SD-NAE: GENERATING NATURAL ADVERSARIAL EXAMPLES WITH STABLE DIFFUSION

Yueqian Lin*
Duke Kunshan University
yueqian.lin@duke.edu

Jingyang Zhang*, Yiran Chen, Hai Li
Duke University
{jz288, yiran.chen, hai.li}@duke.edu

ABSTRACT

Natural Adversarial Examples (NAEs), images arising naturally from the environment and capable of deceiving classifiers, are instrumental in robustly evaluating and identifying vulnerabilities in trained models. In this work, unlike prior works that *passively* collect NAEs from real images, we propose to *actively* synthesize NAEs using the state-of-the-art Stable Diffusion. Specifically, our method formulates a controlled optimization process, where we perturb the token embedding that corresponds to a specified class to generate NAEs. This generation process is guided by the gradient of loss from the target classifier, ensuring that the created image closely mimics the ground-truth class yet fools the classifier. Named SD-NAE (Stable Diffusion for Natural Adversarial Examples), our innovative method is effective in producing valid and useful NAEs, which is demonstrated through a meticulously designed experiment. Code is available at https://github.com/linyueqian/SD-NAE.

1 INTRODUCTION

Robustly evaluating deep image classifiers is challenging, as existing standard test sets such as ImageNet (Deng et al., 2009) often feature simpler image compositions (Recht et al., 2019) and "spurious features" (Geirhos et al., 2019), which can lead to an overestimate of model performance. To address this issue, Hendrycks et al. (2021) introduce "Natural Adversarial Examples" (NAEs), where they employ adversarial filtration over extensive real images to pinpoint challenging natural images that deceive classifiers. NAEs are valuable in assessing worst-case performance and uncovering model limitations. Their approach to obtaining NAEs, however, is limited by its passive nature and lack of control over the selection of specific types of challenging examples, thereby restricting the ability to fully explore classifier weaknesses in diverse scenarios.

In this work, we propose to *actively* synthesize NAEs with a controlled optimization process. Leveraging a class-conditional generative model, particularly Stable Diffusion (Rombach et al., 2021), we optimize the class token embedding in the condition embedding space. This process is guided by the gradients of classification loss from the target image classifier to ensure the adversarial nature of the generated examples. Our method, termed SD-NAE (Stable Diffusion for Natural Adversarial Example), not only achieves a non-trivial fooling rate against an ImageNet classifier but also offers greater flexibility and control compared to previous methods, highlighting SD-NAE's potential as a tool for evaluating and enhancing model robustness.

2 METHODOLOGY

We introduce the SD-NAE method (Figure 1), which is motivated by the concept of NAEs and uses Stable Diffusion to approximate natural-looking images (see Appendix A for background introduction). Our exploration focuses on how adversarial optimization can enhance this approach, comparable to the creation of pixel-perturbed adversarial examples (Szegedy et al., 2013). The core of SD-NAE lies in the strategic optimization of class-relevant token embedding to trick the classifier into misclassifying the generated image. Consider, for instance, an image of a cat generated by Stable Diffusion G following the text condition "A high-quality image of a cat". Initially, the

*Equal contribution.

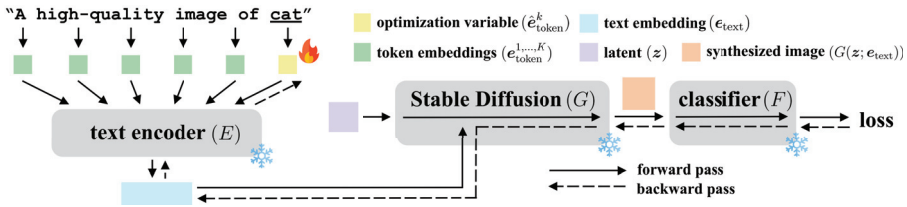

Figure 1: Guided by the loss gradient backpropagated from the classifier, SD-NAE generates NAEs by optimizing only the class-related token embedding, while keeping all models frozen. The letters in the parentheses are notations used in Equation (1).

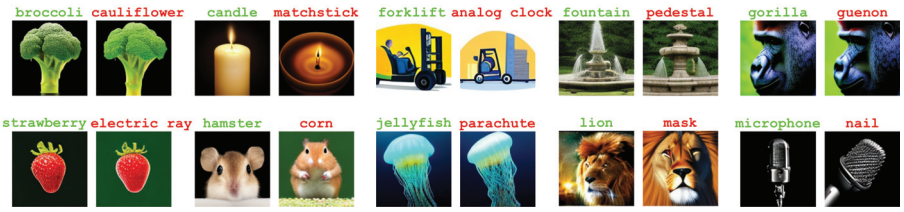

Figure 2: NAEs generated by SD-NAE. In each pair, the left initialization image is correctly classified by the model, yet the right one optimized by our method gets misclassified with the wrong prediction marked in red. See more samples in Figure 4.

image can be correctly identified as a cat by the classifier F, given Stable Diffusion's accurate generation capability. However, through subtle alterations of the token embedding of "cat", we expect to induce misclassification (*e.g.*, towards a target class y, $y \neq$"cat") over the generated image while maintaining its ground-truth as "cat". This process is governed by the optimization equation:

$$\min_{\hat{e}^k_{\text{token}}} L(F(G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}})), y) + \lambda \cdot R(\hat{e}^k_{\text{token}}, e^k_{\text{token}}), \text{where } \boldsymbol{e}_{\text{text}} = E(e^0_{\text{token}}, ..., \hat{e}^k_{\text{token}}, ..., e^{K-1}_{\text{token}}). \quad (1)$$

We mark the notations in Figure 1, while leaving a detailed discussion of Equation (1) in Appendix B. In general, the optimization variable $\hat{e}^k_{\text{token}}$ is the class-relevant token embedding (corresponding to "cat" in our example). The first term encourages the produced image to be adversarial, while the second term makes sure that the perturbation on $\hat{e}^k_{\text{token}}$ is only moderate, retaining the natural appearance of the generated image and preserving its ground-truth label as "cat".

3 Experiment

We evaluate SD-NAE using a carefully designed experiment. Please see details in Appendix C. Essentially, we focus on 10 categories of ImageNet whose semantics is clear. We take them as the ground-truth and generate 20 samples using SD-NAE for each of the 10 classes, resulting in 20 * 10 = 200 total optimization processes. An optimization is deemed successful if the image at any optimization step gets misclassified by the target classifier, which is a ResNet-50 pretrained on ImageNet. Importantly, we make sure that all initialization images (prior to optimization by SD-NAE) are correctly classified. In such a setting, our SD-NAE achieves a noteworthy success rate of 43.5% (which is actually a lower bound; see Appendix C), demonstrating its capability to effectively generate NAEs. Furthermore, as can be seen in Figure 2, the images generated by SD-NAE display variations in color, background, view angle, and style, underscoring its potential as a tool for examining model generalization among various covariate shifts. For comparison with the prior work of Song et al. (2018), please see Appendix D.

4 Conclusion

SD-NAE, by leveraging Stable Diffusion, effectively generates Natural Adversarial Examples and demonstrates its significant potential in the field of robustness research. As deep learning models continue to evolve, we believe that SD-NAE presents a novel approach for evaluating and understanding these complex systems, thereby emphasizing its profound role in future research. We discuss related works and limitations of SD-NAE in Appendix A and Appendix E, respectively.

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of the ICLR 2024 Tiny Papers Track.

ACKNOWLEDGEMENTS

The research was supported in part by NSF 1822085, the NSF IUCRC for ASIC and the memberships contributed by our industrial partners (https://asic.pratt.duke.edu/).

REFERENCES

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Xuelong Dai, Kaisheng Liang, and Bin Xiao. Advdiff: Generating unrestricted adversarial examples using diffusion models. *arXiv preprint arXiv:2307.12499*, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bygh9j09KX.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15262–15271, June 2021.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pp. 5389–5400. PMLR, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/8cea559c47e4fbdb73b23e0223d04e79-Paper.pdf.

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.

Jingyang Zhang, Nathan Inkawhich, Randolph Linderman, Yiran Chen, and Hai Li. Mixture outlier exposure: Towards out-of-distribution detection in fine-grained environments. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 5531–5540, January 2023a.

Jingyang Zhang, Jingkang Yang, Pengyun Wang, Haoqi Wang, Yueqian Lin, Haoran Zhang, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, et al. Openood v1. 5: Enhanced benchmark for out-of-distribution detection. *arXiv preprint arXiv:2306.09301*, 2023b.

APPENDIX

A BACKGROUND AND RELATED WORK

In this section, we provide background information and discuss related works to facilitate the presentation of our method. We first introduce the definition of NAEs, then distinguish our study from prior works on generating NAEs, and finally give an overview of the functionality of Stable Diffusion.

A.1 NATURAL ADVERSARIAL EXAMPLES

Formally, NAEs are defined as a set of samples w.r.t. a target classifier F (Hendrycks et al., 2021; Song et al., 2018):

$$\mathcal{A} \triangleq \{\boldsymbol{x} \in \mathcal{S} | O(\boldsymbol{x}) \neq F(\boldsymbol{x})\}. \tag{2}$$

Here, $\mathcal{S}$ contains all images that naturally arise in the real world and look realistic to humans. O is an oracle that yields the ground-truth semantic category of the image and relies on human evaluation.

Notice that NAEs are less restricted than pixel-perturbed adversarial examples (often referred to as "adversarial examples") (Szegedy et al., 2013), which are samples *artificially* crafted by adding minor perturbations to image pixels. Both NAEs and adversarial examples (AEs) can expose the vulnerability of a given classifier. However, since AEs are artificial rather than natural, they are mostly studied in the security context where there is assumed to be an attacker that intentionally attempts to compromise a model. In contrast, studies of NAEs, including ours, focus on a broader setting where the samples naturally occur within the environment but are misclassified by the classifier.

A.2 GENERATING NAES

As previously mentioned, we contend that the passive filtering of NAEs from real images, as demonstrated by Hendrycks et al. (2021), lacks flexibility; in contrast, we utilize a generative model to synthesize NAEs. In this regard, our work is closely related to but also has essential distinctions with the work by Song et al. (2018). While they optimize the latent of a class-conditional GAN with a fixed condition, we propose perturbing the condition while keeping the latent fixed within Stable Diffusion. In fact, it is later found that GAN can be sensitive to the latents, and generated images may be of low quality when the optimized latents land outside the well-defined support (Dai et al., 2023). We will show that applying their concept to Stable Diffusion is less effective in producing NAEs compared to our method.

Building upon this comparison, it is pertinent to discuss a concurrent work by Dai et al. (2023), who also apply the diffusion model to generate NAEs. However, their method is to enforce classifier guidance (Dhariwal & Nichol, 2021) to be adversarial, which requires sophisticated modification to the default classifier-free guidance sampling (Ho & Salimans, 2021) and may need extra care to adapt to different samplers. In contrast, our method can readily generalize to various diffusion models as it only perturbs the condition embedding without interfering with the sampling process.

A.3 STABLE DIFFUSION

Stable Diffusion represents a family of latent diffusion models (Rombach et al., 2021) with the capability of conditional generation. Using G to represent the Stable Diffusion model, the formulation that best describes its functionality in the context of our work is

$$\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}}), \tag{3}$$

where $\boldsymbol{x}$ is the synthesized image, $\boldsymbol{z}$ is a (random) latent vector, and $\boldsymbol{e}_{\text{text}}$ is the text embedding which serves as the condition for the generation. More specifically, $\boldsymbol{e}_{\text{text}}$ is the output of a transformer-based text encoder E where the input is the token embedding sequence $\boldsymbol{e}_{\text{token}}^{0:K-1}$ corresponding to the raw text description after tokenization, *i.e.*,

$$\boldsymbol{e}_{\text{text}} = E(\boldsymbol{e}_{\text{token}}^{0}, \boldsymbol{e}_{\text{token}}^{1}, ..., \boldsymbol{e}_{\text{token}}^{K-1}). \tag{4}$$

where K denotes the maximum padded length specified by the encoder E.

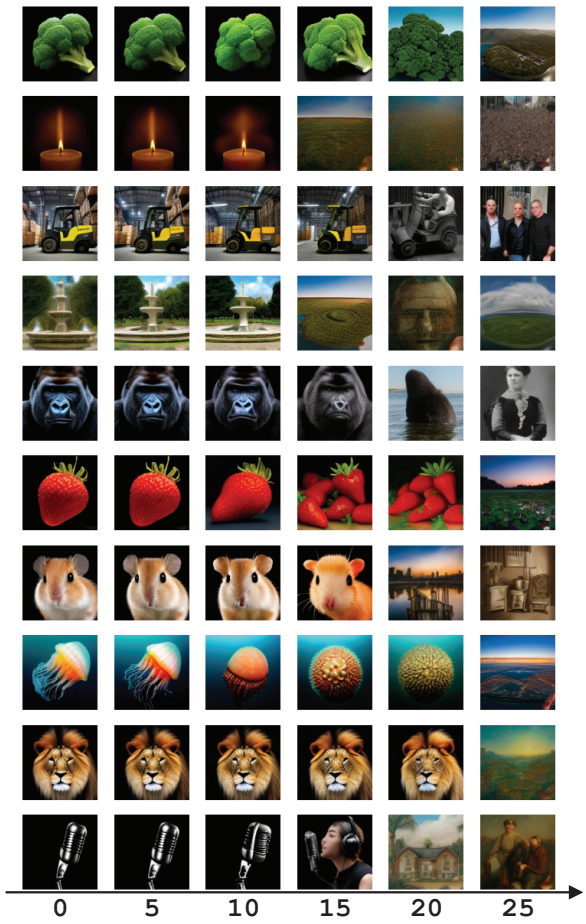

Figure 3: Empirical evidence that constraining the magnitude of token embedding perturbation can help preserve the image ground-truth. From left to right of each row, we move the initialized class token embedding along a random yet fixed Rademacher vector (*i.e.*, each element has equal probability of being +1 or –1) with increasing magnitude. The bottom axis denotes the relative magnitude of the perturbation, and the real magnitude has a factor of 1e-3. It can be noticed that the image semantic is well-preserved when the perturbation is small.

B Expanded Discussion on SD-NAE

First, we provide a more detailed explanation of our optimization objective in Equation (1). Let us explain the variables with a concrete example for clarity. Suppose we want to generate an image of a cat with a text condition being "A high-quality image of a cat" (other prompts can also work here as long as it contains the keyword "cat"), such that the image is misclassified by F as some category other than "cat". Equivalently and more formally, the goal is $O(G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}})) =$ "cat" $\neq F(G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}}))$.

The optimization variable, denoted as $\hat{\boldsymbol{e}}_{\text{token}}^{k}$ in Equation (1), corresponds to the token embedding of the word "cat" We initialize $\hat{\boldsymbol{e}}_{\text{token}}^{k}$ with the original token embedding $\boldsymbol{e}_{\text{token}}^{k}$ and optimize it with two

terms. The first term aims to encourage the generated sample $G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}})$ to deceive the classifier F into making an incorrect prediction. Specifically, if we want to induce a targeted misclassification towards a specific class y ($y \neq$ "cat"), we can simply use the cross-entropy loss as L. For untargeted misclassification, one can set y to "cat" and use negative cross-entropy as L (*i.e.*, maximizing the classification loss of class "cat").

The second term serves to regularize $\hat{\boldsymbol{e}}^k_{\text{token}}$, ensuring that the ground truth $O(G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}}))$ remains unchanged during optimization; otherwise, the unintended equality $O(G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}})) = F(G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}}))$ may occur, contradicting the definition of NAEs. To achieve this, we can let the regularization R be a distance metric (*e.g.*, Euclidean distance or cosine similarity) to enforce $\hat{\boldsymbol{e}}^k_{\text{token}}$ to stay in the vicinity of its unmodified counterpart $\boldsymbol{e}^k_{\text{token}}$. In other words, we are only inducing moderate perturbation to the token embedding, which intuitively helps $O(G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}}))$ remain unchanged (*e.g.*, being "cat" all the time in our example). We empirically justify this intuition and design in Figure 3. The λ that accompanies the second term is just a weighting factor. It is worth noting, however, that in practice, when the number of optimization steps is small, it often suffices to set $\lambda = 0$ since the overall magnitude of the perturbation (*i.e.*, the distance between $\hat{\boldsymbol{e}}^k_{\text{token}}$ and $\boldsymbol{e}^k_{\text{token}}$) is bounded.

Why token embedding? We next discuss why we choose the token embedding $\boldsymbol{e}^k_{\text{token}}$ instead of the latent z or the text embedding $\boldsymbol{e}_{\text{text}}$ as the optimization variable here. Empirically, we find that perturbing the latent or text embedding is significantly less effective and efficient in generating NAEs compared to perturbing the token embedding, which has fooling rates of 10%, 20%, and 43.5% respectively (refer to Appendix C for the definition of fooling rate and experiment setup). Our hypothesis for this observation is as follows. Firstly, in a diffusion model, the latent undergoes an iterative multi-step reverse diffusion process, potentially impeding the gradient flow from the classifier back to the latent. Secondly, as the text embedding integrates all tokens, perturbations on the text embedding might disperse across all tokens. Intuitively, perturbing some class-irrelevant tokens (*e.g.*, the meaningless padding tokens) is not likely to induce significant change to the image content, meaning that there is less chance the generated sample will fool the classifier. In contrast, perturbing the class-relevant token (*i.e.*, $\boldsymbol{e}^k_{\text{token}}$) directly targets the semantic-related content of the image, which we will further demonstrate to be effective with the following experiment results.

Other application scenarios. Lastly, notice that SD-NAE can be easily adapted to create other types of NAEs beyond in-distribution (ID) misclassification. For instance, a deployed model in the real world will inevitably encounter Out-of-Distribution (OOD) samples, which are samples not belonging to any known category (Zhang et al., 2023a;b), necessitating an accurate OOD detector to flag unknown inputs. With SD-NAE, one can generate NAEs that fool the OOD detector into ID $\rightarrow$ OOD or OOD $\rightarrow$ ID misclassification by playing with the loss L in Equation (1). Specifically, notice that an OOD detector often operates by thresholding the maximum softmax probability. To generate an OOD image predicted as ID by the detector, we can employ cross-entropy loss as L and use any one-hot label as y, thereby encouraging the classifier to make a confident prediction on the synthesized image. Reversely, the ID $\rightarrow$ OOD misclassification is also achievable if we minimize the maximum classification probability by minimizing the cross-entropy between the softmax probability distribution and uniform distribution (Zhang et al., 2023a). Overall, SD-NAE demonstrates flexibility in producing NAEs for various purposes.

C Experiment Details

Models and setup. The target classifier, which we aim for the NAEs to fool is an ImageNet-pretrained ResNet-50 hosted by Microsoft on Hugging Face (model tag: "microsoft/resnet-50"). We utilize a nano version of Stable Diffusion, finetuned from the official 2.1 release (model tag: "bguisard/stable-diffusion-nano-2-1"). We generate 128x128 images to ensure the optimization is manageable with a single 24GB GPU; these images are then resized to 224x224 before being fed to the classifier, matching its default resolution. DDIM sampler with 20 sampling steps is used for Stable Diffusion. The guidance scale is set to the default value of 7.5. In the optimization of SD-NAE, we use Adam as the optimizer with a learning rate of 0.001. The number of iterations or gradient steps is 20.

Workflow and metric. We use *fooling rate* as the quantitative metric for SD-NAE. To ensure a fair and meaningful evaluation, we first do several careful preprocessing as follows. We start with 100 random classes from ImageNet. For each class, we generate 100 samples from Stable Diffusion with

random latent and measure the accuracy of the classifier on those samples. Subsequently, we remove the classes whose accuracy is lower than 90%, which leaves us 25 classes. After that, we manually pick ten classes whose semantics are clear and unambiguous to our human evaluators (the authors of this work) to make it easy for later human inspection (to get the oracle prediction $O(\boldsymbol{x})$). The selected classes are `broccoli`, `candle`, `forklift`, `fountain`, `gorilla`, `strawberry`, `hamster`, `jellyfish`, `lion`, and `microphone`. Finally, for each of the ten classes, we prepare 20 different random latent vectors z, with which the generated image $G(\boldsymbol{z}; \boldsymbol{e}_{\text{text}})$ (without SD-NAE optimization yet) is correctly classified by the ResNet-50. This step ensures that the initialized sample is not already an NAE, allowing us to isolate the effect and confidently attribute the NAEs to our SD-NAE optimization rather than to other factors inherent in the generative model.

After the preprocessing, we perform 20 * 10 = 200 optimization processes, each corresponding to one class and one prepared latent. In each multi-step optimization process, if any one of the sample $\boldsymbol{x}$ generated at a certain step satisfies $O(\boldsymbol{x}) =$ current desired class $\neq F(\boldsymbol{x})$, we count this optimization as success in fooling the classifier. The final fooling rate is calculated as the ratio of successful deceptions to the total number of optimizations, amounting to 200 in our study. It is noteworthy that we adopt a stricter definition of NAE than that in Equation (2) to explicitly demonstrate the efficacy of SD-NAE. In practice, one does not need to enforce $O(\boldsymbol{x}) =$ current desired class: Even if $O(\boldsymbol{x})$ deviates from the expected class, *e.g.*, the synthesized image is not a `broccoli` while the current text prompt is "An image of a broccoli", $\boldsymbol{x}$ is still a valid NAE as long as $O(\boldsymbol{x}) \neq F(\boldsymbol{x})$. Therefore, the fooling rate reported in our experiment represents only a lower bound of the actual fooling rate.

Result. As discussed in the main text, SD-NAE achieves a non-trivial 43.5% fooling rate/success rate. The generated NAEs are visualized in Figure 4.

D Comparison with Prior Work

Here, we compare our SD-NAE with the method proposed by Song et al. (2018). As mentioned in Appendix A.2, they perturb the latent vector of class-conditional GANs to curate NAEs, while our design is to optimize the conditional token embedding of Stable Diffusion models (and keep the latent fixed). In Appendix C, we have shown by empirical results that directly applying the previous method (*i.e.*, updating the latent of Stable Diffusion) yields a much worse attack success rate/fooling rate, indirectly justifying our design. Here, we perform a straight comparison between SD-NAE and the work of Song et al. (2018).

Setup. We use a class-conditional BigGAN (Brock et al., 2019) pre-trained on ImageNet, which to our knowledge is one of the most powerful GANs for ImageNet. The experiment workflow remains the same as with our method: Denoting the GAN as G and the target ResNet-50 classifier we want to attack as F, for each image category we prepare 20 randomly-initialized latent vectors $\boldsymbol{z}$ where the prediction on each generated image $F(G(\boldsymbol{z}))$ matches the ground-truth or the oracle prediction $O(G(\boldsymbol{z}))$. Then using the same loss function as in Song et al. (2018), we optimize the latent vector of the GAN to generate NAEs. We try our best to vary the hyperparameters and report the best result that we observe. It is also worth noting that in the original work of Song et al. (2018), they only did experiments on small-scale and simple datasets like MNIST, SVHN, and CelebA, while here we are looking at ImageNet with images consisting complex, real-world objects/scenes.

Result. The best fooling rate/attack success rate of Song et al. (2018) is 14.0%, which is much lower than ours (43.5%). Specifically, in some cases the optimized image does not change much from the initialization and thus fails to deceive the classifier. In other cases, the optimization goes wild and leads to nonsensical images. The latter case is in line with the finding that GANs can be sensitive to perturbed latents (Dai et al., 2023), since they might be "out-of-distribution" w.r.t. the well-regularized latent distribution that the model sees during the training. We visualize the synthesized samples in Figure 5. Qualitatively, SD-NAE results in higher-quality samples than the compared method.

E Limitations

Since SD-NAE is based upon Stable Diffusion, it inherits a few limitations from its underlying framework. First, the computational cost of SD-NAE could be high and the optimization could be

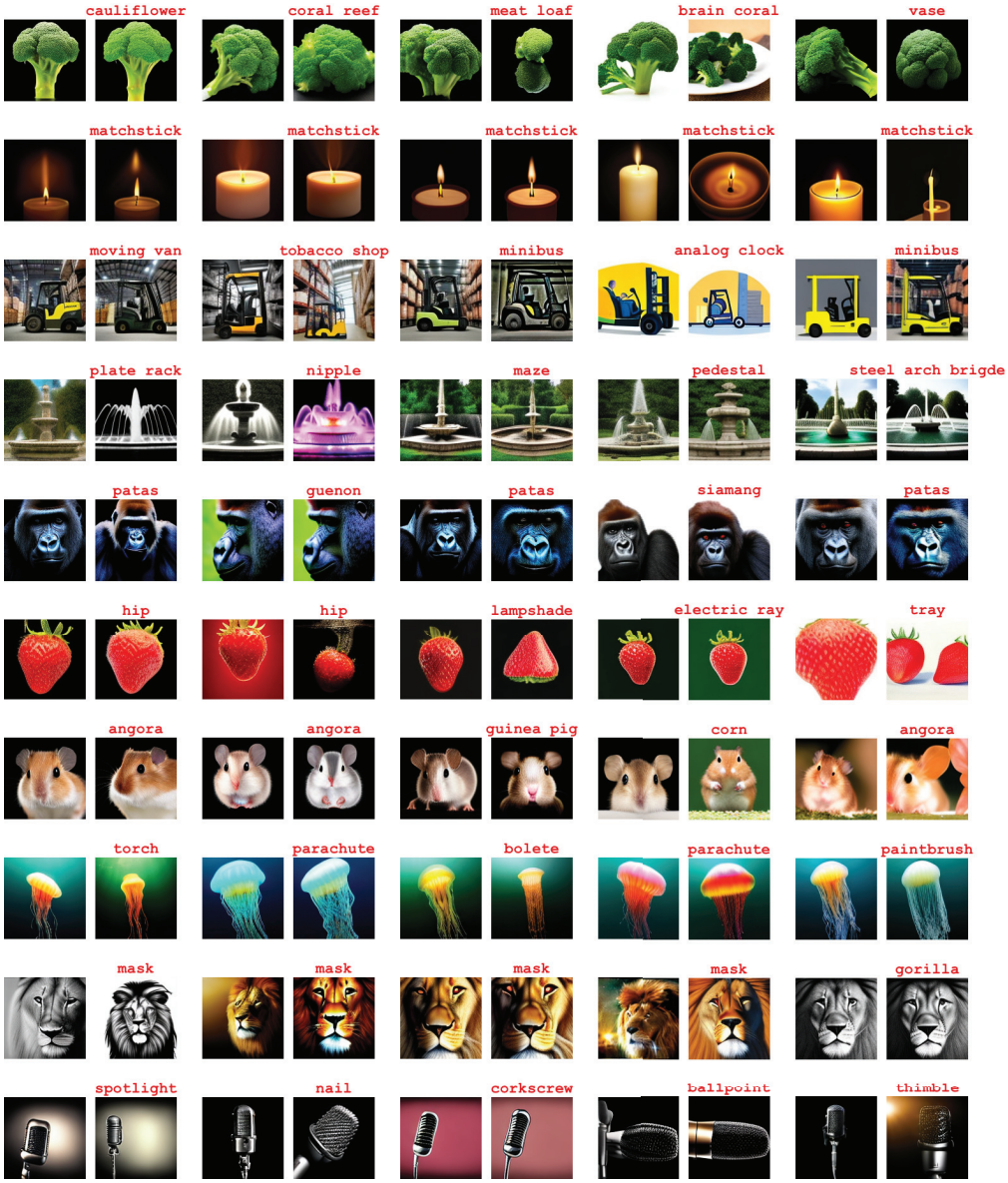

Figure 4: Examples generated by SD-NAE. From top to bottom, the ground-truth is `broccoli`, `candle`, `forklift`, `fountain`, `gorilla`, `strawberry`, `hamster`, `jellyfish`, `lion`, and `microphone`, respectively. In each pair, the left one is generated with the initialized token embedding. Importantly, we make sure that all left images are correctly classified by the ImageNet ResNet-50 model in the first place. The right ones are the result of SD-NAE optimization when using the corresponding left one as initialization, and we mark the classifier's prediction in red above the image.

slow. For instance, generating a single 128x128 natural adversarial example with SD-NAE under our experiment setting (*i.e.*, 20 steps for diffusion sampling and 20 steps for SD-NAE's optimization) requires approximately 22GB of GPU memory and takes about 1 minute. However, we note that both the memory footprint and time cost can be significantly reduced if sampling-efficient diffusion models are used, *e.g.*, Latent Consistency Models (Luo et al., 2023) and SD-turbo (Sauer et al., 2023) which only require 1 to 4 diffusion sampling steps. Meanwhile, empirically we find that SD-NAE does *not* really require as many as 20 optimization steps to succeed: In our experiment, the average number of steps for finding the first adversarial example is around 10 (9.66).

Figure 5: Examples generated by Song et al. (2018) using GAN. Following our experiment setup, each initialized image is correctly classified by the target ResNet-50 classifier. The first row shows examples that we count as successfully generated NAEs, whereas the second row shows failure cases where the optimized images exhibit unnatural looking. Note that some successful NAEs here actually do not look that natural, and the quality in general lags behind those generated by SD-NAE (Figure 4). Still, despite counting them as success, we observe a mere 14.0% success rate compared with 43.5% achieved by SD-NAE.

Second, in some cases, we find that the generated image is absurd and diverges significantly from a natural appearance. Such cases can arise either inherently from Stable Diffusion or from our SD-NAE optimization process. Taking the category `broccoli` as an example, out of the 100 initialization images (generated by Stable Diffusion with random latents), there are 8 of them which exhibits a weird, unnatural looking of a broccoli (an 8% "failure" rate). Then, during SD-NAE optimization, there are 24 out of 400 images that fail to present the normal looking of a broccoli (an 6% "failure" rate). However, we remark that having unnatural images at a few steps does not mean that SD-NAE is compromised; instead, it can be considered success as long as there is at least one natural-looking adversarial example produced during the multi-step optimization, which is typically the case in our experiment.