CAG: A REAL-TIME LOW-COST ENHANCED-ROBUSTNESS HIGH-TRANSFERABILITY CONTENT-AWARE ADVERSARIAL ATTACK GENERATOR

by

VAN NHAT HUY PHAN

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Bo Yuan

and approved by

New Brunswick, New Jersey

May, 2020

ABSTRACT OF THE THESIS

CAG: A Real-time Low-cost Enhanced-robustness High-transferability Content-aware Adversarial Attack Generator

By Van Nhat Huy Phan

Thesis Director:

Bo Yuan

Deep neural networks (DNNs) are vulnerable to adversarial attack despite their tremendous success in many artificial intelligence fields. Adversarial attack is a method that causes the intended misclassfication by adding imperceptible perturbations to legitimate inputs. To date, researchers have developed numerous types of adversarial attack methods. However, from the perspective of practical deployment, these methods suffer from several drawbacks such as long attack generating time, high memory cost, insufficient robustness and low transferability. To address the drawbacks, we propose a Content-aware Adversarial Attack Generator (CAG) to achieve real-time, low-cost, enhanced-robustness and high-transferability adversarial attack. First, as a type of generative model-based attack, CAG shows significant speedup (at least 500 times) in generating adversarial examples compared to the state-of-the-art attacks such as PGD and C&W. Furthermore, CAG only needs a single generative model to perform targeted attack to any targeted class. Because CAG encodes the label information into a trainable embedding layer, it differs from prior generative model-based adversarial attacks that use n different copies of generative models for n different targeted classes.

As a result, CAG significantly reduces the required memory cost for generating adversarial examples. Moreover, CAG can generate adversarial perturbations that focus on the critical areas of input by integrating the class activation maps information in the training process, and hence improve the robustness of CAG attack against the state-of-art adversarial defenses. In addition, CAG exhibits high transferability across different DNN classifier models in black-box attack scenario by introducing random dropout in the process of generating perturbations. Extensive experiments on different datasets and DNN models have verified the real-time, low-cost, enhanced-robustness, and high-transferability benefits of CAG.

Acknowledgements

I would like to express my gratitude to my primary supervisor, Bo Yuan, who guided me throughout this project. I would also like to thank my colleagues, Yi Xie, and Siyu Liao who offered deep insight into the study. Finally, I wish to extend my special thanks to my friends and family for all the support I received from them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep neural networks (DNNs) have achieved unprecedented success in many artificial intelligence fields, such as computer vision, natural language processing and speech recognition [8], [11], [12]. Despite their current popularity and prosperity, DNNs are still facing several severe challenges, especially their high vulnerability to adversarial attack [5], [17], which adds well-designed tiny perturbations to the legitimate inputs to cause the intended misclassification of DNN models. Such attacks could cause severe safety, economic and social problems if launched to the DNNs deployed in practical applications ranging from face recognition, autonomous driving to speech authentication.

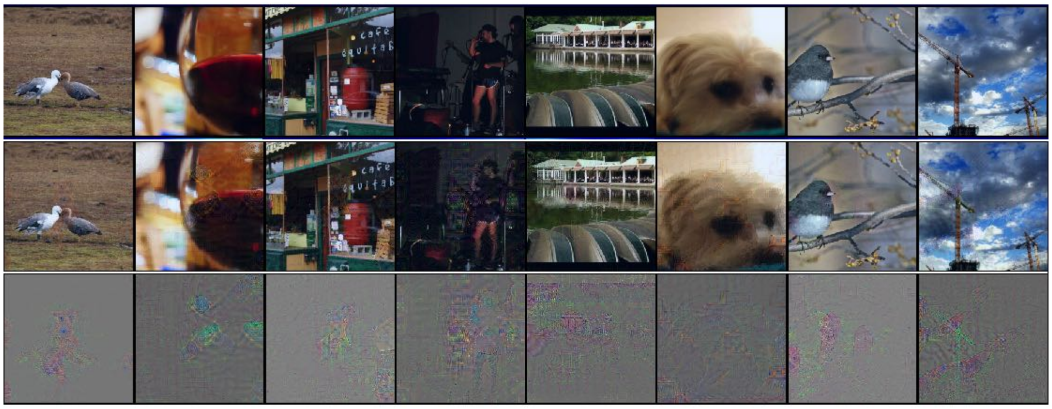


Figure 1.1: Adversarial images generated with CAG using ImageNet dataset. From top row to bottom row: legitimate images, adversarial images, perturbations (enhanced).

In order to address this critical challenge, the machine learning community has conducted extensive researches on the vulnerability of DNNs, from both the attack and defense aspects. Adversarial attack technique was pioneered by Szegedy *et al.* [4]. Since then, researchers have developed various adversarial attacking algorithms, targeting different types of DNN models including convolutional neural networks, recurrent neural networks and graph neural networks, and also different application scenarios, ranging from image classification, machine translation, to graph classification etc. Among those algorithms, one popular class of attack techniques is fast gradient sign method (FGSM), which performs one-step gradient computation to craft untargeted adversarial examples [5]. Considering the relatively weak attack performance of FGSM, the machine learning community has proposed several iterative optimization-based techniques including C&W, I-FGSM and PGD that deliver the state-of-the-art attack performance [5], [17], [20]. Furthermore, some recent work has also proposed to use generative models, e.g., GAN and U-Net, to generate adversarial examples [6], [10], [24], [26].

Although the existing adversarial attack methods can already exhibit high attack success rate (ASR), especially in white-box attack scenario, from the perspective of practical deployment, they are still suffering one or more drawbacks, namely long adversarial example generating time, high memory cost for launching adversarial attack, insufficient robustness against defense methods and low transferability in black-box attack scenario.

Aiming to overcome these drawbacks, in this paper we propose a Content-aware Adversarial Attack Generator (CAG), to achieve real-time, low-cost, enhanced-robustness and high-transferability adversarial attack. We show some adversarial images generated by CAG in Figure 1.1. The features and benefits of CAG are summarized as follows:

- CAG is a generative model-based attack, so it can avoid time-consuming iterative optimization procedure to generate adversarial examples. Compared with the state-of-the-art iterative attacks such as PGD and C&W, CAG achieves significant speedup (at least 500 times), and hence makes real-time attack possible.

- CAG utilizes a trainable embedding layer to encode all label information to one single model, unlike prior generative model-based methods which require different generative models for different targeted classes. In n-class targeted attack scenario, the number of the required generative models is reduced from n to 1, thereby drastically reducing the memory cost for launching attacks.

- CAG integrates the class activation maps (CAMs) information into the training process, in contrast to many other attack methods that generate adversarial perturbations over the entire input. Consequently, CAG is able to generate adversarial perturbations that focus on the critical areas of input, and thus improves the attack's robustness against the state-of-art defense approaches.

- CAG exhibits high transferability across different DNN classifier models in black-box attack scenario. CAG can generate adversarial perturbations with better generality by introducing random dropout in the perturbations-generation process. As a result, CAG's adversarial examples have higher transferability when attacking unseen classifiers.

The rest of this paper is organized as follows. Section 2 introduces the related work on adversarial attack and defense methods. Section 3 discusses our motivation. Section 4 describes the technical details of CAG. The experimental results are presented and analyzed in Section 5. Section 6 draws the conclusions of all findings in our paper.

Chapter 2

Related Work

2.1 Adversarial Attacks

To define an adversarial attack, let $X = \{x_1, ..., x_m\}$ be a set of the valid inputs from the dataset, $y \in \{1, ..., L\}$ be the valid class label, and $F(\cdot)$ be the well-trained DNN classifier. Let (x_i, y_i) denote the i-th benign instance and the corresponding true label. The goal of an adversarial attack is to create the $x_i' = x_i + \delta$, where δ is imperceptible adversarial perturbation. A nontargeted attack requires $F(x_i) \neq y_i$ and a targeted attack specifies $t \neq i$ such that $F(x_i') = y_t$.

2.2 FGSM

Fast gradient sign method (FGSM) is a one-step fast-adversarial-example-generation approach [5]. It aims to linearize loss function in L_∞ neighborhood of a legitimate input and to find the exact maximum of the linearized loss function. Correspondingly, its adversarial example generation formula is as follows:

$$x' = x + \epsilon \cdot sign(\Delta J(x, y_{true})),$$

where y_{true} denotes the true label, $\Delta J(.,.)$ computes the gradient of the loss function, and $sign$ denotes the sign function. Notice that here ϵ is the attack strength parameter to control the balance between the attack performance and the norm of the perturbations.

2.3 I-FGSM & PGD

Although FGSM is fast, its attack performance is relatively weak. Researchers have proposed various approaches to achieve stronger attack by improving the vanilla FGSM method. Kurakin *et al.* [14] propose to take multiple steps of FGSM (I-FGSM) with smaller attack strength α in an iterative way:

$$x'_{N+1} = Clip_{\epsilon}\{x'_N + \alpha \cdot sign(\Delta J(x'_N, y_{true})\},$$

where x'_N is the adversarial image at the N-th iteration, and $Clip\{\cdot\}$ clips the overall attack strength back to ϵ at the end of the iteration. Notice that in the case of using L_∞ norm, I-FGSM is equivalent to another popular iteration-based attack method (PGD) [20].

2.4 C&W

C&W [17] is an optimization-based attack method. It aims to optimize the loss function as follows:

$$\|x' - x\|_p + c \cdot \max(\max_{i \neq t} f(x')_i - f(x')_t, -\kappa),$$

where t is the targeted class, $f(\cdot)$ denotes the softmax function, c is a constant set by binary search, and κ is an adjustable parameter that encourages the attacker to find an adversarial example being classified as class t with high confidence. By minimizing the above loss function using Adam optimizer in an iterative way, C&W can achieve high ASR with low perturbation norm.

2.5 Generative Model-based

One drawback of the iterative methods mentioned above is long generating time. Hence another method to generate adversarial examples is to use a generative model, such as GAN, Autoencoder [6] or U-Net. For instance, Xiao *et al.* [26] apply AdvGAN to craft perceptually realistic adversarial examples. Moreover, Baluja *et al.* [16] develop an adversarial transformation network to convert inputs into adversarial examples. Poursaeed *et al.* [24] propose a method they name Generative Adversarial Perturbations

(GAP) that uses a ResNet-based generative model [13] to perform adversarial attack.

2.6 Adversarial Defenses

2.7 Pixel Deflection

The key idea of pixel deflection defense is to randomly replace pixels with nearby pixels [25]. To achieve the replacement, this method uses CAMs of the top-5 predictions to guide the update of the pixels [15]. In this scenario, the probability of a pixel being updated is inversely proportional to the likelihood that the area contains the object. After the pixel replacement, a denoising operation is applied to recover the classification accuracy.

2.8 Randomization

The mitigation of adversarial attack effects can also be achieved by using randomization. For instance, Xie *et al.* proposes to first resize the input to random size [22]. After that, a random padding operation is performed to pad zeros around the resized image. Though it may seem simple, this method can significantly improve the robustness of DNN models against adversarial attack.

2.9 Input Transformation

Another type of popular defense methods is input transformation. Its key idea is to perform various transformations, such as bit-depth reduction, lossy compression and variance minimization on adversarial examples to mitigate the attack effects [18], [23]. The reported experimental results show that these methods can achieve balance between robustness against attack and computation overhead.

Chapter 3

Motivations

Despite the abundance of researches on adversarial attack methods described in Section 2, existing approaches still suffer from several inherent drawbacks–in particular from the perspective of practical deployment.

Long Generation Time Iteration-based approaches predominate among current state-of-the-art methods, including PGD and C&W. Consequently, generating adversarial examples using iteration is time-expensive and requires extensive computational resources, especially for the targeted attack. For example, to achieve a high ASR, C&W method takes hours to generate 100 large-size adversarial examples on a GPU. Such long generation time makes launching the adversarial attack in real-time setting infeasible.

High Memory Cost Using an iteration-free generative model-based attack promises to avoid long generation time [26], [24], [16]. However, in these existing works if the attackers wants to achieve targeted attack to a specific class, they have to train different generator models for different targeted classes. For example, to prepare for the targeted attack to 1000 classes in the ImageNet dataset, in total 1000 different generator models have to be trained and stored, thereby causing massive memory cost.

Insufficient Robustness To date, most adversarial example generation is based on the search over the entire input size instead of focusing on the critical part of legitimate object content. Noticing this phenomenon, many defense methods have been developed to improve defense performance via integrating this information into the defense scheme. For instance, Luo *et al.* propose to mask out the background regions with little transformation performed on the critical areas [9]. Similarly, Prakash *et al.* propose to use pixel deflection to denoise and reconstruct the input by locally redistributing pixels under the guidance of the object position [25]. Consequently, such well-designed defense schemes make the existing adversarial attack exhibit insufficient robustness.

Low Transferability Most adversarial attack methods can achieve high ASR in the white-box attack scenario. However, in real-world applications, black-box attack is a more common environment setting. In such cases, the transferability of the generated adversarial examples is important to ensure a successful attack. However, to date on large-scale datasets and large DNN models, the existing adversarial attack approaches exhibit low transferability, thereby impeding the feasibility of launching real-life black-box attack.

Our Motivation Motivated to redress the above challenges plaguing the existing adversarial attack methods, we aim to develop an adversarial attack method that can 1) generate each adversarial example in a real-time manner; 2) require only one model for different targeted classes; 3) exhibit strong robustness against the-state-of-the-art defense techniques; and 4) exhibit high transferability in the black-box attack scenario. To fulfill those requirements, we develop CAG, an attack method with fast generation speed, low memory cost, improved robustness and high transferability. Next, we describe the model training and attack generation schemes of CAG in detail.

Chapter 4

CAG: Content-aware Adversarial Attack Generator

4.1 Overall Architecture

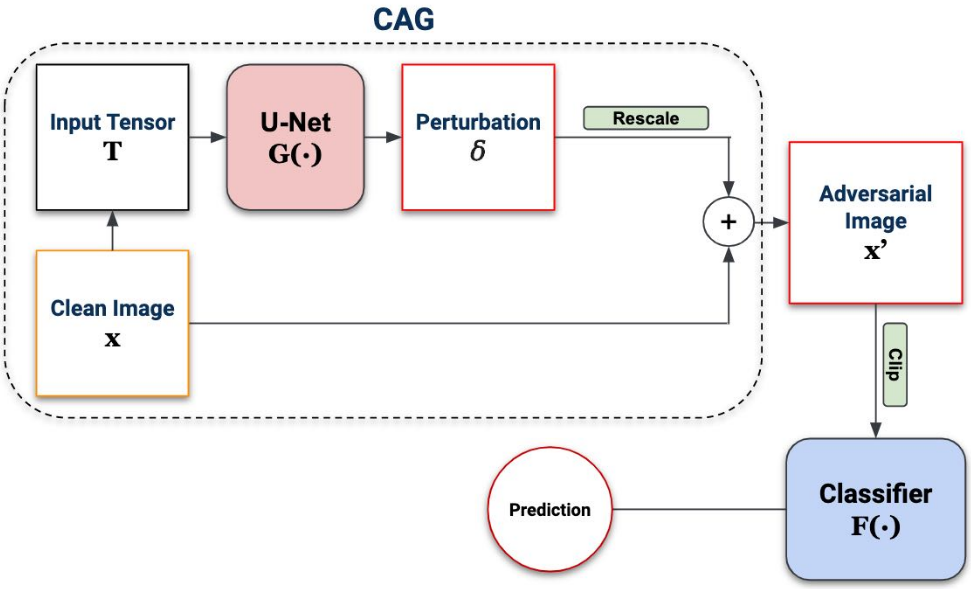


Figure 4.1: Overall architecture of CAG.

Figure 4.1 illustrates the overall architecture of CAG. To generate an adversarial image, an input tensor T is first constructed based on the given clean image x, true label i and targeted label t. Then a generator model $G(\cdot)$, in the format of U-Net, is used to generate the perturbations δ from T. After that, δ is scaled to a fixed L_2 norm and added to x. Finally, after clipping out-of-range values, the adversarial image x' is ready to mislead the classifier from original true class i to the targeted class t.

Fast Generation Speed using U-Net CAG utilizes U-Net as the underlying generative model. Therefore, when compared with other iteration-based attack methods, U-Net-based approach avoids time-expensive iterative procedure, and hence makes real-time generation of adversarial examples possible.

4.2 Building Input Tensors

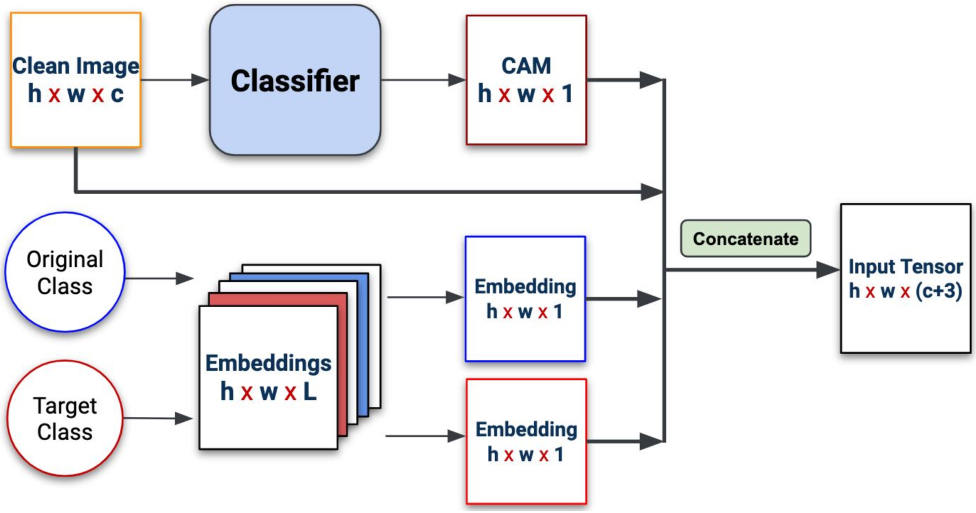


Figure 4.2: Building input tensor for CAG. The target class is randomly selected and given in training and testing phase, respectively.

Single Generative Model via Label Embedding As mentioned in Section 3, the main drawback of generative model-based attack is a need for massive amount of models for different target classes. To address this problem, we encode the class label information into the input tensor T for U-Net. Figure 4.2 shows the overall procedure of constructing T. Here the dimension of the clean image is denoted as $h \times w \times c$ where h, w, c represent height, width and number of channels, respectively. Then during training phase, an embedding layer E_{hwL} with the size of $h \times w \times L$, where L is the number of valid classes, is trained to encode the label information. Specifically, in the forward propagation pass, a targeted class t is randomly selected for each training data x_i. The target label t, as well as the true label i, are used to extract the corresponding slices E_t and E_i from E_{hwL}, where $E_k = E_{::k}$ denotes the k-th front slice of the embedding layer E_{hwL}. Then in the backward propagation phase, E_t and E_i are updated to help E_{hwL} capture more class information for this training data. After being trained on the entire dataset, the embedding layer E_{hwL} learns the important class encoding information and thereby ensuring only one U-Net model is sufficient for different target classes.

Enhanced Attack Robustness using CAM Besides using an embedding layer, the construction of input tensor T also utilizes the information of CAMs. Classifiers make decisions based heavily on the hot areas of CAM because they contain the most discriminative information of an image. Therefore, defense methods cannot make substantial modifications in these critical area, otherwise they can easily cause misclassifications. Taking advantage of this behavior, we place the perturbations only on the hot areas of CAM to enhance the robustness of our attack against many defense schemes. To achieve this increment in robustness, the position of the object in the image needs to be integrated into the input tensor, which can be reflected by the CAM. As shown in Figure 4.2, another component of input tensor T is CAM_x^i with the size of $h \times w \times 1$, which is the CAM with respect to input x_i and its true label y_i. Consequently, we denote τ as the concatenating operation, and the final input tensor is constructed as follows:

$$T = \tau(x_i, E_i, E_t, CAM_x^i),$$

where the size of T is $h \times w \times (c + 3)$.

4.3 Training CAG

Next, we describe the details of CAG training procedure. Our objective is to get $G(\cdot)$ and E_{hwL} to achieve:

$$F(clip(G(T) + x_i)) = y_t.$$

In this scenario, the embedding layer E_{hwL} is treated as a model parameter that can be learned, so that y_t can be any selected label from $\{1, ..., L\}$. Therefore, we can formulate an effective loss function $Loss$, and use existing optimization algorithms to perform training as follows.

First, in order to keep the perturbations imperceptible, we scale the perturbations using the L_2 distance metric. In other words, we keep all the perturbations at a fixed L_2 norm to constrain the attack strength of the noise in a fixed amount.

Then we feed the generated adversarial example x_i' to the classifier $F(x_i')$ to produce

Algorithm 1: CAG Training Algorithm

```
Input: dataset $X = \{x_1, ..., x_m\}$, true labels $y \in \{1, ...L\}$, classifier $F(\cdot)$, input
  tensor $T$, desired perturbation $L_2$ norm.
Result: trained CAG $G(\cdot)$, embedding layer $E_{hwL}$.
Random initialize $G(\cdot)$, $\mathrm{E}_{hwL}$.
for $x_i$, $y_i$ in dataset do
    $y_t$ = get_random_target($y_i$), $t \neq i$;
    $E_i = (E_{::i})$, $E_t = (E_{::t})$; $h \times w \times 1$;
    $CAM_x^i$ = cam_generator($x$, $y_i$);
    $T$ = concat($x_i$, $E_i$, $E_t$, $CAM_x^i$);
    $\delta$ = drop_out ($G(T)$);
    $\delta$ = L2_norm_adjust($\delta$);
    adversarial_img = $x'_i$ = clip($x_i + \delta$) ;
    $y_{pred} = F(x'_i)$;
    $CAM_{x'}^t$ = cam_generator($x'$, $y_t$);
    Loss = CrossEntropy($y_t, y_{pred}$) + $\beta \cdot \|CAM_x^i - CAM_{x'}^t\|^2$;
    update($G(\cdot)$); update($E_{hwL}$);
end
```

the prediction y_{pred}. We define $Loss_{target}$ as the cross-entropy with respect to the one-hot label of the targeted class. Therefore, to ensure the generated adversarial examples can fool the classifier, $Loss_{target}$ is formulated as:

$$Loss_{target} = CrossEntropy(y_t, y_{pred}).$$

Meanwhile, the CAM of the targeted class t for x' is computed and denoted as $CAM_{x'}^t$. We aim to concentrate the adversarial noise on the critical areas which contain the legitimate object content, so that the $CAM_{x'}^t$ for the adversarial examples would not be significantly changed compared to CAM_x^i. In other words, to satisfy the similarity between $CAM_{x'}^t$ and $CAM_{x'}^i$, we need to minimize the their L_2 distance. Therefore, $Loss_{CAM}$ is defined to lead the distribution of the noise:

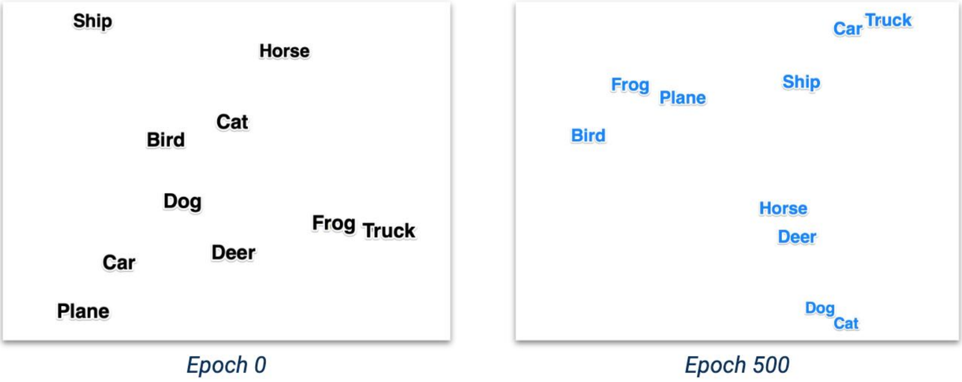
$$Loss_{CAM} = \|CAM_x^i - CAM_{x'}^t\|^2.$$

Finally, the new loss function is formulated as:

$$Loss = Loss_{target} + \beta \cdot Loss_{CAM},$$

where β controls the magnitude of $Loss_{CAM}$. We then iteratively optimize the CAG as well as E_{hwL} by minimizing the $Loss$ function. The details of our approach to train the CAG are summarized in Algorithm 1.

Improve Transferability via Noise Dropout It is worth noting that before directly adding the noise on x_i, we propose to apply a dropout layer with probability p in the training phase. As a result, dropout layer can eliminate over-fitting problem to the current classifier and achieve better performance in black-box attack scenario by increasing the transferability. The extensive experimental results are given in the next section.



Epoch 0

Epoch 500

Figure 4.3: Embeddings of different classes after using T-SNE to reduce the dimension (CIFAR-10).

After preparing CAG and E_{hwL} to perform attacks, we visualize the E_{hwL} to demonstrate the effectiveness of this embedding layer. We show the examples using CIFAR-10 dataset, thus the size of the embedding layer is $32 \times 32 \times 10$ for 10 classes [3]. For better visualization, T-SNE is applied to reduce the each class embedding's dimension to 2 [1]. As we can see from Figure 4.3, at epoch 0, class embeddings are initialized and distributed randomly. However, at epoch 500, embeddings of similar classes are close to each other, such as car-truck, horse-deer, and dog-cat. Therefore, the local distance between similar classes suggests that our approach creates a useful set of embeddings.

We also show the attention regions using CAM for adversarial examples generated by different attack methods. As shown in Figure 4.4, compared with the clean images, I-FGSM and GAP achieve targeted attack by misleading the network's attention. However, we believe that changing the attention would make adversarial images vulnerable to designed defense mechanisms. Interestingly, as can be seen in the last row of Figure

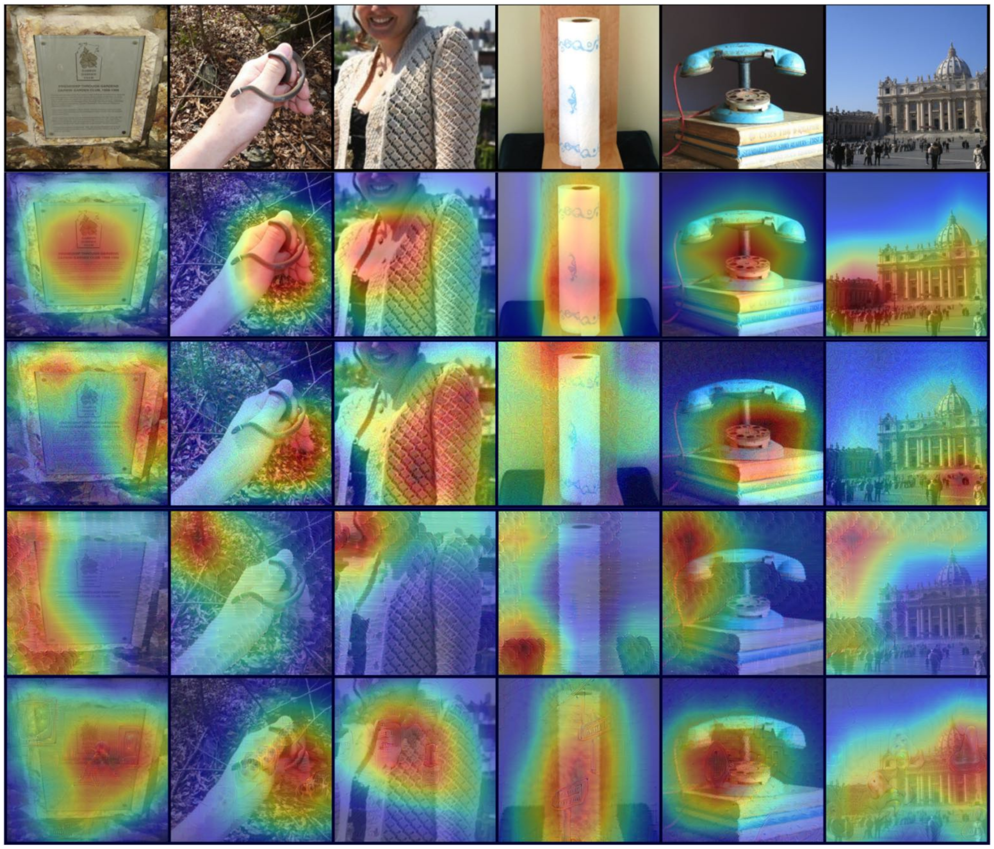


Figure 4.4: CAM attention visualization. From top row to bottom row: clean images, CAM of clean images, CAM of PGD's adversarial images, CAM of GAP's adversarial images, CAM of CAG's adversarial images.

4.4, the adversarial images generated by CAG do not suffer from this problem. Malicious perturbations are constrained to locate in the discriminative areas, so that CAG's adversarial examples are robust enough circumvent detection and defense methods.

Chapter 5

Experimental Results

5.1 Experiment Design

To evaluate the effectiveness of CAG, we conduct extensive experiments on CIFAR-10 [3] and ImageNet [2] dataset. We perform white-box and black-box attacks by using a pool of 6 different classifiers: ResNet-18 (RN-18), ResNet-34 (RN-34); VGG-11, VGG-13; DenseNet-121 (DN-121), DenseNet-169 (DN-169) [7], [12], [19]. The top-1 classification accuracy is above 92% (CIFAR-10) and above 70% (ImageNet) for all classifiers. We use ResNet-18 to generate CAMs for all experiments. We set $\beta = 3$ for both datasets because higher β enforces too much restriction and can reduce ASR. Then we train CAG using SGD with Nesterov momentum. The initial learning rate is set to $5e^{-2}$ and gradually decayed to $1e^{-6}$ using a cosine annealing curve. During training, a target label is randomly picked from all incorrect classes for each data point. On CIFAR-10, the CAG is trained for a total of 500 epochs using the batch size of 256. On ImagetNet, we train the CAG for 20 epochs with batch size of 64. The L_2 norm of adversarial perturbations is set to 0.1 for both datasets.

We compare our proposed method with other existing attack algorithms: I-FGSM, PGD, and C&W. We use FoolBox in PyTorch [21] to generate these adversarial examples. Our experiments are performed on NVIDIA Tesla V100 GPU.

5.2 CIFAR-10

We first evaluate our proposed CAG on CIFAR-10 in white-box scenario. The classifier is set to be ResNet-18, and the classification accuracy on clean images achieves 93.48% for 10,000 validation images. To evaluate the targeted attack algorithms, ASR is used

	ASR	**Acc.**	L_2	**Time**
I-FGSM	99.53%	0.05%	0.106	13m24s
PGD	99.56%	0.22%	0.106	12m56s
C&W	**99.85%**	**0.14%**	**0.009**	>10h
CAG	97.29%	1.4%	0.100	**1.44s**

Table 5.1: Comparison of adversarial examples generated by CAG and other methods on ResNet-18 (CIFAR-10).

Attacks / Classifiers	RN-34	VGG-11	VGG-13	DN-121	DN-169	**Average**
I-FGSM	30.45%	21.39%	23.88%	29.01%	28.70%	26.69%
PGD	45.38%	28.46%	34.90%	41.15%	40.05%	37.99%
C&W	7.57%	8.88%	8.40%	6.61%	7.70%	7.85%
CAG $p = 0.0$	86.47%	66.46%	94.09%	83.64%	85.74%	83.78%
CAG $p = 0.1$	89.02%	70.51%	**95.10%**	87.01%	88.52%	85.92%
CAG $p = 0.2$	90.83%	74.31%	94.93%	88.93%	90.37%	87.85%
CAG $p = 0.3$	**91.83%**	**77.81%**	94.89%	**90.49%**	**91.31%**	**89.24%**

Table 5.2: Comparison of transferbility (ASR) of various attack methods and CAG with different dropout rate p on ResNet-18 in black-box scenario (CIFAR-10).

as the performance metric.

5.2.1 Low Computation Time

We generate 10,000 adversarial examples in CIFAR-10 validation set, and each image is targeted to a randomly incorrect class. The ASR can reach 97.29% on the ResNet-18. We compare our proposed CAG with other state-of-art targeted attack methods. Similar to the procedure we use to evaluate CAG, we also choose attack targets in random manner. As for C&W, we only report first 1000 images targeted on random classes. Since the L_2 norm for CAG is set to be 0.1, for fair comparison, we try to keep L_2 norm around similar range for I-FGSM and PGD. Therefore, ϵ and α is set to 0.1 and 0.035, respectively. The maximum iteration is set to 50. When using C&W attack, we perform 10 iterations of binary search and run 10,000 iterations of gradient descent with learning rate at 0.005 using the Adam optimizer. We only generate 1,000 images using C&W attack. As can be seen from the Table 5.1, our attack achieves comparable results compared with I-FGSM, PGD, and C&W. However, our attack has much lower inference time of only 1.44 seconds compared of 12 minutes 56 seconds of PGD and more than 10 hours of C&W attack–a more than 500-fold speedup. The

	Storage	**White-box**				**Black-box**			
Attacks / Classifiers		RN-18	VGG-11	DN-121	**Average**	RN-34	VGG-13	DN-169	**Average**
GAP Unet (5T)	30 MB × 5	97.98%	**98.45%**	**97.85%**	**98.09%**	82.97%	85.69%	88.31%	85.66%
GAP ResNet (5T)	30 MB × 5	91.02%	94.25%	90.58%	91.95%	76.40%	86.27%	78.33%	80.33%
GAP (1000T)	30 MB × 1000	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CAG (5T)	222 MB	**98.52%**	97.71%	96.91%	97.71%	**95.45%**	**94.34%**	**94.06%**	**94.62%**
CAG (1000T)	222 MB	97.79%	97.01%	96.62%	97.14%	93.38%	94.28%	92.61%	93.42%

Table 5.3: Storage and ASR comparison of adversarial examples generated by CAG and GAP (ImageNet). Both are trained on ensemble of models: RN-18, VGG-11 and DN-121. 5T and 1000T represents 5 and 1000 targeted classes, respectively. Due to the limitation of storage and impractical training time, we can not report the attack results on GAP with 1000T.

	White-box					**Black-box**				
Defense Methods	RN-18	VGG-11	DN-121	**Average**	**Average I-FGSM**	RN-34	VGG-13	DN-169	**Average**	**Average I-FGSM**
None	0.59%	1.03%	1.50%	**1.04%**	4.58%	2.10%	1.51%	3.85%	**2.49%**	42.17%
Pixel Deflection	24.75%	26.39%	31.76%	27.63%	**14.76%**	33.83%	23.44%	43.45%	**33.57%**	57.25%
Randomization	3.06%	3.07%	6.80%	**4.31%**	22.19%	5.26%	2.77%	10.23%	**6.09%**	43.90%
Bit Depth Reduction	4.80%	7.33%	10.41%	**7.51%**	10.22%	11.12%	10.18%	17.85%	**12.96%**	50.08%
JPEG Compression	5.81%	7.63%	11.05%	**8.16%**	12.62%	12.17%	10.48%	17.78%	**13.48%**	49.36%

Table 5.4: Classification accuracy of CAG's adversarial images versus I-FGSM's after applying defense mechanisms (ImageNet).

ability to generate a large number of adversarial images in a such a small time makes our attack method practical in real-time applications.

5.2.2 High Transferability

CAG always has ASR greater than 95% in white-box attack scenario. However, considering black-box attack, when attackers have no access to architecture and parameters of the classifier, ASR is not as high as the white-box scenario. To address this high transferbility requirement, we propose to drop out part of the perturbation before adding it on the benign image during training phase. As a result, CAG generalizes better and is less prone to over-fitting to a particular classifier. Hence, the transferability of the adversarial examples to new classifiers increases. We train 4 CAG models using ResNet-18 with dropout probability $p = 0.0$, $p = 0.1$, $p = 0.2$ and $p = 0.3$. The ASR for 10,000 validation images (only 1000 images for C&W) targeted on random incorrect classes are reported. Table 5.2 reveals that even without dropout, CAG still has better performance in black-box results compared with other methods. Furthermore, the transferability of adversarial examples improves with increasing dropout probability.

5.3 ImageNet

We also evaluate the CAG on ImageNet. In our experiments, CAG takes a long time to converge when trained with a single classifier. Thus to accelerate the training process and perform stronger attack, we train CAG with an ensemble of ResNet-18, VGG-11 and DenseNet-121. When training with an ensemble of classifiers, we observe that the CAG does not suffer from over-fitting as much as training with only one classifier. Hence, unlike the best configuration in CIFAR-10 where $p = 0.3$, we reduce the perturbation dropout to $p = 0.1$ in this case.

To explicitly demonstrate the performance of our proposed method, we compare our results with GAP [24]. To create a fair comparison, we implement GAP with two architectures and keep the configuration the same as our method. The first GAP uses identical generative architecture to ours, so we denote it GAP U-Net. The second GAP has the same architecture used in GAP's original paper, which we denote it GAP ResNet. However, to perform targeted attack, GAP requires 1 model for each targeted class. Because we do not have enough resources to train 1,000 GAP models to have a comprehensive evaluation, we train 5 models for each architecture targeted at these following random chosen classes: *black swan, Tibetan terrier, tiger beetle, cliff dwelling, hook.*

5.3.1 Low Memory Cost

The comparison result is shown in Table 5.3. 10,000 benign images are randomly picked from the validation dataset to do the evaluation. We use CAG to generate adversarial examples targeted at the same 5 selected labels for fair comparison. In addition, since our proposed CAG can perform the comprehensive targeted attack on all 1,000 classes, we also generate adversarial images crossing all classes. In the table, 5T means ASR are evaluated on a pool of the same 5 targeted classes using 10,000 images in ImageNet evaluation dataset. In the last row, 1000T means that 10,000 images are targeted to any randomly selected label from all 1,000 classes. As can be seen from the table, to perform comprehensive attacks to all 1000 classes of ImageNet, our model takes 222MB of storage: 30MB for model's weights, and 192MB for the embeddings. However, other generative models can take up to 30MB × 1000 ≈ 30GB for storage to attack all classes. Moreover, as shown in Table 5.3, for seen classifiers, ASR is above 90% for all approaches. On the one hand, while targeting 5 selected classes, adversarial images generated by GAP U-Net and CAG have comparable performance. On the other hand, by analyzing the result of unseen classifiers, we can see that CAG outperforms GAP. ASR of CAG can reach to 93.42% for 1,000 target labels in black-box scenario. To sum up, our proposed CAG is more practical to perform general targeted attack while keeping high ASR and transferability.

5.4 Breaking Defenses

5.4.1 Enhanced Robustness

Finally, we study the robustness of adversarial examples generated by CAG on ImageNet. We prepare CAG trained on the ensemble of ResNet-18, VGG-11 and DenseNet-121. Using the optimal setting, the dropout probability is set to 0.1. Since it is meaningless to protect images that are originally mis-classified, we evaluate 10,000 (ImageNet) images that are correctly classified by all three classifiers. We use the following configurations:

5.4.2 Pixel Deflection

To achieve the strongest defense performance we provide the CAMs of the true class of correctly classified images to guide the pixel deflection (unlike using CAMs of top-5 predictions as the original paper suggests). We set the parameters following the original paper with window=10, deflections=100.

5.4.3 Randomization

To perform this defense with optimal parameters, we keep the scale ratio the same as the ratio reported in the original paper. Thus the image size is modified from $224 \times 224 \times 3$ to $253 \times 253 \times 3$ in our implementation.

5.4.4 Bit-depth Reduction

In our experimental setting, we reduce images to 3 bits as the original paper [22].

5.4.5 JPEG Compression

We perform JPEG compression at quality level 75 out of 100.

Classification accuracy after applying defense methods is shown in Table 5.4. As a result of using CAM guidance in proposed CAG, our attack is robust against defense methods that aim to modify the non-discriminative regions such as pixel deflection and randomization. After using pixel deflection, classifiers accuracy on CAG's generated adversarial images is still low at 27.63% (white-box) and 33.34% (black-box). In addition, CAG's adversarial images can bypass the defense effects of input transformation. Bit depth reduction and JPEG compression can not improve the accuracy more than 10% for white-box and 14% for black-box setting. Compared with I-FGSM, our attack achieves lower classification accuracy in almost all categories. To sum up, our attack is robust against many defense mechanisms.

Chapter 6

Conclusions

In this work, we propose a generative model to perform targeted adversarial attacks called CAG. With the help of the trainable embedding layer, the supervision of CAMs and random dropout, CAG is able to produce robust adversarial examples with state-of-art attacking performance and high transferability, while still maintaining low computation time and low memory cost. CAG has many desirable properties of an adversarial attack method, and therefore outperforms many other methods and can launch a real-time robust attack against many modern DNN systems.

Bibliography

[1] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[3] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] D. Yu and L. Deng, *AUTOMATIC SPEECH RECOGNITION.* Springer, 2014.

[9] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao, "Foveation-based mechanisms alleviate adversarial examples," *arXiv preprint arXiv:1511.06292*, 2015.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[11] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," *arXiv preprint arXiv:1606.01781*, vol. 2, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, Springer, 2016, pp. 694–711.

[14] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[15] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[16] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017.

[17] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 39–57.

[18] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[19] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[21] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017. arXiv: `1707.04131`. [Online]. Available: `http://arxiv.org/abs/1707.04131`.

[22] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[23] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[24] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4422–4431.

[25] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8571–8580.

[26] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018.