AFRL-RI-RS-TR-2022-001

CULTIVATING ROBUSTNESS FOR DEEP LEARNING

RESEARCH FOUNDATION OF THE CITY UNIVERSITY OF NEW YORK

JANUARY 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ AIR FORCE MATERIEL COMMAND ■ UNITED STATES AIR FORCE ■ ROME, NY 13441

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC)(http://www.dtic.mil).

AFRL-RI-RS-TR-2022-001 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
NOAH MCDONALD
Work Unit Manager

/ S /
GREGORY HADYNSKI
Assistant Technical Advisor
Computing &Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
JANUARY 2022	FINAL TECHNICAL REPORT	January 2018	July 2021

4. TITLE AND SUBTITLE

CULTIVATING ROBUSTNESS FOR DEEP LEARNING

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-18-2-0058	N/A	62788F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
T2EB	CU	NY

6. AUTHOR(S)

Myung Lee, Bo Yuan and Xue Lin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

RESEARCH FOUNDATION OF THE CITY UNIVERSITY OF NEW YORK
160 Convent Ave
New York NY 10031-9101

8.PERFORMING ORGANIZATIONREPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITB
525 Brooks Road
Rome NY 13441-4505

10.SPONSOR/MONITOR'S ACRONYM(S)

RI

11.SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-RI-RS-TR-2022-001

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. **SUPPLEMENTARY NOTES**

14. ABSTRACT

In this project we report our research activities on exploring the vulnerability and improving the robustness of deep learning against adversarial attacks. To be specific, we systematically analyze and investigate the efficient attack and defense mechanism of DNN models across different domains for different types of models with different constraints . In addition, the efficient interpretation and verification are also studied and developed, thereby providing useful perspective to detect and check the correctness of the input and DNN models.

15. SUBJECT TERMS

Robustness, Deep learning models, real-time adversarial attacks, defense mechanisms, graph neural networks, universal attack, fast and universal adversarial attacks in audio domain, interpretation and validation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	73

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
NOAH MCDONALD	315-330-2248

Contents

List of Figures

List of Tables

Publication Outcomes

Total **11** papers published with **261** Citations:

[1] Z. Li, C. Shi, T. Zhang, Y. Xie, J. Liu, B. Yuan and Y. Chen, "Robust Detection of Machine-induced Audio Attacks in Intelligent Audio Systems with Microphone Array," ACM Conf. on Computer and Communications Security (CCS), Nov. 2021.

[2] Z. Xiao, Y Xie, J. Chen and B. Yuan, "Graph Universal Adversarial Attacks: A Few Bad Actors Ruin Graph Learning Models," International Joint Conferences on Artificial Intelligence (IJCAI), Aug. 2021.

[3] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C. Hsieh, "Fast and complete: enabling complete neural network verification with rapid and massively parallel incomplete verifiers," International Conference on Learning Representations (ICLR), April 2021.

[4] Y. Xie, Z. Li, C. Shi, J. Liu, Y. Chen and B. Yuan, "Real-time, Robust and Adaptive Universal Adversarial Attacks Against Speaker Recognition Systems," Springer Journal of Signal Processing System (JSPS), Feb. 2021.

[5] Y. Xie, Z. Li, C. Shi, J. Liu, Y. Chen and B. Yuan, "Enabling Fast and Universal Audio Adversarial Attack Using Generative Model," AAAI Conference on Artificial Intelligence (AAAI), Feb. 2021.

[6] X. Wang, S. Wang, P. Chen, X. Lin, and P. Chin, "ADVMS: a multi-source multi-cost defense against adversarial attacks," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2020.

[7] Y. Xie, Z. Li, C. Shi, J. Liu, Y. Chen and B. Yuan, "Real-time, Universal, and Robust Adversarial Attacks against Speaker Recognition Systems," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2020.

[8] K. Xu, H. Chen, S. Liu, P. Chen, T. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," International Joint Conference on Artificial Intelligence (IJCAI). Aug. 2019.

[9] H. Phan, Y. Xie, S. Liao, J. Chen and B. Yuan, “CAG: A Real-time Low-cost Enhanced-robustness High-transferability Content-aware Adversarial Attack Generator,” AAAI Conference on Artificial Intelligence (AAAI), Feb. 2020.

[10] X. Wang, S. Wang, P. Chen, X. Lin, and P. Chin, “Block Switching: A Stochastic Approach for Deep Learning Security,” KDD Workshop on Adversarial Learning Methods for Machine Learning and Data Mining (AdvML), 2019.

[11] K. Xu, S. Liu, P. Zhao, P. Chen, H. Zhang, Q. Fan, D. Erdogmus, Y. Wang, and X. Lin, “Structured adversarial attack: Towards general implementation and better interpretability,” International Conference on Learning Representations (ICLR), April 2019.

Abstract

Deep neural networks (DNNs) have been widely used in many important applications, such as computer vision, speaker recognition, natural language processing etc. Despite their current popular adoptions, DNN models are facing security issues when considering their practical use in many critical systems. In particular, the vulnerability of DNN models under adversarial attack, an emerging attack approach that only performs imperceptible perturbation on the input, has emerged as a potential challenge that may hinder the further deployment of deep learning.

In this project we report our research activities on exploring the vulnerability and improving the robustness of deep learning against adversarial attacks. To be specific, we systematically analyze and investigate the efficient attack and defense mechanism of DNN models across different domains (image and audio), for different types of models (convolutional neural networks and graph neural networks) with different constraints (memory and timing). In addition, the efficient interpretation and verification are also studied and developed, thereby providing useful perspective to detect and check the correctness of the input example and the functional validity of the deployed DNN models. The research outcomes and the delivered approaches will facilitate the deep understanding and development of DNN models when considering the efficient deployment in the practical systems.

Section 1
Summary

Deep neural networks (DNNs) have served as the backbone machine learning technique in many intelligent systems. Considering their widespread adoption in many important applications, especially for those critical and security-sensitive scenarios, the vulnerability and robustness of DNNs should be carefully examined. In particular, the performance of DNN models under the adversarial attack, a popular attack strategy that only perform small perturbation to the inputs yet causing severe misclassification, should be evaluated and investigated in a comprehensive way.

In order to obtain the deep understanding of the vulnerability of DNN models and cultivate their robustness, this project performs systematic investigation and study on the adversarial attack and defense of deep learning across different domains and different types of neural network. The research outcomes and technical contributions from this project are summarized as follows:

- To explore the vulnerability of DNN models, we develop advanced optimization-based structured adversarial attack method, which can bring better attacking performance and imperceptible perturbation. In addition, we also develop generative model-based fast adversarial example generation scheme, which can launch real-time adversarial attack with low memory footprint requirement and high attacking performance.
- To efficiently defend the DNN models against adversarial attacks, we develop a model switching mechanism that can stochastically and dynamically modify the DNN models without affecting task accuracy, thereby providing very strong defense against the adversary. In addition, we also develop a multi-cost multi-source defend strategy to further improve the robustness of the trained model.
- Considering the wide application of graph neural networks (GNNs), we develop efficient attack and defense approaches for GNNs. Specifically, we develop first-order gradient-based attack methods and the optimization-based topology defense solution. In addition, we also study and develop universal attack to the GNN models, which can only change very few nodes and then affect the classification results. To the best of our knowledge, this is the first universal attack to GNN models.

- We also study the vulnerability issue of DNN models in the audio domain. In particular, our proposed approaches fully consider the complicated nature of audio signals and the unique constrains in the audio applications. To be specific, we develop a universal audio adverbial attack method that can be directly applied to many audio clips with high transferability. In addition, we also develop a real-time fast audio adversarial example generation scheme, which is meanwhile robust to the over-the-air propagation.
- Besides the throughout analysis and empirical experiments, we also investigate the efficient interpretability and verification of adversarial examples and DNN models, respectively. We develop a class activation mapping-based approach to provide better understanding of the roles of adversarial perturbations. Also, we propose a complete and rapid formal verification for DNN models, which can bring order-of-magnitude speed improvement for DNN verification.

Section 2
Introduction

Deep learning has become ubiquitous in various intelligence-demanded applications, including object recognition, machine translations, speech recognition, scene understanding, etc. Based on its unprecedentedly powerful feature extraction and representation capability, deep learning is believed to be the most important and fundamental technique for building the current and future artificial intelligence infrastructure. Nowadays, deep learning researchers and domain experts are collaborating to promote the deployment of deep learning to a broad range of everyday applications, such as medical diagnostics, autonomous driving, smart office/household assistants etc. There is no doubt that the wide adoption of deep learning will revolutionize and reshape many aspects of the behaviors of people, community and society.

As deep learning has become an underlying data-analysis and decision-making tool in many fields, the incentives for adversaries to manipulate deep learning systems have arisen, such as to tamper with the correct outputs, or to produce their intended results. In particular, the recently reported black-box attacks using adversarial examples, which are legitimate for human-being while causing misclassification, have proved the vulnerability of deep learning systems. Such vulnerability, if not being properly protected, would cause severe economic, social, and safety problems in various applications, such as automatic fraud transaction detection, security surveillance systems, self-driving cars etc. This report summarizes several techniques developed in the project, aiming to investigate the vulnerability and improve the robustness of deep learning systems. To be specific, Section 3.1 presents several high-performance real-time adversarial attacks that can be launched in the practical applications. Correspondingly, Section 3.2 investigates the efficient defense mechanism against adversarial attack. Section 3.3 extends the attack and defense solution for image-domain neural networks to graph neural networks. Considering the wide adoption of deep learning in the existing audio applications, Section 3.4 studies the high-performance real-time adversarial attack approaches in the audio domain. Finally, to have better understanding of adversarial examples and robustness of deep learning, Section 3.5 further investigates the efficient interpretability and verification of neural networks,

Section 3 Methods, Assumptions, and Procedures

3.1 High-performance Real-time Attack

In this section, we report our research outcomes on the attack aspect. To be specific, we study and propose the efficient structured attack and real-time attack approaches against the deep neural networks used in the practical applications. These proposed methods show great interpretability and speed performance.

3.1.1 Structured Adversarial Attack

Given with datasets and neural network models, we develop a strong attack that achieves both high sparsity and low l_p norm (a measure of difference between an original image and an adversarial example, $p \in \{1, 2, \infty\}$) in the perturbations. This attack method is helpful for the broader applications of deep neural networks in the security-critical areas and provides interpretability of the adversarial examples.

To the best of our knowledge, this is the first attempt towards exploring group-wise sparse structures when implementing adversarial attacks, but without losing l_p distortion performance when compared to state-of-the-art attacking methods. The proposed attack model covers many norm-ball based attacks such as C&W [8] and EAD [11]. To be specific, we develop an efficient algorithm to generate structured adversarial perturbations by leveraging the alternating direction method of multipliers (ADMM). The generated adversarial perturbations demonstrate the clear correlations and interpretations between original and target images.

More specifically, we first define the group sparsity. Let $\Delta \in \mathbb{R}^{W \times H \times C}$ be an adversarial perturbation added to an original image X_0, where $W \times H$ gives the spatial region, and C is the depth, e.g., $C = 3$ for RGB images. A sliding mask M finally divides Δ into a set of groups $\{\Delta G_{p,q}\}$ for $p \in [P]$ and $q \in [Q]$, where $P = (W - r)/S + 1, Q = (H - r)/S + 1$, and $[n]$ denotes the integer set $\{1, 2, \ldots, n\}$. Given the groups $\{\Delta G_{p,q}\}$, the group sparsity canbe characterized through the following sparsity-inducing function [3, 16, 28], motivated by the problem of group Lasso [41]:

$$g(\Delta) = \sum_{p=1}^{P} \sum_{q=1}^{Q} \| \boldsymbol{\Delta}_{\mathbf{G_{p,q}}} \|_2, \tag{3.1}$$

where $\Delta G_{p,q}$ denotes the set of pixels of Δ indexed by $G_{p,q}$, and $\| \|_2$ is the l_2 norm.

Then we propose a general formulation of designing adversarial attacks taking into account group sparsity and l_p norm. Given an original image $x_0 \in \mathbb{R}^n$, we aim to design the optimal adversarial perturbation $\boldsymbol{\delta} \in \mathbb{R}^n$ so that the adversarial example $(x_0 + \boldsymbol{\delta})$ misleads DNNs trained on the natural images. Throughout this paper, we use vector representations of the adversarial perturbation Δ and the original image x_0 without loss of generality. A well-designed perturbation $\boldsymbol{\delta}$ can be obtained by solving the following optimization problems:

$$\min_{\delta} f(x_0 + \delta, t) + \gamma D(\delta) + \tau g(\delta) \tag{3.2}$$

$$subject\ to\ (x_0 + \delta) \in [0,1]^n, ||\delta||_\infty \leq \epsilon,$$

where $f(x, t)$ denotes the loss function for crafting adversarial example given a target class t, $D(\boldsymbol{\delta})$ is a distortion function that controls the perceptual similarity between an image and a perturbed image.

In problem equation 3.2, the 'hard' constraints ensure the validness of created adversarial examples with ϵ-tolerant perturbed pixel values. And the non-negative regularization parameters γ and τ place our emphasis on the distortion of an adversarial example (to an original image) and group sparsity of adversarial perturbation.

Next, we reformulate problem equation 3.2 into the form of ADMM,

$$\min_{\delta,z,w,y} f(\boldsymbol{z} + x_0) + \gamma D(\delta) + \tau \sum_{i=1}^{PQ} |\, \boldsymbol{y}_{\mathcal{D}_i}|_2 + h(\boldsymbol{w}) \tag{3.3}$$

$$subject\ to\ \boldsymbol{z} = \delta, \boldsymbol{z} = y, \boldsymbol{z} = w,$$

where $\boldsymbol{z}, \boldsymbol{y}$ and $\boldsymbol{w}$ are newly introduced variables, for ease of notation let $D_{(q-1)P+p} = \mathrm{G}p, q$,and $h(\boldsymbol{w})$ is an indicator function with respect to the constraints of problem equation 3.2,

$$h(\boldsymbol{w}) = \begin{cases} 0 & if\ (x_0 + w) \in [0,1]^n, ||w||_\infty \leq \epsilon, \\ \infty & otherwise. \end{cases} \tag{3.4}$$

ADMM is performed by minimizing the augmented Lagrangian of problem equation 3.3,

$$L(z, \delta, y, w, u, v, s) = f(\boldsymbol{z} + x_0) + \gamma D(\boldsymbol{\delta}) + \tau \sum_{i=1}^{QP} \|\boldsymbol{y}_{D_i}\|_2 + h(\boldsymbol{w}) + \boldsymbol{u}^T(\boldsymbol{\delta} - \boldsymbol{z})$$

$$+\boldsymbol{v}^T(\boldsymbol{y}-\boldsymbol{z})+\boldsymbol{s}^T(\boldsymbol{w}-\boldsymbol{z})+\frac{\rho}{2}\|\boldsymbol{\delta}-\boldsymbol{z}\|_2^2+\frac{\rho}{2}\|\boldsymbol{y}-\boldsymbol{z}\|_2^2+\frac{\rho}{2}\|\boldsymbol{w}-\boldsymbol{z}\|_2^2, \tag{3.5}$$

where $\boldsymbol{u}, \boldsymbol{v}$ and $\boldsymbol{s}$ are Lagrangian multipliers, and $\rho > 0$ is a given penalty parameter. And ADMM splits all of optimization variables into two blocks and adopts the following iterative scheme:

$$\{\delta^{k+1}, w^{k+1}, y^{k+1}\} = \underset{\delta,w,y}{\operatorname{argmin}} L(\delta, z^k, w, y, u^k, v^k, s^k), \tag{3.6}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} L(\delta^{k+1}, z, w^{k+1}, y^{k+1}, u^k, v^k, s^k), \tag{3.7}$$

$$\begin{cases} u^{k+1} = u^k + \rho(\delta^{k+1} - z^{k+1}), \\ v^{k+1} = v^k + \rho(y^{k+1} - z^{k+1}), \\ s^{k+1} = s^k + \rho(w^{k+1} - z^{k+1}), \end{cases} \tag{3.8}$$

where k is the iteration index, equation 3.6-equation 3.7 are used for updating primal variables, and the last step equation 3.8 is known as the dual update step.

3.1.2 Real-time Adversarial Attack

Besides structured adversarial attack, we also study the efficient real-time adversarial attack. To date, the existing iteration-based approaches predominate among current state-of-the-art attack methods, including PGD and C&W. Consequently, generating adversarial examples using iteration is time-expensive and requires lots of computational resources, especially for the targeted attack. For example, to achieve a high attack successful rate, 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.

Motivated by this challenge, we aim to develop an adversarial attack method that can generate each adversarial example in a real-time manner and exhibit strong robustness against the-state-of-the-art defense techniques. Following those requirements, we propose content-aware generator (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.

Overall Architecture. 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 on the 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 making the real-time generation of adversarial examples possible.

Single Generative Model via Label Embedding. One 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. Next, we will introduce 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 represents height, width and number of channels, respectively. Then during training phase, an embedding layer E_{hwL} with thesize 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 differenttarget classes.

Enhanced Attack Robustness using CAM. Besides using an embedding layer,the construction of input tensor T also utilizes the information of CAMs [43]. Classifiers make decisions base heavily on the hot areas of CAM because they contain the most discriminative information of an image. Therefore, defense methods cannot make huge modification in these critical areas, otherwise they could easily cause misclassification. 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. Another component of input tensor T is CAM^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:

$$T = \tau\left(x_i,\ E_i, E_t, CAM_x^i\right), \tag{3.9}$$

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

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

$$F\left(clip(G(T) + x_i)\right) = y_t. \tag{3.10}$$

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

$$Loss_{target} = CrossEntropy\left(y_t, y_{pred}\right). \tag{3.11}$$

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, we need to minimize the difference between $CAM_{x'}^t$ and CAM_x^i to qualify the similarity. Therefore, $Loss_{CAM}$ is defined to lead the distribution of the noise:

$$Loss_{CAM} = \left\| CAM_x^i - CAM_{x'}^t \right\|^2. \tag{3.12}$$

Finally, the new loss function is formulated as:

$$Loss = Loss_{target} + \beta \cdot Loss_{CAM}, \tag{3.13}$$

where β controls the magnitude of $Loss_{CAM}$. We then iteratively optimize the CAG as well as E_{hwL} by minimizing the $Loss$ function.

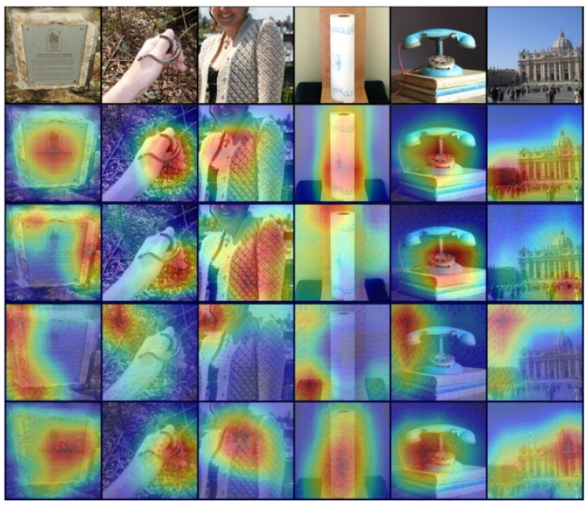


Figure 1: CAM attention visualization. From top row to bottom row: clean images, CAM of clean images, CAM of PGD's adversarial images, CAM of GAP's adversarial images, CAM of CAG's adversarial images.

We show the attention regions using CAM for adversarial examples generated by different attack methods. As illustrated in Figure 1, comparing with the clean images, I-FGSM and GAP achieve targeted attack by misleading the network's attention. However, we believe that this would make adversarial images vulnerable to designed defense mechanisms. Interestingly, as can be seen in the last row of Figure 1, the adversarial images generatedby our proposed CAG do not suffer this problem. Malicious perturbations are constrained to locate in the discriminative areas, so that CAG's adversarial examples are robust to circumvent detection and defense methods.

3.2 Efficient DNN Defense Mechanism

Built on the obtained understanding for the vulnerability of DNN under adversarial attack, in this section we further study the efficiency defense mechanism. To be specific, we propose two defense approaches: model switching and multi-source multi-cost defense.

3.2.1 Switching Model

Given with datasets and adversarial attack methods, we train a new network model that can resist even white-box attacks generated using the adversarial attack methods. The

research project enables the broader applications of deep neural networks for some security-critical areas.

In general, we introduce the models with the Block Switching strategy, which shows effectiveness in defending against adversarial attacks. The switching model is trained in the following process, which is shown in Fig. 2. To be specific, a number of regular models, which we refer to as sub-models, are trained in a conventional manner. With the same model hyper-parameters, training approaches and data, they tend to have similar classification accuracy while the value of weight parameters are different because of the random initialization and stochastic factors in the training. Then, the trained sub-models are cut into two parts in the same division manner. We call the part containing the input layer the lower part, and the one containing output layer the upper part. We discard the upper parts of all sub-models but keep all lower parts. Also, we connect the outputs of all lower parts to a new common upper model which has the same architecture as the upper parts of the sub-models but has the weights randomly initialized. We refer to the lower parts of the sub-models as different "channels" of the overall switching model. An input switcher is added as the common input gate of all channels which assign the input to one of the channels randomly at the time of forward propagation. We call a channel "activated" if it is selected to taking the input by the random switcher.

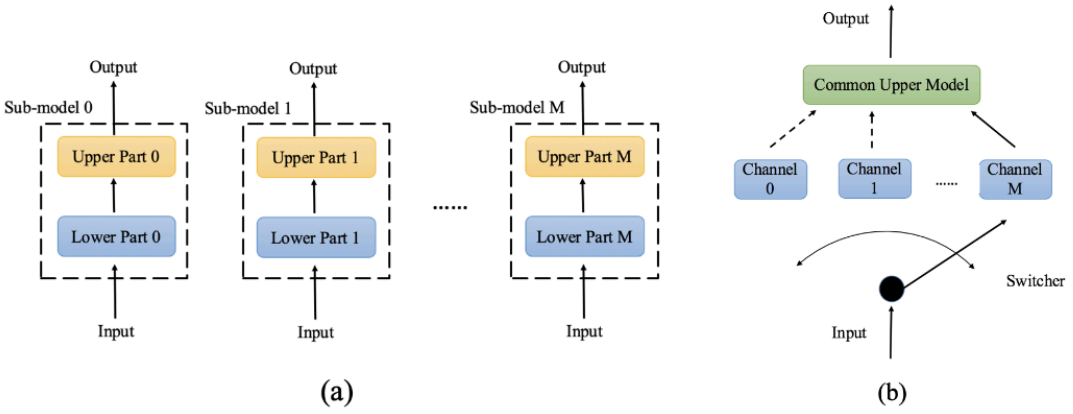


Figure 2: The steps of assembling a block switching. (a): Sub-models are trained individually. (b): The lower parts of sub-models are used to initialize parallel channels of block switching.

3.2.2 A Multi-source Multi-cost Defense

Conventional defense methods, although shown to be promising, are largely limited by their single-source single-cost nature: The robustness promotion tends to plateau when

the defenses are made increasingly stronger while the cost tends to amplify. We propose a multi-source and multi-cost defense scheme, Adversarially Trained Model Switching (AdvMS), which inherits advantages from two leading schemes: adversarial training and random model switching.

We show that the multi-source nature of AdvMS mitigates the performance plateau issue and the multi-cost nature enables improving robustness at a flexible and adjustable combination of costs over different factors which can better suit specific restrictions and needs in practice.

We first explain the terms source and cost of a defense scheme using adversarial training [24] as an example, as it is considered one of the most advanced one in the literature. Adversarial training improves the worst-case robustness of a neural network by incorporating adversarial examples in the training process. As a result, the adversarially trained weights learn to adapt inputs with adversarial perturbations. We thus term the weights of adversarially trained models the source of the defense, as they are the surface of performing defense efforts. The cost of adversarial training is the test accuracy of the model on natural examples: the defense can be made stronger by using larger adversarial perturbations for training, at the cost of worse test accuracy on clean examples.

We call defense schemes in this fashion single-source single-cost defenses, and most existing defenses fall into this category. As they are implemented alone, the drawbacks are obvious. Since there is only one scheme boosting robustness, the benefit in robustness tends to plateau with increased defense strength. Another issue is that the cost on a particular factor can easily exceed the acceptable range, making the defense strategy infeasible.

A natural solution to the above limitations is to design multi-source multi-cost defense schemes. Although many proposed defenses claim to be compatible with others, designing complex defense is not trivial. We anticipate the following principles of combining multiple defense components as a complex defense. First, components need to have different defense source. This ensures the defenses are not intertwined which may cause degraded performance. Second, components need to have different defense cost. This avoids trading robustness with a single cost factor which leads to decreased marginal defense improvement, and at the same time it allows a more flexible combination of costs over multiple factors.

Following the above guidelines, we propose a complex defense AdvMS by merging adversarial training and switching-based stochastic defense. The scheme of model

switching [44] protects the model by using a group of parallel sub-models trained from different random initialization, where the active one that processes model inputs is ever-switching in the run time. It will not degrade test accuracy but increases the memory consumption in the testing time. It thus has different source (stochasticity) and cost (memory consumption) compared to adversarial training.

Implementing AdvMS with M sub-models contains 3 steps: *step 1*, randomly initialize the weights of M sub-models of the same architecture and train them individually with the same training settings (ϵ_{train}), and training data; *step 2*, save all M adversarially trained models and construct a pool of parallel sub-models; *step 3*, add the stochastic scheme that randomly activate one model from the pool in each round of inference.

3.3 Attack and Defense for Graph Neural Networks

In this section, we extend our study on deep neural network to the graph neural network and investigate the corresponding attack and defense approaches. To be specific, we first propose topology-based adversarial attack and the corresponding defense mechanism for graph neural network, and we then investigate the feasibility of universal attack.

3.3.1 Topology Attack and Defense for Graph Neural Network

Graph structured data are ubiquitous with examples ranging from proteins, power grids, traffic networks, to social networks. Deep learning models for graphs, in particular, graph neural networks (GNN) [17][19][34] attracted much attention recently and have achieved remarkable success in several tasks, including community detection, link prediction, and node classification. Their success is witnessed by many practical applications, such as content recommendation [38], protein interaction [33], and blog analysis [12].

As a type of neural network, graph neural network may also be vulnerable to adversarial attack. For instance, Wikipedia hoax articles can effectively disguise through modifying their links in a proper manner [20]. For another example, frauds may hide them selves through building plausible friendship in a social network, to confuse the prediction system.

At first, we present a novel gradient-based attack method that facilitates the difficulty of tackling discrete graph data. When comparing to current adversarial attacks on GNNs, the results show that by only perturbing a small number of edge perturbations, including

addition and deletion, our optimization-based attack can lead to a noticeable decrease in classification performance.

After the proposed gradient-based attacks on GNNs, we next introduce the robust training as an effective defense against topology attacks on GNNs. The research project enables the broader applications of deep graph neural networks with more robustness and security.

We propose a general first-order attack generation framework under two attacking scenarios: a) attacking a pre-defined GNN and b) attacking a re-trainable GNN. We first define attack loss (beyond the conventional cross-entropy loss) under different attacking scenarios. We then develop two efficient attack generation methods by leveraging first-order optimization. We call the resulting attacks projected gradient descent (PGD) topology attack and min-max topology attack, respectively.

With the aid of first-order attack generation methods, we now introduce our adversarial training for GNNs via robust optimization. Similar approach was applied to convolutional neural networks [24]. Here we solve a min-max problem for robust optimization. We use the same attack loss as in the proposed min-max topology attack. Following the idea of adversarial training for image classifiers, we restrict the loss function f as the CE-type loss (cross-entropy loss). The min-max problem tries to minimize the training loss at the presence of topology perturbations.

$$\min_{W} \max_{s \in S} -f(s, W), \tag{5.1}$$

This min-max problem share a similar form as that for the CE-type attack, however, they are not equivalent since the loss function here is neither convex with respect to s nor concave with respect to W, namely, lacking saddle point property. However, we build the connections between them through Proposition 2 in our IJCAI 2019 paper [39].

The solution to the min-max problem for robust optimization is we perform multiple inner minimization steps with respect to s within each iteration t to have a solution towards minimizer during alternating optimization. This improves stability of convergence in practice.

3.3.2 Universal Attack to Graph Neural Networks

Here we study the vulnerability of GNNs and show that it is possible to attack them if a few graph nodes serve as the bad actors: when their links to a certain node are flipped,

the node will likely be misclassified. Such attacks are akin to universal attacks because the bad actors are universal to any target. We propose a graph universal adversarial attack method, GUA, to identify the bad actors.

Notice that the universal attack to GNN is different from conventional adversarial attack to GNN in the attack setting. Prior work focuses on poisoning attacks (injecting or modifying training data as well as labels to foster a misbehaving model) and evasion attacks (modifying test data to encourage misclassification of a trained model). For graphs, these attacks could modify the graph structure and/or node features in a target-dependent scenario. The setting we consider, on the other hand, is a single and universal modification that applies to all targets. One clear advantage from the attack point of view is that computing the modification incurs a lower cost, as it is done once for all.

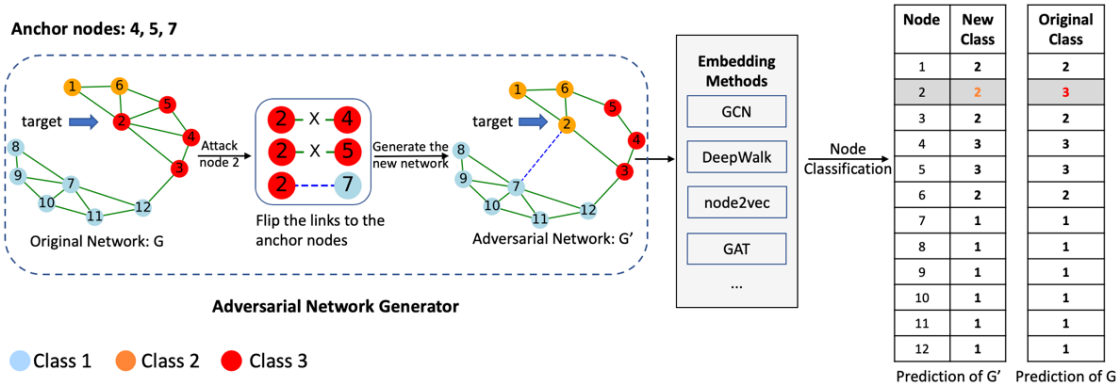


Figure 3: Illustration of GUA. A small number of anchor nodes (4, 5, and 7) is identified. To confuse the classification of a target node (e.g., 2), their connections to this node are flipped.

Figure 3 illustrates the universal attack setting we consider. A few bad-actor nodes (4, 5, and 7) are identified; we call them *anchor nodes*. When an adversary attempts to attack the classification of a target node (say, 2), the existing links from the anchor nodes to the target node are removed while non-existing links are created. The identification method we propose, GUA, is conducted on a particular classification model (here, GCN), but the found anchors apply to other models as well (e.g., DeepWalk, node2vec, and GAT).

As a type of attack, universal attacks may be preferred by the adversary for several reasons. First, the anchor nodes are computed only once and there incurs no extra cost when attacking individual targets. Second, the number of anchors can be very small (it is easier tocompromise fewer nodes). Third, attacks are less noticeable when only a limited number of links are flipped.

Notation. Assume a graph is denoted as $G = (V, E)$, where V is the node set and E is the edge set. An unweighted graph is represented by the adjacency matrix$A = \{0,1\}^{|V|\times|V|}$. The graph nodes have d-dimensional features, which collectively form the feature matrix X, whose dimension is $|V| \times d$. In GCN, one normalizes the adjacency matrix into $\hat{A} = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$, where $\tilde{A} = A + I$ and $\widetilde{D}$ is the diagonal adjusted degree matrix with diagonal entries $\widetilde{D}_{ii} = \sum_{j=1}^{|V|} A_{ij}$. Then, the neural network is

$$Z = f(A, X) = softmax\big(\hat{A}ReLU\big(\hat{A}XW^{(0)}\big)W^{(1)}\big), \tag{5.2}$$

where $W^{(0)}$ and $W^{(1)}$ are model parameters. The training of the parameters uses the cross-entropy loss.

Graph Universal Adversarial Attack. Following the introduced notation introduced, given the graph adjacency matrix A and node feature matrix X, we let $f\ (A, X)$ be the classification model and let $\hat{l}(A, X, i)$ be the predicted label of node i; that is, $\tilde{l}(A, X, i) = \arg\max f(A, X)_i$. Given a trained model , the goal is for each node i to modify the adjacency matrix A into A' such that $\hat{l}(A, X, i) \neq \hat{l}(A', X, i)$. Note that the modified A' is independent in our attack setting.

Attack Vector and Matrix. Let the graph have n nodes. We use a length-n binary vector p to denote the attack vector to be determined, where 1 means an anchor node and 0 otherwise. Hence, A' is a function of three quantities: the original adjacency matrix A, the target node i, and the attack vector p. To derive an explicit form of the function, we extend the vector p to an $n \times n$ matrix P, which starts as a zero matrix, with the i-th row and i-th column replaced by the attack vector p. Thus, the (i, j) element of the attack matrix P indicates whether the connection of the node pair (i, j) is flipped: 1 means yes and 0 means no.

It is then not hard to see that one may write the function

$$A' \coloneqq g(A, i, p) = (\mathbf{1} - P) \circ A + P \circ (\mathbf{1_0} - A), \tag{5.3}$$

where $\mathbf{1}$ denotes the matrix of all ones and $\mathbf{1}_0$ is similar except that the diagonal is replaced by zero; $\circ$ means element-wise multiplication. The term $(\mathbf{1} - P)$ serves as the mask that preserves the connections of all node pairs other than those between the anchors j and the target node i. The term $(\mathbf{1_0} - A)$ intends to flip the whole A (except diagonal) but the P in the front ensures that only the involved (i, j) pairs are actually flipped. Moreover, one can verify that the diagonal of the new adjacency matrix remains zero. For gradient based

optimization, the binary elements of the attack vector p may be relaxed into real values between 0 and 1 . In this case, the connections of all node pairs other than those between the anchors j and the target node i remain the same. On the other hand, the connections between the involved (i, j) pairs are fractionally changed. The jth element of p indicates the strength of change.

Outer Procedure: GUA. Let V_L be the training set with known node labels. Given an attack success rate threshold δ, we formulate the problem as finding a binary vector p such that

$$Err(V_L) \coloneqq \frac{1}{|V_L|}\sum_{i=1}^{|V_L|} \mathrm{b1}\{\hat{l}(A', X, i) \neq \hat{l}(A, X, i)\} \geq \delta. \tag{5.4}$$

To effectively leverage gradient-based tools for adversarial attacks, we perform a continuous relaxation on p so that it can be iteratively updated. Now elements of p stay in the interval [0, 1]. The algorithm proceeds as follows. We initialize p with zero. In each epoch, we begin with a binary p and iteratively visit each training node i. If i is not misclassified by the current p, we seek a minimum continuous perturbation Δp to misclassify it. In otherwords,

$$\Delta p = \underset{r}{\operatorname{argmin}} \left|\left|r\right|\right|_2, s.t.\, \hat{l}(g(A, i, p + r), X, i) \neq \hat{l}(A, X, i). \tag{5.5}$$

We will elaborate in the next subsection an algorithm to find such Δp. After all training nodes are visited, we perform a hard threshold at 0.5 and force back p to be a binary vector. Then, the next epoch begins. We run a maximum number of epochs and terminate when equation 5.4 is satisfied.

The updated $p \leftarrow p + \Delta p$ found through solving equation 5.5, if unbounded, may be problematic because (i) it may incur too many anchor nodes; and (ii) its elements may be outside [0, 1]. We perform an l_2-norm projection to circumvent the first problem and a clipping between interval [0, 1] to circumvent the second. The rationale of l_2-norm projection is to suppress the magnitude of p and encourage that eventually few entries are greater than 0.5. The maximum number of anchor nodes grows quadratically with the projection radius ξ. A small ξ clearly encourages fewer anchors.

Through experimentation, we find that clipping is crucial to obtain a stable result. Later we will illustrate an experiment to show that the attack success rate may drop to zero in several random trials if clipping is not performed.

Inner Procedure: IMP. To solve equation 5.5, we adapt DeepFool [26] to find a minimum perturbation that sends the target node *i* to the decision boundary of another class.

Denote by v the minimum $pred = \hat{l}(A, X, i)$ perturbation. To find the closest decision boundary other than that of the original class, we first select the closest class $k = argmin_{c \neq pred} \frac{|\Delta f_k|}{||\Delta w_k||_2^2} \Delta w_k$, where $\Delta f_c = f(A, X)_{i,c} - f(A, X)_{i,pred}$ and $\Delta w_c = \nabla f(A, X)_{i,c} - \nabla f(A, X)_{i,pred}$. Then, we update v by adding to it Δv:

$$\Delta v = \frac{|\Delta f_k|}{\left||\Delta w_k|\right|_2^2} \Delta w_k. \tag{5.6}$$

We iteratively update v until $(1 + overshoot) \times v$ successfully attacks node i, where *overshoot* is a small factor that ensures the node passes the decision boundary. We also clip the new A' to ensure stability, in a manner similar to the handling of p.

3.4 Fast and Universal Adversarial Attack to Deep Neural Network in the Audio Domain

Deep network networks (DNNs), with its superiority over current state-of-the- art models (e.g., universal background model-Gaussian mixture model) [21] [25], has been becoming the computation core of the speaker recognition systems. However, audio-domain DNN models are also vulnerable to adversarial voice input. In this section, we study two practical audio-domain adversarial attacks, namely robust universal audio attack and generator-based fast audio attack.

3.4.1 Robust Universal Audio Adversarial Attack

Threat Model. We consider the white box threat model where the adversary has full knowledge of the target speaker recognition model as well as its parameters. In order to build a robust adversarial attack considering the sound distortions in the room where the attack will be launched, we assume the adversary has access to the room's layout. We aim to find a single audio-agnostic universal perturbation that can be applied on arbitrary enrolled speakers' input audio to mislead the speaker recognition system causing it to output the specific adversary-desired speaker label. Additionally, we expect to build a more

robust adversarial perturbation that can remain effective while being played over-the-air in acoustic room simulated environment.

Challenges. Generating such a real-time, universal, and robust adversarial example against speaker recognition system in practice raises a number of challenges. *(1) Real-time Adversarial Attack.* To craft an adversarial noise with respect to the speaker's speech, using conventional optimization-based approach is usually very time-consuming, which makes many practical attack scenarios impossible, such as playing the adversarial noise on a hidden speaker in a real-time manner along with the speaker's voice input. *(2) Universal Targeted Adversarial Example.* Using an audio-agnostic universal perturbation to deceive the speaker recognition system, which causes it to misclassify any enrolled speaker's input audio as the adversary-desired speaker, needs to build a universal mapping from the audio sources to the adversary-desired target. The proposed algorithm needs to be general enough to various length audio inputs spoken by different speakers with various accents. *(3) Robust Adversarial Example.* The attack performance would be inevitably impacted by thesound distortions due to the attenuation and multi-path effects while playing the adversarialexamples over the air. Thus, the generated adversarial perturbation needs to be robust enough to remain effective under this kind of real-world distortions.

We explore how to build a single universal perturbation that can be directly applied to arbitrary speaker's any utterance, making the speaker recognition system output the adversary-desired speaker label. Such a universal perturbation would greatly shorten the attack launching time, making real-time attacks possible.

To clearly present the steps of our perturbation generation, we model the target speaker recognition system, X-vector, as a function $F(x)$, which takes as input an utterance x and outputs a predicted speaker label. We define $P(x)$ as the function of all DNN layers to compute the probabilities of classifying x as each of the profiledspeakers. We can recognize the voice as the speaker with highest calculated probability,$F(x) = Argmax(P(x))$. Therefore, to launch a universal targeted adversarial attack, where targeted speaker label is t, we aim to find a perturbation δ that could achieve $F(x+\delta) = Argmax(P(x+\delta)) = t$ for arbitrary x. To build such a universal attack, we need to find a general solution that can make the generated perturbation effective for all the utterances regardless of their speakers, accents, speech content and length. To overcome the issue of varying utterance length, we dynamically construct the universal perturbation δ based on the length of the input utterance x:

$$\delta = Crop([\Delta\delta - \cdots - \Delta\delta], x), \tag{6.1}$$

where $\Delta\delta$ is a short-length adversarial perturbation (e.g., 1s in our work), and [$\Delta\delta$ - ... -$\Delta\delta$] is a vector constructed by repeating $\Delta\delta$. $Crop(\cdot,\cdot)$ crops the first input to the length of the second input. With this process, the derived perturbation δ could be applied to the audio input with any length. To minimize the distortion between the adversarial example and the original voice, δ would be clipped to a pre-defined range. The generated adversarial example with the clipped δ could be formulated as:

$$x' = x + Clip_{\epsilon}(\delta), \tag{6.2}$$

where $Clip_{\epsilon}(\delta)$ is the function to perform element-wise clipping of δ. Values of δ outside the interval $[-\epsilon, \epsilon]$ would be clipped to the interval edges, and ϵ is our pre-defined attack strength. Moreover, to preserve the effectiveness of the adversarial example while being played over the air, we first mimic the sound distortions during playback and recording by estimating room impulse response (RIR), *r*, which characterizes the acoustic propagation (e.g., reverberations) in a room environment. The details of how to estimate RIR (i.e., *r*) based on the room setting are provided. Then, we could iteratively derive the targeted adversarial example through the following objective function:

$$Argmax\big(P(x' * r)\big) = t, \tag{6.3}$$

where t is the targeted speaker label, $*$ denotes the convolution operation, and $x' * r$ is the estimated adversarial example recorded by the microphone. It is important to note that the estimated RIR represents a certain mapping from the played sound to the recorded sound as per specific location of the loudspeaker and microphone in the room. To make the generated adversarial examples robust in various environmental settings, we estimate multiple RIRs **r** in various environments. To make the adversarial perturbation survive all these environments, we randomly select one RIR in **r** for each training step when updating the perturbation based on each training utterance. In addition, as directly solving the non-linear constrained non-convex problem is difficult, we iteratively solve the following optimization problem [9]:

$$minimize\ max(max\{P(x' * r)i: i \neq t\} - P(x' * r)_t, -\kappa), \tag{6.4}$$

where $\{P(x' * r)i: i \neq t\}$ represents the output probabilities of all speakers except the targeted speaker, while $P(x' * r)_t$ denotes the predicted probability to the targeted speaker. κ is a configurable parameter which represents attack confidence and is set to 0

in our implementation. To generate the universal perturbation, we iteratively modify the trainable sequence, $\Delta\delta$, which is used for constructing δ, with the entire training dataset until satisfying the desired attack success rate. For each training utterance, if the predicted probability of the targeted class is larger than other classes, the update of the perturbation $\Delta\delta$ is skipped on the next sample.

3.4.2 Generator-based Fast Audio Adversarial Attack

Despite the current progress of the existing audio adversarial attacks, one of the most challenging limitations is their inherent slow generation process for adversarial perturbations. This is because: 1) the current commonly adopted underlying adversarial perturbation-generating approaches, such as PGD [24], C&W [10] and genetic algorithms [1], are built on numbers of iterations to optimize or search the perturbations. Although this iterative mechanism can bring high attack performance, the corresponding required generation time is prohibitively long, such as seconds or even hours for producing one well-crafted perturbation. 2) Reducing the number of iterations to make generation time satisfy the real-time requirement is an alternative solution; however, as shown in our experiments that will be reported later, when the iteration-based attack method is performed in a restricted time budget, the corresponding attack performance is severely degraded. 3) On the other hand, the existing one-step perturbation generation methods, e.g., FGSM [15], though enjoy the advantage on fast generation, suffer from the poor attack performance limitation – they typically have much lower attack success rates than their iteration-based counterparts.

Overall Architecture. We propose fast adversarial perturbation generator (FAPG), a fast audio adversarial perturbation generator, to launch the audio-domain adversarial attack in a rapid, high-performance and low-memory-cost way. Figure 4. illustrates the overall architecture of FAPG, which contains a generative model ($G(\cdot)$), e.g., Wave-U-Net [31], and multiple class-wise embedding feature maps. During the training phase, both the generative model and embedding feature maps are jointly trained on the training dataset. After proper training, given a benign audio input and a target class label y_t that the adversary plans to mislead the DNN classifier ($F(\cdot)$) to, the corresponding audio adversarial perturbation can be quickly generated via performing inference of the benign input over the well-trained generative model, where the embedding feature map for the target class y_t is concatenated to one intermediate feature map of $G(\cdot)$. Next, we describe the details of the used generative model and the set of embedding feature maps as follows.

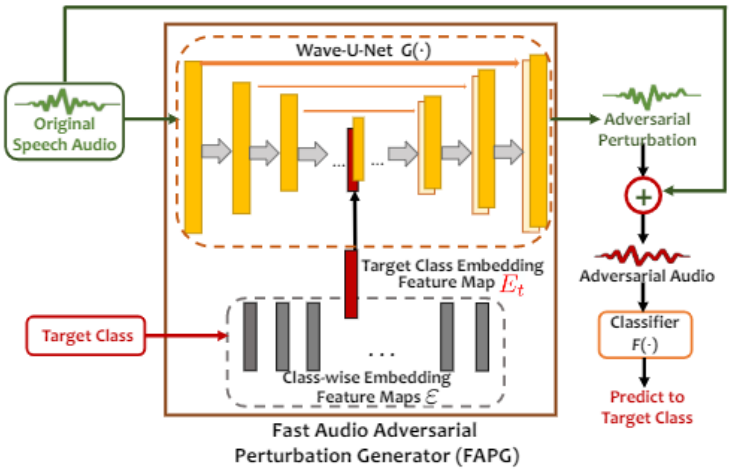


Figure 4: Overall architecture of the proposed FAPG.

Audio-specific Generative Model. Generative model is the core component of FAPG. Although various types of generative models have been widely used in image-domain applications, they are not well-suited for the use in FAPG due to the inherent difference between image and audio signals (e.g., sequence order and varying length). To address these challenges, we adopt Wave-U-Net [31], which was originally used for audio source separation, as the underlying generative model of FAPG. Wave-U-Net is a special type ofCNN containing 1-D convolutional, decimal down-sampling blocks and linear interpolation up-sampling blocks. Such inherent encoder-decoder structure makes Wave-U-Net exhibit strong distribution modeling capability. Meanwhile, its unique design of first-layer 1-D convolution and up/down sampling blocks also enables Wave-U-Net can naturally capture the temporal information from 1-D varying-length data.

Class-wise Embedding Feature Maps. The purpose of using k-class embedding feature maps is to ensure that a single generative model can be re-used for attacking against different target classes instead of class-specific design. To this end, those class-aware embedding feature maps, denoted as $\varepsilon = \{E_1, E_2, \ldots E_k\}$, are designed to be trainable, and each of them corresponds to one target class. After joint training of generative model $G(\cdot)$ and these embedding feature maps ε, the label information for class y_t is encoded in the corresponding feature map E_t. Then during the generation phase E_t is concatenated with one intermediate feature map of $G(\cdot)$ to craft the adversarial perturbation for the target class y_t. In our design, E_t has the exact same shape of the intermediate feature map to which it will be concatenated. To be specific, E_t is typically aligned with the intermediate feature map at the intersection between the encoder and decoder parts of

Wave-U-Net. This is because the feature map has the smallest size at this position, and thereby minimizing the storage cost of the corresponding E_t.

Training Procedure of FAPG. Next we describe the training procedure of FAPG, or more specifically, the joint training for $G(\cdot)$ and ε. In the forward propagation phase of the entire training procedure, for each batch of input voice data $\boldsymbol{X}$, we first randomly select one target class y_t, and fetch the corresponding embedding feature map E_t. This selected feature map is concatenated into the generative model $G(\cdot)$ to form an overall model $G_t(\cdot)$. A forward pass on $G(\cdot)$ will be performed with input $\boldsymbol{X}$. The result, denoted as $\boldsymbol{\delta_t}$, is clipped to the range of $\{-\tau, +\tau\}$ to constrain the generated perturbation $\boldsymbol{\delta_t}$ to be imperceptible, where τ is a threshold parameter. Notice that according to our experiments, τ should be set as a relatively large value initially, and gradually decreased during the training procedure. Empirically such adjusting scheme can bring better training convergence. After perturbation $\boldsymbol{\delta_t}$ is calculated from the generative model, it is imposed on the benign data to form the adversarial input, which can cause the misclassification of DNN classifier $F(\cdot)$. Then, the loss function, which is the key of the entire training procedure, is formulated as follows:

$$Loss(\boldsymbol{X}, y_t) = -y_t \cdot log\left(F\left(\boldsymbol{X} + G_t(\boldsymbol{X})\right)\right) + \beta \cdot ||G_t(\boldsymbol{X})||_2, \tag{6.5}$$

where the first and second terms are the cross-entropy loss and l_2 loss, respectively, and β is a pre-set coefficient. The existence of l_2 loss in the entire loss function is to control the attack strength and make the generated adversarial perturbation imperceptible.

Consequently, in the backward propagation phase both the generative model $G(\cdot)$ and the current selected embedding feature map E_t are updated simultaneously by minimizing the loss function. Notice that for each batch of data, E_t is randomly selected. Therefore, after rounds of iterations the generative model $G(\cdot)$ itself learns the general distribution of adversarial perturbations, and different E_t learns the encoded information for each specific target class.

3.5 Interpretation and Verification

In addition to efficient attack and defense methods, deep understanding about the interpretation of adversarial example is also very important. Meanwhile, how to verify a deep neural network is another critical task for DNN security. In this section we report our research outcomes on these two aspects.

3.5.1 Interpreting Adversarial Examples

Interpretability of imperceptible adversarial perturbations is less explored in the literature. We aim to better understand the roles of adversarial perturbations and provide visual explanations by linking them to discriminative image regions through class activation mapping (CAM) [43], a network interpretation technique. The fundamental questions are shown as follows: a) How to interpret the mechanism of adversarial perturbations at pixel and image levels? b) Rather than attack generation, how to explain the effectiveness of different adversarial attacks? c) How to explore the adversarial effects on the internal response of CNNs? And d) how does the interpretability of adversarial examples help robustness?

Seeing Effects of Adversarial Perturbations from Network Dissection. We begin by reviewing the main idea of network dissection; see more details in [4]. Interpretability measured by network dissection refers to the alignment between individual hidden unit and a set of semantic concepts provided by the broadly and densely labeled dataset *Broden*. Different from other datasets, examples in Broden contain pixel-level concept annotation, ranging from low-level concepts such as *color* and *texture* to higher-level concepts such as *material*, *part*, *object* and *scene*. Network dissection builds a correspondence between a hidden unit's activation and its interpretability on semantic concepts. More formally, the interpretability of unit (IoU) k w.r.t. the concept c is defined by [4]

$$IoU(k,c) = \frac{\sum_{x\in D}|M_k(x) \cap L_c(x)|}{\sum_{x\in D}|M_k(x) \cup L_c(x)|}, \tag{7.1}$$

where D denotes Broden, and $|\cdot|$ is the cardinality of a set. In equation 7.1, $\mathbf{M}_k(\boldsymbol{x})$ is a binary segmentation of the activation map of unit k, which gives the representative region of $\boldsymbol{x}$ at k. Here the activation is scaled up to the input resolution using bilinear interpolation, denoted by $\boldsymbol{S}_k(\boldsymbol{x})$, and then truncated using the top 5% quantile (dataset-level) threshold T_k. That is, $\mathbf{M}_k(\boldsymbol{x}) = \boldsymbol{S}_k(\boldsymbol{x}) \geq T_k$, namely, the (i,j)th element of $\mathbf{M}_k(\boldsymbol{x})$ is 1 if the (i,j)th element of $\boldsymbol{S}_k(\boldsymbol{x})$ is greater than or equal to T_k, and 0 otherwise. In equation 7.1, $\boldsymbol{L}_c(\boldsymbol{x})$ is the input-resolution annotation mask, provided by Broden, for the concept c w.r.t. $\mathbf{x}$. Since one unit might detect multiple concepts, the interpretability of a unit is summarized as $\text{IoU}(k) = (1/|C|)\Sigma_c \text{IoU}(k,c)$, where $|C|$ denotes the total number of concept labels.

We next investigate the effect of adversarial perturbations on the internal response of CNNs by leveraging network dissection. We produce adversarial examples D' from Broden using the PGD untargeted attack method [24]. Given adversarial examples $\{\boldsymbol{x}' \in D'\}$, we characterize the sensitivity of unit k (to adversarial perturbations) via the change of activation segmentation

$$v(k) \coloneqq E_{(x,x')}[||M_k(x) - M_k(x')||_2], \tag{7.2}$$

where $(\boldsymbol{x}, \boldsymbol{x}')$ is a pair of natural and adversarial examples, and the expectation is taken overall certain distribution of our interest, e.g., the entire dataset or data of fixed source-target labels. In equation 7.2, we adopt the activation segmentation $\mathbf{M}_k$ rather than the activation map $\boldsymbol{S}_k$ since the former highlights the representative region of an activation map without inducing the layer-wise magnitude bias.

3.5.2 Complete and Rapid Neural Network Verification

Formal verification of neural networks (NNs) is a challenging and important problem. Existing efficient complete solvers typically require the branch-and-bound (BaB) process, which splits the problem domain into sub-domains and solves each sub-domain using faster but weaker incomplete verifiers, such as Linear Programming (LP) on linearly relaxed sub-domains. The neural network verification problem can be cast into the following decision problem:

Given a neural network $f(\cdot)$, an input domain C, and a property P. $\forall x \in C$, does $f(x)$ satisfy P?

We aim to use fast and typically weak incomplete verifiers for complete verification. Specifically, we focus on a class of incomplete verifiers using efficient bound propagation operations, referred to as linear relaxation based perturbation analysis (LiRPA)algorithms [40]. Representative algorithms in this class include convex outer adversarial polytope [37], CROWN [42] and DeepPoly [30] LiRPA algorithms exhibit high parallelism as the bound propagation process is similar to forward or backward propagation of NNs, which can fully exploit machine learning accelerators (e.g., GPUs and TPUs).

We will first introduce our proposed efficient optimization of LiRPA bounds on GPUs that can allow us to achieve tight approximation on par with LP or even tighter for some cases but in a much faster manner. In Fig. 5, we provide a two-layer NN example to illustrate how our optimized LiRPA can improve the performance of BaB verification.

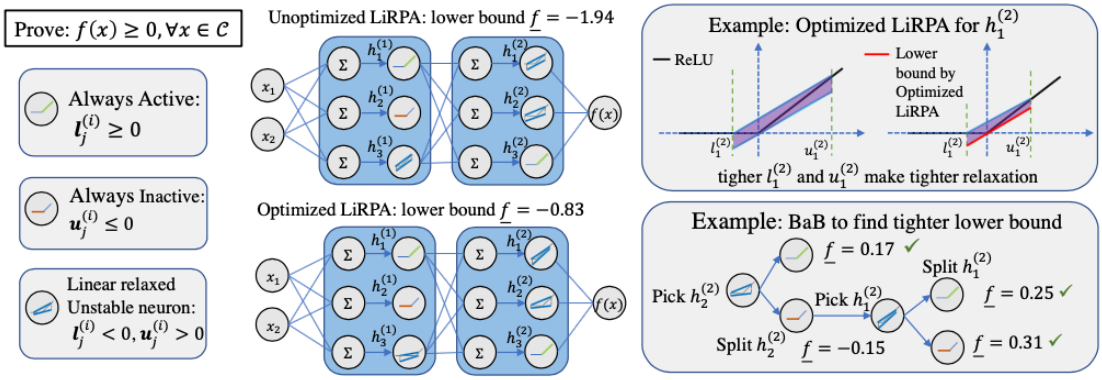


Figure 5: Illustration of our optimized LiRPA bounds and the BaB process. Given a two-layer neural network, we aim to verify output $f(x) \geq 0$. Optimized LiRPA chooses optimized slopes for ReLU lower bounds, allowing tightening the intermediate layer bounds $\boldsymbol{l}_j^{(i)}$ and $\boldsymbol{u}_j^{(i)}$ and also the output layer lower bound $\underline{f}$. BaB splits two unstable neurons $h_2^{(2)}$ and $h_1^{(2)}$ to improve $\underline{f}$ and verify all sub-domains ($\underline{f} \geq 0$ for all cases.

Our LiRPA based complete verification framework is presented in Alg. 1. The algorithm takes a target NN function f and a domain C as inputs. We run optimized LiRPA to get initial bounds $(\underline{f}, \overline{f})$ for $x \in C$ (Line 2). Then we utilize the power of GPUs to split in parallel and maintain a global set P storing all the sub-domains which cannot be verified with optimized LiRPA (Line 5-10).

Specifically, *batch_pick_out* improves BaBSR [7] in a parallel manner to select n sub-domains in P and determine the corresponding ReLU neuron to split for each of them. If the length of P is less than n, then we reduce n to the length of P *batch_split* split each selected C_i to two sub-domains C_i^l and C_i^u by forcing the selected unstable ReLU neuron to be positive and negative, respectively. *optimize_LiRPA* runs optimized LiRPA in parallel as a batch and returns the lower and upper bounds for n selected sub-domains simultaneously. *Domain_Filter* filters out verified sub-domains (proved with $(\underline{f}_{C_i} \geq 0)$) and we insert the remaining ones to P. The loop breaks if the property is proved ($\underline{f} \geq 0$) or a counter example is found in any sub-domain ($\overline{f} < 0$).

To avoid excessive splits, we set the maximum length of the sub-domains to η (Line 12). Once the length of P reaches this threshold, *compute_bound_LP* will be called. It solves these η sub-domains by LP (one by one in a loop, or in parallel if using multiple CPUs is allowed) with optimized LiRPA computed intermediate layer bounds. If a subdomain $C_i \in P$ (which previously cannot be verified by LiRPA) is proved or detected to be infeasible by LP, as an effective heuristic, we will backtrack and prioritize to check

its parent node with LP. If the parent sub-domain is also proved or infeasible, we can prune all its child nodes to greatly reduce the size of the search tree.

Algorithm 1 Parallel BaB with optimized LiRPA bounding (we highlight the differences between our algorithm and regular BaB [7] in blue. COMMENTs are in red.

Inputs: $f, \mathcal{C}, n$ (batch size), η (threshold to switch to LP)
Outputs: $\underline{f}, \overline{f}$
$(\underline{f}, \overline{f}) \leftarrow$ `optimized_LiRPA`$(f, [\mathcal{C}])$
$\mathbb{P} \leftarrow [(\underline{f}, \overline{f}, \mathcal{C})]$ {$\mathbb{P}$ is the set of all unverified sub-domains}
while $\underline{f} < 0$ **and** $\overline{f} \geq 0$ **do**
  $(\mathcal{C}_1, \ldots, \mathcal{C}_n) \leftarrow$ `batch_pick_out`$(\mathbb{P}, n)$ {Pick sub-domains to split and removed them from $\mathbb{P}$}
  $[\mathcal{C}_1^l, \mathcal{C}_1^u, \ldots, \mathcal{C}_n^l, \mathcal{C}_n^u] \leftarrow$ `batch_split`$(\mathcal{C}_1, \ldots, \mathcal{C}_n)$ {Each $\mathcal{C}_i$ splits into two sub-domains $\mathcal{C}_i^l$ and $\mathcal{C}_i^u$}
  $\left[\underline{f}_{\mathcal{C}_1^l}, \overline{f}_{\mathcal{C}_1^l}, \underline{f}_{\mathcal{C}_1^u}, \overline{f}_{\mathcal{C}_1^u}, \ldots, \underline{f}_{\mathcal{C}_n^l}, \overline{f}_{\mathcal{C}_n^l}, \underline{f}_{\mathcal{C}_n^u}, \overline{f}_{\mathcal{C}_n^u}\right] \leftarrow$ `optimized_LiRPA`$(f, [\mathcal{C}_1^l, \mathcal{C}_1^u, \ldots, \mathcal{C}_n^l, \mathcal{C}_n^u])$ {Compute lower and upper bounds using LiRPA for each sub-domain on GPUs in a batch}
  $\mathbb{P} \leftarrow \mathbb{P} \bigcup$ `Domain_Filter` $\left([\underline{f}_{\mathcal{C}_1^l}, \overline{f}_{\mathcal{C}_1^l}, \mathcal{C}_1^l], [\underline{f}_{\mathcal{C}_1^u}, \overline{f}_{\mathcal{C}_1^u}, \mathcal{C}_1^u], \ldots, [\underline{f}_{\mathcal{C}_n^l}, \overline{f}_{\mathcal{C}_n^1}, \mathcal{C}_n^l], [\underline{f}_{\mathcal{C}_n^u}, \overline{f}_{\mathcal{C}_n^u}, \mathcal{C}_n^u]\right)$ {Filter out verified sub-domains, insert the left domains back to $\mathbb{P}$}
  $\underline{f} \leftarrow \min\{\underline{f}_{\mathcal{C}_i} \mid (\underline{f}_{\mathcal{C}_i}, \mathcal{C}_i) \in \mathbb{P}\}, i = 1, \ldots, n$ {To ease notation, $\mathcal{C}_i$ here indicates both $\mathcal{C}_i^u$ and $\mathcal{C}_i^l$}
  $\overline{f} \leftarrow \min\{\overline{f}_{\mathcal{C}_i} \mid (\overline{f}_{\mathcal{C}_i}, \mathcal{C}_i) \in \mathbb{P}\}, i = 1, \ldots, n$
  if `length`$(\mathbb{P}) > \eta$ {Fall back to LP for completeness} **then**
    $\left[\underline{f}_{\mathcal{C}_1^l}, \overline{f}_{\mathcal{C}_1^l}, \underline{f}_{\mathcal{C}_1^u}, \overline{f}_{\mathcal{C}_1^u}, \ldots, \underline{f}_{\mathcal{C}_n^l}, \overline{f}_{\mathcal{C}_n^l}, \underline{f}_{\mathcal{C}_n^u}, \overline{f}_{\mathcal{C}_n^u},\right] \leftarrow$ `compute_bound_LP`$(f, [\mathcal{C}_1^l, \mathcal{C}_1^u, \ldots, \mathcal{C}_n^l, \mathcal{C}_n^u])$
    $\mathbb{P} \leftarrow \mathbb{P} \bigcup$ `Domain_Filter` $\left([\underline{f}_{\mathcal{C}_1^l}, \overline{f}_{\mathcal{C}_1^l}, \mathcal{C}_1^l], [\underline{f}_{\mathcal{C}_1^u}, \overline{f}_{\mathcal{C}_1^u}, \mathcal{C}_1^u], \ldots, [\underline{f}_{\mathcal{C}_n^l}, \overline{f}_{\mathcal{C}_n^1}, \mathcal{C}_n^l], [\underline{f}_{\mathcal{C}_n^u}, \overline{f}_{\mathcal{C}_n^u}, \mathcal{C}_n^u]\right)$
  end if
end while

Section 4
Results and Discussion

4.1 High-performance Real-time Attack

4.1.1 Structured Adversarial Attack

Fig. 4.1.1 compares adversarial examples generated by our proposed StrAttack and C&W attack on each dataset. We observe that the perturbation of the C&W attack has poor group sparsity, i.e., many non-zeros groups with small magnitudes. However, the ASR of the C&W attack is quite sensitive to these small perturbations. As applying a threshold to have the same l_0 norm as our attack, we find that only 6.7% of adversarial examples generated from C&W attack remain valid. By contrast, StrAttack can highlight the most important group structures (local regions) of adversarial perturbations without attacking other pixels. For example, StrAttack misclassifies a natural image (4 in MNIST) as an incorrect label 3. That is because the pixels that appears in the structure of 3 are more significantly perturbed by our attack (see the top right plots of Fig. 6). Furthermore, the 'goose-sorrel' example shows that misclassification occurs when we just perturb a small number of non-sparse group regions on goose's head, which is more consistent with human perception.

By quantitative analysis, we report l_p norms and ASR in Table 1 for $p \in \{0, 1, 2, \infty\}$. We show that StrAttack perturbs much fewer pixels (smaller l_0 norm), but it is comparable to or even better than other attacks in terms of l_1, l_2, and l_∞ norms. Specifically, the FGM attack yields the worst performance in both ASR and l_p distortion. On MNIST and CIFAR-10, StrAttack outperforms other attacks in l_0, l_1 and l_∞ distortion. On ImageNet, StrAttack outperforms C&W attack in l_0 and l_1 distortion. Since the C&W attacking loss directly penalizes the l_2 norm, it often causes smaller l_2 distortion than StrAttack. We also observe that the overlapping case leads to the adversarial perturbation of less sparsity (in terms of l_0 norm) compared to the non-overlapping case. This is not surprising, since the sparsity of the overlapping region is controlled by at least two groups. However, compared to C&W attack, the use of overlapping groups in StrAttack still yields sparser perturbations. Unless specified, otherwise, we focus on the case of non-overlapping groups to generate the most sparse adversarial perturbations. We highlight that although a so-called one-pixel attack [32] also yields very small l_0 norm, it is at the cost of

very large l_∞ distortion. Unlike one-pixel attack, StrAttack achieves the sparsity *without losing* the performance of l_∞, l_1 and l_2 distortion

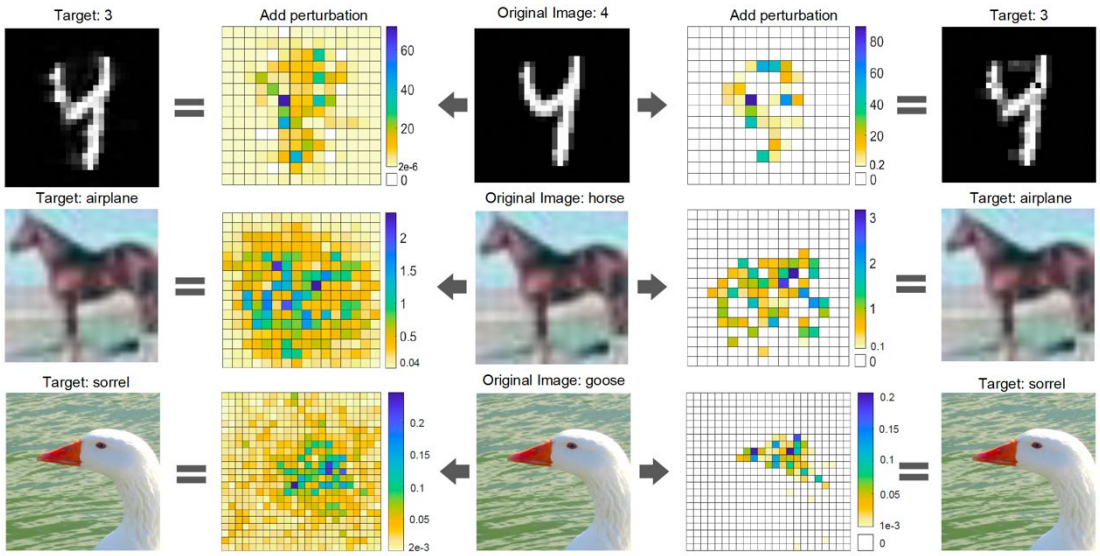


Figure 6: C&W attack vs StrAttack. Here each grid cell represents a 2 × 2, 2 × 2, and 13 × 13 small region in MNIST, CIFAR-10 and ImageNet, respectively. The group sparsity of perturbation isrepresented by heatmap. The colors on heatmap represent average absolute value of distortion scaleto [0, 255]. The left two columns correspond to results of using C&W attack. The right two columns show results of StrAttack.

Table 1: Adversarial attack success rate (ASR) and l_p distortion values for various attacks.

Data Set	Attack Method	Best Case*					Average Case*					Worst Case*				
		ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞	ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞	ASR	ℓ_0	ℓ_1	ℓ_2	ℓ_∞
MNIST	FGM	99.3	456.5	28.2	2.32	0.57	35.8	466	39.4	3.17	0.717	0	N.A.**	N.A.	N.A.	N.A.
	IFGSM	100	549.5	18.3	1.57	0.4	100	588	30.9	2.41	0.566	99.8	640.4	50.98	3.742	0.784
	C&W	100	479.8	13.3	1.35	0.397	100	493.4	21.3	**1.9**	0.528	99.7	524.3	29.9	**2.45**	0.664
	StrAttack	100	**73.2**	10.9	1.51	**0.384**	100	**119.4**	**18.05**	2.16	**0.47**	100	**182.0**	26.9	2.81	**0.5**
CIFAR-10	FGM	98.5	3049	12.9	0.389	0.046	44.1	3048	34.2	0.989	0.113	0.2	3071	61.3	1.76	0.194
	IFGSM	100	3051	6.22	0.182	0.02	100	3051	13.7	0.391	0.0433	100	3060	22.9	0.655	0.075
	C&W	100	2954	6.03	0.178	**0.019**	100	2956	12.1	0.347	**0.0364**	99.9	3070	16.8	**0.481**	**0.0536**
	StrAttack	100	**264**	**3.33**	0.204	0.031	100	**487**	**7.13**	0.353	0.050	100	**772**	**12.5**	0.563	0.075
ImageNet	FGM	12	264917	152	0.477	**0.0157**	2	263585	51.3	**0.18**	**0.00614**	0	N.A.	N.A.	N.A.	N.A.
	IFGSM	100	267079	299.32	0.9086	0.02964	100	267293	723	2.2	0.0792	98	267581	1378	4.22	0.158
	C&W	100	267916	127	**0.471**	0.016	100	263140	198	0.679	0.03	100	265212	268	**0.852**	**0.041**
	StrAttack	100	**14462**	**55.2**	0.719	0.058	100	**52328**	**152**	1.06	0.075	100	**80722**	**197**	1.35	0.122

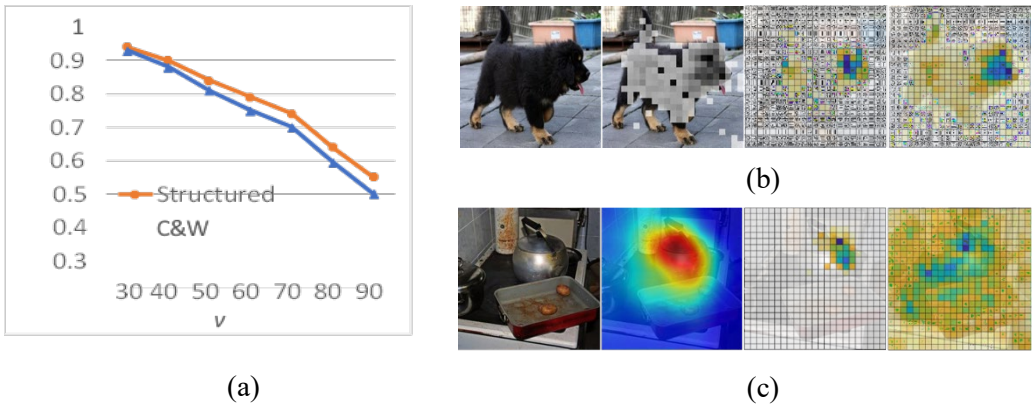


Figure 7: Interpretability comparison of StrAttack and C&W attack. (a) ASM-based IS vs ν, given from the 30th percentile to the 90th percentile of ASM scores. (b) Overlay ASM and $\mathbf{B}_{\mathrm{ASM}}{}^{\circ}\,\boldsymbol{\delta}$ on top of image with the true label 'Tibetan Mastiff' and the target label 'streetcar'. From left to right: original image, ASM (darker color represents larger value of ASM score), $\mathbf{B}_{\mathrm{ASM}}{}^{\circ}\,\boldsymbol{\delta}$ under StrAttack, and $\mathbf{B}_{\mathrm{ASM}}{}^{\circ}\,\boldsymbol{\delta}$ under C&W attack. Here ν in $\mathbf{B}_{\mathrm{ASM}}$ is set by the 90th percentile of ASM scores. (c) From left to right: original image with true label 'stove', CAM of 'stove', and perturbations with target label 'water ouzel' under StrAttack and C&W.

4.1.2 Real-time Adversarial Attack

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 as the performance metric. 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 Table 4.1.2, our attack achieves competitive results compared with momentum 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. The speedup is more than 500×. The ability to generate a large number of adversarial images quickly makes our attack method practical in real-time applications.

Table 2: Comparison of adversarial examples on ResNet-18(CIFAR-10).

	ASR	**Acc.**	**L2**	**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 3: Storage and ASR comparison of adversarial examples generated by CAG and GAP. CAG and GAP 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 andimpractical training time, we cannot report the attack results on GAP with 1000T.)

	Storage	**White-box**				**Black-box**			
Attacks / Classifiers		RN-18	VGG-11	DN-121	**Average**	RN-34	VGG-13	DN-169	**Average**
GAP Unet (5T)	150MB	97.98%	**98.45%**	**97.85%**	**98.09%**	82.97%	85.69%	88.31%	85.66%
GAP ResNet (5T)	150 MB	91.02%	94.25%	90.58%	91.95%	76.40%	86.27%	78.33%	80.33%
GAP (1000T)	30GB	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%

CAG always has ASR greater than 95% on seen classifiers for white-box attack. 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 transferability requirement, we propose to dropout part of the perturbation before adding it on the benign image during training phase. As a result, CAG generalizes better and is overfits less to a particular classifier. Hence, the transferability of the adversarial examplesto new classifiers will increase. We train 4 CAG models base on 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. The Table 3 reveals that even though 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.

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 bad as training with only one classifier. Hence, unlike the best configuration in CIFAR-10 where $p = 0.3$, we reduce perturbation dropout to $p = 0.1$.

To explicitly demonstrate the performance of our proposed method, we compare our results with Generative Adversarial perturbation (GAP) [28]. To have a fair comparison, we implement GAP with two architectures and keep the configuration same with our method. The first GAP is applied Unet which has the identical architecture with our model, so we named it GAP Unet. The second architecture is the one used in GAP's original paper. We call 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.* The comparison result is shown in Table 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 attacked target on all 1,000 classes, we also generate adversarial images crossing all classes. In the table, 5T means ASRs are evaluatedon 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 $\times$ 1000 $\approx$ 30GB for storage to attack all classes. Moreover, as shown in Table 3, for seen classifiers, ASR is above 90% for all approaches. While targeting on 5 selected classes, adversarial images generated by GAP Unet and CAG have comparable performance. On the other hand, analyzing the result of unseen classifiers, we can see that CAG outperformsGAPs. 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.

4.2 Efficient DNN Defense Mechanism

4.2.1 Switching Model

We use FGSM [15] and C&W [8] attacks to generate adversarial examples targeting the regular model, the stochastic activation pruning (SAP) [13] model (state-of-the-art),and the proposed switching model. As shown in Table 4, on MNIST dataset, for example,the fooling ratio by C&W attack is reduced from 100% (the regular model) to 21%, while SAP can only reduce fooling ratio to 32.1%. As shown in Table 5, on CIFAR-10 dataset,the proposed switching model reduces the fooling ratio to 22.2%, while the SAP only reduces the fooling ratio to 93.3%.

Table 4: Fooling ratio (FR) and distortion of FGSM and C&W attacks on MNIST dataset.

Attack	Regular		SAP		Switching	
	FR	L2	FR	L2	FR	L2
FGSM $\epsilon = 0.1$	3.9%	2.73	3.7%	2.73	1.6%	2.73
FGSM $\epsilon = 0.25$	34.0%	6.84	32.8%	6.84	20.3%	6.84
CW	100.0%	2.28	32.1%	2.28	21.0%	2.37

Table 5: Fooling ratio (FR) and distortion of FGSM and C&W attacks on CIFAR-10.

Attack	Regular		SAP		Switching	
	FR	L2	FR	L2	FR	L2
FGSM $\epsilon = 0.01$	25.0%	0.55	24.8%	0.55	8.1%	0.55
CW	100.0%	0.54	93.3%	0.52	22.2%	0.69

We visualize some gradients distribution during C& W attack on an SAP model and a proposed switching model in Fig. 8. The gradient distribution on the switching model is wider and has a clear multimodal pattern, while the gradient on the SAP model has a unimodal bell shape. The stronger defense effects of the switching model are due to the wider gradient distribution making the generation of adversarial attacks harder for the adversaries.

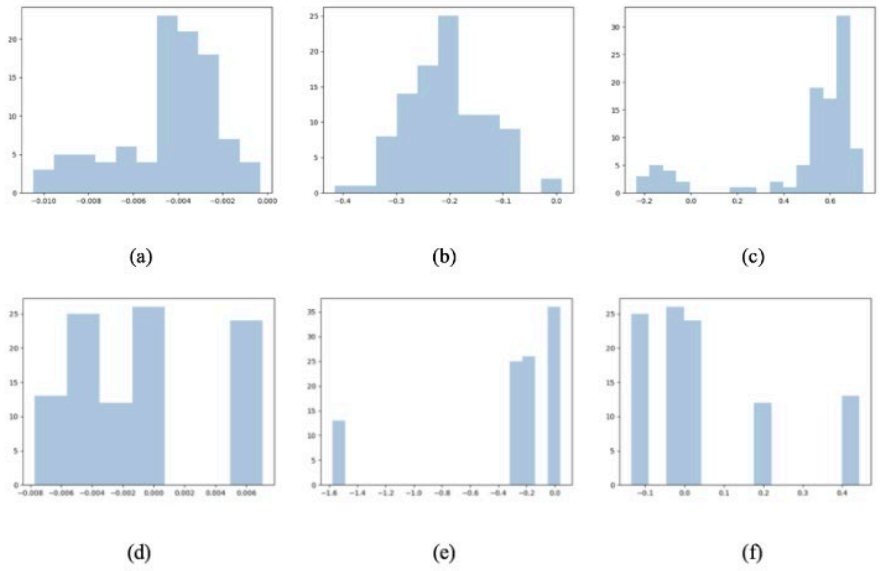


Figure 8: We use three images (a-c): Gradient distributions of C&W attack on an SAP model. (d-f): Corresponding gradient distributions on a switching model. Distributions in the same column belongto the same input dimension. Each distribution is sampled for 100 times.

4.2.2 A Multi-source Multi-cost Defense

We compare our proposed AdvMS with its building blocks: adversarial training [24] and random model switching [29,44] The purpose of this experiment is to show that the proposed complex defense outperforms each of its defending component performed alone. For a fair comparison, all defenses are implemented on the same base model.

Base model architectures for MNIST and CIFAR-10 are summarized in Table 6. The defending performance is measured by attack success rate (ASR) where lower ASRs indicate stronger defense. We perform three attacks: FGSM, PGD, and CW-PGD which are among the strongest baselines in the literature. All attacks are conducted in multiple strengths (controlled by the l_∞ norm ϵ of the perturbation) and in both white-box setting,

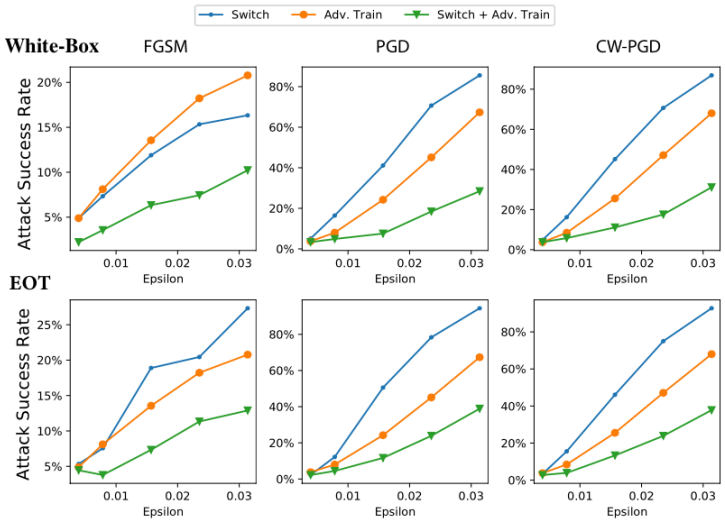


Figure 9: Attack success rate on CIFAR-10 dataset using *above:*(White-Box) FGSM, PGD and CW-PGD attacks, *below:* FGSM + EOT, PGD + E OT, and CW-PGD + EOT attacks.

where we assume attackers know all information of the target model, and EOT (Expectation Over Transformation) [2, 35] which is a counter-measure of attacks against randomized defense schemes by using the expectation of stochastic gradients. Fig.9 and Fig.10 summarize the defending performance on CIFAR-10 and MNIST datasets, respectively.

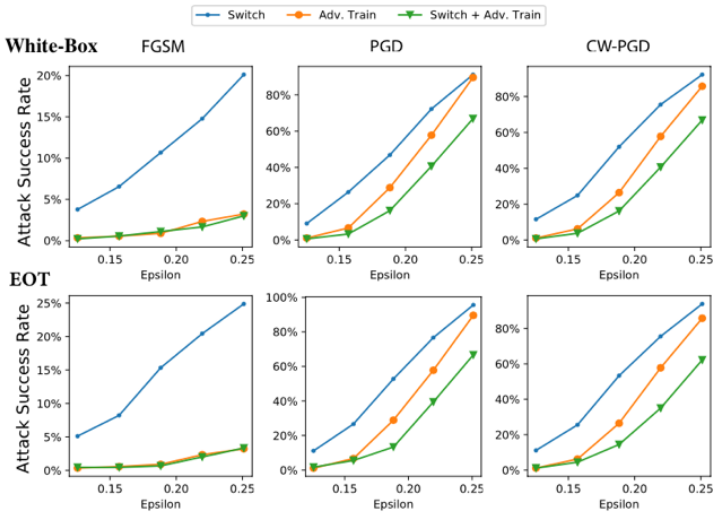


Figure 10: Attack success rate on MNIST dataset using *above:* (White-Box) FGSM, PGD, and CW-PGD attacks, *below:* FGSM + EOT, PGD + EOT, and CW-PGD + EOT attacks.

As we can observe, the proposed AdvMS achieves superior defense performance under all attack settings and performs even better under stronger attacks such as PGD and CW-PGD, where model switching and adversarial training are less effective given the same level of attack strength. We also perform a quantitative analysis by controlling each of the two strength factors of AdvMS: ϵ, which is the l_∞ distortion of adversarial examples used in training, and$M$ which is the number of models for switching. This experiment illustrates how enforcing one defending component contributes to boost the performance on the top of the other.

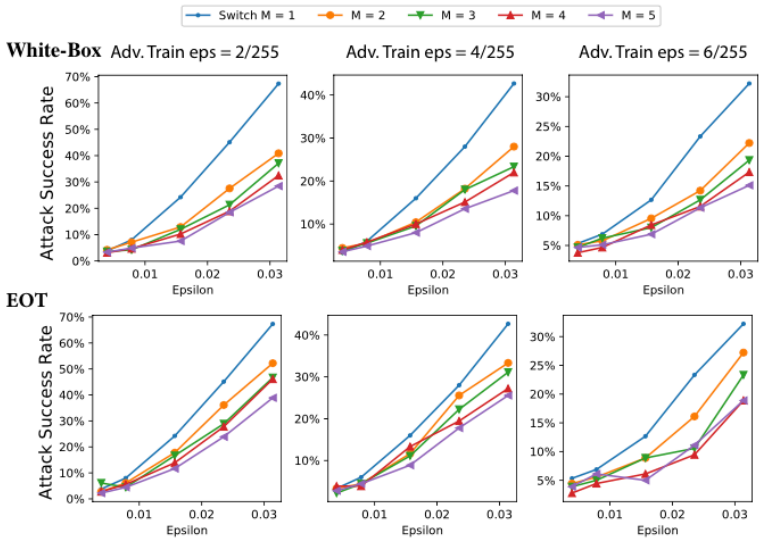


Figure 11: Attack success rate on CIFAR-10 dataset using different (white-box or EOT) PGD attack settings with fixed number of models M in AdvMS.

We further perform quantitative analysis on changing the number of sub-models in AdvMS. Fig.12 exhibits our findings that AdvMS with more adversarially trained models shows more resistance against adversarial attacks. While the gain in the defense rate when changing the deterministic model ($M = 1$) to a stochastic model with $M = 2$ is sufficiently large, the marginal gain in defense rate also tends to decease when M is further increased.

In Fig.11, we fix the number of models in AdvMS and compare the robustness performance obtained by changing the ϵ_{train} for adversarially trained models. In all cases we can observe the trend that, AdvMS trained with larger ϵ_{train} value shows stronger resiliency to adversarial attacks in both attack settings. However, it is also observed that although the gap between no adversarial training and adversarial training with $\epsilon_{train} = 2/255$ is large, the benefits gained by increasing ϵ_{train} tend to saturate with large ϵ_{train}.

Table 6: Base Model Architectures for MNIST and CIFAR-10.

	MNIST	CIFAR-10
Conv layer	32 × (3,3)	64 × (3,3)
Conv layer	32 × (3,3)	64 × (3,3)
Pooling layer	pool size (2,2)	pool size (2,2)
Conv layer	64 × (3,3)	128 × (3,3)
Conv layer	64 × (3,3)	128 × (3,3)
Pooling layer	pool size (2,2)	pool size (2,2)
Fully connected 1	200 units	256 units
Fully connected 2	200 units	256 units
Output layer	10 units	10 units

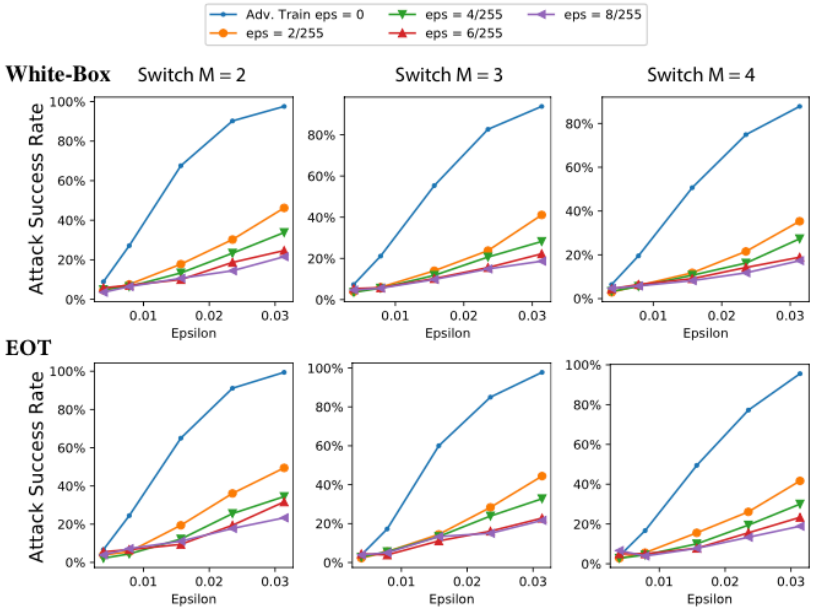


Figure 12: CIFAR-10 with fixed ϵ_{train} PGD Attack. Attack success rate on CIFAR-10 dataset using different (white-box or EOT) PGD attack settings with fixed adversarial training strength ϵ_{train} in AdvMS.

4.3 Attack and Defense for Graph Neural Networks

4.3.1 Topology Attack and Defense for Graph Neural Network

We compare our four attack methods (CE-PGD, CW-PGD, CE-min-max, CW-min-max) with DICE ("delete edges internally, connect externally") [36], Meta-Self attack [46] and greedy attack, a variant of Meta-Self attack without weight re-training for GCN. The greedy attack is considered as a fair comparison with our CE-PGD and CW-PGD attacks, which are generated on a fixed GCN without weight re-training. In min-max attacks (CE-min- max and CW-min-max), we show misclassification rates against both natural and retrained models and compare them with the state-of-the-art Meta-Self attack. For a fair comparison,we use the same performance evaluation criterion in Meta-Self, testing nodes' predicted labels (not their ground-truth label) by an independent pre-trained model that can be used during the attack.

In Table 7, we present the misclassification rate of different attack methods against both natural and retrained model. Here we recall that the retrained model arises dueto the scenario of attacking an interactive GCN with re-trainable weights. For comparison,we also show the misclassification rate of a natural model with the true topology (denotedby "clean"). As we can see, to attack the natural model, our proposed attacks achieve better misclassification rate than the existing methods. We also observe that compared to min-max attacks (CE-min-max and CW-min-max), CE-PGD and CW-PGD yield better attacking performance since it is easier to attack a pre-defined GCN. To attack the model that allows retraining, we set 20 steps of inner maximization per iteration. The results show that our proposed min-max attack achieves very competitive performance compared to Meta-Self attack. Note that evaluating the attack performance on the retrained model is not quite fair since the retrained weights could be sub-optimal and induce degradation in classification.

In Figure 13, we present convergence of our robust training. As we can see, the loss drops reasonably and the 1000 iterations are necessary for robust training rather than normal training process which only need 200 iterations. We also observe that our robust training algorithm does not harm the test accuracy when ϵ=5%, but successfully improves the robustness as the attack success rate drops from 28.0% to 22.0% in Cora dataset as shown in Table 8, After showing the effectiveness of our algorithm, we explore deeper in adversarial training on GCN. We aim to show how large ϵ we can use in robust training.

Table 7: Misclassification rates (%) under 5% perturbed edge.

		Cora	Citeseer
fixed natural model	clean	18.2 ± 0.1	28.9 ± 0.3
	DICE	18.9 ± 0.2	29.8 ± 0.4
	Greedy	25.2 ± 0.2	34.6 ± 0.3
	Meta-Self	22.7 ± 0.3	31.2 ± 0.5
	CE-PGD	$\mathbf{28.0 \pm 0.1}$	36.0 ± 0.2
	CW-PGD	27.8 ± 0.4	$\mathbf{37.1 \pm 0.5}$
	CE-min-max	26.4 ± 0.1	34.1 ± 0.3
	CW-min-max	26.0 ± 0.3	34.7 ± 0.6
retrained model	Meta-Self	29.6 ± 0.4	$\mathbf{39.7 \pm 0.3}$
	CE-min-max	$\mathbf{30.8 \pm 0.2}$	37.5 ± 0.3
	CW-min-max	30.5 ± 0.5	39.6 ± 0.4

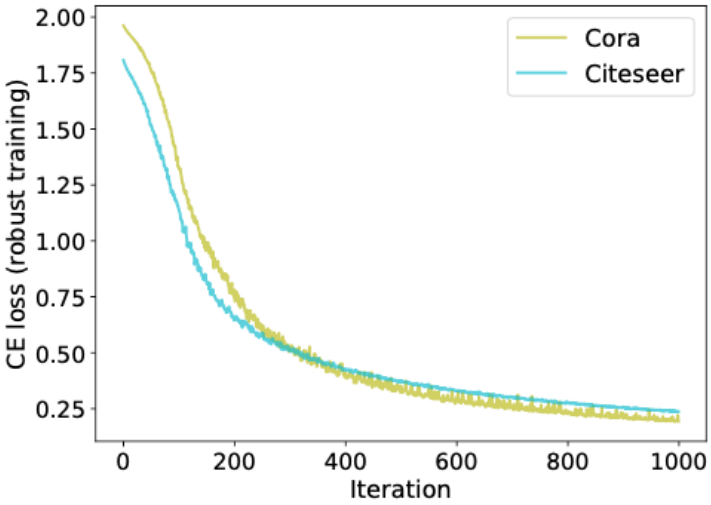


Figure 13: Robust training loss on Cora and Citeseer datasets.

So, we set ϵ from 5% to 20% and apply CE-PGD attack following the same ϵ setting. The results are presented in Table 9. Note that when $\epsilon = 0$, the first row shows misclassification rates of test nodes on natural graph as the baseline for *lowest* misclassification rate we can obtain; the first column shows the CE-PGD attack

misclassification rates of natural model as the baseline for *highest* misclassification rate we can obtain. We can conclude that when a robust model trained under an ϵ constraint, the model will gain robustness under this ϵ distinctly. Considering its importance to keep the original graph test performance, we suggest generating robust model under $\epsilon = 0.1$. Moreover, please refer to Figure 14 that a) our robust trained model can provide universal defense to CE-PGD, CW-PGD and Greedyattacks; b) when increasing ϵ, the difference between both test accuracy and CE-PGD attack accuracy increases substantially, which also implies the robust model under larger ϵ is harder to obtain.

Table 8: Misclassification rates (%) of robust training (smaller is better for defense task) with at most 5% of edge perturbations. **A** means the natural graph, $\mathbf{A}'$ means the generated adversarial graph under $\epsilon = 5\%$. $\mathbf{X}/M$ means the misclassification rate of using model M on graph $\mathbf{X}$.

	Cora	Citeseer
A/natural model	18.2 ± 0.1	28.9 ± 0.1
A/robust model	18.1 ± 0.3	28.7 ± 0.4
$\mathbf{A}'$/natural model	28.0 ± 0.1	36.0 ± 0.2
$\mathbf{A}'$/robust model	22.0 ± 0.2	32.2 ± 0.4

Table 9: Misclassification rates (%) of CE-PGD attack against robust training model versus (smaller is better) different ϵ (%) on Cora dataset. Here $\epsilon = 0$ in training means natural model and $\epsilon = 0$ in attack means unperturbed topology.

		ϵ in robust training (in %)				
		0	5	10	15	20
ϵ in attack (in %)	0	18.1	18.2	19.0	20.2	21.3
	5	27.9	22.0	23.9	24.8	26.5
	10	32.7	32.1	26.4	27.7	31.0
	15	36.7	36.2	33.4	29.7	32.9
	20	40.2	40.1	36.3	36.3	33.5

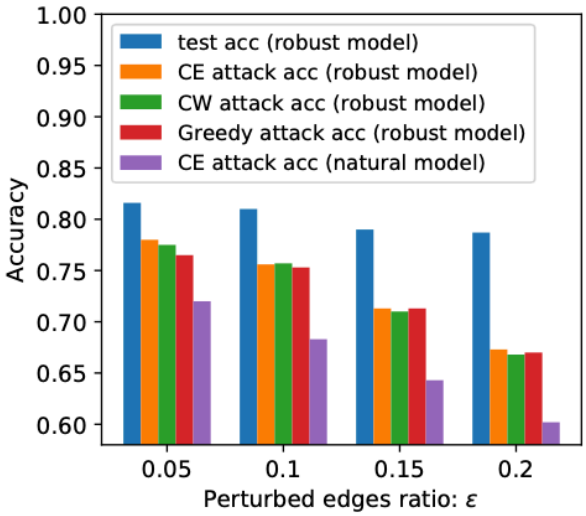


Figure 14: Test accuracy of robust model (no attack), CE-PGD attack against robust model, CW-PGD attack against robust model, Greedy attack against robust model and CE-PGD attack against natural model for different ϵ used in robust training and test on Cora dataset.

4.3.2 Universal Attack to Graph Neural Networks

Baseline Methods. Because graph universal attacks were barely studied, we design four relevant baselines for GUA. The first two are basic while the last two are more sophisticated.

- Global Random: Each node has a probability *Prob* to become an anchor node. In other words, each element of the attack vector p is an independent sample of Bernoulli (*Prob*).
- Victim-Class Attack (Victim Attack): We sample a prescribed number of anchor nodes without replacement from nodes of a particular class. This baseline originates from a finding that the anchor nodes computed by GUA often belong to the same class (see more details later).
- High-Degree (HD) Global Random: We strengthen the Global Random baseline by picking random anchors uniformly from top 10% nodes with highest degrees.
- Top-Confidence (TC) Victim Attack: the anchor set is composed of nodes with the highest prediction probability from the victim class.

The evaluation metric is attack success rate (ASR). Another quantity of interest is the number of modified links (ML). For universal attacks, it is equivalent to the anchor set size.

Comparison with baselines. We compare GUA with four baseline methods, together with two non-universal attacks and one global attack. Since we cannot choose the number of anchor nodes for GUA, we obtain this value based on the results in when $\xi = 4, 4, 8$ on Cora, Citeseer and Pol.Blogs, respectively. In this case, the average ML for these datasets is respectively 7.9, 7.7, and 26.6. Therefore, we set the number of anchor nodes for all baselines, but for Global Random and Meta-Self, to be the ceiling of these values. For Global Random, *Prob* is set such that the expected number of anchor nodes is these values. For Meta-Self, the MLs are 53, 47 and 167 for Cora, Citeseer and Pol.Blogs, respectively, which are 1% of the number of original edges in corresponding datasets. From Table 10, one sees that GUA significantly outperforms other universal attack methods. Among them,Top-Confidence Victim-Class Attack is the most effective, but it is still inferior to GUA. This result suggests that GUA leverages more information (in this case, node features) than the class labels, although we have seen strong evidence that anchor nodes computed by GUA mostly belong to the same class. One also sees that GUA is inferior to FGA and Nettack if only ASR is concerned. FGA and Nettack are not universal attack methods; they find different anchors for each target node. Thus, it is possible to optimize the number of anchors (possibly different for each target) to aim at a certain ASR, or equivalently, to achieve a better ASR given a certain number of anchors. However, it is also because they are non-universal attacks, that the total number of anchors for all targets soars. For example, FGA modifies links with 1406 anchors on Cora and 1359 anchors on Citeseer in total. GUA also significantly outperforms Meta-Self, since it only attacks the graph once, instead of attacking each node individually.

Table 10: Average ASR. For a fair comparison, all universal attack methods except Global Random uses the same number of anchor nodes. FGA and Nettack are not universal attacks and we set their ML per node to be the same as the number of anchor nodes.

Attack Method	Cora	Citeseer	Pol.Blogs
GUA	84.21%	77.07%	43.06%
Global Random	22.00%	26.58%	17.58%
Victim Attack	62.64%	54.47%	32.45%
HD Global Random	26.68%	51.74%	17.28%
TC Victim Attack	79.64%	73.45%	36.09%
FGA (not univ.)	89.70%	84.82%	57.67%
Nettack (not univ.)	86.09%	77.06%	76.91%
Meta-Self (global)	16.21%	30.37%	14.25%

Effect of removing anchor nodes. Once a set of anchor nodes is identified, a natural question asks if the set contains redundancy. We perform the following experiment: we randomly remove a number of anchor nodes and recompute the ASR. Because on Cora and Citeseer, the average number of anchor nodes for $\xi = 4$ is 7.9 and 7.7 respectively, we use an anchor set of size eight to conduct the experiment. For each case, we randomly remove 1–7 nodes from the anchor set and report the corresponding average ASR. The results are shown in Figure. From the figure, one sees that the average ASR gradually decreases to zero as more and more anchor nodes are removed. This result indicates that there exist no redundancy in the anchor set. The decrease is faster when more nodes are removed, but the average ASR is still quite high even when removing half of the nodes. This finding is another evidence that supports the trade-off between anchor set size and ASR.

Speeding up training through sampling. The cost of finding the anchors is $O(n \cdot |V_L| \cdot max\_epoch)$, because in each epoch the attack vector p is kept being updated through iterating $|V_L|$ training nodes. For a given graph with fixed n, we are interested in seeing whether reducing the training set size affects the attack performance. We randomly sample a portion of the training set in each epoch and report in Table 4.3.5 the resulting ASR. One sees that the ASR barely changes by using 40% of the data to train each epoch. Further reducing the size starts to hurt, but even using 5% of the training data, the ASR drops by only 5% to 13%. Moreover, GUA is efficient compared to the state-of-art attack method Nettack [45] whose complexity is $O(n^2(E \cdot T + F))$ to attack all nodes, where E and F represent the number of edge and feature perturbations, respectively, and T is the average size of a 2-hop neighborhood. In practice, Nettack is slower to run, due to the n^2 factor.

Table 11: Average ASR through sampling the training set per epoch.

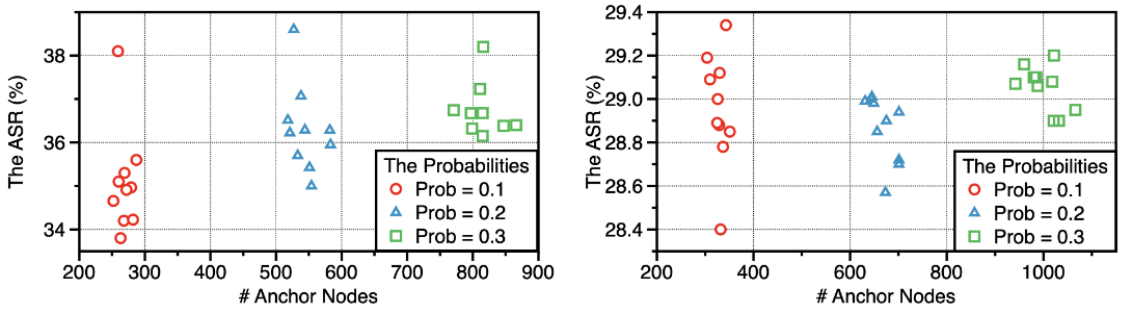
Dataset	100%	40%	20%	10%	5%
Cora	84.2%	83.7%	83.2%	82.6%	79.9%
Citeseer	77.1%	77.4%	69.7%	69.7	65.2%
Pol.Blogs	43.1%	40.1%	35.7%	37.7%	30.8%

Transferability. We have already seen that GUA is quite effective in attacking GCN. Such an attack belongs to the white-box family, because knowledge of the model to be attacked is assumed. In reality, however, the model parameters may not be known at all, not even the model form. Attacks under this scenario is called black-box. One approach to

conducting black-box attack is to use a surrogate model. In our case, if one is interested in attacking graph deep learning models other than GCN, GCN may serve as the surrogate. The important question is whether anchors found by attacking the surrogate can effectively attack other models as well. We perform an experiment with three such models: DeepWalk [27], node2vec [16], and GAT [34]. The first two compute, in an unsupervised manner, node embeddings that are used for downstream classification, whereas the last one is a graph neural network that directly performs classification. In Table 12, we list the ASR for these models. One sees that the ASRs are similarly high as that for GCN; sometimes even surpassing. Specifically, GAT is developed based on GCN through incorporating the attention mechanism, while Node2vec and DeepWalk update node embeddings by exploring the local structure via random walk. Since GUA modifies the neighborhood of the target, it is reasonable that all the other methods can also be misled efficiently. This finding concludes that the results of GUA are well transferable.

Table 12: Average ASR after applying the anchor nodes found by GUA when $\xi = 4$ on Cora and Citeseer, and $\xi = 8$ on Pol.Blogs.

Dataset	GCN	DeepWalk	node2vec	GAT
Cora	84.21%	85.80%	80.84%	85.15%
Citeseer	77.07%	81.71%	74.07%	77.02%
Pol.Blogs	43.06%	33.21%	41.62%	38.85%



(a) Cora. (b) Citeseer.

Figure 15: Performance of global random attack, repeated ten times.

4.4 Fast and Universal Adversarial Attack to Audio-domain DNN

4.4.1 Robust Universal Audio Adversarial Attack

Effectiveness of Universal Targeted Attack. To evaluate the effectiveness of our proposed universal targeted attack, we alternatively choose one of the 109 enrolled speakers as the targeted speaker and the rest 108 speakers as victims. In total, we generated 109 universal adversarial perturbations, trying to make the speaker recognition system classify the victims' utterances as the targeted speakers. As shown in Table 1, by adjusting attack strength ϵ, the noise level ranges from $-18.84dB$ to $-33.96dB$. As discussed in the previous study [10], such noise level is considered to be quasi-imperceptible to humans. For instance, $-33.96dB$ is comparatively the difference between a person talking and the ambient noise in a quiet room. For each ϵ value, the minimum, maximum, and average attack success rate among all attacks attempts targeting on 109 speakers are calculated. We can observe that when the noise level is $-18.84dB$, a high average attack success rate of 99.95% can be reached. When the noise level decreases to −33.96 dB, the average attack success rate still remains over 80%,which illustrates the effectiveness of our proposed universal targeted attack.

Table 13: Results of universal targeted attack.

Attack Strength	Noise Level	Min. Attack Success Rate	Max. Attack Success Rate	Avg. Attack Success Rate
ϵ=0.05	-18.84dB	98.47%	100%	99.95%
ϵ=0.03	-23.27dB	95.31%	99.91%	98.40%
ϵ=0.01	-33.96dB	53.32%	95.48%	83.82%

Robustness Analysis Using Room Simulator. An acoustic room simulator toolkit is used to simulate the audio propagation in a room environment. Specifically, a modeled room with a size of $5m \times 5m \times 3m$ is used, and 120 locations of the loudspeaker and the microphone are chosen randomly in the room for RIR estimation. For the estimated RIRs, 100 locations are used to build the universal, targeted and robust adversarial perturbation, and the rest 20 locations are used for testing.

Table 14: Results of robust universal targeted attack using acoustic room simulator.

	Noise Level	Min. Attack Success Rate	Max. Attack Success Rate	Avg.Attack Success Rate
Without RIR	-18.84dB	0.7%	3.52%	1.33%
With RIR	-18.84dB	74.68%	98.05%	90.19%
	-23.27dB	66.54%	96.81%	86.17%
	-33.96dB	54.48%	90.83%	78.25%

Table 14 summarizes the results of our practical universal adversarial perturbation. We can observe that the universal adversarial perturbations trained with RIRs still remain effective after the over-the-air simulation. In particular, the practical universal perturbation generated with a noise level of $-18.84dB$ can still achieve an average attack success rate of 90.19%. For comparison, we test the adversarial perturbation of the same noise level and without RIR in the simulated room environment. However, the average attack success rate decreases significantly to 1.33%. This shows that our approach can efficiently improve the robustness of the generated adversarial examples.

Speedup on Attack Time. Unlike conventional individual attacks that require to build adversarial perturbation for each individual voice input, our proposed universal attack could generate a single perturbation that makes arbitrary speaker's utterances to be identified as the adversary-desired speaker. Thus, simply playing the pre-generated universal perturbation nearby the victim speaker becomes possible for launching adversarial attacks. For showing the possibility of launching real-time attacks, we compare the attack launching time of using the conventional individual targeted attack method [10] and our proposed universal attack for a given audio signal. Particularly, the conventional targeted attack requires at least $15s$ to deploy, measured on a Tesla V100 GPU with $32GB$ memory, while our proposed universal method only takes an average of $0.015s$, which results in a 100× speedup.

4.4.2 Generator-based Fast Audio Adversarial Attack

FAPG Generator Implementation. We use model $M1$ of Wave-U-Net to construct our FAPG. Specifically, our model contains 5 down-sampling blocks and 5 up-sampling

blocks. The feature map size of the last encoding layer is also the size of each additional class-wise embedding feature map. For FAPG, a total of 10,000 training steps are conducted using Adam optimizer with the batch size of 100. The initial learning rate is set to $1e^{-4}$ and gradually decayed to $1e^{-6}$. β is set as 0.1 for all datasets. τ is initially set as 0.1 and reduces to 0.05 and 0.03 at step of 3,000 and 7,000 for command recognition and speaker recognition, and it stops reducing as 0.05 for sound classification model, which leads to an approximate noise level of −30 dB and −18 dB respectively.

Attack Speedup and Performance. To demonstrate the ability of the proposed FAPG in terms of achieving high success rate while maintaining a short attack generation time, we conduct experiments on the three aforementioned target models under different time conditions. Table 15 compares the attack performance of the proposed FAPG with commonly-used attacks, i.e., FGSM [15], PGD [24] and C&W [8] under constrained time budget scenario, which requires to generate adversarial example with no more than 0.065s (the approximate execution time for one iteration in PGD and C&W attack. For fair comparison, we constrain the perturbations generated by these attacks with an infinity norm of 0.03 for speech command classification and speaker recognition, and 0.05 for environmental sound classification, which are the same as used in FAPG implementation. As shown in Table 15, the proposed FAPG can achieve high attack success rate (SR) (over 90%) under the short time budget for all the three target models, while FGSM, PGD and C&W attack can only achieve less than 15% SR with limited attack time budget.

Table 15: Success rate (SR) of audio-dependent targeted attacks under constrained time budget (0.065s).

	FGSM	PGD	C&W	FAPG
Command Recognition	11.89%	11.96%	13.26%	**97.77%**
Speaker Recognition	1.65%	0.96%	11.09%	**98.35%**
Sound Classification	14.46%	10.08%	11.42%	**92.93%**

Additionally, we also conduct experiments when sufficient time budget is granted. As shown in Table 16 though PGD and C&W achieve the very similar SRs to our proposed FAPG, they require much longer adversarial perturbation generation time. For instance,

for speaker recognition task PGD needs 4.33s and C&W even requires more than 10s to launch the attack, while the time period for each data is only 1.75s. Such huge gap makes the PGD and C&W-based attacks infeasible in the practical real-time attack scenarios. On the other hand, with achieving the very similar high SR, our proposed FAPG only needs 0.05s to generate adversarial perturbation, thereby bringing very high speedup (up to 86× and 214× as compared with PGD and C&W, respectively). Also, compared with another fast generation approach FGSM, FAPG achieves much higher SR.

Table 16: Success rate (SR) and the corresponding attack generation time of audio-dependent targeted attacks under sufficient time budget.

	Metric	FGSM	PGD	C&W	FAPG
Command Recognition	SR(%)	11.89	96.03	**97.92**	97.77
	Time	0.05s	1.36s	9.16s	**0.05s**
Speaker Recognition	SR(%)	1.65	97.47	98.08	**98.35**
	Time	0.05s	4.33s	10.74s	**0.05s**
Sound Classification	SR(%)	14.46	91.74	92.55	**92.93**
	Time	0.05s	1.85s	4.69s	**0.05s**

Memory Cost Reduction. Our proposed trainable class-wise feature maps can reduce the memory cost significantly. Without the class-wise feature embedding maps, launching targeted attack requires to train one generative model for each target class, which results in a memory consumption of 23.8 MB, 259 MB, and 23.8 MB for the speech command recognition, speaker recognition, and sound classification model, respectively. In contrast, by utilizing the class-wise embedding feature maps, our proposed FAPG only requires training a single generative model and a set of embedding maps, regardless of the number of target classes, and therefore only takes up 2.4 MB, 3.53 MB, and 2.44 MB for these three target models respectively. This leads to a memory cost reduction of 9.9×, 73.5×, and 9.8×, respectively.

4.5 Interpretation and Verification

4.5.1 Interpreting Adversarial Examples

Seeing Effects of Adversarial Perturbations from Network Dissection. We present the significance test of the interpretability of top $N \in \{10, 20, 30, 50, 80, 100\}$ sensitive units against the layer index of Resnet-152 (Figure 4.5.1-a). We also show the number of concept detectors among top $N = 100$ sensitive units versus layers for every concept category (Figure 4.5.1-b). Here we denote by conv*i_j* the last convolutional layer of the *j*-th building block at the *i*-th layer in Resnet-152 [18]. It is

seen from Figure 16-a that there exists a strong connection between the sensitivity of units and their interpretability since $p < 0.05$ in most of cases. By fixing the layer number, such a connection becomes more significant as N increases: Most of the top 100 sensitive units are interpretable, although the top 10 sensitive units might not be the same top 10 interpretable units. By fixing N, we observe that deep layers (conv4_36 and conv5_3) exhibit stronger connection between sensitivity and interpretability compared to shallow layers (conv2_3 and conv3_8). That is because the change of activation induced by adversarial attacks at shallow layers could be subtle and are less detectable in terms of interpretability. Indeed, Figure 16-b shows that more high-level concept detectors (e.g., object and part) emerge in conv4_36 and conv5_3 while low-level concepts (e.g., color and texture) dominate at lower layers.

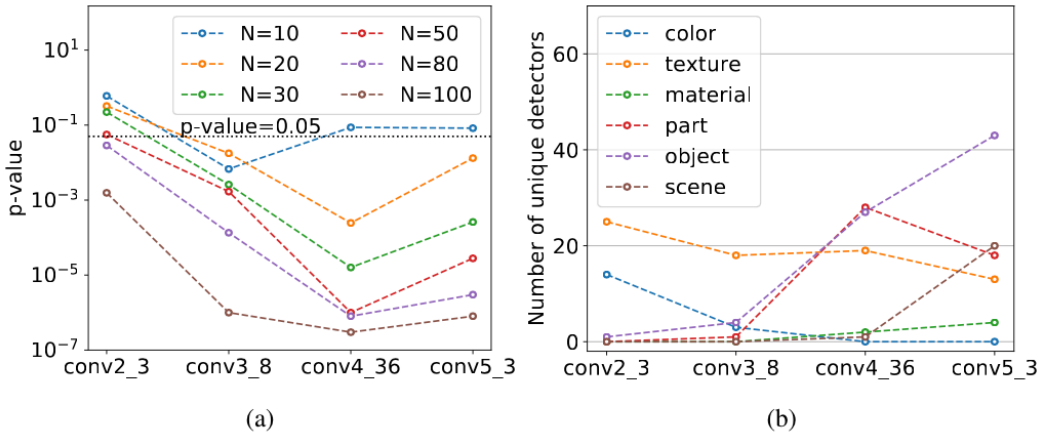


Figure 16: Sensitivity and interpretability. (a) p-value of interpretability of top N sensitive units to adversarial attacks in Resnet_152, where the presented layers include conv2_3 (256 units), conv3_8 (512 units), conv4_36 (1024 units) and conv5_3 (2048 units). (b) Number of concept detectors among top N = 100 sensitive units per layer for each concept category.

To peer into the impact of adversarial perturbations on individual images, we examine how the representation of concept detectors change while facing adversarial examples by attacking images from the same true class t_0 to the same target class t. Here the representation of a concept detector is visualized by the segmented input image determined by $\mathbf{M}_k(\boldsymbol{x})$. In Figure 16, we show two examples of attacks: "table lamp"-to-"studio couch, day bed" and "airliner"-to-"seashore, seacoast". We first note that most of low-level concepts (e.g., color and texture) are detected at shallow layers, consistent with Figure 16b. In the attack "table lamp"-to-"studio couch, day bed", the color "orange" detected at conv2_3 is less expressed for the adversarial image against the natural image.

This aligns with human perception since “orange” is related to “light” and thus “table lamp”. By contrast, in the attack “airliner”-to-“seashore, seacoast”, the color “blue” is well detected at both natural and adversarial images, since “blue” is associated with both “sky” for “airliner” and “sea” for “seashore”. We also note that high-level concepts (e.g., part and object) dominate at deeperlayers. At conv5_3, the expression of object concepts (e.g., lamp and airplane) relevant to the *true* label is *suppressed*. Meanwhile, the expression of object concepts (e.g., sofa and beach) relevant to the *target* label is *promoted*. This precisely reflects the activation promotion-suppression effect induced by adversarial perturbations

Figure 17: Visualizing impact of original (Ori) & adversarial (Adv) examples on the response of concept detectors identified by network dissection at 4 representative layers in Resnet. (top) attack “table lamp”-to-“studio couch, day bed”, (bottom) attack “airliner”-to-“seashore, seacoast”. In both top and bottom sub-figures, the first row presents unit indices together with detected top ranked concept labels and categories (in the format “concept label”-“concept category”). The last two rows present the response of concept detectors visualized by the segmented input image, where the segmentationis given by $\mathbf{M}_k(\mathbf{x})$ corresponding to the top ranked concept.

In Figure 18, we connect images in Figure 17 to PSR and CAM based visual explanation. For example, the suppressed image region identified by PSR (white color) corresponds to the interpretable activation of object concept airplane in Figure 17. And the promoted image region identified by PSR (black color) corresponds to the interpretable activation of scene concept beach.

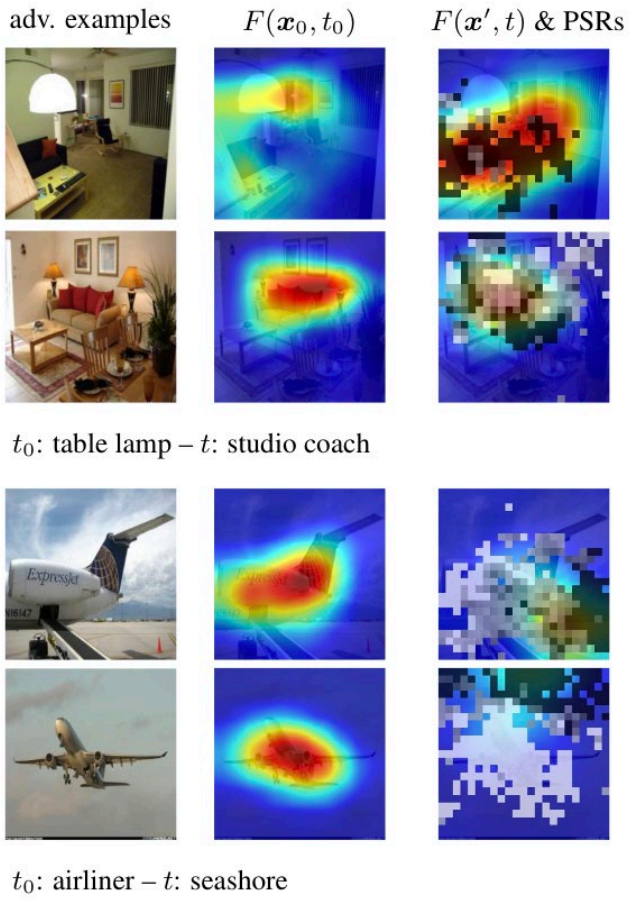
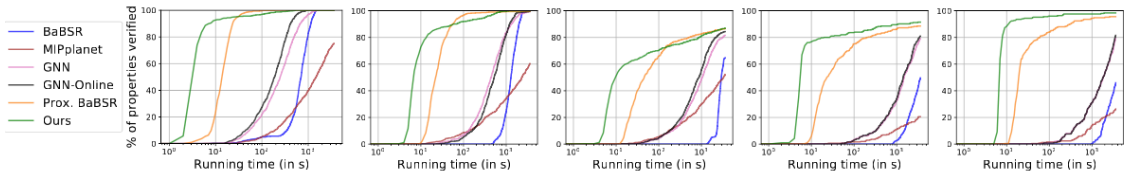


Figure 18: Interpreting adversarial perturbations via CAM and PSR. Image examples are from Figure 4.5.2. For PSR, only the top 70% most significant perturbed grids ranked by $\{s_i\}$ are shown. The white and black colors represent the suppression-dominated regions ($r_i < -1$) and the promotion-dominated regions ($r_i > 1$), respectively. The gray color corresponds to balance-dominated perturbations ($r_i \in [-1, 1]$).

4.5.2 Complete and Rapid Neural Network Verification

We include five different methods for comparison: (1) BABSR [6], a BaB and LP based verifier using a simple ReLU split heuristic; (2) MIPPLANET [14], a customized MIP solver for NN verification where unstable ReLU neurons are randomly selected for split; (3) GNN [23] and (4) GNN-ONLINE [23] are BaB and LP based verifiers using a learned graph neural network (GNN) to guide the ReLU split. (5) BDD+BABSR [5] is a very recently proposed verification framework based on Lagrangian decomposition which also supports GPU acceleration without solving LPs. All methods use 1 CPU with 1 GPU. The timeout threshold is 3600 seconds.

For the Base model in different difficulty levels, Easy, Medium and Hard, Table 17 shows that we are around 5 ∼ 40X faster than baseline BaBSR and around 2 ∼ 20X faster than GNN split baselines. The accumulative solved properties with increasing runtime are shown in Figure 19. In all our experiments, we use the basic heuristic in BaBSR for branching and do not use GNNs, so our speedup comes purely from the faster LiRPA based bounding procedure. We are also competitive against Lagrangian decomposition on GPUs.



(a)Base-Easy (b)Base-Medium (c)Base-Hard (d)Wide (e)Deep

Figure 19: Cactus plots for our method and other baselines in Base (Easy, Medium and Hard), Wide and Deep models. We plot the percentage of solved properties with growing running time.

Table 17: Performance of various methods on different models. We compare each method's avg. solving time, the avg. number of branches required, and the percentage of timed out (TO) properties.

	Base - Easy			Base - Medium			Base - Hard			Wide			Deep		
Method	time(s)	branches	%TO	time(s)	branches	%TO	time(s)	branches	%TO	time(s)	branches	%TO	time(s)	branches	%TO
BABSR	522.5	585	0.0	1335.4	1471	0.0	2875.2	1843	35.2	3325.7	455	50.3	2855.2	365	54.0
MIPPLANET	1462.2	-	16.5	1912.2	-	43.5	2172.2	-	46.2	3088.4	-	79.4	2842.5	-	73.6
GNN	312.9	301	0.0	624.1	635	0.9	1468.7	931	15.6	1791.5	375	19.0	1870.6	198	18.4
GNN-ONLINE	207.43	269	0.0	638.15	546	0.4	1255.4	968	15.6	1642.0	389	19.0	1845.7	196	18.4
BDD+ BABSR	15.68	1371	0.0	51.88	6482	0.4	**627.96**	91880	13.4	510.55	45855	11.4	230.06	6721	4.4
OURS	**11.86**	2589	0.0	**42.04**	9233	**0.0**	633.85	96755	**13.0**	**375.23**	53481	**8.5**	**81.55**	1439	**1.6**

Section 5
Conclusion

In this report we present and summarize our study for the systematic investigation on the vulnerability and robustness of deep learning system. The key research outcomes include the high-performance real-time adversarial attack approaches for the image-domain DNN models, stochastic switching-based and multi-cost multi-source-based defense mechanism, topology attack and defense for graph neural networks and the corresponding universal attack approach. We also extend the application domain from image to audio and develop several fast and real-time audio-domain adversarial attack methods. Moreover, we study the efficient interpretation of adversarial example to better detect the potential attack, and we also develop rapid and complete formal verification scheme for deep neural networks. The proposals of these approaches pave the way for better and deeper understanding of the vulnerability and robustness of various deep learning models in various application domains.

References

[1] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprintarXiv:1801.00554*, 2018.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283, 2018.

[3] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[5] Rudy Bunel, Alessandro De Palma, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip H. S. Torr, and M. Pawan Kumar. Lagrangian decomposition for neural network verification. *Conference on Uncertainty in Artificial Intelligence (UAI) 2020*, 2020.

[6] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, P Kohli, P Torr, and P Mudigonda. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(2020), 2020.

[7] Rudy R Bunel, Ilker Turkaslan, Philip Torr, Pushmeet Kohli, and Pawan K Mudigonda. A unified view of piecewise linear neural network verification. In *Advances in Neural Information Processing Systems*, pages 4790–4799, 2018.

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Securityand Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[9] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX Security Symposium (16)*, pages 513–530, 2016.

[10] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[11] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. *arXiv preprint arXiv:1709.04114*, 2017.

[12] Michael D Conover, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonçalves, Filppo Menczer, and Alessandro Flammini. Political polarization on twitter. In *Fifth international AAAI conference on weblogs and social media*, 2011.

[13] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[14] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[17] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[20] Srijan Kumar, Robert West, and Jure Leskovec. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th International conference on World Wide Web*, pages 591–602. 2016.

[21] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1695–1699. IEEE, 2014.

[22] S. Liu, S. Kar, M. Fardad, and P. K. Varshney. Sparsity-aware sensor collaboration for linear coherent estimation. *IEEE Transactions on Signal Processing*, 63(10):2582– 2596, 2015.

[23] Jingyue Lu and M Pawan Kumar. Neural network branching for neural network verification. *International Conference on Learning Representation (ICLR)*, 2020.

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

[25] Mitchell McLaren, Yun Lei, and Luciana Ferrer. Advances in deep neural network approaches to speaker recognition. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4814–4818. IEEE, 2015.

[26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[28] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.

[29] Sailik Sengupta, Tathagata Chakraborti, and Subbarao Kambhampati. Mtdeep: boosting the security of deep neural nets against adversarial attacks with moving target defense. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[30] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):41, 2019.

[31] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *arXiv preprint arXiv:1806.03185*, 2018.

[32] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*, 2017.

[33] Masashi Tsubaki, Kentaro Tomii, and Jun Sese. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35(2):309–318, 2018.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[35] Xiao Wang, Siyue Wang, Pin-Yu Chen, Yanzhi Wang, Brian Kulis, Xue Lin, and Peter Chin. Protecting neural networks with hierarchical random switching: towards better robustness-accuracy trade-off for stochastic defenses. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6013–6019. AAAI Press, 2019.

[36] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139, 2018.

[37] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

[38] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session- based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353, 2019.

[39] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[40] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.

[41] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[42] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pages 4939–4948, 2018.

[43] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[44] Yan Zhou, Murat Kantarcioglu, and Bowei Xi. Breaking transferability of adversarial samples with randomness. *arXiv preprint arXiv:1805.04613*, 2018.

[45] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856. ACM, 2018.

[46] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

List of Abbreviations

- ADMM: Alternating direction method of multipliers
- Adv: Adversarial
- AdvMS: Adversarially trained model switching
- ASR: Attack success rate
- BaB: branch-and-bound
- CAG: Content-aware generator
- CAM: Class activation mapping
- CE: Cross-entropy
- CE-PGD: Cross-entropy - Projected Gradient Descent
- CIFAR: Canadian Institute for Advanced Research
- CNN: Convolutional Neural Network
- CPU: Central Processing Unit
- C&W: Carlini & Wagner
- CW-PGD: Carlini & Wagner - Projected Gradient Descent
- DICE: Delete edges internally, connect externally
- DNNs: Deep neural networks
- EAD: Elastic-net Attack to DNNs
- EOT: Expectation over transformation
- FAPG: Fast adversarial perturbation generator
- FGA: Fast Gradient Attack

- FGM: Fast Gradient Method
- FR: Fooling ratio
- GAP: Generative adversarial perturbation
- GAT: Graph Attention Network
- GCN: Graph Convolutional Network
- GNNs: Graph neural networks
- GPU: Graphics Processing Unit
- HD: High-degree
- I-FGSM: Iterative Fast Gradient Sign Method
- IJCAI: International Joint Conferences on Artificial Intelligence
- IoU: Interpretability of unit
- LiRPA: Linear relaxation based perturbation analysis
- LP: Linear programming
- MIP: Mixed Integer Programming
- ML: Modified links
- MNIST: Modified National Institute of Standards and Technology
- NNs: Neural networks
- Ori: Original
- PGD: Projected gradient descent
- ReLu: Rectified Linear Unit
- RIR: Room impulse response
- SAP: Stochastic activation pruning

- SR: Success rate
- TO: Timed out
- TC: Top-confidence
- TPU: Tensor Processing Unit