Dynamic Generative Targeted Attacks with Pattern Injection

Weiwei Feng[1,*], Nanqing Xu[1,*] , Tianzhu Zhang[1,2,†], Yongdong Zhang[1]
[1] University of Science and Technology of China, [2] Deep Space Exploration Lab
fengww@mail.ustc.edu.cn, xnq@mail.ustc.edu.cn, {tzzhang, zhyd73}@ustc.edu.cn

To further demonstrate the effectiveness of our dynamic generative attack method, we present detailed mathematical proofs and extra experiments in this **Supplementary Material**. Section 1 mainly offers detailed proofs of Section 3.4 **Theoretical Analyses** in our paper. Detailed results and analysis of extra experiments are available in Section 2, and Section 3 provides visualization of the generated adversarial examples and adversarial perturbations.

1. Theoretical Analyses

1.1. Preliminaries

Firstly, we consider a model of Gaussian Binary Classification, where the model's goal is to classify samples drawn from $\mathcal{C}_1 \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathcal{C}_2 \sim N(-\boldsymbol{\mu}, \boldsymbol{\Sigma})$ according to a maximum likelihood rule. Therefore, for an input $\boldsymbol{x} \in \mathcal{R}^k$, we have posterior probabilities $P(C_1|\boldsymbol{x})$ and $P(C_2|\boldsymbol{x})$ via Bayes rule, which can be indicated as follows:

$$\begin{aligned} P(C_1|\boldsymbol{x}) &= \frac{P(C_1)P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x})} \\ P(C_2|\boldsymbol{x}) &= \frac{P(C_2)P(\boldsymbol{x}|C_2)}{P(\boldsymbol{x})} \end{aligned} \tag{1}$$

where the likelihood probabilities can be formulated as:

$$P(\boldsymbol{x}|C_1) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}},$$

$$P(\boldsymbol{x}|C_2) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}+\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}+\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}.$$

$P(C_1|\boldsymbol{x})$ and $P(C_2|\boldsymbol{x})$ are the probabilities of the sample $\boldsymbol{x}$ as C_1 and C_2, respectively. Thus, we can know that if $P(C_1|\boldsymbol{x}) > P(C_2|\boldsymbol{x})$, then the sample $\boldsymbol{x}$ is classified as C_1, and vice versa $\boldsymbol{x}$ is classified as C_2. Figure 1 illustrates the classification rule. Supposing that $P(C_1) = P(C_2)$, we classify $\boldsymbol{x}$ as C_1, if the posterior probabilities hold the following inequality:

$$\frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})} = \frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)} > 1. \tag{2}$$

* Equal Contribution
† Corresponding Author

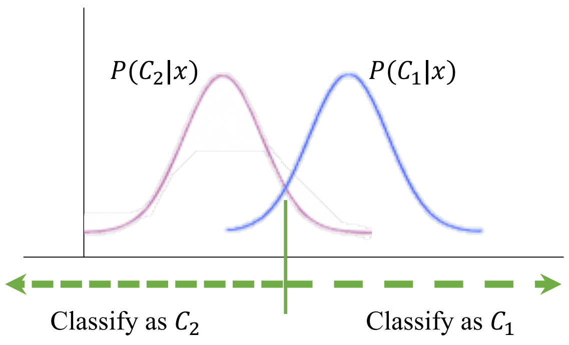


Figure 1. The illustration of the learned classification rule.

Further, we just need to solve the following problem:

$$\begin{aligned} & \left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) - \left(-\tfrac{1}{2}(\boldsymbol{x}+\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}+\boldsymbol{\mu})\right) > 0, \\ \Leftarrow \quad & \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} > 0 \\ \Leftarrow \quad & \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} > 0. \end{aligned} \tag{3}$$

Therefore, we know that the learned classification rule is:

$$\mathbb{1}\{\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} > 0\}, \tag{4}$$

where $\mathbb{1}$ is the indicate function.

1.2. Proofs

This section provides comprehensive proofs of Section 3.4 **Theoretical Analyses** in our paper.

Setup. We consider a simple problem of maximum likelihood classification, similar to that of [4], between two Gaussian distributions.

$$\mathcal{X}_s \sim N\left(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s\right), \quad \mathcal{X}_t \sim N\left(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\right), \tag{5}$$

where $\boldsymbol{\Sigma}_s, \boldsymbol{\Sigma}_t = \operatorname{diag}\left\{\sigma_{s_1}^2, \cdots \sigma_{s_n}^2\right\}, \operatorname{diag}\left\{\sigma_{t_1}^2, \cdots \sigma_{t_n}^2\right\}$, respectively. To simplify the following derivation, we introduce a mapping and a translation operation to transform the model into a standard Gaussian binary classification model that is explained in Section 1.1, which can be denoted as:

$$\begin{aligned} \mathcal{F}_s &= \boldsymbol{A}\mathcal{X}_s - \tfrac{\boldsymbol{A}\boldsymbol{\mu}_s + \boldsymbol{\mu}_t}{2} \sim N\left(-\boldsymbol{\mu}, \boldsymbol{\Sigma}\right), \\ \mathcal{F}_t &= \boldsymbol{E}\mathcal{X}_t - \tfrac{\boldsymbol{A}\boldsymbol{\mu}_s + \boldsymbol{\mu}_t}{2} \sim N\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right), \end{aligned} \tag{6}$$

where the mapping matrix $\boldsymbol{A} = \operatorname{diag}\left\{\frac{\sigma_{t_1}}{\sigma_{s_1}}, \frac{\sigma_{t_2}}{\sigma_{s_2}} \cdots, \frac{\sigma_{t_n}}{\sigma_{s_n}}\right\}$, $\boldsymbol{E}$ is the identity matrix and $\boldsymbol{\mu} = \frac{-\boldsymbol{A}\boldsymbol{\mu}_s + \boldsymbol{\mu}_t}{2}$, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_t$. To perform targeted attacks against this model with a given sample $\boldsymbol{x}_s \in \mathcal{X}_s$ and the target label t, where $\boldsymbol{A}\boldsymbol{x}_s - \frac{\boldsymbol{A}\boldsymbol{\mu}_s + \boldsymbol{\mu}_t}{2} = \boldsymbol{f}_s \in \mathcal{F}_s$ and $\boldsymbol{f}_s + \boldsymbol{\delta}^\star = \boldsymbol{A}(\boldsymbol{x}_s + \boldsymbol{\delta}) - \frac{\boldsymbol{A}\boldsymbol{\mu}_s + \boldsymbol{\mu}_t}{2}$, according to Equation (4), we aim to solve the following optimization problem:

$$\boldsymbol{\delta} = \boldsymbol{A}^{-1}\boldsymbol{\delta}^\star, \quad \boldsymbol{\delta}^\star = \underset{\|\boldsymbol{\delta}'\|_2^2 \leqslant \varepsilon^2}{\operatorname{argmax}} \left(\boldsymbol{f}_s + \boldsymbol{\delta}'\right)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}. \tag{7}$$

Taking advantage of the method of Lagrange multipliers, we can get the following problem:

$$\begin{cases} \nabla_{\boldsymbol{\delta}'}\left(\boldsymbol{\delta}'^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) = \lambda \nabla_{\boldsymbol{\delta}'}\left(\|\boldsymbol{\delta}'\|_2^2 - \varepsilon^2\right) \\ \|\boldsymbol{\delta}'\|_2^2 = \varepsilon^2 \end{cases} \tag{8}$$

Then, we can easily get the optimal solution as follows:

$$\begin{aligned} \boldsymbol{\delta}^\star &= \frac{1}{\lambda}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \\ \boldsymbol{\delta} &= \frac{\boldsymbol{\Sigma}^{-1}}{2\lambda}\left[\boldsymbol{A}^{-1}\Big(\boldsymbol{A}\left(\boldsymbol{x}_s - \boldsymbol{\mu}_s\right) + \boldsymbol{\mu}_t\Big) - \boldsymbol{x}_s\right]. \end{aligned} \tag{9}$$

Going a step further, we rewrite the solution of $\boldsymbol{\delta}$ into a more concise formula as:

$$\boldsymbol{\delta} = C_1 \left[\begin{pmatrix} \frac{\sigma_{t_1}}{\sigma_{s_1}} & & \\ & \ddots & \\ & & \frac{\sigma_{t_n}}{\sigma_{s_n}} \end{pmatrix} \left(\boldsymbol{x}_s - \boldsymbol{\mu}_s\right) + \boldsymbol{\mu}_t \right] - C_2 \boldsymbol{x}_s, \tag{10}$$

where $C_1 = \frac{\boldsymbol{\Sigma}^{-1}\boldsymbol{A}^{-1}}{2\lambda}$ and $C_2 = \frac{\boldsymbol{\Sigma}^{-1}}{2\lambda}$. In fact, note that the item of 🟥 represents the target pattern or style injection, which is consistent with the previous works [3, 5]. Therefore, the formulation of Equation (10) shows a close underlying correlation between the optimal targeted adversarial perturbation and the embedding of target pattern or style, which also theoretically guarantees the effectiveness of our proposed generative model for targeted attacks.

2. Extra Experiments

2.1. Attack Google Cloud Vision API

To further demonstrate the effectiveness of our proposed method, we also evaluate our adversarial examples against real-world systems, Google Cloud Vision API. Note that the API outputs a list of semantic labels along with confidence scores, which are presented in Figure 2. Because all predicted labels are with relatively high confidence ($\geq 50\%$), successful attacks mean that as long as the target class label appears in the top-10 prediction list. Besides, since the semantic label set predicted by the API does not exactly correspond to the 1000 ImageNet classes, we regard semantically similar classes as the same class. For example, in

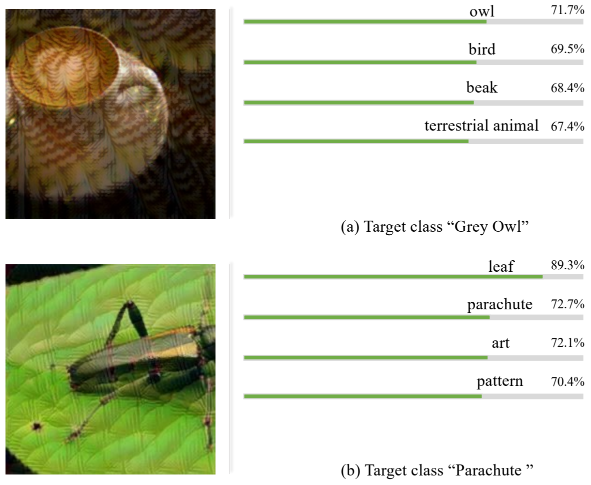


Figure 2. Successful targeted adversarial images on Google Cloud Vision generated by the our method with various target classes.

Table 1. Targeted transfer success rates of different methods against Google Cloud Vision with different substitute models.

Substitute model $\rightarrow$	Inc-v3	Res-152	Vgg-16
DIM [8]	8.3%	9.2%	8.6%
TIM [2]	8.6%	9.5%	8.6%
SIM [6]	8.6%	9.1%	8.6%
MIM [1]	7.8%	7.7%	8.1%
Ours	**19.6%**	**37.5%**	**25.9%**

Figure 2(a), we think that the predicted labels "owl", "bird" and "beak" are semantically similar to the target class "Grey Owl", which indicates that our method generates successful targeted adversarial examples. Table 1 reports the attack success rate against Google Cloud Vision API with various white-box models. In particular, our method achieves the best transferability in all cases.

2.2. Evaluation on ImageNet Validation Dataset

For a consistent and fair comparison, we also follow [7] to evaluate our method on ImageNet validation dataset. Figure 3 presents Top-1 target attack success rates (%) averaged across 10 various target classes on 49.95K ImageNet validation images (except 50 images from the target class). Obviously, our method can show better transferability than TTP [7], where our method outperforms TTP [7] by 2.02% on average.

3. Visualization

Adversarial examples and corresponding perturbations are shown in Figure 4, Figure 5, Figure 6, Figure 7, Figure 8, Figure 9, Figure 10, Figure 11, Figure 12 and Fig-

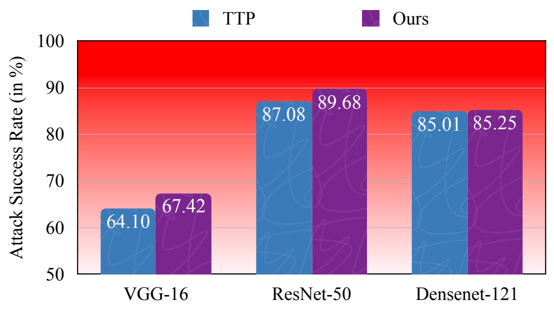


Figure 3. Target attack success rates (%) averaged across 10 targets with 49.95K ImageNet validation images. Note that the substitute model is ResNet-152, and target models includes VGG-16, ResNet-50 and DenseNet-121.

ure 13. The examples are randomly selected from the dataset. Evidently, our method not only demonstrates an underlying dependency between the input instance and the corresponding adversarial perturbation, but also can generate adversarial perturbations with strong targeted semantic patterns.

References

[1] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018. 2

[2] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4312–4321, 2019. 2

[3] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 2

[4] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 2

[6] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2019. 2

[7] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. On generating transferable targeted perturbations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2021. 2

[8] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2730–2739, 2019. 2

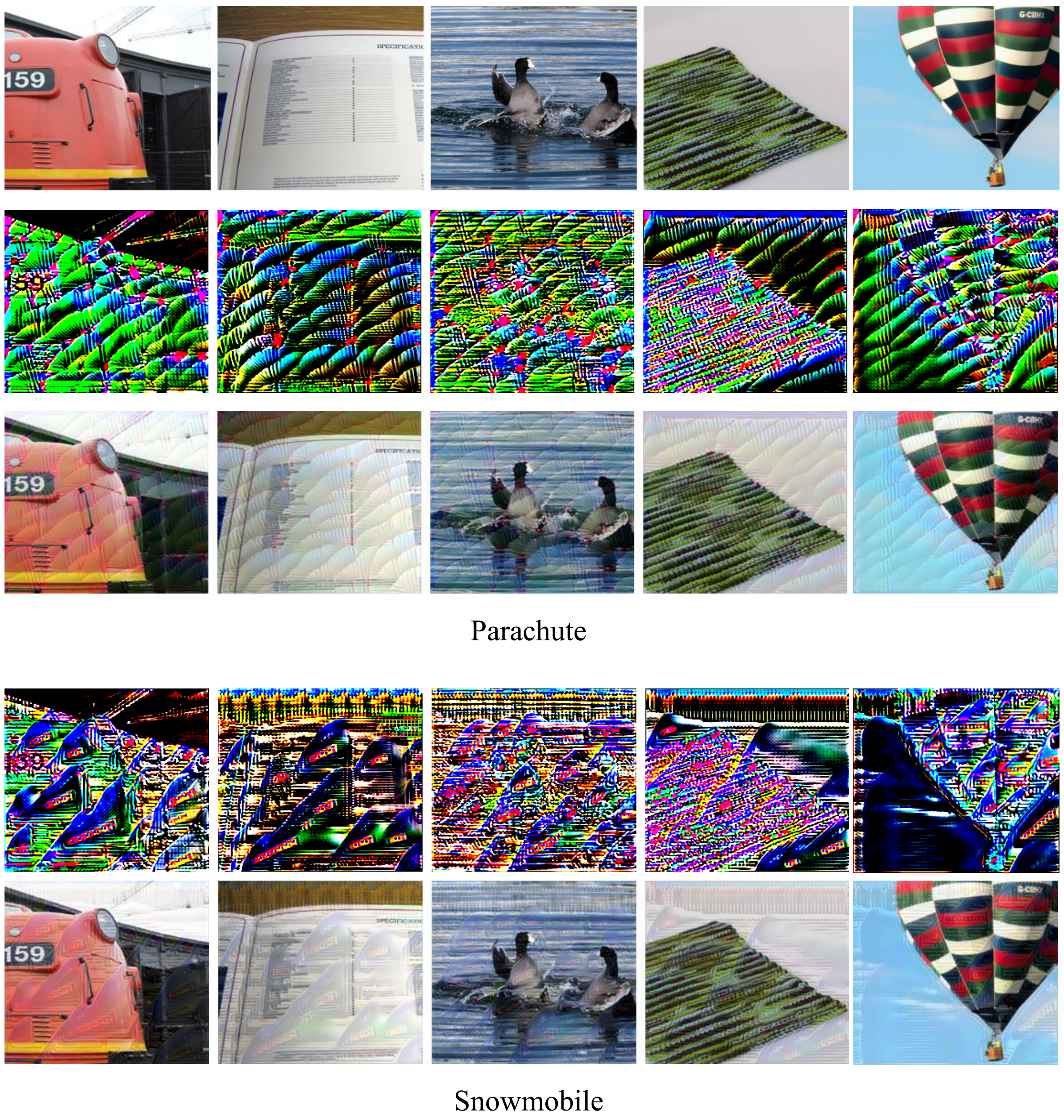


Figure 4. Targeted adversarial examples and perturbations generated by our method trained against Res-152. Note that different target classes lead to different patterns of adversarial perturbations. And there is an underlying dependency between the input instance and the corresponding adversarial perturbation as well.

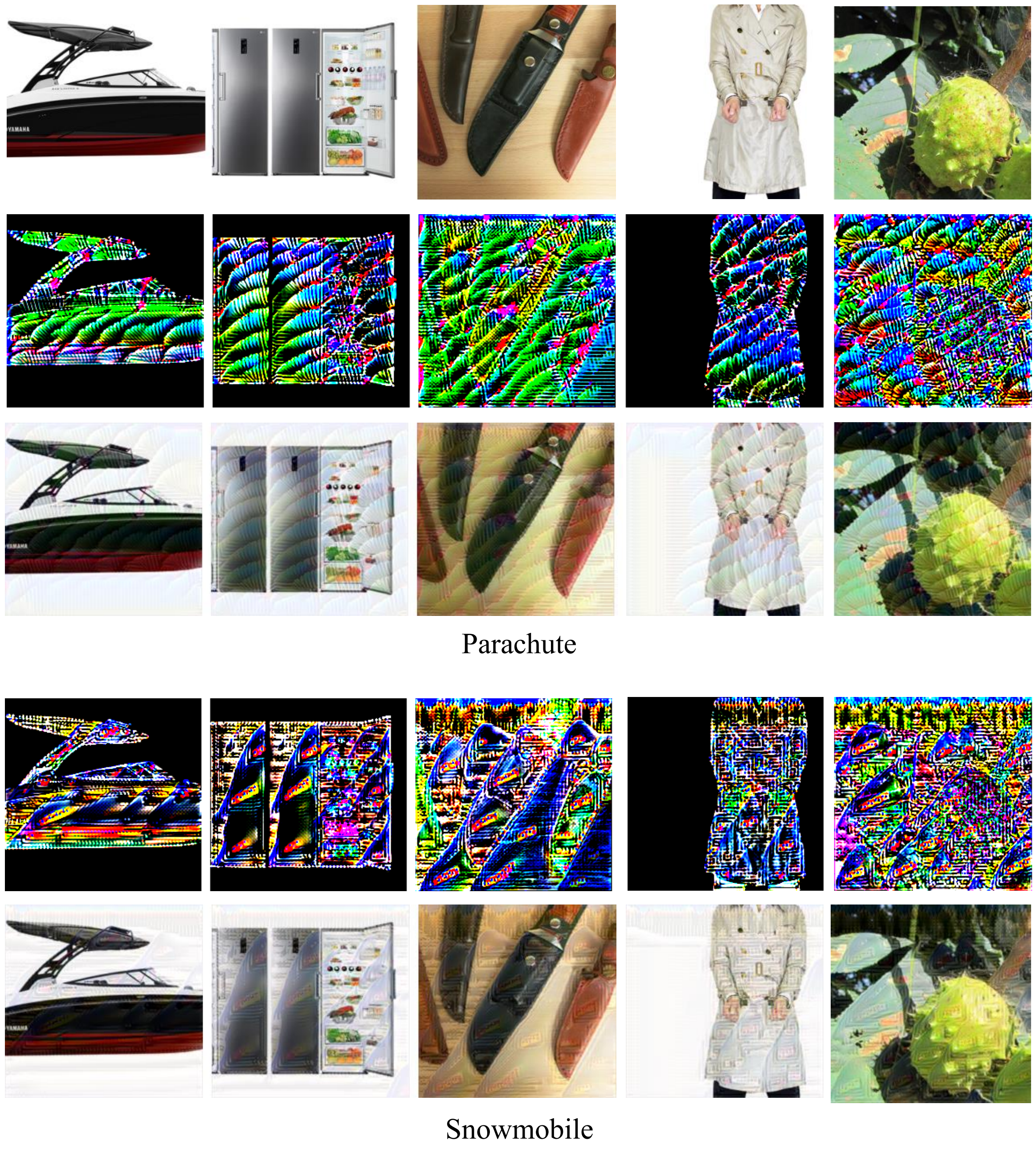


Figure 5. Targeted adversarial examples and perturbations generated by our method trained against Res-152. Note that different target classes lead to different patterns of adversarial perturbation. And there is an underlying dependency between the input instance and the corresponding adversarial perturbation as well.

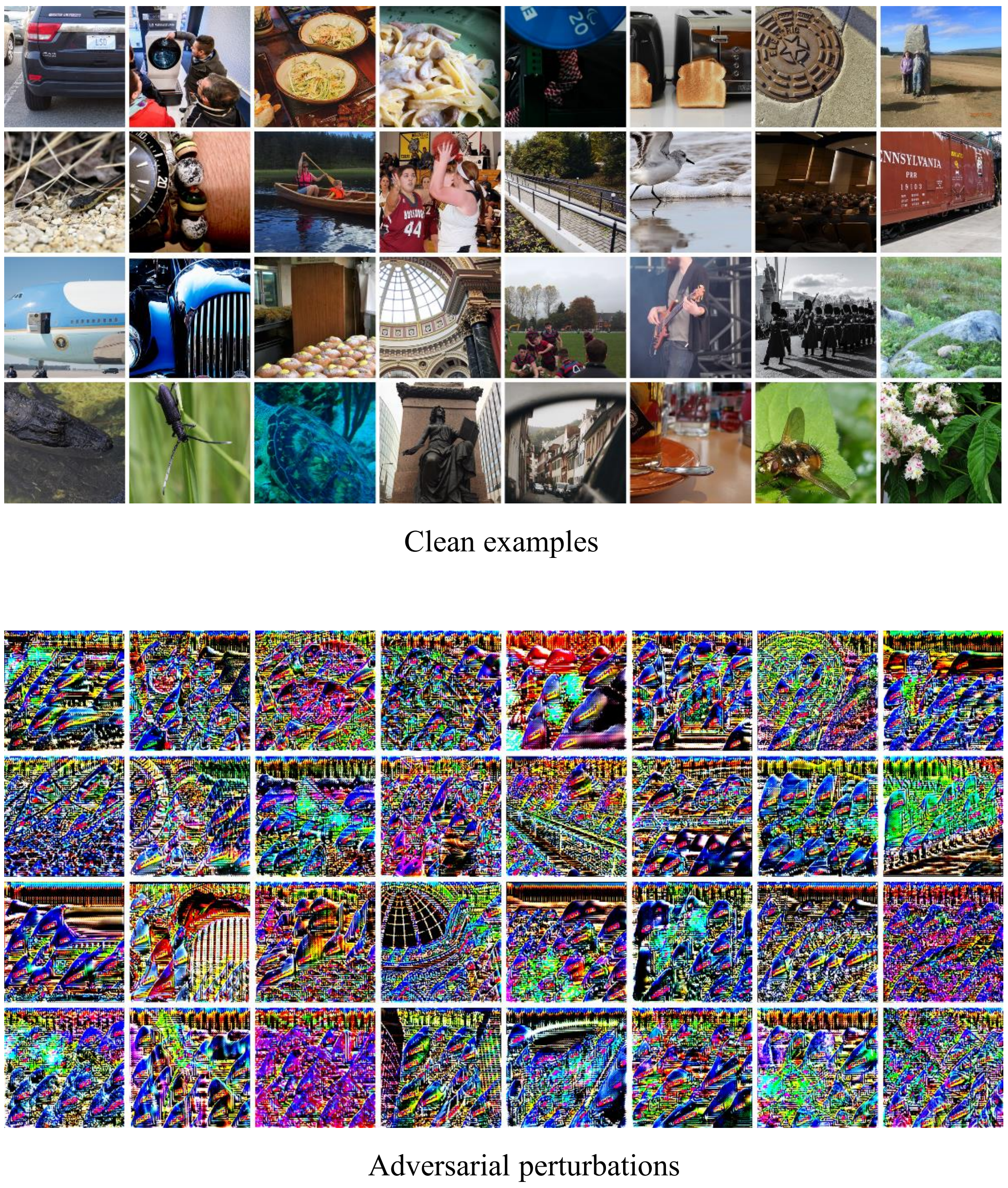
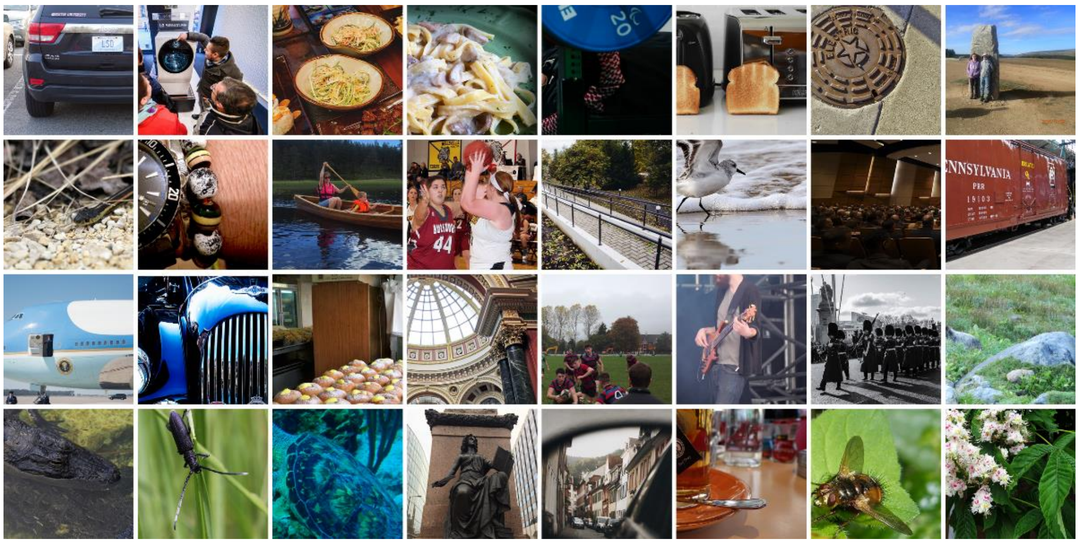


Figure 6. Targeted adversarial perturbations generated by our method trained against Res-152 with target class “Snowmobile”. Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.


Clean examples

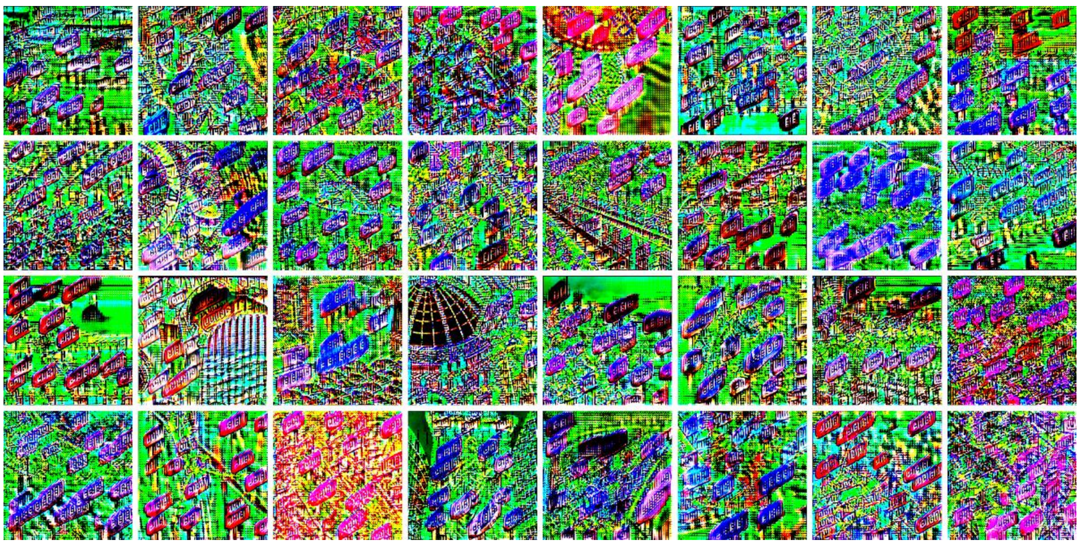
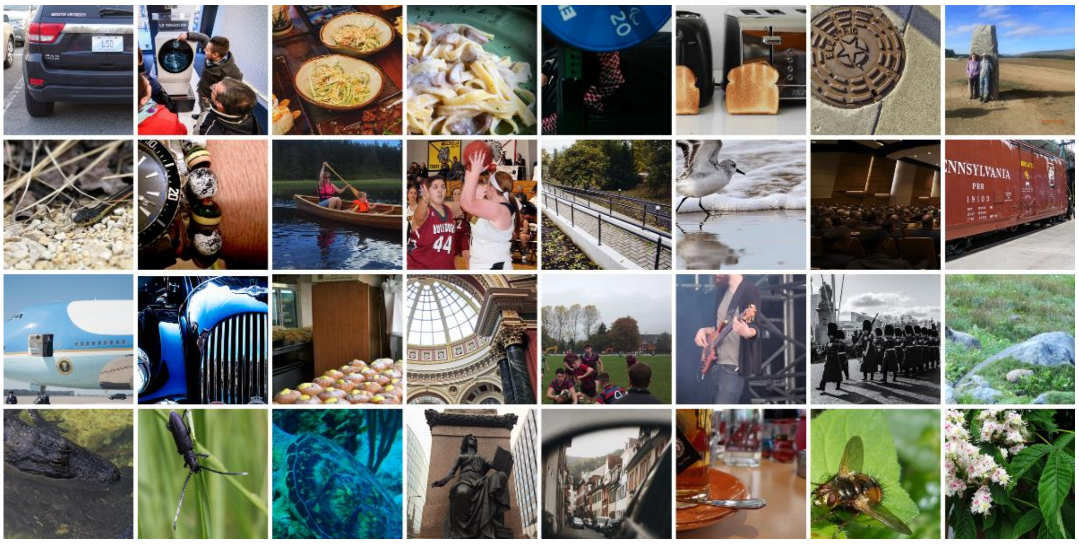

Adversarial perturbations

Figure 7. Targeted adversarial perturbations generated by our method trained against Res-152 with target class “Street Sign”. Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.



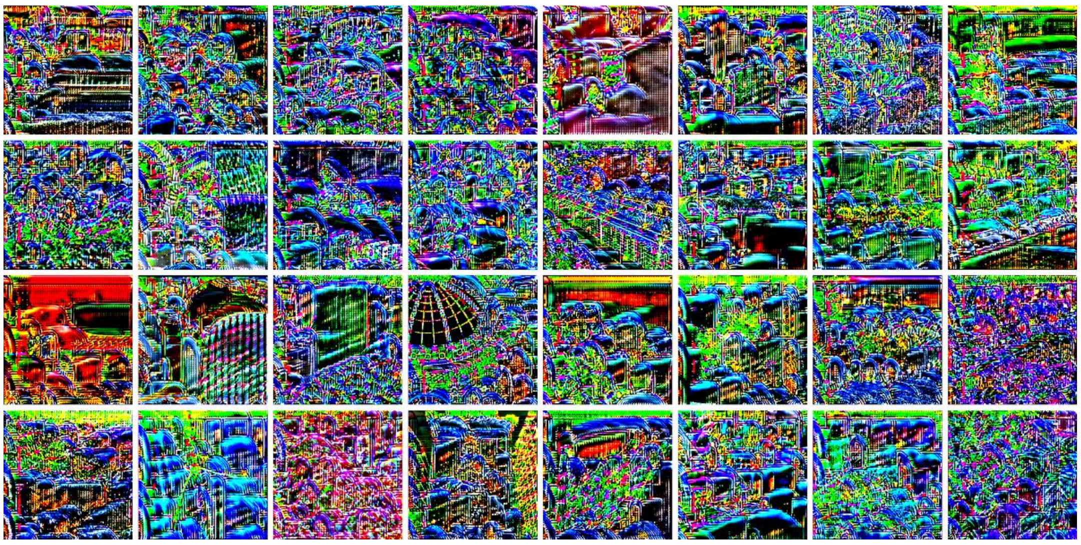


Figure 8. Targeted adversarial perturbations generated by our method trained against Res-152 with target class "Model T". Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.

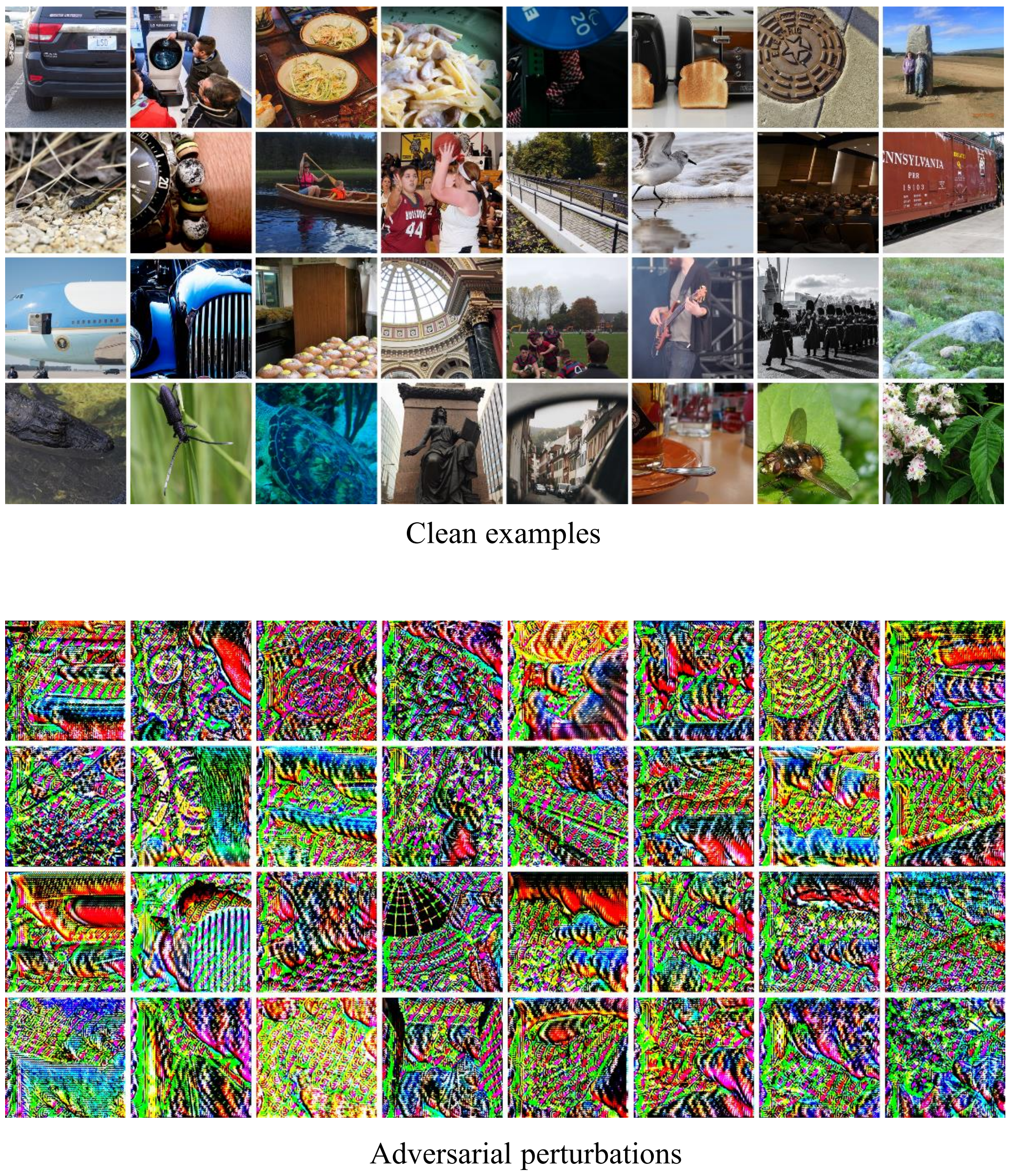


Figure 9. Targeted adversarial perturbations generated by our method trained against Res-152 with target class "French Bulldog". Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.

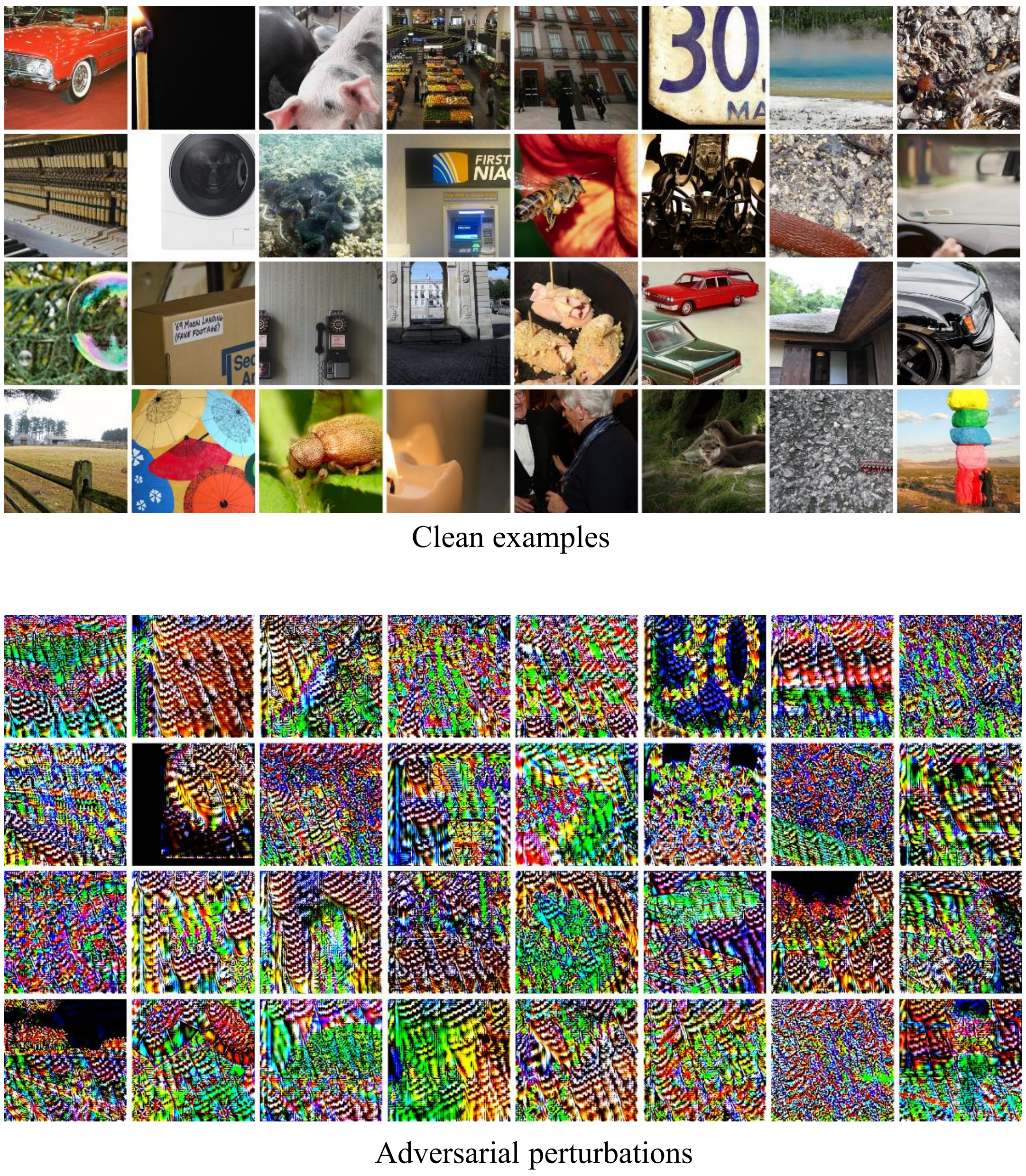


Figure 10. Targeted adversarial perturbations generated by our method trained against Res-152 with target class "Grey Owl". Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.

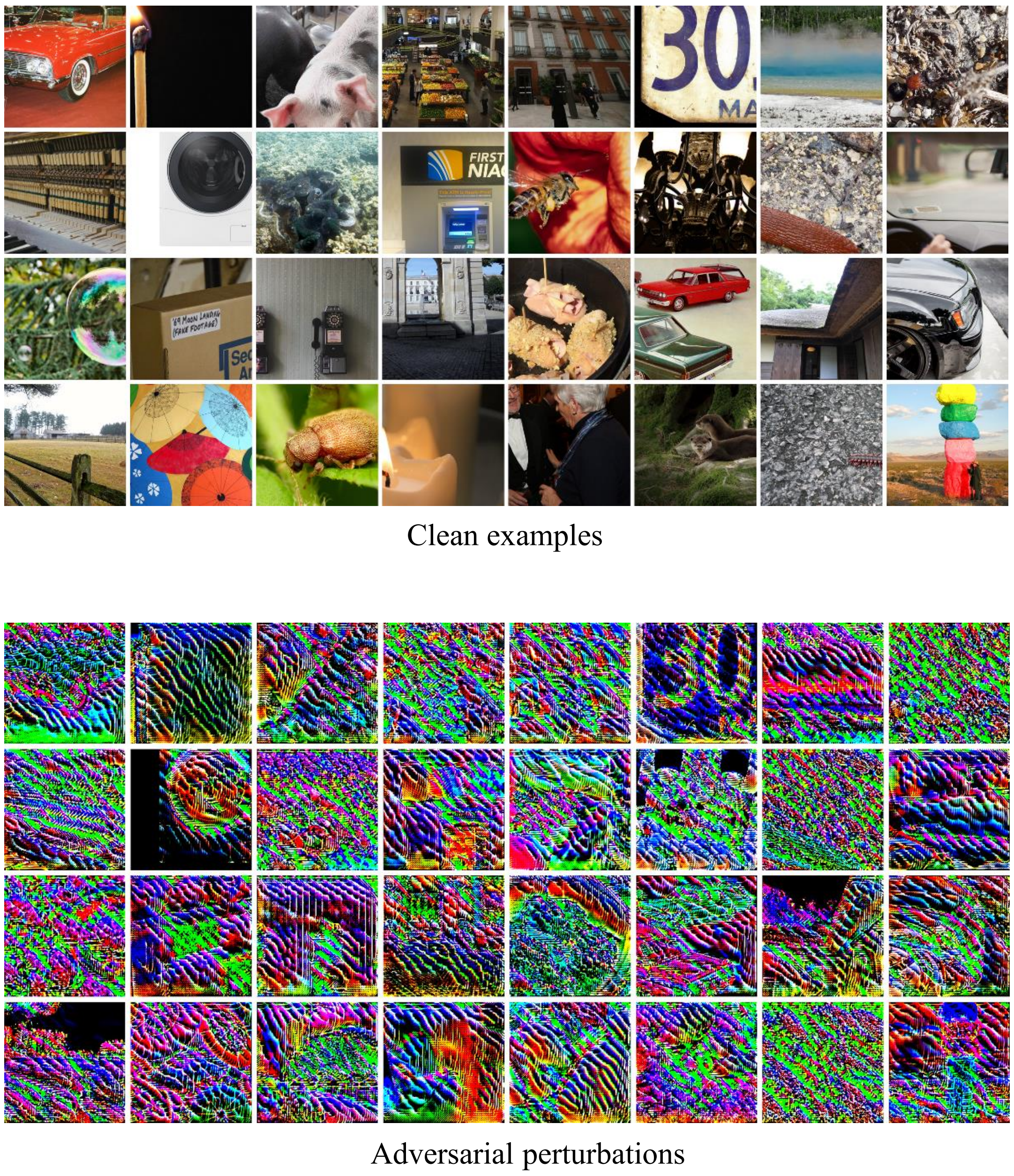


Figure 11. Targeted adversarial perturbations generated by our method trained against Res-152 with target class "Goose". Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.

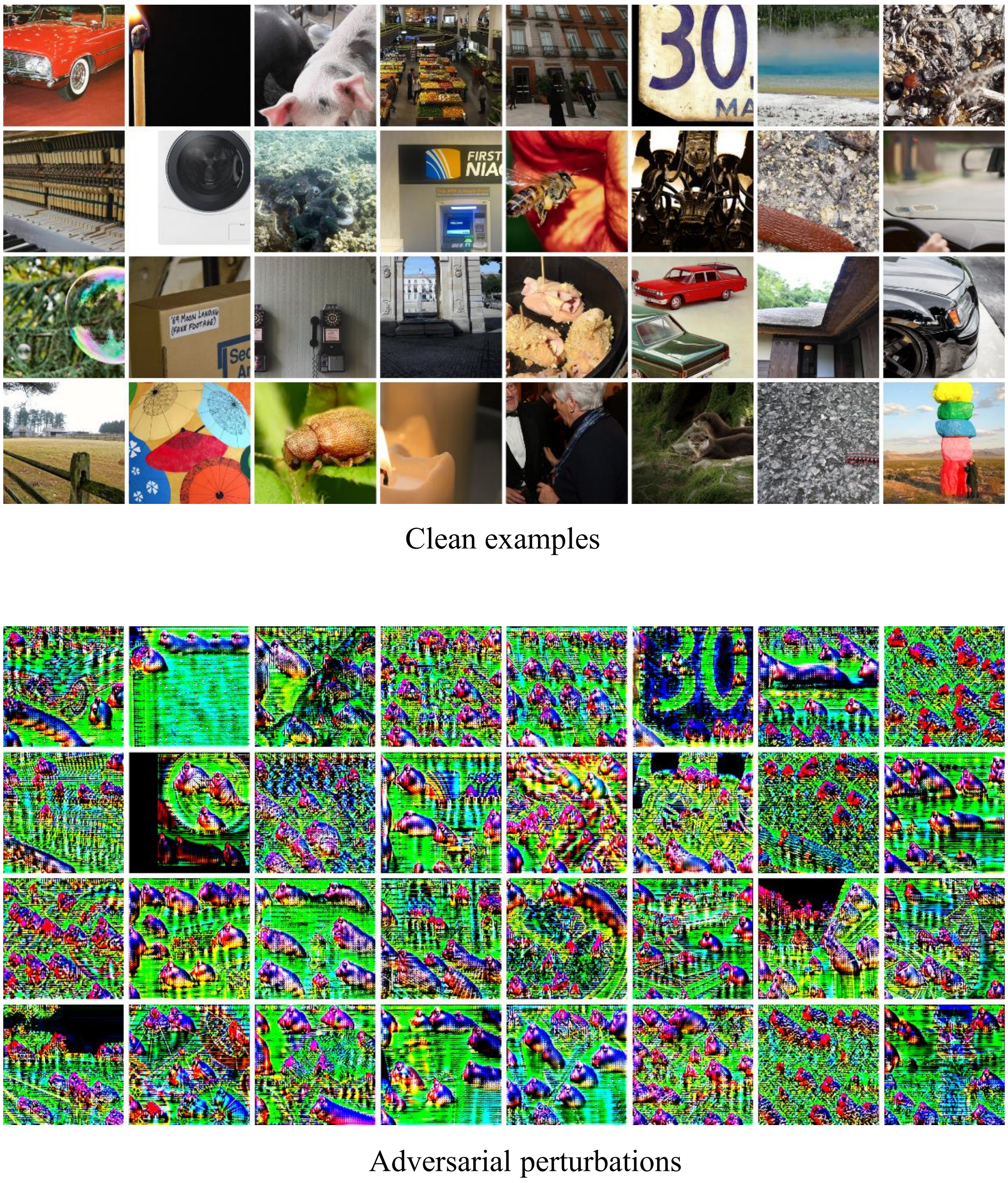


Figure 12. Targeted adversarial perturbations generated by our method trained against Res-152 with target class “Hippopotamus”. Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.

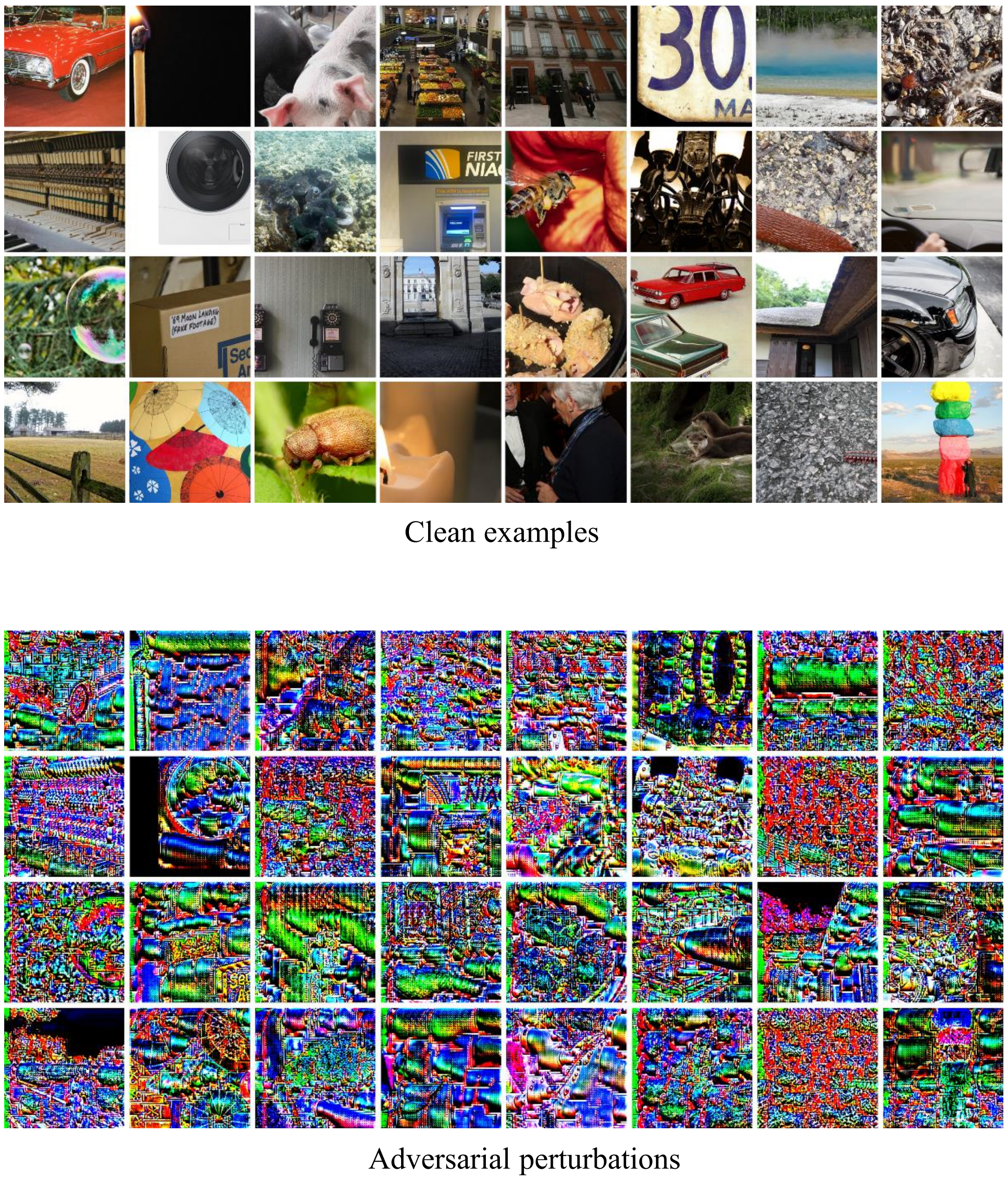


Figure 13. Targeted adversarial perturbations generated by our method trained against Res-152 with target class “Cannon”. Note that there is an underlying dependency between the input instance and the corresponding adversarial perturbation. And the perturbations are produced with strong targeted semantic patterns.