x}'_{\text{bdry}}) - \Psi(\boldsymbol{x})\right\|_p = \epsilon_{\mathscr{F}}$, which proves the claim.

■

D.3 PROOF OF PROPOSITION 2

Proof. Let $\Gamma\colon [0, 1] \to \mathbb{R}^D$ be an arbitrary continuous curve so that $\Gamma(0) = \boldsymbol{x}_0$ and $\Gamma(1) = \boldsymbol{x}_1$. We recall that the interior, the boundary and the exterior of a set can partition the entire space. That is

$$\mathbb{R}^D = \text{int}(\mathcal{R}) \sqcup \partial\mathcal{R} \sqcup \text{ext}(\mathcal{R}), \tag{5}$$

where $\sqcup$ denotes the disjoint union. Also notice that both $\text{int}(\mathcal{R})$ and $\text{ext}(\mathcal{R})$ are open sets in $\mathbb{R}^D$. Suppose on the contrary that $\Gamma([0, 1]) \cap \partial\mathcal{R} = \emptyset$. If we apply intersection of $\Gamma([0, 1])$ with Eq. 5, we obtain

$$\begin{aligned}\Gamma([0, 1]) &= \Big(\text{int}(\mathcal{R}) \cap \Gamma([0, 1])\Big) \sqcup \Big(\partial\mathcal{R} \cap \Gamma([0, 1])\Big) \sqcup \Big(\text{ext}(\mathcal{R}) \cap \Gamma([0, 1])\Big) \\ &= \Big(\text{int}(\mathcal{R}) \cap \Gamma([0, 1])\Big) \sqcup \Big(\text{ext}(\mathcal{R}) \cap \Gamma([0, 1])\Big).\end{aligned}$$

That is, $\Gamma([0, 1])$ can be expressed as a disjoint union of nonempty open sets (in a relatively topology of $\Gamma[0, 1]$) $\text{int}(\mathcal{R}) \cap \Gamma([0, 1])$ and $\text{ext}(\mathcal{R}) \cap \Gamma([0, 1])$. This implies $\Gamma([0, 1])$ is disconnected (Rudin et al., 1976). However, it contradicts to the assumption on the connectivity of $\Gamma([0, 1])$ as it is the image of a connected set $[0, 1]$ under a continuous function Γ. Therefore, there must exist $t_0 \in [0, 1]$ such that $\Gamma(t_0) \in \partial\mathcal{R}$. ■

E FURTHER DISCUSSION ON INVERTIBLE GENERATIVE MODELING $\boldsymbol{f}$

In PadvFlow, the invertible generative model $\boldsymbol{f}$ is modeled by pre-trained NFs. However, we explore other plausible candidates for a replacement. Song et al. (2020) introduces probability flow ODE, a deterministic process with trajectories sharing the same marginal density as the diffusion process (Song & Ermon, 2019; Ho et al., 2020) modeled via stochastic differential equations. Probability flow ODE relates the latent and data space in an invertible way by solving forward/backward ordinary differential equation (ODE) via ODE solvers. We testify PadvFlow with

$\boldsymbol{f} =$ probability flow ODE on WRN34-Free by randomly sampling 100 images from CIFAR-10. We found out its success rate degrades significantly to 13.7% while it slightly improves on perceptual measurements. We claim that the degradation of success rate is due to numerical errors of solving ODE which leads to inaccurate invertibility. To be precise, let $\boldsymbol{x}$ be a sample in the image space. We found out that $\boldsymbol{x}'$, which is obtained by first transforming via the inverse of probability flow ODE to a latent variable and then transforming the latent back to the image space via probability flow ODE, usually has unignorable error $\|\boldsymbol{x}' - \boldsymbol{x}\|_\infty$ compared to the standard ℓ_∞ threshold $8/255$. Such a shift of $\boldsymbol{x}'$ may accumulate and affect the search of adversarial candidates via perceptual map. On the other hand, the improvement on perceptual quality from the probability flow ODE may due to that diffusion models are more expressive. However, the computation cost of probability flow ODE is extremely expensive even for a single forward-backward transform from the data space to latent space as it requires a ODE solving for each pass.

Nevertheless, Probability flow ODE still seems to be a potential workaround of NFs as NFs are generally difficult to train on a large scale dataset. We leave the improvement of increasing accuracy and reducing computation costs for $\boldsymbol{f} =$ probability flow ODE as a future work.