Generate (Non-Software) Bugs to Fool Classifiers

Hiromu Yakura,[1,2*] **Youhei Akimoto,**[1,2] **Jun Sakuma**[1,2]
[1]University of Tsukuba, Japan
[2]RIKEN Center for Advanced Intelligence Project, Japan
hiromu@mdl.cs.tsukuba.ac.jp, {akimoto, jun}@cs.tsukuba.ac.jp

Abstract

In adversarial attacks intended to confound deep learning models, most studies have focused on limiting the magnitude of the modification so that humans do not notice the attack. On the other hand, during an attack against autonomous cars, for example, most drivers would not find it strange if a small insect image were placed on a stop sign, or they may overlook it. In this paper, we present a systematic approach to generate natural adversarial examples against classification models by employing such natural-appearing perturbations that imitate a certain object or signal. We first show the feasibility of this approach in an attack against an image classifier by employing generative adversarial networks that produce image patches that have the appearance of a natural object to fool the target model. We also introduce an algorithm to optimize placement of the perturbation in accordance with the input image, which makes the generation of adversarial examples fast and likely to succeed. Moreover, we experimentally show that the proposed approach can be extended to the audio domain, for example, to generate perturbations that sound like the chirping of birds to fool a speech classifier.

1 Introduction

Despite the great success of deep learning in various fields (LeCun, Bengio, and Hinton 2015), recent studies have shown that deep learning methods are vulnerable to adversarial examples (Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015). In other words, an attacker can make deep learning models misclassify examples by intentionally adding small perturbations to the examples, which are referred to as adversarial examples. Following a study by Szegedy et al. (2014) on image classification, adversarial examples have been demonstrated in many other domains, including natural language processing (Jia and Liang 2017), speech recognition (Carlini and Wagner 2018), and malware detection (Grosse et al. 2017). Moreover, some studies (Eykholt et al. 2018; Chen et al. 2018) have demonstrated a practical attack scenario based on adversarial examples to make autonomous driving systems misclassify stop signs by placing stickers on them.

[*]Also with Teambox Inc., Japan.

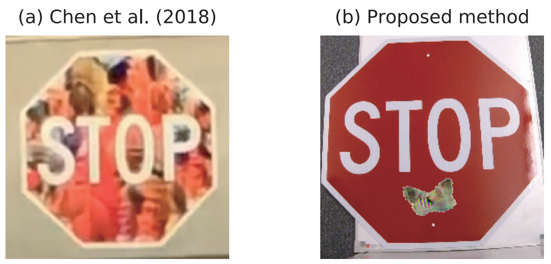

Figure 1: Because road signs are commonly located outside, adding a bug image to such a sign is more natural than perturbing with an arbitrary pattern and is more difficult to notice.

These adversarial examples are most commonly generated by perturbing the input data, in a way that limits the magnitude of the perturbation so that humans do not notice the difference between a legitimate input sample and an adversarial example. When image classification models are attacked, regularization by L_2- or L_∞-norm is often used to make the generated adversarial examples unnoticeable to humans. In another approach, some studies (Xiao et al. 2018; Zhao, Dua, and Singh 2018) introduced generative adversarial networks (GAN) (Goodfellow et al. 2014) to prepare adversarial examples that are likely to appear as natural images. In these methods, GAN is used to generate adversarial examples that are close to the distribution of pre-given natural images.

One drawback of the GAN-based approach is that natural adversarial examples do not always necessarily resemble the input or natural images in some attack scenarios. For example, to attack autonomous cars, placing small bugs on a stop sign would be more natural than modifying the entire sign, as shown in Figure 1. More generally, by overlaying a perturbation that imitates a natural object consistent with the attack scenario on part of the input image, we can create adversarial examples less likely to be noticed. Such a strategy can be extended to domains other than images, for example, to attack speech classification models by generating a perturbation that sounds like environmental noise.

Given the widespread use of image classification and speech recognition in real-world applications to replace human involvement, it is important to analyze such gaps between these models and human perception. In particular, if we can investigate them systematically, it would be possible to design models and training processes to overcome such gaps automatically.

In this paper, we propose a systematic approach to generate natural adversarial examples against classification models by making perturbations that mimic objects or sound signals that are unnoticeable. In this method, we use GAN to generate perturbations so that they imitate collections of reference objects or signals while fooling the target model. Here, because the attack against an image classifier uses a perturbation that is small in relation to the overall image, it is expected that the classification results from the model would be affected not only by the content of the perturbation but also by its location in the image. In other words, optimizing the location can increase the success rate of the attack, but it has the problem that the gradient at the location easily vanishes. Thus, we also introduce a policy gradient method (Sehnke et al. 2010), which is typically used for reinforcement learning, to optimize the location without a gradient.

To confirm the effectiveness of the proposed method, we first conducted experiments with a road sign classifier and an ImageNet classifier. The results confirmed that the obtained adversarial examples could fool an ImageNet classifier, even with bug image perturbations measuring 32×32 pixels, by optimizing the placement location. Moreover, we demonstrate that the proposed approach can also be applied to a speech command classifier with a perturbation that sounds like the chirping of birds. This paper presents a new strategy for creating unnoticeable adversarial attacks that manipulate the content of the perturbation to match the attack scenario.

2 Background

As explained in Section 1, most approaches use optimization algorithms to generate adversarial examples. Let us consider a scenario in which an attacker wishes to modify input image $\boldsymbol{x}$ so that the target model f classifies it with the specific label t. The generation process can be represented as follows:

$$\hat{\boldsymbol{v}} = \operatorname{argmin}_{\boldsymbol{v}} \mathcal{L}_f(\boldsymbol{x} + \boldsymbol{v}, t) + \epsilon \|\boldsymbol{v}\| \quad , \tag{1}$$

where $\mathcal{L}_f$ denotes a loss function that represents how distant the input data are from the given label under f and $\boldsymbol{v} \mapsto \|\boldsymbol{v}\|$ is a norm function to regularize the perturbation so that $\boldsymbol{v}$ becomes unnoticeable to humans. Then, $\boldsymbol{x} + \hat{\boldsymbol{v}}$ is expected to form an adversarial example that is classified as t while it looks similar to $\boldsymbol{x}$.

Earlier approaches, such as Szegedy et al. (2014) and Moosavi-Dezfooli, Fawzi, and Frossard (2016), used L_2-norm to limit the magnitude of the perturbation. In contrast, Su, Vargas, and Sakurai (2017) used L_0-norm to limit the number of modified pixels and showed that even modification of a one-pixel could generate adversarial examples.

More recent studies introduced GAN instead of directly optimizing perturbations (Xiao et al. 2018; Zhao, Dua, and Singh 2018) for the purpose of ensuring the naturalness of adversarial examples. For example, Xiao et al. (2018) trained a discriminator network to distinguish adversarial examples from natural images so that the generator network produced adversarial examples that appeared as natural images. Given the distribution $p_{\boldsymbol{x}}$ over the natural images and the trade-off parameter α, its training process can be represented similarly to that in Goodfellow et al. (2014) as follows:

$$\begin{aligned} \min_{\mathcal{G}} \max_{\mathcal{D}} \; & \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} [\log \mathcal{D}(\boldsymbol{x})] \\ & + \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} [\log(1 - \mathcal{D}(\boldsymbol{x} + \mathcal{G}(\boldsymbol{x})))] \\ & + \alpha\, \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} [\mathcal{L}_f(\boldsymbol{x} + \mathcal{G}(\boldsymbol{x}), t)] \quad . \end{aligned} \tag{2}$$

Then, we can obtain the generator network $\mathcal{G}$ that outputs a perturbation over the entire region of the input image so that the overlaid image fools the target model.

However, we want to generate adversarial examples by placing small objects in the image, not by modifying the entire image. In that respect, Brown et al. (2017) proposed an adversarial patch, which changes the classification results by placing the patch at an arbitrary location in the input image. Given that $A(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{\theta})$ represents an operation of placing a patch $\boldsymbol{p}$ in image $\boldsymbol{x}$ with the location and rotation specified by the parameter $\boldsymbol{\theta}$, its generation process is represented as follows:

$$\hat{\boldsymbol{p}} = \operatorname{argmin}_{\boldsymbol{p}} \mathbb{E}_{\boldsymbol{\theta} \sim p_{\boldsymbol{\theta}}} [\mathcal{L}_f(A(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{\theta}), t)] \quad , \tag{3}$$

where $p_{\boldsymbol{\theta}}$ is the distribution of all possible locations and rotations. We note that this method generates patches with ignoring the attack context, and thus the resulting patches appear to be arresting artifacts that would be instantly recognized by humans.

In contrast, Sharif et al. (2016) proposed an attack against a face recognition model by generating perturbations within a fixed frame that look like eyeglasses. As a result, the person in their adversarial example appears to wear glasses with a strange pattern. Though their approach for making the attack inconspicuous is quite similar to our idea, they only focused on the shape of the perturbation, not its content. Thus, the application of their method is limited to cases in which a strange pattern is acceptable, such as using eccentrically colored eyeglasses to attack a facial recognition model. In addition, this method requires human attackers to design the shape and location of the frame specifically and is not applicable to the audio domain.

Based on the articles reviewed above, we concluded that existing methods minimize the perceptibility by humans mainly by focusing on limited aspects of a perturbation, such as its magnitude or size. In contrast, our approach focuses on the content of perturbations themselves and makes them hard to notice by imitating a natural object or signal, such as bug images or bird songs, to disguise the deception. In addition, unlike Sharif et al. (2016), the proposed method can be systematically used in a wide range of attack scenarios because the GAN can mimic arbitrary objects. We believe that a new strategy of unnoticeable attacks can be created by manipulating the content of the perturbations to conform with the attack situation.

3 Proposed Method

In this paper, we propose a method to generate adversarial examples by placing small perturbations that look similar to a familiar object, such as bugs. We propose two approaches to cover a wide range of attack scenarios. The first one is a patch-based method, which makes the obtained adversarial examples robust against different placement locations. The second one is based on the policy gradient method, which is designed to increase the success rate instead of preserving the robustness of the location change.

3.1 Patch-based Method

As described in Section 2, Xiao et al. (2018) presented a method for making the adversarial examples similar to natural images of any class. Our method extends their approach, shown in Equation (2), so that the obtained perturbations become similar to certain reference images (e.g., bug images).

At the same time, our method has a degree of freedom with regard to the location to place perturbations, unlike that embodied in Equation (2), because our aim is not to modify the entire region but to overlay small patches. Thus, inspired by Brown et al. (2017), we introduce a mechanism to make the perturbations robust against changes in location for the same manner as that in Equation (3). In other words, given that $p_{\boldsymbol{v}}$ represents a distribution over the reference images of the specific object, the proposed method is presented as follows:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{\boldsymbol{v}\sim p_{\boldsymbol{v}}}\left[\log \mathcal{D}\left(\boldsymbol{v}\right)\right] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\left[\log\left(1-\mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}\right)\right)\right)\right] + \alpha\,\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}},\boldsymbol{\theta}\sim p_{\boldsymbol{\theta}}}\left[\mathcal{L}_f\left(A\left(\mathcal{G}\left(\boldsymbol{z}\right),\boldsymbol{x},\boldsymbol{\theta}\right),t\right)\right] \quad , \qquad (4)$$

where $p_{\boldsymbol{z}}$ represents the prior distribution for the generator network in the same manner used in Goodfellow et al. (2014).

3.2 PEPG-based Method

Section 3.1 introduced Equation (4) to produce a perturbation that works without regard to the location in which it is placed. However, compared to Equation (3), its objective variable is changed from the patch itself to the parameter of the network generating the patch. Considering such complexity, the generation process of adversarial examples by the patch-based method is expected to be much harder to deal with.

Therefore, we then optimize the location of perturbations using the policy gradient method, as mentioned in Section 1. This is based on the observation that image classification models often leverage local information in the input image (Ng, Yang, and Davis 2015). In other words, it suggests that an optimal perturbation location exists for each input image to fool the target model. If we can find such a location, the generation process would become easier.

One potential approach is to use a subdifferentiable affine transformation (Jaderberg et al. 2015). In other words, we optimize the parameter of the affine transform to apply it to the perturbation instead of optimizing the location directly. However, this idea is not applicable to our situation because the gradient of the parameter is almost zero. This can be understood by the fact that a very small change in the parameter places the perturbation in the same location. That is, the output probability from the target model is not affected by a small change, consequently, producing zero gradient for the parameter.

Algorithm 1 PEPG-based adversarial example generation

```
Hyperparameters: the batch size $m$, the importance of the loss from the target model $\alpha$, the step size of PEPG $\beta$, and the initial value of the distribution in PEPG $\boldsymbol{\mu}_{\text{init}}, \boldsymbol{\sigma}_{\text{init}}$
Initialize $\boldsymbol{\mu} = \boldsymbol{\mu}_{\text{init}}, \boldsymbol{\sigma} = \boldsymbol{\sigma}_{\text{init}}$
repeat
    // Training of the discriminator
    • Sample $m$ noises $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from $p_{\boldsymbol{z}}$
    • Sample $m$ examples $\{\boldsymbol{v}^{(1)}, \dots, \boldsymbol{v}^{(m)}\}$ from $p_{\boldsymbol{v}}$
    • Update the parameter of the discriminator using the following gradient:
      $\nabla \frac{1}{m}\sum_{i=1}^{m}\left[\log \mathcal{D}\left(\boldsymbol{v}^{(i)}\right) + \log\left(1-\mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$
    // Generation of adversarial examples
    • Sample $m$ noises $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from $p_{\boldsymbol{z}}$
    • Sample $m$ location and rotation parameters $\{\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}\}$ from $\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\sigma}\right)$
    • Create adversarial examples for $i = 1, \dots, m$:
      $\tilde{\boldsymbol{x}}^{(i)} = \left(A\left(\mathcal{G}\left(\boldsymbol{z}^{(i)}\right), \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\right)\right)$
    // Training of the generator
    • Calculate loss value for the adversarial examples $\ell^{(i)} = \mathcal{L}_f\left(\tilde{\boldsymbol{x}}^{(i)}, t\right)$ for $i = 1, \dots, m$
    • Update the parameter of the discriminator using the following gradient:
      $\nabla \frac{1}{m}\sum_{i=1}^{m}\left[\log\left(1-\mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}^{(i)}\right)\right)\right) + \alpha\ell^{(i)}\right]$
    // Training of the PEPG distribution
    • Update $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ using $\{-\ell^{(1)}, \dots, -\ell^{(m)}\}$ as the rewards and $\frac{1}{m}\sum_{i=1}^{m} -\ell^{(i)}$ as the baseline reward based on the PEPG algorithm
until a sufficient number of adversarial examples recognized as $t$ are generated
```

Therefore, we use a parameter-exploring policy gradient (PEPG) method (Sehnke et al. 2010) to optimize the location and rotation of the perturbation. This method assumes the distribution over the parameters and trains the hyperparameters of the distribution instead of the parameters themselves. The advantage of the method is that it can explore a wide range of parameters in addition to not needing the gradients of the parameters.

When using the PEPG method, our method initially applies various locations sampled from the prior Gaussian distribution. Then, based on the loss value from the target model during the trials, it gradually updates the hyperparameters so that the locations with small losses are likely to be sampled from the distribution. By doing so, we can train the hyperparameters and the generator network simultaneously.

The detailed process is shown in Algorithm 1. Here, we used symmetric sampling and reward normalization in the PEPG to accelerate convergence, as suggested by Sehnke et al. (2010).

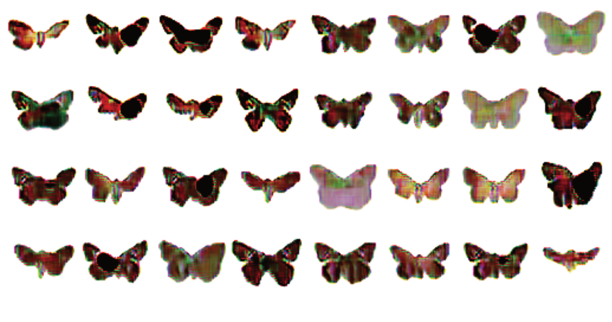
Figure 2: Examples of moth images generated by the pretrained GAN.

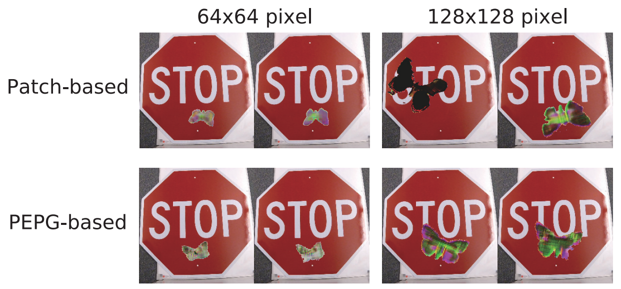

Figure 3: Examples of adversarial examples that made the road sign classifier recognize them as "Speed Limit 80."

4 Experimental Results

To test our approaches, we first performed a preliminary experiment with a road sign classifier to investigate the feasibility of the proposed methods. Then, we conducted an experiment with an ImageNet classifier to confirm the availability of the methods against a wide range of input images and target classes. In both experiments, we compared the patch-based and PEPG-based methods[1].

For the architecture of the GAN, we used WGAN-GP (Gulrajani et al. 2017) and changed the size of the output perturbations among 32×32, 64×64, and 128×128 pixels to evaluate the effect of the perturbation size. For all tests, we used an image dataset of moths from Costa Rica (Rodner et al. 2015) for the reference images, that is, p_v in Equation (4), to make the perturbations look like moths. To ensure the stability of the training, we pretrained the GAN without the loss value from the target model and confirmed that it outputs moth-like images, as shown in Figure 2.

4.1 Road Sign Classifier

We first explored the feasibility of the proposed methods with a road sign classifier trained on the German Traffic Sign Recognition Benchmark (Stallkamp et al. 2012), as done by Eykholt et al. (2018). We used their convolutional neural network-based classifier for the target model[2], which was reported to show 95.7% accuracy. For the input image, we used the same images of stop signs measuring 256×256 pixels and tried to make them be recognized as Speed Limit 80 in the same manner as done by Eykholt et al. (2018). Regarding each combination of the perturbation size and generation approach, we examined whether we could obtain adversarial examples in a given number of iterations.

The obtained adversarial examples are presented in Figure 3. The result indicates that the success of the generation highly depends on the size of the perturbation; that is, we could not generate an adversarial example after 20,000 iterations when the perturbation size was limited to 32×32. This corresponds to the result from Brown et al. (2017): the larger the adversarial patch becomes, the higher the success rate became.

By comparing the patch-based and PEPG-based methods, we found that the PEPG-based method took much less time. For example, in the case of 128×128 pixels, the PEPG-based method took about 6 minutes and 753 iterations to generate 100 adversarial examples, whereas the patch-based method took about an hour and 5,340 iterations. This difference suggests that optimization of the location and rotation of the perturbation helps achieve success by finding the sensitive region of the input image.

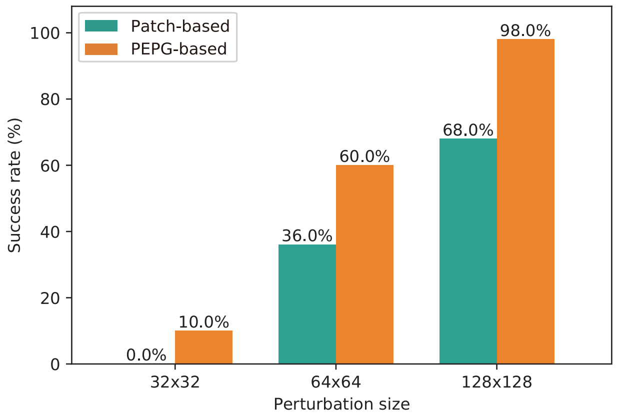

Figure 4: Success rate for each combination of perturbation size and generation approach against the ImageNet classifier.

4.2 ImageNet Classifier

We then evaluated the proposed methods using an Inception-V3 classifier (Szegedy et al. 2016) pretrained on ImageNet, which is distributed by TensorFlow[3]. For the combination of the input image and target class, we used the same tasks as the NIPS '17 competition on adversarial examples (Kurakin et al. 2018). Here, because the proposed methods involve a training process for each combination and require some time, we selected the first 50 tasks.

In the same manner described in Section 4.1, we examined whether we could obtain 100 adversarial examples in 50,000 iterations for each combination of perturbation size

[1]The source code for both experiments is available at https://github.com/hiromu/adversarial_examples_with_bugs.

[2]https://github.com/evtimovi/robust_physical_perturbations

[3]https://github.com/tensorflow/models/tree/master/research/slim

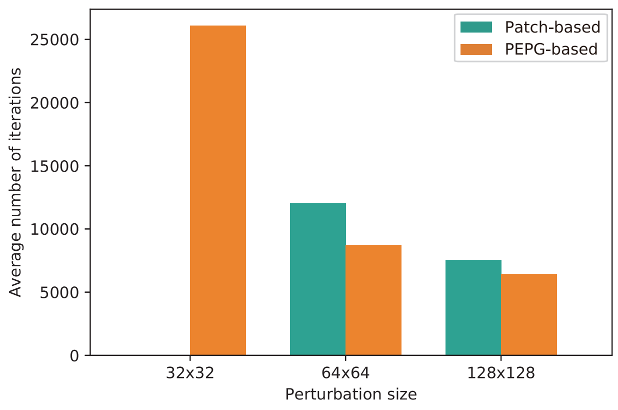

Figure 5: Average number of iterations required to generate 100 successful adversarial examples.

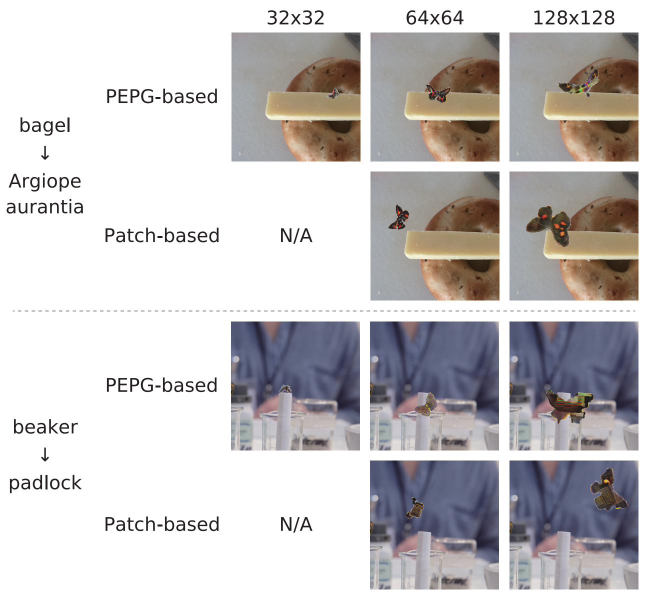

Figure 6: Examples of adversarial examples generated against the ImageNet classifier.

and generation approach. Then, we compared the success rate over 50 tasks and the average number of iterations in successful cases.

The results are shown in Figure 4 and Figure 5. Some images of the obtained adversarial examples are also shown in Figure 6. We confirmed that the larger size of the perturbation helps the generation of adversarial examples; that is, it increases the success rate and decreases the required number of iterations.

We also confirmed that the optimization of the perturbation location by the PEPG algorithm helps the generation. In particular, the PEPG-based method succeeded in generating adversarial examples with perturbations of only 32×32 pixels in five tasks, whereas the patch-based method completely failed. In addition, in the case of 128×128 pixels, the PEPG-based method succeeded in generating 100 adversarial examples for 49 tasks. We also note that, in the remaining (50th) task, our method could not generate 100 examples within 50,000 iterations but obtained 91 adversarial examples. From these points, the proposed approach of optimizing the location is shown to be effective for increasing the success rate.

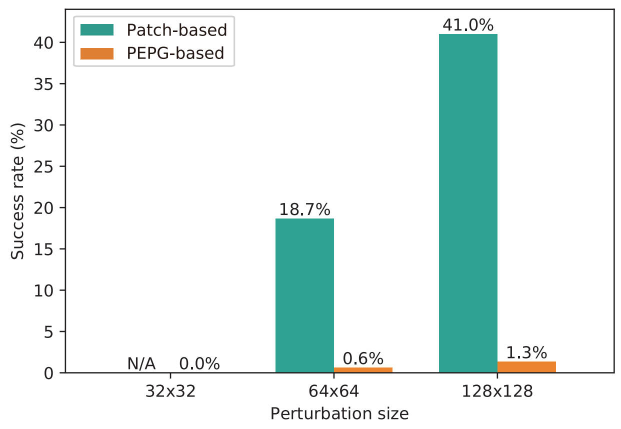

Figure 7: Ratio of samples crafted by random relocation of the perturbations and classified into the target label.

4.3 Analysis

In this section, we analyze characteristics of the proposed method based on the above results and additional investigations. We first examine the robustness of the perturbations obtained by the patch-based method. Then, we examine the effectiveness of the PEPG algorithm for the successful generation.

Robustness of the Patch-based Perturbations As discussed in Section 3.1, the patch-based method generates perturbations that are robust against change in location. Thus, it is expected that we can produce new adversarial examples by relocating the perturbations in a fashion similar to that described in Brown et al. (2017). We tested the idea by crafting 10 samples from each adversarial example obtained as described in Section 4.2 with a random relocation and verifying their classification result.

Figure 7 shows the success rate for adversarial example multiplication, that is, the ratio of the crafted samples that are classified to the same target label as the original adversarial example. These results suggest that the perturbations obtained by the patch-based method are more robust than the results for the PEPG-based method. In particular, we found that successful samples among the adversarial examples obtained by the patch-based method (Figure 8) have perturbations in various locations, whereas the samples obtained from the PEPG-based method (Figure 9) exhibit a limited number of successful cases where the perturbation location is similar to that in the original adversarial examples. We note that a success rate is roughly comparable to the adversarial patch (Brown et al. 2017), which showed the success rate of about 20% when modifying 3% of the pixels in the

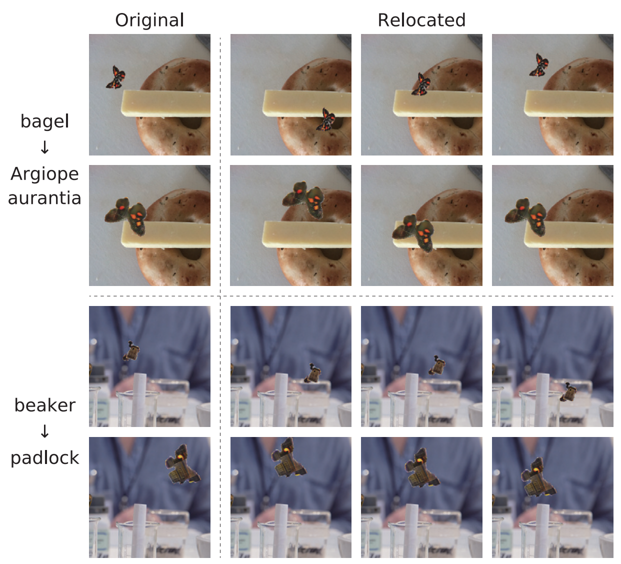

Figure 8: Examples of successful samples crafted from adversarial examples obtained by the patch-based method.

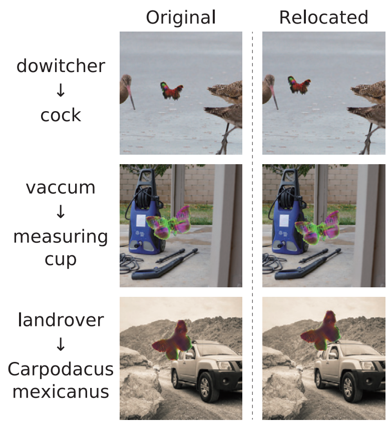

Figure 9: Examples of successful samples crafted from adversarial examples obtained by the PEPG-based method.

image[4], although the proposed method uses a GAN to generate perturbations instead of optimizing them directly.

Effectiveness of the PEPG Algorithm We confirmed that the perturbations obtained by the PEPG-based method show limited robustness regarding the relocation. Conversely, it implies that the PEPG algorithm successfully finds the limited area where the perturbation functions as an adversarial example. For example, in Figure 6, we found that all adversarial examples obtained by the PEPG-based method have perturbations in similar locations in the same input image.

Thus, we examined how the classification results are changed when we move the perturbations to a different location. The heatmaps of Figure 10 show the confidence that the image obtained by shifting the center of the perturbation

[4]The perturbation of 64×64 pixels accounts for only 4.6% of the image of 299×299 pixels.

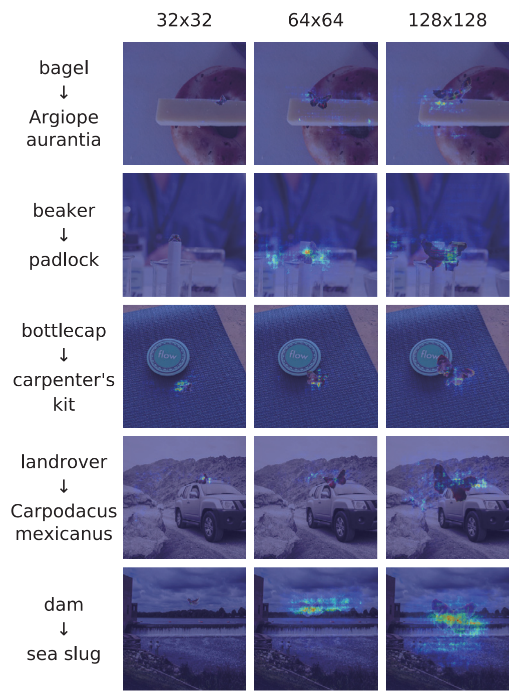

Figure 10: Confidence of the target label obtained when the center of the perturbation is shifted to each pixel.

to each pixel is classified as the target label. The results confirm that the PEPG algorithm successfully finds the limited location where the perturbation functions as an adversarial example, especially when the perturbation is small.

In addition, we compared these adversarial examples with the activation heatmaps of the corresponding input image obtained by Grad-CAM (Selvaraju et al. 2017), which shows the class-discriminative regions. The results are shown in Figure 11. We found that the perturbations were placed near the most discriminative regions in the input images, regardless of the size of the perturbations. These analyses support the effectiveness of the PEPG-based method for optimizing the perturbation placement, which would contribute to its higher success rate compared to the patch-based method, as presented in Section 4.2.

In summary, our analysis suggests the existence of some trade-off between the robustness and the success rate when applying the patch-based and PEPG-based method, as discussed in Section 3. In other words, we can choose an appropriate method from them to fit different attack situations.

5 Application: Audio Adversarial Example

In this paper, we presented a new approach for generating adversarial examples by making perturbations imitate a specific object. Up to this point, we discussed attacks that targeting image classification models, but this idea can be applied to other domains. For example, if we can generate an adversarial example that fools speech recognition models but sounds like the chirping of small birds, we would be able to control personal assistants based on speech interactions

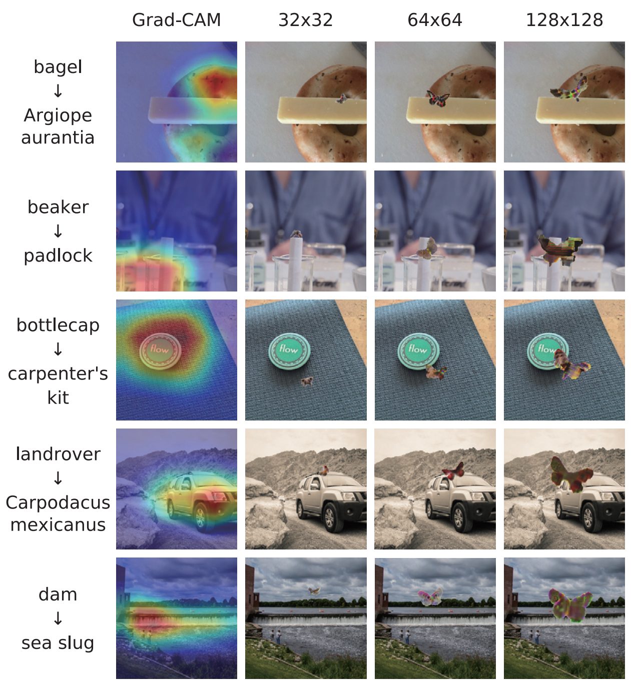

Figure 11: Comparison between the activation maps of the input image and the location of the perturbation optimized by the PEPG-based method.

(e.g., Siri or Google Assistant) without being noticed by humans. To investigate the potential of the proposed approach, we tested whether we can generate such audio adversarial examples using GAN in the same manner as used for images (Section 3).

As with image adversarial examples, there are many proposed methods for generating audio adversarial examples (Alzantot, Balaji, and Srivastava 2017; Carlini and Wagner 2018; Yakura and Sakuma 2019; Schönherr et al. 2019). For example, Alzantot, Balaji, and Srivastava (2017) generated adversarial examples against a speech command classifier and attained a success rate of 87%. Carlini and Wagner (2018) proposed a generation method that targets DeepSpeech (Hannun et al. 2014) by extending their previous work (Carlini and Wagner 2017b) from image adversarial examples. Yakura and Sakuma (2019) realized a physical attack against DeepSpeech by simulating noise and reverberation in the physical world during audio generation. With respect to perceptibility by humans, Schönherr et al. (2019) optimized less noticeable perturbations by exploiting a psychoacoustic model. However, none of them tried to manipulate the content of the perturbation to match the attack scenario, such as camouflaging the perturbation with environmental noise like birds chirping.

5.1 Settings

For the target model, we used the same speech command classifier (Sainath and Parada 2015) as Alzantot, Balaji, and Srivastava (2017), which is distributed by TensorFlow[5]. We employed the architecture WaveGAN (Donahue, McAuley, and Puckette 2019), which is based on WGAN-GP (Gulrajani et al. 2017), to generate perturbations[6].

For the reference audio, we used the VB100 Bird Dataset (Ge et al. 2016) to make perturbations that sounded like chirping birds. The generated perturbations were added to one of two audio clips from the Speech Commands Dataset (Warden 2018), which says “yes” or “no.” Then, we examined whether the obtained adversarial examples were classified as the target label; in this case, we chose “stop” among 12 labels of the model output that included ”yes” and ”no.”

5.2 Results

We succeeded in generating adversarial examples that were classified as “stop,” although they sound like someone saying “yes” or “no” with chirping birds in a background. Some of the obtained adversarial examples are available at https://yumetaro.info/projects/bugs-ae/.

We also conducted a listening experiment with 25 human participants using Amazon Mechanical Turk in a manner similar to that in Alzantot, Balaji, and Srivastava (2017). First, we presented two obtained adversarial examples made from the clips saying “yes” and ”no” and asked the participants to write down the words they heard. Then, we asked them to write down anything abnormal that they detected. The responses are summarized as follows:

- For the transcription tasks:
  - All participants transcribed the contents of the original clips.
  - No participant identified the target word.
- In the collected comments:
  - Six participants noted the existence of birds chirping in the background.
  - No participant felt suspicious about the clips.

These responses suggest that the proposed approach for making perturbations mimic unnoticeable objects also works in the audio domain and has very low perception for humans.

6 Limitations and Future Directions

While our results confirmed the effectiveness of the new approach of generating adversarial examples by making perturbations imitate a specific object or signal, there are still some limitations. In particular, though we experimentally investigated the effectiveness of the PEPG-based method in Section 4.3, its theoretical background is still unclear. One possible clue would be a transition of the reward of the PEPG algorithm, considering that literature on reinforcement learning often use it to examine the training process in detail.

[5]https://www.tensorflow.org/tutorials/sequences/audio_recognition

[6]The source code is available at https://github.com/hiromu/adversarial_examples_with_bugs.

In this respect, further investigations are needed to clarify the prospects of this new idea of applying the PEPG algorithm to the generation of adversarial examples. For example, though we fixed the size of the perturbations in Section 4, it would be possible also to optimize the size if we can design an appropriate penalty term. For the audio adversarial examples, the start timing of synthesizing the perturbation can be optimized so as to find the best position to hide the birds chirping.

The discussion of the attack scenarios is also an important research direction. As discussed in Section 1, the proposed approach can increase the magnitude of the perturbation without being noticed by humans. Given that many defense methods have been defeated with a relatively small magnitude of the perturbation (Carlini and Wagner 2017a), this approach potentially opens up further attack possibilities, which we must discuss to ensure the safety of socio-technical systems based on machine learning.

7 Conclusion

In this paper, we proposed a systematic approach for generating natural adversarial examples by making perturbations imitate specific objects or signals, such as bugs images. We presented its feasibility for attacking image classifiers by leveraging GAN to generate perturbations that imitated the reference images and fooled the target model. We also confirmed that the optimization of the perturbation location using the PEPG algorithm led to the successful generation of adversarial examples. Furthermore, we experimentally showed that the proposed approach could be extended to the audio domain, such as generating a perturbation that sounds like the chirping of birds. Our results provide a new direction for creating natural adversarial examples by manipulating the content of the perturbation instead of trying to make the perturbation imperceptible by limiting its magnitude.

Acknowledgments

This work was supported by KAKENHI 19H04164 and 18H04099.

References

Alzantot, M.; Balaji, B.; and Srivastava, M. B. 2017. Did you hear that? adversarial examples against automatic speech recognition. In *Proceedings of the 2017 NIPS Workshop on Machine Deception*, 1–6. San Diego, CA: NIPS Foundation.

Brown, T. B.; Mané, D.; Roy, A.; Abadi, M.; and Gilmer, J. 2017. Adversarial patch. In *Proceedings of the 2017 NIPS Workshop on Machine Learning and Computer Security*, 1–5. San Diego, CA: NIPS Foundation.

Carlini, N., and Wagner, D. A. 2017a. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 3–14. New York, NY: ACM.

Carlini, N., and Wagner, D. A. 2017b. Towards evaluating the robustness of neural networks. In *Proceedings of the 38th IEEE Symposium on Security and Privacy*, 39–57. Washington, DC: IEEE Computer Society.

Carlini, N., and Wagner, D. A. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *Proceedings of the 1st IEEE Deep Learning and Security Workshop*, 1–7. Washington, DC: IEEE Computer Society.

Chen, S.; Cornelius, C.; Martin, J.; and Chau, D. H. P. 2018. Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector. In *Proceedings of the 2018 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, volume 11051, 52–68. Cham, Switzerland: Springer.

Donahue, C.; McAuley, J. J.; and Puckette, M. S. 2019. Adversarial audio synthesis. In *Proceedings of the 7th International Conference on Learning Representations*, 1–16. La Jolla, CA: ICLR.

Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; and Song, D. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1625–1634. Washington, DC: IEEE Computer Society.

Ge, Z.; McCool, C.; Sanderson, C.; Wang, P.; Liu, L.; Reid, I. D.; and Corke, P. I. 2016. Exploiting temporal information for dcnn-based fine-grained object classification. In *Proceedings of the 2016 International Conference on Digital Image Computing: Techniques and Applications*, 1–6. New York, NY: IEEE.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial nets. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, 2672–2680. San Diego, CA: NIPS Foundation.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations*, 1–11. La Jolla, CA: ICLR.

Grosse, K.; Papernot, N.; Manoharan, P.; Backes, M.; and McDaniel, P. D. 2017. Adversarial examples for malware detection. In *Proceedings of the 22nd European Symposium on Research in Computer Security*, volume 10493, 62–79. Cham, Switzerland: Springer.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, 5769–5779. San Diego, CA: NIPS Foundation.

Hannun, A. Y.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; and Ng, A. Y. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv* 1412.5567:1–12.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; and Kavukcuoglu, K. 2015. Spatial transformer networks. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, 2017–2025. San Diego, CA: NIPS Foundation.

Jia, R., and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2021–2031. Stroudsburg, PA: ACL.

Kurakin, A.; Goodfellow, I. J.; Bengio, S.; Dong, Y.; Liao, F.; Liang, M.; Pang, T.; Zhu, J.; Hu, X.; Xie, C.; Wang, J.; Zhang, Z.; Ren, Z.; Yuille, A. L.; Huang, S.; Zhao, Y.; Zhao, Y.; Han, Z.; Long, J.; Berdibekov, Y.; Akiba, T.; Tokui, S.; and Abe, M. 2018. Adversarial attacks and defences competition. In *The NIPS '17 Competition: Building Intelligent Systems*. Cham, Switzerland: Springer. 195–231.

LeCun, Y.; Bengio, Y.; and Hinton, G. E. 2015. Deep learning. *Nature* 521(7553):436–444.

Moosavi-Dezfooli, S.; Fawzi, A.; and Frossard, P. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*, 2574–2582. Washington, DC: IEEE Computer Society.

Ng, J. Y.; Yang, F.; and Davis, L. S. 2015. Exploiting local features from deep networks for image retrieval. In *Proceedings of the 2015 CVPR Workshop on Deep Vision*, 53–61. Washington, DC: IEEE Computer Society.

Rodner, E.; Simon, M.; Brehm, G.; Pietsch, S.; Wägele, J.; and Denzler, J. 2015. Fine-grained recognition datasets for biodiversity analysis. In *Proceedings of the 2015 CVPR Workshop on Fine-grained Visual Classification*, 1–4. Washington, DC: IEEE Computer Society.

Sainath, T. N., and Parada, C. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association*, 1478–1482. Baixas, France: ISCA.

Schönherr, L.; Kohls, K.; Zeiler, S.; Holz, T.; and Kolossa, D. 2019. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, 1–15. Reston, VA: The Internet Society.

Sehnke, F.; Osendorfer, C.; Rückstieß, T.; Graves, A.; Peters, J.; and Schmidhuber, J. 2010. Parameter-exploring policy gradients. *Neural Networks* 23(4):551–559.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the 16th IEEE International Conference on Computer Vision*, 618–626. Washington, DC: IEEE Computer Society.

Sharif, M.; Bhagavatula, S.; Bauer, L.; and Reiter, M. K. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 1528–1540. New York, NY: ACM.

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 32:323–332.

Su, J.; Vargas, D. V.; and Sakurai, K. 2017. One pixel attack for fooling deep neural networks. *arXiv* 1710.08864:1–15.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations*, 1–10. La Jolla, CA: ICLR.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826. Washington, DC: IEEE Computer Society.

Warden, P. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv* 1804.03209:1–11.

Xiao, C.; Li, B.; Zhu, J.; He, W.; Liu, M.; and Song, D. 2018. Generating adversarial examples with adversarial networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 3905–3911. IJCAI.

Yakura, H., and Sakuma, J. 2019. Robust audio adversarial example for a physical attack. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 5334–5341. IJCAI.

Zhao, Z.; Dua, D.; and Singh, S. 2018. Generating natural adversarial examples. In *Proceedings of the 6th International Conference on Learning Representations*, 1–15. La Jolla, CA: ICLR.