Simple iterative method for generating targeted universal adversarial perturbations

algorithms

MDPI

Article

Simple Iterative Method for Generating Targeted Universal Adversarial Perturbations

Hokuto Hirano and Kazuhiro Takemoto *

Department of Bioscience and Bioinformatics, Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan; hirano.hkt@gmail.com

* Correspondence: takemoto@bio.kyutech.ac.jp

Received: 8 September 2020; Accepted: 20 October 2020; Published: 22 October 2020

Abstract: Deep neural networks (DNNs) are vulnerable to adversarial attacks. In particular, a single perturbation known as the universal adversarial perturbation (UAP) can foil most classification tasks conducted by DNNs. Thus, different methods for generating UAPs are required to fully evaluate the vulnerability of DNNs. A realistic evaluation would be with cases that consider targeted attacks; wherein the generated UAP causes the DNN to classify an input into a specific class. However, the development of UAPs for targeted attacks has largely fallen behind that of UAPs for non-targeted attacks. Therefore, we propose a simple iterative method to generate UAPs for targeted attacks. Our method combines the simple iterative method for generating non-targeted UAPs and the fast gradient sign method for generating a targeted adversarial perturbation for an input. We applied the proposed method to state-of-the-art DNN models for image classification and proved the existence of almost imperceptible UAPs for targeted attacks; further, we demonstrated that such UAPs can be easily generated.

Keywords: deep neural networks; adversarial attacks; image classification; security and privacy

1. Introduction

Deep neural networks (DNNs) are widely used for image classification, a task in which an input image is assigned a class from a fixed set of classes. For example, DNN-based image classification has applications in medical science (e.g., medical image-based diagnosis [1]) and self-driving technology (e.g., detecting and classifying traffic signs [2]).

However, DNNs are known to be vulnerable to adversarial examples [3], which are input images that cause misclassifications by DNNs and are generally generated by adding specific, imperceptible perturbations to the original input images that have been correctly classified using DNNs. Many methods for generating adversarial examples have been proposed [4,5]. For example, the fast gradient sign method (FGSM) [3] generates adversarial examples with a linear approximation; specifically, it performs a one step gradient update along the direction of the sign of the gradient of the loss function at each pixel. The basic iterative method [6] is an iterative version of the FGSM; in particular, it repeats the one step gradient update (i.e., FGSM) while the size of the adversarial perturbation is limited by a specified upper bound. DeepFool [7] considers a linear approximation and generates adversarial examples using the fact that the minimal perturbation of an affine classifier is the distance to the separating affine hyperplane. Carlini and Wagner [8] proposed an optimization algorithm for generating adversarial examples; specifically, they considered an optimization problem for finding the smallest adversarial perturbation that can fool the DNNs and solved the problem using gradient descent. The existence of adversarial examples questions the generalization ability of DNNs, reduces model interpretability, and limits the applications of deep learning in safety- and security-critical environments [9].

However, these methods have only considered input-dependent adversarial attacks (i.e., an individual adversarial perturbation is used such that each input image is misclassified). Such adversarial attacks are difficult tasks because they require high computational costs. More realistic adversarial attacks must be further considered. Notably, a single perturbation that can induce DNN failure in most image classification tasks is also generatable as a universal adversarial perturbation (UAP) [10,11]. UAP-based adversarial attacks can be more straightforward to implement by adversaries in real-world environments. As UAPs are image agnostic, adversaries do not need to consider input images when attacking DNNs; as a result, they can result in failed DNN-based image classification at lower costs. The vulnerability in DNNs to adversarial attacks (UAPs, in particular) is a security concern for practical applications of DNNs [10–12]. The development of methods for generating UAPs is required to evaluate the reliability and safety of DNNs against adversarial attacks.

A simple iterative method [10], a representative method, for generating UAPs has been proposed. To generate a UAP, this method iteratively updates a perturbation by additively obtaining individual adversarial perturbation for an input image randomly selected from an input image set. Khrulkov and Oseledets [13] proposed a method for generating UAPs based on computing the singular vectors of the Jacobian matrices of the hidden layers of a DNN. Mopuri et al. [14] proposed a network for adversary generation, a generative approach to model the distribution of UAPs, inspired by generative adversarial networks (GANs) [15]. However, these methods are limited to non-targeted attacks that cause misclassification (i.e., a task failure resulting in an input image being assigned an incorrect class).

More realistic cases need to consider targeted attacks, wherein generating a UAP would cause the DNN to classify an input image into a specific class (e.g., into the "diseased" class in medical diagnosis) [16]. The existence of targeted UAPs indicates that adversaries can control DNN-based image classification and may result in more significant security concerns compared with non-targeted attacks.

Several methods for generating UAPs for targeted attacks have been proposed. Hayes and Danezis [17] proposed a method based on a generative network model. Specifically, the method uses the universal adversarial network, trained similarly to the generator network in a GAN [15]. However, the generative network model is difficult to apply to large images (e.g., the ImageNet image dataset [18]) because it requires high computational costs. Brown et al. [19] proposed the targeted adversarial patch approach for targeted universal adversarial attacks. This approach adds an image patch to the input images to generate adversarial examples and seeks the location and transformations (e.g., rotations and scaling) of the patch on the input images that maximize the average confidence score for a target class of the adversarial examples. However, such adversarial patches are perceptible; thus, defenders can easily detect adversarial attacks.

A simpler algorithm for generating almost imperceptible UAPs for targeted attacks is required for practical applications. Thus, herein, we propose a simple iterative method to generate such targeted UAPs by extending the iterative method for generating non-targeted UAPs [10] and show the validity of the proposed method using representative DNN models for image classification.

2. Materials and Methods

2.1. Targeted Universal Adversarial Perturbations

Our algorithm (Algorithm 1) for generating UAPs for targeted attacks is an extension of the simple iterative algorithm for generating UAPs for non-targeted attacks [10]. Similar to the non-targeted UAP algorithm, our algorithm considers a classifier $C(\boldsymbol{x})$ that returns the class or label (with the highest confidence score) for an input image $\boldsymbol{x}$. The algorithm starts with $\boldsymbol{\rho} = \mathbf{0}$ (no perturbation) and iteratively updates the UAP $\boldsymbol{\rho}$ under the constraint that the L_p norm of the perturbation is equal to or less than a small value ξ (i.e., $\|\boldsymbol{\rho}\|_p \leq \xi$) by additively obtaining an adversarial perturbation for an input image $\boldsymbol{x}$, which is randomly selected from an input image set $\boldsymbol{X}$ without replacement. These iterative updates continue up till the termination conditions have been satisfied. Unlike the non-targeted UAP algorithm, our algorithm uses the fast gradient sign method for targeted attacks (tFGSM) to generate targeted

UAPs, whereas the non-targeted UAP algorithm uses a method (e.g., DeepFool [7]) that generates a non-targeted adversarial example for an input image.

tFGSM generates a targeted adversarial perturbation $\psi(\boldsymbol{x}, y)$ that causes an image $\boldsymbol{x}$ to be classified into the target class y using the gradient $\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}, y)$ of the loss function with respect to pixels [3,20]. The gradient can be computed efficiently using backpropagation (i.e., by computing the chain rule backwards) [3]. Specifically, the forward pass computes values from the inputs (pixels) to the output. The backward pass then performs backpropagation that starts at the end of the DNN with the loss value based on the output and recursively applies the chain rule to compute the gradients all the way to the inputs.

For the L_∞ norm, the perturbation is calculated as:

$$\psi(\boldsymbol{x}, y) = -\epsilon \cdot \mathrm{sign}\left(\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}, y)\right), \tag{1}$$

where ϵ (> 0) is the attack strength. For the L_1 and L_2 norms, the perturbation is obtained as:

$$\psi(\boldsymbol{x}, y) = -\epsilon \frac{\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}, y)}{\|\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}, y)\|_p}. \tag{2}$$

The adversarial example $\boldsymbol{x}_{\mathrm{adv}}$ is obtained as follows:

$$\boldsymbol{x}_{\mathrm{adv}} = \boldsymbol{x} + \psi(\boldsymbol{x}, y). \tag{3}$$

At each iteration step, our algorithm computes a targeted adversarial perturbation $\psi(\boldsymbol{x} + \boldsymbol{\rho}, y)$, if the perturbed image $\boldsymbol{x} + \boldsymbol{\rho}$ is not classified into the target class y (i.e., $C(\boldsymbol{x} + \boldsymbol{\rho}) \neq y$); however, the non-targeted UAP algorithm obtains a non-targeted adversarial perturbation that satisfies $C(\boldsymbol{x} + \boldsymbol{\rho}) \neq C(\boldsymbol{x})$ if $C(\boldsymbol{x} + \boldsymbol{\rho}) = C(\boldsymbol{x})$. After generating the adversarial example at this step (i.e., $\boldsymbol{x}_{\mathrm{adv}} \leftarrow \boldsymbol{x} + \boldsymbol{\rho} + \psi(\boldsymbol{x} + \boldsymbol{\rho}, y)$), the perturbation $\boldsymbol{\rho}$ is updated if $\boldsymbol{x}_{\mathrm{adv}}$ is classified into the target class y (i.e., $C(\boldsymbol{x}_{\mathrm{adv}}) = y$), whereas the non-targeted UAP algorithm updates the perturbation $\boldsymbol{\rho}$ if $C(\boldsymbol{x} + \boldsymbol{\rho}) \neq C(\boldsymbol{x})$. Note that tFGSM does not ensure that adversarial examples are classified into a target class. When updating $\boldsymbol{\rho}$, a projection function $\mathrm{project}(\boldsymbol{x}, p, \xi)$ is used to satisfy the constraint that $\|\boldsymbol{\rho}\|_p \leq \xi$ (i.e., $\boldsymbol{\rho} \leftarrow \mathrm{project}(\boldsymbol{x}_{\mathrm{adv}} - \boldsymbol{x}, p, \xi)$. This projection is defined as follows:

$$\mathrm{project}(\boldsymbol{x}, p, \xi) = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \ \text{ s.t. } \ \|\boldsymbol{x}'\|_p \leq \xi \tag{4}$$

This update procedure terminates when the targeted attack success rate r_{ts} for input images (i.e., the proportion of input images classified into the target class; $|\boldsymbol{X}|^{-1} \sum_{\boldsymbol{x} \in \boldsymbol{X}} \mathbb{I}\left(C(\boldsymbol{x} + \boldsymbol{\rho}) = y\right)$) equals 100% (i.e., all input images are classified into the target class due to the UAP $\boldsymbol{\rho}$) or the number of iterations reaches the maximum $i_{\max}$.

The pseudocode of our algorithm is shown in Algorithm 1. Our algorithm was implemented using Keras (Version 2.2.4; keras.io) and the Adversarial Robustness 360 Toolbox [20] (Version 1.0; github.com/IBM/adversarial-robustness-toolbox).

Algorithm 1 Computation of a targeted UAP.

Input: Set $\boldsymbol{X}$ of input images, target class y, classifier $C(\cdot)$, cap ξ on the L_p norm of the perturbation, norm type p (1, 2, or ∞), maximum number $i_{\max}$ of iterations.

Output: Targeted UAP vector $\boldsymbol{\rho}$.

$\boldsymbol{\rho} \leftarrow \mathbf{0}$, $r_{st} \leftarrow 0$, $i \leftarrow 0$
while $r_{st} < 1$ and $i < i_{\max}$ **do**
  for $\boldsymbol{x} \in \boldsymbol{X}$ in random order **do**
    if $C(\boldsymbol{x} + \boldsymbol{\rho}) \neq y$ **then**
      $\boldsymbol{x}_{\text{adv}} \leftarrow \boldsymbol{x} + \boldsymbol{\rho} + \psi(\boldsymbol{x} + \boldsymbol{\rho}, y)$
      if $C(\boldsymbol{x}_{\text{adv}}) = y$ **then**
        $\boldsymbol{\rho} \leftarrow \text{project}(\boldsymbol{x}_{\text{adv}} - \boldsymbol{x}, p, \xi)$
      end if
    end if
  end for
  $r_{st} \leftarrow |\boldsymbol{X}|^{-1} \sum_{\boldsymbol{x} \in \boldsymbol{X}} \mathbb{I}\left(C(\boldsymbol{x} + \boldsymbol{\rho}) = y\right)$
  $i \leftarrow i + 1$
end while

2.2. Deep Neural Network Models and Image Datasets

To evaluate targeted UAPs, we used two DNN models that were trained to classify the CIFAR-10 image dataset (www.cs.toronto.edu/~kriz/cifar.html). The CIFAR-10 dataset includes 60,000 RGB color images with the size of 32 × 32 pixels classified into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Six-thousand images are available in each class. The dataset comprises 50,000 training images (5000 images per class) and 10,000 test images (1000 images per class). In particular, we used the VGG-20 [21] and ResNet-20 models [22] for the CIFAR-10 dataset obtained from a GitHub repository (github.com/GuanqiaoDing/CNN-CIFAR10); their test accuracies were 91.1% and 91.3%, respectively.

Moreover, we also considered three DNN models trained to classify the ImageNet image dataset [18] (www.image-net.org). The ImageNet dataset is comprised of RGB color images with the size of 224 × 224 pixels classified into 1000 classes. In particular, we used the VGG-16 [21], VGG-19 [21], and ResNet-50 models [22] for the ImageNet dataset available in Keras (Version 2.2.4; keras.io), and their test accuracies were 71.6%, 71.5%, and 74.6%, respectively.

2.3. Generating Targeted Adversarial Perturbations and Evaluating Their Performance

Targeted UAPs were generated using an input image set obtained from the datasets. The parameter p was set to two. We generated targeted UAPs with various magnitudes by adjusting the parameters ϵ and ξ. The magnitude of a UAP was measured using a normalized L_2 norm of the perturbation; in particular, we used the ratio ζ of the L_2 norm of the UAP to the average L_2 norm of an image in a dataset. The average L_2 norms of an image were 7381 and 50,135 in the CIFAR-10 and ImageNet datasets, respectively.

TO compare the performance of the targeted UAPs generated by our method with random controls, we also generated random vectors (random UAPs) sampled uniformly from the sphere of a given radius [20].

The performance of the UAPs was evaluated using the targeted attack success rate r_{ts}. In particular, we considered the success rates r_{ts} for input images. In addition to this, we also computed the success rates r_{ts} for test images to experimentally evaluate the performance of UAPs for unknown images. A test image set was obtained from the dataset and did not overlap with the input image set.

3. Results and Discussion

3.1. Case of the CIFAR-10 Models

For the CIFAR-10 models, we used 10,000 input images to generate the targeted UAPs. The input image set was obtained by randomly selecting 1000 images per class from the training images of the CIFAR-10 dataset. All 10,000 test images of the dataset were used as test images for evaluating the UAP performance. We considered the targeted attack for each class. The parameters ϵ and $i_{\max}$ were set to 0.006 and 10, respectively.

For the targeted attacks for each class, the targeted attack success rates r_{ts} for both the input image set and the UAP test image set rapidly increased with the perturbation rate, despite a low ζ (2–6%). In particular, the success rate was >80% for $\zeta = 5\%$ (Figure 1); moreover, it reached ∼100% for $\zeta > 10\%$. The success rates of the targeted UAP with the same ζ for both the input image set and the UAP test image set were almost similar. This result indicates no overfitting of the generated UAPs to the input images.

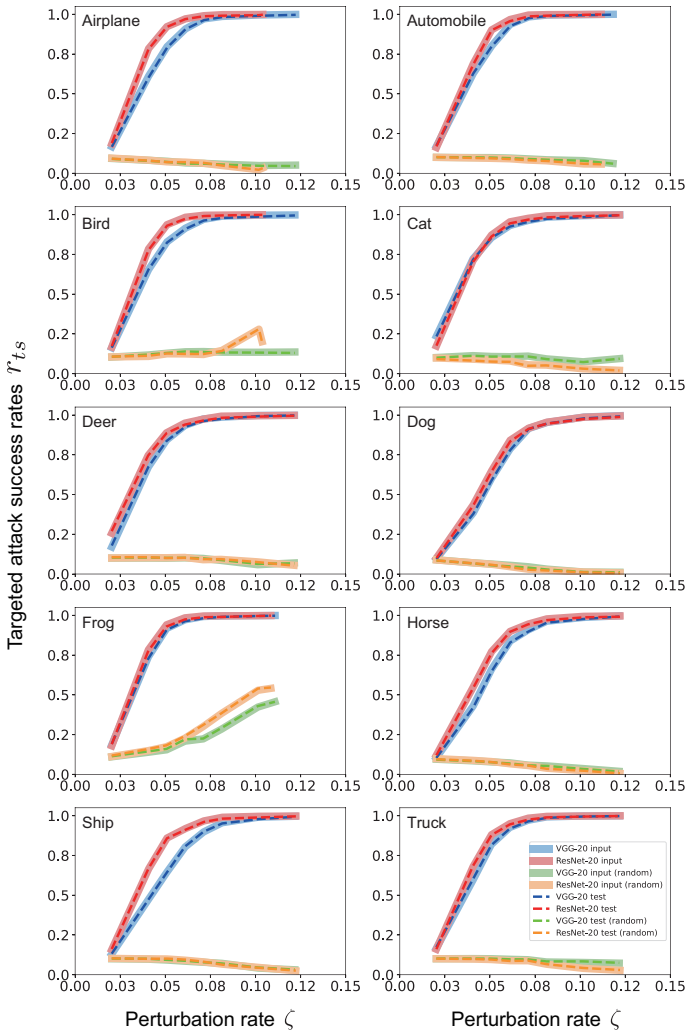

Figure 1. Line plot of the target attack success rate r_{ts} versus the perturbation rate for targeted attacks for each class of the CIFAR-10 dataset. The legend label indicates the DNN model and image set used for computing r_{ts}. For example, "VGG-20 input" indicates the r_{ts} of targeted UAPs against the VGG-20 model computed using the input image set. The additional argument "(random)" indicates that random UAPs were used instead of targeted UAPs.

The targeted UAPs with $\zeta = 5\%$ were almost imperceptible (Figure 2). Moreover, the UAPs seemed to represent the object shapes of each target class. The success rates of the targeted UAPs were significantly higher than those of random UAPs. These tendencies were observed both in the VGG-20 model and in the ResNet-20 model. These results indicate that non-random patterns are required for targeted attacks and that the proposed method applies to various DNN architectures.

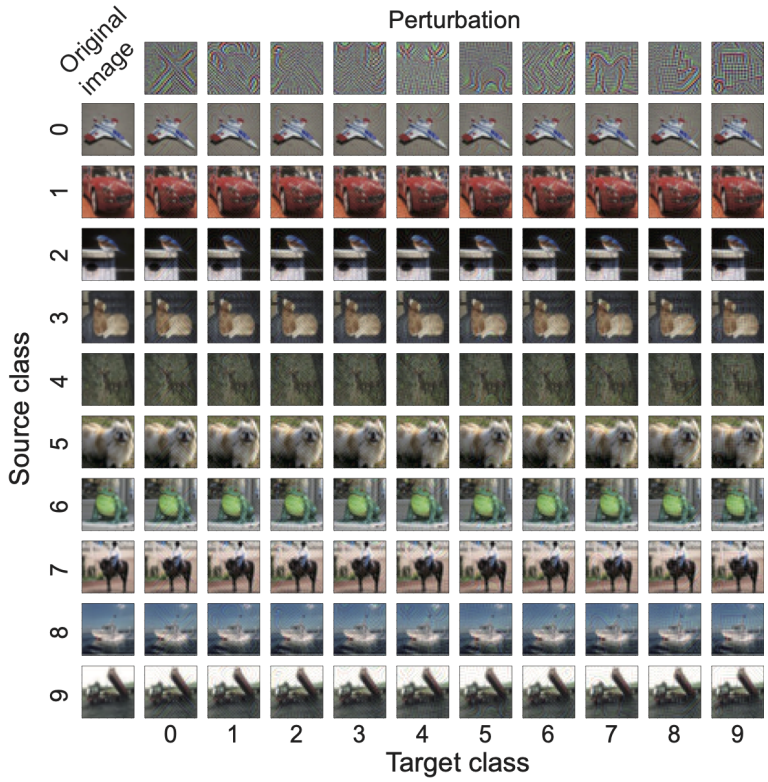

Figure 2. Targeted UAPs (top panel) with $\zeta = 5\%$ against the VGG-20 model for the CIFAR-10 dataset and their adversarial attacks for an original (i.e., non-perturbed) image (left panel) randomly selected from the images that, without perturbation, was correctly classified into each source class and, with the perturbations, correctly classified into the target classes: airplane (0), automobile (1), bird (2), cat (3), deer (4), dog (5), frog (6), horse (7), ship (8), and truck (9). Note that the UAPs are emphatically displayed for clarity; in particular, each UAP was scaled with the maximum of one and the minimum of zero.

3.2. Case of ImageNet Models

For the ImageNet models, we used the validation dataset used in the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012; www.image-net.org/challenges/LSVRC/2012/) to generate the targeted UAPs. The dataset is comprised of 50,000 images (50 images per class). We used 40,000 images as the input images. The input image set was obtained by randomly selecting 40 images per class. The rest (10,000 images; 10 images per class) was used as test images for evaluating the UAPs. The parameters ϵ and $i_{\max}$ were set to 0.5 and five, respectively.

In this study, we considered targeted attacks for three classes (golf ball, broccoli, and stone wall) that were randomly selected from 1000 classes in a previous study [17], to avoid redundancy.

We generated targeted UAPs with $\zeta = 6\%$ ($\xi = 3000$) and $\zeta = 8\%$ ($\xi = 4000$). The target attack success rates r_{ts} were between ~30% and ~75% and between ~60% and ~90% when $\zeta = 6\%$ and $\zeta = 8\%$, respectively (Table 1). The success rates for the input images were almost similar to those for the test images, indicating no overfitting of the generated UAPs to the input image set. The success

rates of the targeted UAPs were significantly higher than those of random UAPs, which were less than 1% in all cases. This indicates that non-random patterns are required for targeted attacks

Table 1. Targeted attack success rates r_{ts} of targeted UAPs against the DNN models for each target class. r_{ts} for input images and test images are shown.

Target Class	Model	$\zeta = 6\%$		$\zeta = 8\%$	
		Input	**Test**	**Input**	**Test**
Golf ball	VGG-16	58.0%	57.6%	81.6%	80.6%
	VGG-19	55.3%	55.2%	81.3%	80.1%
	ResNet-50	66.8%	66.5%	90.3%	89.8%
Broccoli	VGG-16	29.3%	29.0%	59.7%	59.5%
	VGG-19	31.2%	30.5%	59.7%	59.4%
	ResNet-50	46.4%	46.6%	74.6%	73.9%
Stone wall	VGG-16	47.1%	46.7%	75.0%	74.5%
	VGG-19	48.4%	48.1%	73.9%	72.9%
	ResNet-50	74.7%	74.4%	92.0%	91.3%

A higher perturbation magnitude ζ led to a higher targeted attack success rate r_{ts}. The success rates r_{ts} depend on the image classes. For example, the targeted attacks for the class "golf ball" were more easily achieved than those for the class "broccoli". The success rates r_{ts} also depended on the DNN architectures; in particular, the ResNet-50 model was more easy to fool than the VGG models. Overall, however, we confirmed that the proposed method was applicable to various DNN architectures.

The targeted UAPs with $\zeta = 6\%$ and $\zeta = 8\%$ were almost imperceptible (Figure 3); however, they were partly perceptible for whitish images (e.g., trimaran). Moreover, the UAPs seem to reflect object shapes of each target class.

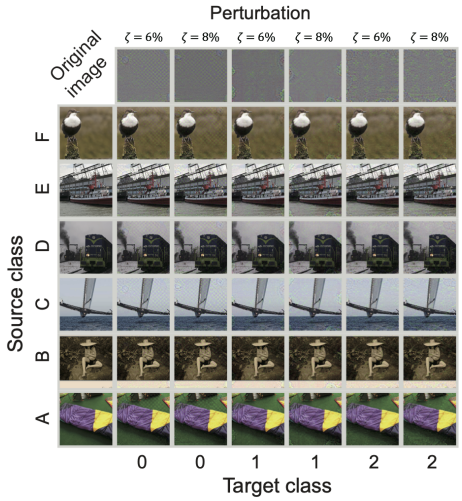

Figure 3. Targeted UAPs (top panel) against the ResNet-50 model for the ImageNet dataset and their adversarial attacks for the original (i.e., non-perturbed) images (left panel) randomly selected from the images that, without perturbation, were correctly classified into the source class and, with the perturbation, correctly classified into each target class under the constraint that the source classes are not overlapping each other and with the target classes. The source classes displayed here are sleeping bag (**A**), sombrero (**B**), trimaran (**C**), steam locomotive (**D**), fireboat (**E**), and water ouzel, dipper (**F**). The target classes are golf ball (0), broccoli (1), and stone wall (2). The UAPs with $\zeta = 6\%$ and $\zeta = 8\%$ are shown. Note that the UAPs are emphatically displayed for clarity; in particular, each UAP was scaled with the maximum of one and the minimum of zero.

The targeted attack success rates in the ImageNet models were relatively lower than those in the CIFAR-10 models. This is because the ImageNet dataset has a larger number of classes than the CIFAR-10 dataset. In short, it is more difficult to exactly classify an input image into a specific target class within a larger number of classes. Moreover, the observed lower success rate may be because the validation dataset of ILSVRC2012 was used when generating targeted UAPs. Higher success rates may be obtained when generating targeted UAPs using training images.

These results indicate that the proposed method has several advantages for generating UAPs. As Chaubey et al. [11] mentioned, the previous methods for generating UAPs are limited to non-targeted attacks, except for a generative model-based method [17]. The generative model-based method requires high computational costs; thus, it does not apply to large image datasets such as the ImageNet dataset. On the other hand, our simple iterative method can compute targeted UAPs for large image datasets. Our method can help reliability assessments and increase safety [4,8] in practical applications of DNNs (e.g., medical image-based diagnosis [1,23] and detecting and classifying traffic signs [2,24]).

4. Conclusions

We propose a simple iterative method to generate targeted UAPs for image classification, although the proposed algorithm is a straightforward extension of the non-targeted UAP algorithm [10]. Using the CIFAR-10 and ImageNet models, we demonstrated that a small (almost imperceptible) UAP generated by our method made the models largely classify test images into a target class. Our results indicated the existence of UAPs for targeted attacks and that such UAPs can be easily generated. Our study enhances our understanding of the vulnerabilities of DNNs to adversarial attacks; moreover, it may help accelerate practical applications of DNNs. The source code of our proposed method for generating targeted UPAs is available from our GitHub repository: github.com/hkthirano/targeted_UAP_CIFAR10. Our method is also available in the Adversarial Robustness Toolbox [20] (Version 1.4 and later; github.com/Trusted-AI/adversarial-robustness-toolbox), a Python library for machine learning security.

Author Contributions: Conceptualization, H.H. and K.T.; methodology, H.H. and K.T.; software, H.H. and K.T.; validation, H.H.; investigation, H.H.; resources, H.H.; writing, original draft preparation, K.T.; writing, review and editing, H.H. and K.T.; visualization, H.H. and K.T.; supervision, K.T.; project administration, K.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DNN	Deep neural network
FGSM	Fast gradient signed method
GAN	Generative adversarial network
ILSVRC2012	Large Scale Visual Recognition Challenge 2012
ResNet	Residual network
UAP	Universal adversarial perturbation
VGG	Visual geometry group

References

1. Esteva, A.; Kuprel, B.; Novoa, R.A.; Ko, J.; Swetter, S.M.; Blau, H.M.; Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **2017**, *542*, 115–118. [CrossRef] [PubMed]
2. Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Netw.* **2015**, *32*, 323–332.
3. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2014**, arXiv:1412.6572.
4. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [CrossRef] [PubMed]
5. Machado, G.R.; Silva, E.; Goldschmidt, R.R. Adversarial machine learning in image classification: A survey towards the defender's perspective. *arXiv* **2020**, arXiv:2009.03728
6. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial examples in the physical world. *arXiv* **2016**, arXiv:1607.02533
7. Mohsen, S.; Fawzi, A.; Frossard, P. DeepFool: A simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
8. Carlini, N.; Wagner, D. Towards evaluating the robustness of neural networks. In Proceedings of the 2017 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 22–26 May 2017; pp. 39–57.
9. Matyasko, A.; Chau, L.-P. Improved network robustness with adversary critic. *arXiv* **2018**, arXiv:1810.12576.
10. Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; Frossard, P. Universal adversarial perturbations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
11. Chaubey, A.; Agrawal, N.; Barnwal, K.; Guliani, K.K.; Mehta, P. Universal adversarial perturbations: A survey. *arXiv* **2020**, arXiv:2005.08087
12. Finlayson, S.G.; Bowers, J.D.; Ito, J.; Zittrain, J.L.; Beam, A.L.; Kohane, I.S. Adversarial attacks on medical machine learning. *Science* **2019**, *363*, 1287–1289. [CrossRef] [PubMed]
13. Khrulkov, V.; Oseledets, I. Art of singular vectors and universal adversarial perturbations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
14. Mopuri, K.R.; Ojha, U.; Garg, U.; Babu, R.V. NAG: Network for adversary generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
15. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
16. Kurakin, A.; Goodfellow, I.; Bengio, S.; Dong, Y.; Liao, F.; Liang, M.; Pang, T.; Zhu, J.; Hu, X.; Xie, C.; et al. Adversarial attacks and defences competition. In *The NIPS '17 Competition: Building Intelligent Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 195–231.
17. Hayes, J.; Danezis, G. Learning universal adversarial perturbations with generative models. In Proceedings of the IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 24 May 2018; pp. 43–49.
18. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
19. Brown, T.B.; Mané, D.; Roy, A.; Abadi, M.; Gilmer, J. Adversarial patch. *arXiv* **2017**, arXiv:1712.09665
20. Nicolae, M.-I.; Sinn, M.; Tran, M.N.; Buesser, B.; Rawat, A.; Wistuba, M.; Zantedeschi, V.; Baracaldo, N.; Chen, B.; Ludwig, H.; et al. Adversarial Robustness Toolbox v1.0.1. *arXiv* **2018**, arXiv:1807.01069
21. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

23. Taghanaki, S.A.; Das, A.; Hamarneh, G. Vulnerability analysis of chest X-ray image classification against adversarial attacks. In *Understanding and Interpreting Machine Learning in Medical Image Computing Applications*; Springer: Cham, Switzerland, 2018; pp. 87–94.
24. Morgulis, N.; Kreines, A.; Mendelowitz, S.; Weisglass, Y. Fooling a real car with adversarial traffic signs. *arXiv* **2019**, arXiv:1907.00374