On the Robustness and Interpretability of Deep Learning Models

WU, Weibin

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2021

Thesis Assessment Committee

Abstract of thesis entitled:

On the Robustness and Interpretability of Deep Learning Models

Submitted by WU, Weibin

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in September 2021

The enormous success of deep neural networks (DNNs) popularizes the deployment of DNNs in a broad spectrum of safety- and security-sensitive applications, like autonomous driving and medical diagnosis. Therefore, studying the robustness and interpretability of the underlying DNN models is of paramount importance. Unfortunately, due to the large-scale and black-box nature of DNNs, it is challenging to investigate the robustness and interpretability of DNNs. In this thesis, we endeavor to address these problems from multiple facets. In short, we explore two primary scenarios regarding the robustness of DNNs: accidental failures and intentional ones. We also explore providing global explanations for DNNs in pursuit of promoting their interpretability.

Firstly, we focus on improving the robustness of DNNs against accidental failures, where DNNs frequently manifest erroneous behaviors in real-world corner cases, like abnormal weather conditions. Existing countermeasures usually center on improving the testing and bug-fixing practice. However, it is scarcely viable to build an omnipotent DNN that can handle all possible cases, so anomaly detection is indispensable in practice. Motivated by the idea of data validation in traditional software, we propose Deep Validation (DV), the first framework for detecting real-world error-inducing corner cases in DNNs. Deep Validation can achieve excellent detection results against various corner case scenarios.

Secondly, we evaluate the robustness of undefended DNNs against intentional failures, where attackers craft adversarial samples to fool victim models into wrong predictions. We center on transfer-based attacks against image classifiers, where attackers are restricted to craft adversarial images based on local proxy models without the feedback information from the remote target ones. However, under such a constrained but practical setup, the synthesized adversarial samples often achieve limited success due to overfitting to the employed local model. We propose a novel mechanism to alleviate the overfitting issue, called Attention-guided Transfer Attack (ATA). Experimental results confirm that our method can markedly promote the transferability of adversarial instances.

Thirdly, we turn to assess the robustness of defended DNNs against intentional failures. We also consider transfer-based attacks as before. Since adversarial noises are usually purposeful perturbations of small magnitude, previous attacks hardly survive under defenses, like transformation-based ones. To better evaluate the susceptibility of existing defenses, we propose a novel attack method named Adversarial Transformation-enhanced Transfer Attack (ATTA). It is motivated by the data augmentation methodology to improve the generalization of models. Extensive experiments show that our scheme outshines previous proposals in evaluating the robustness of defended DNNs against transfer-based attacks.

Lastly, we cover how to promote the interpretability of DNNs. In addition to being a scientific problem itself towards building safe and dependable DNNs, elevating the interpretability of DNNs is conducive to spot robustness issues and promote the robustness of DNNs. Specifically, we concentrate on offering global explanations, which contribute to understanding model predictions on a whole category of samples. However, existing methods overwhelmingly conduct separate input attribution or rely on local approximations of models, making them fail to offer faithful global explanations of DNNs. To overcome such drawbacks, we propose a novel two-stage framework: Attacking for Interpretability (AfI), which explains model decisions in terms of the importance of user-defined concepts. Experimental comparisons corroborate that AfI can provide more accurate estimations of concept importance than existing proposals.

論文題目：論深度學習模型的魯棒性與可解釋性
作者　　：吳煒濱
學校　　：香港中文大學
學系　　：計算機科學與工程學系
修讀學位：哲學博士
摘要　　：

深度神經網絡的巨大成功使其在一些對安全和安保性要求很高的領域得到了廣泛應用，比如自動駕駛和醫學診斷。因此，研究這些深度模型的魯棒性和可解釋性顯得格外重要。然而，由於深度模型的規模性和黑盒性，研究深度模型的魯棒性和可解釋性顯得十分困難。本論文致力於從多方面解決這些問題。簡言之，我們探索了涉及深度模型魯棒性的兩個主要情況：意外故障和惡意故障。我們也研究了提供模型的全局解釋，以改善其可解釋性。

首先，我們關注的是改善深度模型在面對意外故障時的魯棒性。意外故障是指深度模型在面對現實的異常情況時，比如異常的天氣情況，經常會產生錯誤的行表現。現有的解決方案通常局限於改善故障測試和修復的過程。然而，構造一個萬能的可以解決所有可能情況的深度模型是幾乎不可能的，因此異常檢測在實際中就顯得不可或缺。受傳統軟件中數據驗證的思路的發，我們首次提出了深度驗證框架（DV），用於檢測現實中易導致模型錯誤的異常情況。該深度驗證框架在多種異常場景下都能達到優異的檢測表現。

第二部分，我們研究了評估未防禦過的深度模型在面對惡意故障時的魯棒性。惡意故障是指攻擊者構建對抗樣本來誤導目標模型使其預測出錯。我們關注的是對圖像分類器的可遷移攻擊，也就是攻擊者在構造對抗樣本的過程中，只基於本地模型而無需遠端目標模型的反饋信息。然而，在這一受約束的實際場景下，構造的對抗樣本常常過擬合於本地模型，使其攻擊成功率很低。我們提出了一個全新的機制來改善這一過擬合的問題，叫做基於注意力的可遷移攻擊（ATA）。實驗結果證明我們的方法可以顯著提升所得到的對抗樣本的可遷移性。

第三部分，我們轉向評估防禦過的模型在惡意故障下的魯棒性。跟之前類似，我們同樣考慮了可遷移攻擊的場景。因為對抗噪聲通常是惡意的小擾動，這使得之前的攻擊方法難以攻破防禦措施，比如以圖形變換為基礎的防禦方法。了準確評估現有防禦措施的魯棒性，我們提出了一種全新的攻擊方法叫做對抗變形强化的可遷移攻擊（ATTA）。該方法是受利用數據增補改善模型泛化能力的思路的發。大量實驗表明，相比於現有方法，我們的工作可以更好地評估防禦過的深度模型在面對可遷移攻擊時的魯棒性。

最後，我們研究了如何提高深度模型的可解釋性。這一問題除了本身是構建安全和可靠的深度模型的過程中一個重要的科學問題外，提高深度模型的可解釋性也有利於發現其魯棒性問題並提高其魯棒性。具體而言，我們關注提供全局解釋，以便於理解模型對整一類樣本的預測結果。然而，當前的方法絕大多數都進行的是對獨立輸入的歸因或依賴於對模型的本地擬合，使得這些方法不能提供對深度模型的準確的全局解釋。了克服這些缺陷，我們提出了一個全新的兩步框架：利用攻擊來解釋（AfI），用於說明各種人類定義的概念對模型決策的重要性。實驗結果證明了，相比現有手段，該方法能提供更準確的概念重要性的估計。

Acknowledgement

First and foremost, I would like to thank my supervisors, Prof. Michael R. Lyu and Prof. Irwin King, for their excellent supervision during my Ph.D. study at CUHK. From choosing the research topic to technical writing, their inspiring guidance and patience help me conduct challenging research work. During the Ph.D. study period, I have learned so much from their knowledge and attitude in doing research.

I am grateful to my thesis assessment committee members, Prof. Wei Meng, Prof. Patrick P. C. Lee, for their constructive comments and valuable suggestions to this thesis and all my term presentations. Great thanks to Prof. Chu-Song Chen from National Taiwan University, who kindly serves as the external examiner for this thesis.

I would like to thank Dr. Yuxin Su, Dr. Xixian Chen, and Dr. Shenglin Zhao for their valuable contributions to the research in this thesis.

I would like to thank Dr. Hui Xu for his insightful discussion and inspiring guidance on the research topic in the early stage of my Ph.D. study.

I am also thankful to my fantastic group fellows, Xiaotian Yu, Shilin He, Wang Chen, Wenxiang Jiao, Zhuangbin Chen, Yue

Wang, Pengpeng Liu, Ken Chan, Jian Li, Haoli Bai, Jingjing Li, Yifan Gao, Yu Kang, Guang Ling, Chen Cheng, Jieming Zhu, Hongyi Zhang, Junjie Hu, Pinjia He, Tong Zhao, Cuiyun Gao, Jichuan Zeng, Jiani Zhang, Han Shao, and Tianyi Yang.

Last but most important, I would like to thank my family. Their deep love and constant support are the driving force during my Ph.D. study.

To my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Deep neural networks (DNNs), as emerging machine learning techniques, have amazingly approached or surpassed human performance on diverse tasks, such as image classification [48,59,70], machine translation [47, 142, 157], and text analysis [26, 52, 91]. These advances have facilitated the application of DNNs in a growing spectrum of safety- and security-critical domains, including self-driving [149,169], biometric authentication [80,127], and medical diagnosis [83, 158]. Therefore, it is imperative to study the robustness and interpretability of DNNs, which are the critical quality attributes to ensure the correct functionality of DNN-based systems [3].

Let us first specify what quality attributes the robustness of DNNs refers to in this thesis. In the IEEE Standard Glossary of Software Engineering Terminology [1], *robustness* is defined as follows.

The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

Therefore, robustness defines a quality attribute of a system or component in terms of its immunity against invalid inputs or stressful environmental conditions. In this thesis, we center on the robustness of DNNs against invalid inputs.

Invalid inputs are undesired data that are likely to originate from stressful environmental conditions and can trigger the malfunction of a system or component [6, 24, 51, 116]. For example, a sorting program may only be able to handle numbers, and thus letters are invalid inputs for this program. Similarly, when it comes to DNNs, invalid inputs are illegitimate inputs that are not expected during the design of the system [176].

To formally define the invalid inputs and robustness, we can denote a deep learning model as $f : X \to Y$, which predicts a label $y \in Y$ for a given input $\mathbf{x} \in X$. Let y^* be the ground-truth label of $\mathbf{x}$, and $\mathcal{V}$ be the training data distribution of f. Then we can define the invalid inputs of DNNs as follows.

Definition 1.1.1. (Invalid input). The invalid inputs for a deep learning model f are a set of data $\tilde{X} \subseteq X$ such that $\tilde{X} \not\sim \mathcal{V}$.

Therefore, we can formulate the robustness of DNNs as follows.

Definition 1.1.2. (Robustness). The robustness of a deep learning model f measures the performance of f on invalid inputs $\tilde{X}$, which can be quantified as: $\mathbb{E}_{\mathbf{x} \in \tilde{X}}[\mathbb{1}(f(\mathbf{x}) = y^*)]$.

The invalid inputs for DNNs can come from two sources. One is natural environmental conditions that are overlooked during the design of the system. The resultant invalid inputs are called *real-world corner cases*, which may incur *accidental failures* of DNNs. For example, a DNN-based self-driving car may hinge on lane mark images for autonomous navigation. When the self-driving car is working on a sunny day, the lane marks can be easily spotted in the captured images, and the car can function correctly. Nevertheless, when the self-driving car runs on a foggy day, the lane marks may become vague and difficult to identify in the captured images. Therefore, the input images become real-world corner cases for the self-driving car and may incur erroneous behaviors of the system.

The other one is the adversarial environmental condition. In this situation, attackers purposefully synthesize invalid inputs, called *adversarial inputs/samples*, to mislead DNNs, resulting in *intentional failures* of DNNs. For example, an attacker may attach a photo of a human face to the lane. A DNN-based self-driving car may mistake it in the captured image for real persons. Therefore, the attacker succeeds in crafting adversarial samples to make the system misbehave.

Similar to traditional software, *testing* and *debugging* play an essential part in evaluating and improving the robustness of DNNs during model development [15, 140]. More concretely, testing aims to generate a diverse set of test cases to spot the bugs of DNNs, while debugging implies fixing the discovered failures, which is usually achieved by retraining models with the error-prone inputs [112,140]. Therefore, testing can facilitate the detection of invalid inputs to DNNs and is usually a prerequisite for debugging. As per the requirements of the developer, testing amounts to generating real-world corner cases or adversarial samples, or both.

Two primary challenges have emerged from the research of DNN testing. One is the *test oracle problem*, which involves automatically attaining the ground-truth labels of the generated test cases [58]. A prevailing workaround is the differential or metamorphic testing technique [20, 90]. Briefly, metamorphic testing resorts to semantic-preserving transformations to synthesize test cases from seed samples while rendering their labels identical. Hence it implies a failure of the system when we can observe different outputs from the system on the test input and the corresponding seed sample. For example, an adversarial image can be constrained to be within a small l_p-norm neighborhood of the corresponding source image, which can thus maintain the ground-truth label of the original image. By this means, we can generate adversarial images from clean source images and directly apply the resultant adversarial samples to test the DNNs without manually labeling these samples.

The other is the *test adequacy* issue, and the obstacle is two-fold. The first one is how to define the test adequacy of a DNN testing scheme. In traditional software testing, test adequacy is measured by the test coverage [186]. For example, statement coverage defines test adequacy by computing the percentage of exercised statements during testing. Unfortunately, unlike traditional software, the adequacy of DNN testing cannot be directly measured by the similar coverage criterion in traditional software testing, since the operations of DNNs are not explicitly programmed. Therefore, in pursuit of quantifying the adequacy of a testing scheme for DNNs, we can first craft a fixed number of test cases. Then we can estimate the test adequacy of a testing technique with the percentage of triggered failures by the synthesized test set. The second obstacle is how to raise the test adequacy of a DNN testing approach. Existing methods usually show limited success [61].

In this thesis, we focus on testing the robustness of deep image classifiers under adversarial environmental conditions. In other words, we aim to devise attack methods to generate adversarial samples that can achieve high attack success rates (test adequacy). From the perspective of simulating an adversarial environmental condition during DNN testing, we can adopt a *threat model* to define the assumptions we make about the potential attackers [159], such as the attackers' goal, the attackers' admissible actions, and the attackers' knowledge of the target model.

The threat model that we adopt is *transfer-based attacks* [162]. In transfer-based attacks, adversaries craft adversarial images based on off-the-shelf local models without the feedback information from the target model, and directly apply the resultant adversarial samples to attack the victim DNNs. Besides, for stealthiness, the adversarial images are restricted to be near the corresponding seed images in the l_p space. Therefore, transfer-based attacks mirror a severe threat to DNNs in practice and have attracted an exploding interest recently [28, 162].

We first consider testing the robustness of undefended DNNs against transfer-based attacks. Unfortunately, under such a restricted but practical setup, existing attack schemes often manifest limited transferability due to overfitting to the exclusive vulnerabilities of the employed local model. *Transferability* refers to the phenomenon that adversarial samples generated from one model can remain malicious to another one [81]. Therefore, low transferability indicates low attack success rates and low test adequacy. In this thesis, we propose to alleviate the overfitting issue by introducing a regularization term [162]. Our method, called the *Attention-guided Transfer Attack* (ATA), features guiding the search of adversarial images towards the common susceptible direction of different classifiers. As a result, we can alleviate overfitting to a specific source model and boost the transferability of resultant adversarial samples.

We then turn to evaluate the robustness of defended DNNs against transfer-based attacks. As attackers craft adversarial samples by prudently attaching small l_p-norm adversarial noise to legitimate images, the resultant adversarial samples become sensitive to image transformations. Motivated by this observation, defenders can resort to semantic-preserving transformations to pre-process inputs to purify adversarial noise. This body of defenses is shown to be highly effective against existing attacks [22, 166]. To better evaluate the robustness of such defended DNNs, we propose an *Adversarial Transformation-enhanced Transfer Attack* (ATTA) [164]. It is inspired by the data augmentation methodology to improve the model generalization capacity. Specifically, we augment the toughest image transformations during the learning of adversarial samples and require the crafted adversarial samples to resist such distortions. As such, we can make the generated adversarial samples more effective and elevate their transferability against the defended DNNs.

Besides testing and debugging, we need detection to make DNNs more safe and dependable during runtime. Although testing and debugging are indispensable during the development of DNNs, we should bear in mind that after deployment, the real-world environmental conditions can vary with many factors, like brightness, contrast, and camera positions for autonomous cars. Therefore, the training data and test cases we possess are just a relatively small fraction of all possible scenarios in practice. Moreover, it is doubtful whether there exists a perfect DNN model that can handle all possible situations in light of the "no free lunch" theorem [156, 160]. Therefore, in addition to testing and debugging that evaluate and improve the robustness of DNNs before their deployment, we need a detection mechanism to further secure the correct functionality of DNNs during runtime. Concretely, *detection* endeavors to identify invalid inputs at inference and thus is the prerequisite of a fail-safe system.

In this thesis, we concentrate on elevating the robustness of deep image classifiers against accidental failures via detecting real-world corner cases during runtime. Our solution is motivated by *data validation*, which is an effective strategy to detect invalid inputs in traditional software engineering [6, 24, 51, 116]. Data validation within conventional software often resorts to specific validation rules. For example, we can define a validation rule that the entered operands should be integers for a calculator program, if the calculator can only process integers. Therefore, inputs that violate the validation rule are viewed as invalid inputs, since they may incur erroneous outputs from the program. Similarly, real-world corner cases are invalid inputs that go beyond the capacity of DNNs. A similar data validation procedure may also help to identify real-world corner cases for DNNs.

However, existing data validation processes appear infeasible for a DNN model due to its distinct design philosophy from traditional software. The proper executions of a program are manually defined and can be expressed as succinct control flow statements. In contrast, the functionality of a DNN model is indeed learned from a large amount of training data spontaneously without much human supervision. As such, its knowledge is encoded in millions of indecipherable parameters and the associated intricate network structures [39, 103].

To tackle these challenges, we propose to resort to the training data of deep image classifiers to model their specifications for data validation, which is also non-trivial. A naive idea is to require the pixel values of the input images to be within [-1, 1], or to require the input images to have occurred in the training data. However, the former validation rule is too sloppy to effectively detect real-world corner cases, while the latter is too rigid to leave room for model generalization.

In this thesis, we propose *Deep Validation* (DV) as the remedy, which is an effective adaption of data validation for DNNs [165]. It first models the valid input range of intermediate layers within DNNs through characterizing reference distributions. We then quantify the validity of input images by estimating their discrepancy to the valid input region. As a result, we can identify real-world corner cases for DNNs at inference and achieve superior detection performance over state-of-the-art baselines.

In addition to the robustness issues, DNNs suffer from a lack of *interpretability*. "Interpretability is the degree to which a human can understand the cause of a decision" [94]. The reasons for the insufficient interpretability of DNNs are as follows. First, in stark contrast to traditional software, whose logic is explicitly programmed, the operations of DNNs are learned automatically from data with indirect human supervision. They thus cannot be explicitly expressed with program statements. Besides, the increasing complexity of modern DNNs makes it more intractable to decipher the black box.

It is crucial to elevate the interpretability of DNNs. First, the interpretability issue of DNNs raises doubts about the safe and dependable deployment of DNNs in practice [43]. Consequently, interpreting and understanding the behaviors of DNNs can justify their decisions to promote model trustworthiness [43]. More critically, improving the interpretability of DNNs is advantageous to spot their latent defects to inspire the design of better models [50], including those with greater robustness [12,37,107].

For understanding the predictions of DNNs, *attribution* is a prevailing methodology in the literature [38, 105]. It endeavors to succinctly summarize how DNNs arrive at their final decisions. The convention is to measure the importance of human-understandable units to model predictions, such as pixels (*i.e., input attribution*) and concepts (*i.e., concept attribution*) [66].

Input attribution has some pitfalls. The outcome of input attribution, called *saliency maps*, can highlight the most responsible parts of input images for model decisions. Unfortunately, despite being intuitive, input attribution also suffers from confining itself to input space. The primary culprit is that the semantic meanings of image pixels are highly diversified and interdependent. Consequently, the saliency maps returned by input attribution are subject to human perceptions before they become a human-readable interpretation. Unfortunately, human judgments are error-prone and can lead to contradicting conclusions [66].

Concept attribution can overcome the ambiguity of input attribution by directly measuring the importance of human-understandable concepts to model decisions. It thus has attracted growing attention recently [38, 66, 184]. There are two explanation interfaces of concept attribution studied in the community: *local explanations* [184] and *global explanations* [66].

In this thesis, we center on increasing the interpretability of deep image classifiers with global concept attribution. The reasons are as follows. Local explanations investigate the rationale of model predictions on individual data points, which are helpful when we only care about a specific instance. In contrast, global explanations center on mining generic decision modes that apply to an entire class of examples. For instance, global explanations can answer to what extent the banded texture is related to a zebra class in model cognition. Therefore, such global explanations are conducive to summarize the model knowledge succinctly and understand the model as a whole [66].

Prevailing methodologies for global concept attribution of DNNs in the literature are not satisfactory. These efforts overwhelmingly hinge on local approximations of models and analyze individual predictions in isolation. As for global explanations, they repeat the analysis of single samples from a category of interest and simply return summary statistics [38, 66]. It is doubtful whether such a strategy to obtain global explanations indeed sees “globally”. The defect primarily originates from the local approximation of DNNs. However, such an approximation holds merely when we deal with the proximity of individual instances or the last linear layer of DNNs. Worse still, inspecting individual predictions separately ignores the connections among examples of the same class, which may not capture the generic properties of the class embedded in the model knowledge.

To alleviate the shortcomings of existing proposals, we propose a novel concept attribution framework for global explanations of image classifiers: *Attacking for Interpretability* (AfI) [163]. It systematizes the process to model explanations in that we make each attribution step grounded and propose to evaluate the intermediate results. More crucially, we extend the methodology of input occlusion to feature occlusion. The feature occlusion analysis features a process of attacking DNNs and probing into model internals to identify critical features. Therefore, our framework enables layer-wise inspections and learns a global explanation. Experimental results confirm that we can afford more accurate explanations of model decisions than existing efforts.

1.2 Thesis Contributions

In summary, we focus on studying the robustness and interpretability of deep learning models. To this end, we make contributions from various perspectives. More specifically, when considering the accidental failures of deep learning models, spurred by traditional software reliability engineering, we employ detection to elevate model robustness against real-world corner cases. When it comes to intentional failures of deep learning models, we resort to testing to evaluate model robustness against adversarial samples. We cover synthesizing adversarial samples against both undefended and defended models. As for the distinct drawback of deep learning models from traditional software, namely, their lack of interpretability, we focus on promoting model interpretability via global concept attribution.

The overall contributions of the thesis are highlighted as follows.

- For accidental failures of deep learning models, we introduce *Deep Validation* (DV) as the first framework to automatically identify error-inducing real-world corner cases for a deployed DNN-based system [165]. It is motivated by the data validation methodology in traditional software. Concretely, we first model the valid input range of intermediate layers within DNNs through characterizing reference distributions with their training data. We then quantify the validity of inputs by estimating their discrepancy to the valid input distributions. We conduct extensive experiments across various datasets and DNN architectures to evaluate our framework. Deep Validation consistently reports prominent detection results on eight different categories of real-world corner cases.

- For intentional failures of undefended deep learning models, we consider adopting the threat model of transfer-based attacks to test model robustness against adversarial samples. In this setup, we propose a novel strategy to boost the transferability of adversarial images: *Attention-guided Transfer Attack* (ATA) [162]. It features an introduction of model attention to regularize the search of deceptive noises, which mitigates overfitting to specific blind spots of the source model. Extensive experiments show that our technology can severely compromise diverse top-performance image classifiers in both white-box and black-box scenarios, confirming the remarkable test adequacy achieved by our attacks.

- For intentional failures of defended deep learning models, we also investigate evaluating their robustness against adversarial samples under the threat model of transfer-based attacks. Under this scenario, we first propose to improve the transferability of adversarial samples against defenses with adversarial transformations. It aims to augment the most harmful image transformations to promote the effectiveness of adversarial samples. We conduct extensive experiments on the ImageNet benchmark to evaluate our approach: *Adversarial Transformation-enhanced Transfer Attack* (ATTA) [164]. Experimental results confirm the superiority of our method over state-of-the-art baselines in attacking defended models.
- For improving the interpretability of deep learning models, we propose a novel concept attribution framework for global explanations of DNNs: *Attacking for Interpretability* (AfI) [163]. Our framework explicitly builds upon a two-stage procedure and employs a novel feature occlusion methodology to learn a global interpretation. Experimental results validate the effectiveness of our approach and showcase that it can afford more accurate model explanations than prior efforts. Moreover, since better interpretability of DNNs can be beneficial to reveal potential robustness

issues [12, 37, 107], it is promising to apply our framework to analyze and elevate the robustness of DNNs.

1.3 Thesis Organization

The remainder of this thesis is organized as follows.

- **Chapter 2**
  In this chapter, we systematically review the background of the robustness and interpretability of deep learning models, incorporating the brief introduction of DNNs, accidental failures of DNNs, intentional failures of DNNs, and the interpretability of DNNs.

- **Chapter 3**
  In this chapter, to improve the robustness of DNNs against accidental failures, we propose Deep Validation (DV) as the first framework to detect real-world corner cases for DNNs [165]. Concretely, we first specify the real-world corner cases well recognized in the community by defining the fault model of DNNs in Section 3.2. Then in Section 3.3, we elaborate on our Deep validation framework, which is inspired by the data validation methodology in traditional software. Finally, we extensively evaluate our framework and compare it with state-of-the-art baselines in Section 3.4.

- **Chapter 4**
  In this chapter, to test the robustness of undefended DNNs against adversaries, we devise a novel transfer-based attack

mechanism to manufacture adversarial samples: Attention-guided Transfer Attack (ATA) [162]. It features the introduction of model attention to regularize the search for adversarial samples, which we describe in Section 4.3. With extensive experiments in Section 4.4, we demonstrate that the proposed strategy can consistently exceed state-of-the-art attacks by a large margin, while only requiring a regularization term.

- **Chapter 5**
  In this chapter, we turn to evaluate the robustness of defended DNNs against adversaries. To this end, we introduce a novel approach to synthesize transferable adversarial samples: Adversarial Transformation-enhanced Transfer Attack (ATTA) [164]. The key is to enhance the effectiveness of the generated adversarial examples against the toughest image transformations, and we elaborate on how to achieve this goal in Section 5.2. In Section 5.3, we conduct extensive experiments to compare the performance of our method with state-of-the-art attacks, and experimental results validate the superiority of our approach.

- **Chapter 6**
  In this chapter, to improve the interpretability of DNNs, we explore global explanations of DNNs with concept attribution. We design a novel framework for learning global explanations of DNNs: Attacking for Interpretability (AfI) [163]. It involves a two-stage procedure: feature attribution and concept attribution, which we present in Section 6.2. Then in Section 6.3, we systematically evaluate the interpretation results of our method both qual-

itatively and quantitatively. Experimental results confirm that our framework can afford more accurate explanations of DNNs than other competitive strategies.

- **Chapter 7**
  In this chapter, we summarize this thesis and provide some promising future research directions about the robustness and interpretability of deep learning models.

□ **End of chapter.**

Chapter 2

Background Review

In this chapter, we review related work to situate our contributions in the literature. Figure 2.1 exhibits the taxonomy. Specifically, this thesis centers around the robustness and interpretability of deep neural networks. Therefore, we first offer the preliminaries of deep neural networks in Section 2.1.

Then we review related work on the robustness of deep neural networks against invalid inputs. According to the categorization of invalid inputs, we discuss the robustness of deep neural networks in two scenarios. One is accidental failures caused by real-world corner cases, and the other is intentional failures resulting from adversarial samples.

Section 2.2 introduces two lines of research on the accidental failures of deep neural networks. One is testing, which aims to synthesize an immense quantity of real-world corner cases to evaluate the robustness of DNNs against accidental failures. DeepXplore and DeepBackground are representative works of this kind [112,181]. The other line of research proposes to detect real-world corner cases during runtime to enhance the robustness

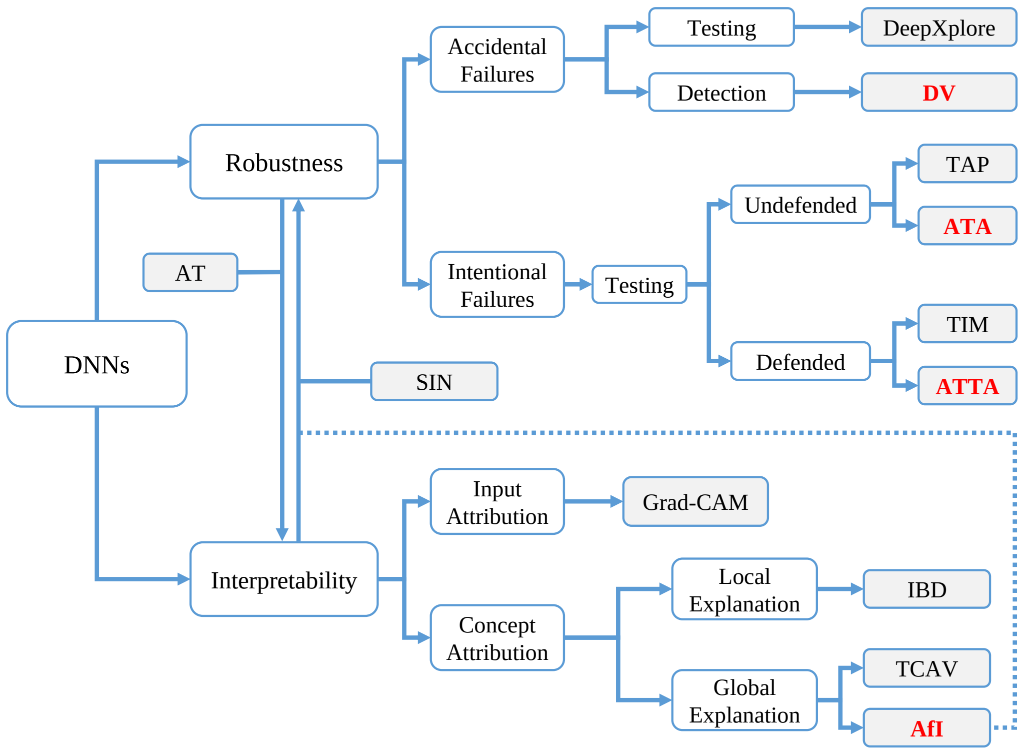


Figure 2.1: Taxonomy of related work and our contributions. The red text indicates our work. The grayed box shows a sample approach.

of DNNs against accidental failures. Our Deep Validation (DV) is the first framework to this end [165].

As for the intentional failures of DNNs, we focus on testing, which amounts to manufacturing adversarial samples to evaluate the robustness of DNNs against intentional failures. Since the most promising defense to date is adversarial training [87], which employs adversarial samples to augment the model's training data, testing is also conducive to defend against intentional failures of DNNs. Section 2.3 presents two bodies of studies on applying adversarial attacks to evaluate the robustness of DNNs against intentional failures. One targets undefended models. The transferable adversarial perturbation

(TAP) [185], transferable targeted perturbations (TTP) [98], and our Attention-guided Transfer Attack (ATA) [162] fall into this category. The other attacks defended models, for example, the Translation-Invariant Method (TIM) [29] and our Adversarial Transformation-enhanced Transfer Attack (ATTA) [164].

Finally, in Section 2.4, we present the recent advance in improving the interpretability of DNNs, which proposes to explain the model's decision in terms of the importance of human-comprehensible units. Such attribution techniques can be further divided into input attribution and concept attribution, which aim to figure out the importance of input pixels and concepts to model's decisions, respectively. Among many proposals, Grad-CAM is a well-established input attribution method [130]. Our work concentrates on concept attribution, since it can overcome the ambiguity of the input attribution results. Two explanation interfaces exist in concept attribution, namely, local explanations and global explanations. The former interprets the model's decision on individual inputs, like Interpretable Basis Decomposition (IBD) [184]. The latter one mines the generic decision mode of DNNs on the same class of samples, such as Testing with Concept Activation Vectors (TCAV) [66], the Causal Concept Effect (CaCE) [44], Integrated Conceptual Sensitivity (ICS) [129], and our Attacking for Interpretability (AfI) [163].

Moreover, we review the progress in investigating the interplay between the robustness and interpretability of DNNs. On the one hand, researchers have discovered that greater robustness of DNNs can facilitate their interpretability. For example, adversarial training (AT) is one scheme born to enhance model robustness, while found to produce more interpretable models as an unexpected emergent benefit [155]. On the other hand, there are attempts to demonstrate the opposite direction. That is, better interpretability of DNNs can assist in analyzing and promoting their robustness. Stylized-ImageNet (SIN) [37] and Permuted Adaptive Instance Normalization (pAdaIN) [104] are exemplars of this kind. Our AfI is also related to this line of work in that we are devoted to producing better interpretations of DNNs, offering a toolkit to identify potential robustness issues.

2.1 Deep Neural Networks

It has been shown either theoretically or empirically that different neural networks structures are more suitable than the others for different tasks [39]. Like in natural language processing, the Long Short Term Memory network (LSTM) proposed by [55] is widely used. It is a variant of recurrent neural networks which is specialized for processing sequential data. However, in image recognition, it is the convolutional neural network - a variant of the feedforward network - that overwhelms the others, because it is quite good at handling data that contain rich spatial information like images. Since we will focus on the DNNs for the image recognition problem, it is useful to introduce the convolutional neural networks here first.

We take a classic image recognition task as an example. The notations are summarized in Table 2.1. Suppose there are totally m mutually exclusive classes for all the input images. Then the convolutional neural network is designed to be an m-class classifier and directly acts on the raw pixel values of grey-scale

Table 2.1: Notations and their descriptions.

Notation	Description
m	Number of classes
$\mathbf{x}$	Image vector
$\mathbb{R}^n$	Real coordinate space of dimension n
h	Height of an image
w	Width of an image
x_i	Intensity of pixel i
$F(\mathbf{x};\theta)$	DNN function with parameter θ
$\mathbf{y}$	Output vector
θ_i	Weight of the i-th layer
$\hat{\theta}_i$	Biases of the i-th layer
$\sigma(\mathbf{x})$	Activation function
$Z(\mathbf{x})$	Logit layer function
$\mathbf{z}$	Logit vector
$C(\mathbf{x})$	Classification function

or RGB images. We can view an image as a vector $\mathbf{x} \in \mathbb{R}^n$. Specifically, for an $h \times w$-pixel grey-scale image, we represent it as a two-dimensional vector $\mathbf{x} \in \mathbb{R}^{hw}$, where x_i denotes the intensity of pixel i and is often scaled to be in the range of [0, 1]. All the pixel values of the image are stored in row-major order and from left to right within a row. While for an RGB image, we encode it as a three-dimensional vector $\mathbf{x} \in \mathbb{R}^{3hw}$, where x_i denotes the intensity of pixel i and is also scaled to be in the range of [0, 1]. The pixel values for three channels - red, green, blue - are stored in a predefined order. Within each channel, the pixel values of the image are also stored in row-major order and from left to right within a row.

As a result, we can regard our convolutional neural network model as a function $F(\mathbf{x};\theta) = \mathbf{y}$, which accepts an input image $\mathbf{x} \in \mathbb{R}^n$ and outputs a vector $\mathbf{y} \in \mathbb{R}^m$. For convenience, we denote the corresponding deep neural network as F, and it is a cascading of different kinds of layers of neurons. Certainly, the model function F depends on the model's parameter θ, which is implicitly encoded in the model architecture.

Now let us introduce the component layer of a typical deep convolutional neural network (CNN) architecture, AlexNet, developed by [70] to get a taste of such kind of neural network model.

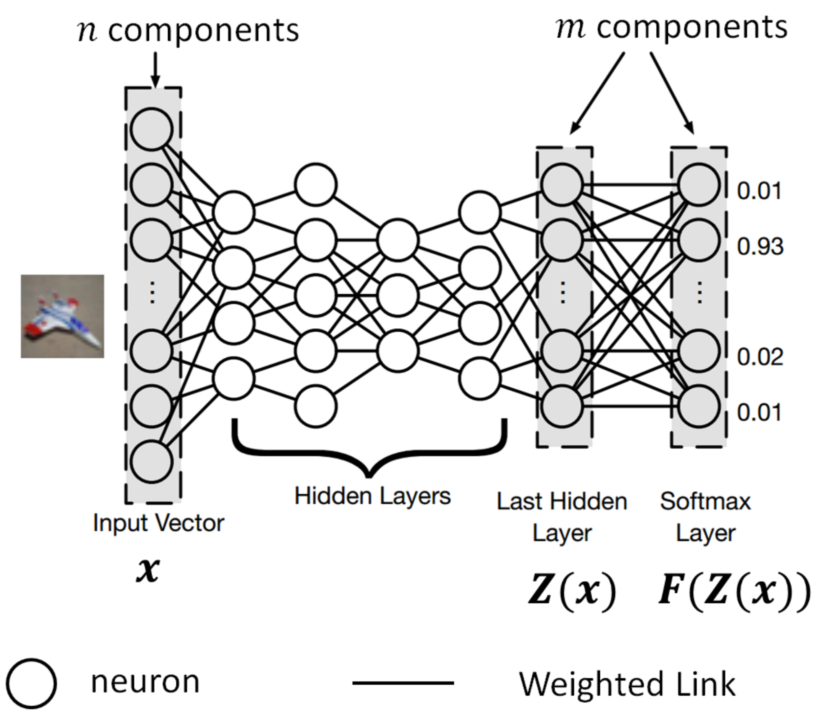


Figure 2.2: A typical deep CNN architecture (modified from the figure in [110]).

2.1.1 Convolutional Layer

Convolutional layers are feature filters, and are used to learn and detect certain features from the input image. In contrast to the fully connected layer, where each neuron has connections with all the neurons of the former layer, convolutional layers apply the ideas of local receptive fields and shared weights. Therefore, convolutional layers tremendously decrease the number of connection weights needed to learn.

2.1.2 Pooling Layer

Pooling layers are often attached to convolutional layers to compress the feature maps output by convolutional layers and extract their main characteristics. The side effect of such abstraction is that it can further cut down the parameters needed to learn and thus the complexity of the model.

2.1.3 Fully Connected Layer

Neurons are fully connected to the neurons in the previous layer in a fully connected layer. It is used to blend different features learned by the former layers so that the final classifier can obtain the overall information of features.

2.1.4 Softmax Layer

A Softmax layer is often used as the final output layer of a deep CNN model. An example of such a deep CNN model is depicted in Figure 2.2, where the model weights are represented as weighted links among neurons.

From Figure 2.2, the final function of a neural network can be seen as a composition of a series of layer functions:

$$F = \text{Softmax} \circ F_n \circ F_{n-1} \circ \cdots \circ F_1, \tag{2.1}$$

where

$$F_i(\mathbf{x}) = \sigma(\theta_i \cdot \mathbf{x}) + \hat{\theta}_i \tag{2.2}$$

is the function of the i-th component layer. Here θ_i summarizes the weights of the i-th component layer, and $\hat{\theta}_i$ is the corresponding layer biases. All these weight matrices and biases vectors make up the complete model parameters. σ is called the activation function, whose common choices are non-linear functions like tanh [93], sigmoid, ReLU [86], or ELU [21].

The last hidden layer is designed to be a fully connected layer, and the number of neurons in this layer equals the total number of classes in our task. The output vector of this layer $Z(\mathbf{x}) = \mathbf{z}$ is called the logits.

The final Softmax layer conducts a kind of normalization for all the elements in $\mathbf{z}$, and computes the i-th component of the final output vector $F(\mathbf{x})$ using the Softmax function:

$$F(\mathbf{x})_i = \frac{e^{z_i}}{\sum_{t=1}^{m} e^{z_i}}. \tag{2.3}$$

Obviously, we can summarize it as:

$$F(\mathbf{x}) = \text{Softmax}(Z(\mathbf{x})) = \mathbf{y}. \tag{2.4}$$

It ensures that the final output vector $\mathbf{y}$ satisfies the following properties:

$$0 \leq y_i \leq 1 \tag{2.5}$$

and

$$\sum_{i=1}^{m} y_i = 1. \tag{2.6}$$

The output vector $\mathbf{y}$ can thus be treated as a probability distribution. In other words, we can regard y_i as the probability of input $\mathbf{x}$ belonging to class i. Therefore, our classifier will infer that $C(\mathbf{x}) = \arg\max_i F(\mathbf{x})_i$ is the label for input $\mathbf{x}$.

2.2 Accidental Failures of DNNs

Despite their marked performance, DNNs often suffer from accidental failures incurred by real-world corner cases. Generally speaking, real-world corner cases are invalid inputs that are overlooked during the design of the system, but can naturally occur in practice. For example, an autonomous Uber vehicle misclassified a pedestrian during road-test at night [154], since the self-driving car is trained mainly based on daytime data. Such misbehavior of DNNs is named accidental failures, and the corresponding culprit input is called real-world corner cases.

In this section, we review two bodies of related works on exploring the robustness of DNNs against real-world corner cases. One is testing, which aims to evaluate the robustness of DNNs. The other is detection, which endeavors to increase their robustness.

There is a growing interest in automatically generating a mass of test cases [112, 113, 150, 176], since regular test data often lack diversity and fail to expose stealthy bugs in DNNs. The synthesized test cases intend to simulate different real-world circumstances. In order to bypass the test oracle problem [58], existing works often resort to differential/metamorphic testing techniques [20, 90]. They generally synthesize test cases by applying image transformation or taking advantage of Generative Adversarial Networks (GANs) [79]. Such efforts are conducive to enhancing the robustness of DNNs before deployment. Nevertheless, in a running DNN-based system, we still need to monitor the status of the system in case of any unexpected situation it cannot handle.

Our detection method is motivated by data validation. In software engineering, data validation seeks to protect the system by disallowing the entry of data that violate predefined validation rules [6, 24, 51, 116]. For example, web applications often turn to data validation to prevent input attacks like buffer overflow, SQL injection, and cross-site scripting [8, 106].

Concurrent with our work, Zhang *et al.* propose to validate the inputs of DNNs via VGGNet features [176], but these two frameworks are distinct. Zhang *et al.* only demonstrate the viability of their method to differentiate images under different weather conditions. Furthermore, they do not investigate the efficacy of their approach in mitigating model misbehavior. So their technique is merely aware of changing weather, no matter whether misclassification occurs, which generally has comparatively limited use in practice. Contrarily, we investigate the general misbehavior of DNN classifiers and propose an effective model validation mechanism to enable fail-safe operations in practice.

2.3 Intentional Failures of DNNs

Apart from accidental failures, DNNs are under the threat of intentional failures when attackers are present. The weapon of attackers is called adversarial samples, which are crafted by purposefully perturbing legitimate images in a human-imperceptible manner. An example of attaching some deliberately engineered

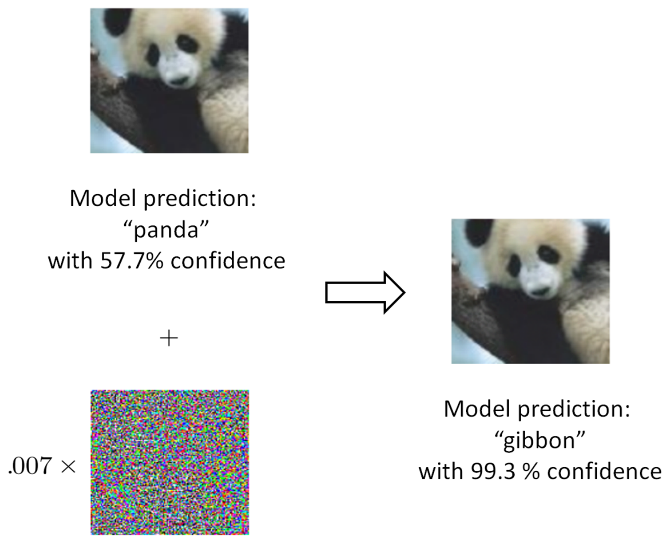


Figure 2.3: An adversarial image for GoogLeNet [144] (modified from the figure in [39]).

"noise" to a legitimate input image so that the model misclassifies the adversarial sample with high confidence is shown in Figure 2.3. Here the difference between the legitimate source image and the adversarial variant is hardly perceptible, while the model is too "sensitive" about the adversarial distortion to make a correct decision.

Therefore, adversarial attacks can serve as a critical surrogate to test the robustness of DNNs against intentional failures. In this section, we review two facets of evaluating the robustness of DNNs against intentional failures: synthesizing adversarial samples against undefended models and synthesizing adversarial samples against defended models.

2.3.1 Synthesizing Adversarial Samples Against Undefended Models

According to the knowledge of attackers, there are generally two categories of threat models in the literature [11]. One is white-box settings where attackers acquire full access to the victim model, for example, the model architecture and parameters. The other is black-box settings where adversaries only obtain query access, namely, image input uploading and prediction output downloading. Under both scenarios, attackers aim to synthesize adversarial samples to mislead learning algorithms by perturbing legitimate images in a human-unnoticeable manner.

Corresponding to the setting that they are tailored for, attacks are classified as white-box attacks and black-box ones [11]. The white-box attack enjoys great popularity among early work on attacking DNNs [18, 42, 72, 147]. Unlike the process of model training, white-box attacks feature an optimization in input space to elevate training loss. The fast gradient sign method (FGSM) alters clean seed images by taking one step along with the sign of the gradient of the model loss function [42]. Its successor, the basic iterative method (BIM), iteratively applies FGSM perturbations of smaller magnitude to improve attack success rates [72]. Projected gradient descent (PGD) extends BIM with a random start to diversify the synthesized adversarial instances [88]. The Carlini and Wagner attacks (C&W) devise a novel attack object to absorb the perturbation budget constraint [18], which also admits the employment of sophisticated optimizers like Adam [67] during the search for deceptive noises. The Jacobian-based Saliency Map Attack (JSMA) [109] is tailored for seeking the adversarial noise with the minimal l_0 norm. Therefore, it proposes to prioritize the modification of the most important image pixels to model decisions.

However, white-box attacks hardly reflect the threat to models in practice since only query access is allowed in most realistic cases. Therefore, black-box attacks have attracted increasing attention recently. There are roughly two sorts of black-box attacks according to the mechanism they adopt. One is query-based [10, 46, 108], and the other one is transfer-based [29, 81, 96, 167, 185].

Query-based black-box attacks can settle the susceptible direction of the victim model as per the response of the target model to given inputs [46]. Alternatively, attackers can approximate the loss gradient of the target model through training a local replica [108] or finite difference techniques [10]. However, such attacks usually require excessive queries before a successful trial and thus have limited applicability in practice [29].

Transfer-based black-box attacks are motivated by the transferability of adversarial samples across different models. Concretely, attackers first launch attacks on off-the-shelf local models to which they have white-box access. Then the deceptive samples are directly transferred to fool the remote victim model. Therefore, attackers can apply any white-box attack algorithm in this task, such as FGSM and BIM. Unfortunately, such a straightforward strategy frequently suffers from overfitting to specific weaknesses of local source models and manifesting limited success. We show that by introducing regularizers into the optimization process of adversarial samples, we can significantly improve the performance of such transfer-based black-box attacks.

There also exist two sorts of methods to promote adversarial transferability. Ensemble-based mechanisms often require the deduced distortion to remain harmful for an ensemble of models [81, 132] or images [29, 96, 167]. More related to our work is the regularization-based approach: transferable adversarial perturbation (TAP) introduced by [185]. TAP injects two regularization terms into the vanilla training loss function of the model to guide the search of adversarial manipulations, which alleviates the issue of vanishing gradient and reduces the variations of resultant adversarial samples. We discover that different models share similar attention when making correct predictions. Therefore, we can exploit this property to boost the transferability of malicious images.

2.3.2 Synthesizing Adversarial Samples Against Defended Models

Enormous efforts have been devoted to defending against adversarial samples, which generally fall into two axes. The first one is termed adversarial training, which remains the state-of-the-art defense to date [73,153]. Adversarial training works by injecting the generated adversarial samples into the training data to retrain the model [42]. Ensemble adversarial training is a refined successor of vanilla adversarial training [153], which employs the adversarial samples synthesized from hold-out models to augment the training data. As such, the adversarially trained models can showcase robustness against transfer-based attacks.

The second line of defenses proceeds by purifying the adversarial samples. Specifically, they pre-process the input images as a potential defense to rectify adversarial perturbations without reducing the classification accuracy on benign images. The state-of-the-art defenses of this kind include applying random resizing and padding [166], a high-level representation guided denoiser [77], randomized smoothing [22], an image compression module [63], and a JPEG-based defensive compression framework [82]. Although being straightforward, such defenses achieve amazing performance against prior attacks.

With the blooming progress of defenses, there is a demand to better evaluate the robustness of defended DNNs. In other words, we need to devise more effective transfer-based attacks so that the generated adversarial samples can achieve better transferability from undefended models to defended ones.

Prevailing solutions usually require the synthesized adversarial samples to remain malicious against certain image transformations that can preserve the image content, such as resizing [167], translation [29], and scaling [78]. However, these approaches bear the deficiency of only considering individual image transformations or their simple combination under fixed distortion strength. It makes the crafted adversarial samples overfit the applied image transformations and hardly survive under unknown distortions, which may lead to sub-par transferability [22, 63, 82].

In this chapter, we propose to address the defects of existing proposals. A straightforward remedy would involve first identifying a large corpus of image transformations that can retain the image content. Then it carefully tunes the combination of image transformations that is appropriate to each image. Unfortunately, such a process can incur prohibitive computational costs.

Inspired by the recent progress in performing image manipulations with convolutional neural networks [36, 85, 187], we propose to exploit a CNN-based adversarial transformation network to mitigate the issue of explicitly modeling the employed image transformations and automate the tuning process. Specifically, our Adversarial Transformation-enhanced Transfer Attack (ATTA) proceeds by training an adversarial transformation network to model the most harmful image transformations to adversarial noises by adversarial learning. Then we require the generated adversarial samples to additionally defeat the adversarial transformation network. As a result, our strategy can improve adversarial transferability against defended DNNs.

2.4 Interpretability of DNNs

In pursuit of improving the interpretability of deep image classifiers, attribution is the mainstream methodology in the literature [105]. Specifically, attribution aims to quantify the importance of human-readable units to model decisions [38,105]. Based on the unit to which it attributes model predictions, there are two attribution techniques: input and concept attribution.

Input attribution explains model behaviors in terms of the importance of different input pixels. The outcome of input attribution, termed as saliency maps, can highlight the most responsible parts of input images for model decisions. There is a vast body of work under this track, such as the gradient-based [105, 130, 133, 134, 136, 137, 141], structure-based [4, 95, 174, 183], proxy model-based [84, 122], and decision-based approaches [25, 27, 35, 123, 174, 179, 188].

Unfortunately, despite being intuitive, input attribution also suffers from confining itself to input space. The primary culprit is that the semantic meanings of image pixels are highly dependent on others and diverse. Consequently, the saliency maps returned by input attribution are subject to human perceptions before they become a human-readable interpretation. Unfortunately, human judgments are error-prone and can lead to contradicting conclusions [66].

Concept attribution, on the other hand, attempts to address this issue by directly measuring the importance of human-understandable concepts to model decisions. It affords two interpretation interfaces: local explanations that work for individual predictions [105, 184] and global explanations that apply for a whole category of examples [38, 66].

Both lines of concept attribution overwhelmingly follow an implicit two-stage procedure. They first conduct feature attribution to derive feature importance, and then translate it to concept importance to accomplish concept attribution. In the feature attribution step, prior schemes coincidentally employ backpropagated gradients to estimate the importance of individual features to a class (the feature importance vector). In the concept attribution phase, they usually exploit concept classification to derive the embedding of a concept in hidden layers of CNNs (the concept vector). Such a concept vector denotes the combinations of feature filters that can best detect the concept. They then project the feature importance vector along the direction of the concept vector to gauge the importance of the corresponding concept to model decisions [38, 66, 184]. For a global explanation, they simply run the above routine for individual samples in isolation and report the average concept importance [38, 66].

Our concept attribution framework defeats the pitfalls of both local approximations of models and separate investigations of samples in existing global explanation approaches. During feature attribution, we devise a novel feature occlusion analysis. It abandons local model approximations, and learns a global interpretation that considers the extensive connections among samples of the same class in model cognition. Motivated by the prior work [7, 34, 66, 97, 184], our concept attribution scheme directly combines feature filters as per their importance. Then we estimate the combined filters' representation capacity of a concept of interest to measure concept importance. Consequently, compared with current attempts, our concept attribution procedure is more general, which also offers the opportunity to integrate prior model visualization techniques [97, 182] into concept attribution.

Like ours, a few other efforts aim to overcome the above shortcomings in existing global explanation methods [44, 178]. However, they possess less general applicability than ours. [44] proposes to perform a direct concept occlusion analysis, whereas it assumes access to the generation process of natural images for given concepts. [178] counts on an inherently more interpretable model, where each feature filter independently and exclusively responds to one concept. In contrast, our technology widely applies to post-training CNN image classifiers, without the need for the data generation mechanism or model modification.

Notably, there is a growing spectrum of efforts on bridging the robustness and interpretability of DNNs. Some focus on the contribution of the robustness of DNNs to their interpretability. [65, 155] are the pioneering work of this kind. They reveal that adversarial training (AT) can produce more human-readable models, although it is primarily designed to enhance the robustness of DNNs via augmenting their training data with adversarial samples.

The others are devoted to demonstrating the opposite direction. That is, better interpretability of DNNs can help to analyze and boost their robustness [37, 163]. [37] examines the relative importance of textures and shapes to models' decisions by measuring the models' performance on samples with texture-shape cue conflicts. Experimental results unearth that ImageNet-trained DNNs intensively hinge on textures for classification, which is an unfavorable bias that may incur robustness issues. Therefore, [37] introduces Stylized-ImageNet (SIN) to modulate the training of DNNs towards learning more shape-based models. As a side effect, the resultant models become robust against a broad array of image distortions. Our AfI is also related to this line of research, since we endeavor to offer more accurate explanations of models' behaviors [163]. Consequently, our framework can serve as a toolkit to uncover potential robustness issues of DNNs and spur the design of more robust models.

□ **End of chapter.**

Chapter 3

Detecting Real-world Corner Cases for Deep Neural Networks

The exceptional performance of Deep neural networks (DNNs) encourages their deployment in safety- and dependability-critical systems. However, DNNs often demonstrate erroneous behaviors in real-world corner cases. Existing countermeasures center on improving the testing and bug-fixing practice. Unfortunately, building a bug-free DNN-based system is almost impossible currently due to its black-box nature, so anomaly detection is imperative in practice.

Motivated by the idea of data validation in a traditional program, we propose and implement *Deep Validation*, a novel framework for detecting real-world error-inducing corner cases in a DNN-based system during runtime. We model the specifications of DNNs by resorting to their training data and cast checking input validity of DNNs as the problem of discrepancy estimation. Deep Validation achieves excellent detection results against various corner case scenarios across three popular datasets. Consequently, Deep Validation greatly complements existing efforts and is a crucial step toward building safe and dependable DNN-based systems.

3.1 Problem and Motivation

Deep neural networks (DNNs), as emerging machine learning techniques, have achieved excellent performance on diverse tasks, such as image classification [48, 59, 70], machine translation [47, 142, 157], and text analysis [26, 52, 91]. These advances have facilitated the application of DNNs in a growing spectrum of safety- and dependability-critical domains, like self-driving [149,169], biometric authentication [80,127], and medical diagnosis [83, 158].

Despite the impressive capacities, researchers recently uncover a prominent issue in the context of image classification that these top performers often exhibit unexpected behaviors facing unforeseen real-world corner cases. For example, a Tesla car in Autopilot mode failed to identify a trailer against a bright sky [138], and an autonomous Uber vehicle misclassified a pedestrian during road-test at night [154], both resulting in deadly accidents. In essence, real-world corner cases are naturally transformed images that will not harm human perception [112, 150]. Therefore, it is dangerous to trust the prediction of a DNN classifier when developers do not include similar scenes in its training data. We focus on coping with such accidental failures of DNN classifiers in this chapter.

Existing solutions to harden DNNs against such flaws mainly follow the idea of model retraining with data augmentation [32, 70, 112, 113, 150]. They assume that the DNN classifier has never seen these difficult corner cases before, and retraining with known corner cases can contribute to a more knowledgeable model. Unfortunately, real-world scenes can vary with many factors, like brightness, camera alignment, and object movements. Hence the training data we possess are just a relatively small fraction of all scenarios in practice. Beyond that, it is also doubtful whether there exists a perfect DNN classifier that can handle all possible images in light of the "no free lunch" theorem [156, 160]. Such methods are also notorious for their painful bug-fixing process since it is computationally intensive to train DNNs, which usually contain millions of parameters.

Therefore, corner case detection should be an indispensable safety tool when deploying DNNs in real-world systems. This kind of anomaly detection is a fundamental building block in many fail-safe systems. It is employed to foresee possible risks, which enables human intervention to correct system errors. However, to our best knowledge, there is *no existing detection method* in the literature tailored for addressing such real-world error-inducing corner cases in DNNs.

We believe a meaningful corner case detector should be *scenario-agnostic*. It is discouraged to build a detector upon known anomalies, which may inherit similar drawbacks of model retraining. We consider that the error-inducing corner cases come from distributions that the classifier has not yet learned to settle. Motivated by the idea of data validation in traditional software engineering [6,24,51,116], we hence infer error-inducing corner cases are out of range of the valid input domain of a DNN model and propose *Deep Validation*. It proceeds by validating intermediate model inputs/states to identify invalid examples that may lead to misbehaviors of the whole system. As a result, the detector is favorably model-dependent rather than scenario-dependent.

However, adapting data validation to validate the input of DNNs is non-trivial. Data validation within a traditional program often resorts to specific validation constraints, because the logic of a program is manually defined and can be expressed as succinct control flow statements. In contrast, a similar validation process appears infeasible for a DNN model due to the distinct design philosophy. The functionality of a DNN model is indeed learned from a large amount of training data spontaneously without much human supervision. Therefore, its knowledge is encoded in millions of indecipherable parameters as well as the associated intricate network structures [39, 103].

The methodology to explore these challenges is as follows. We first justify why we can instead look for support from the training data for the declaration of valid inputs. We then show how to model the valid input range of intermediate layers within DNNs through characterizing reference distributions. We follow by introducing our approach to quantify the validity of input images by estimating their discrepancy to the valid input region. We finally provide extensive experiments to corroborate the high effectiveness and great superiority of our framework in detecting real-world corner cases. Along with other merits we demonstrate, Deep Validation conduces to promoting the safety and dependability of DNN-based systems.

In summary, the main contributions of this chapter are:

- We introduce Deep Validation as the *first framework* to automatically validate internal inputs/states and to identify error-inducing real-world corner cases for a working DNN-based system. It monitors the deviation from the normal functionality of internal components within a DNN and makes sure this black-box system works correctly. As such, Deep Validation contributes to further improving the safety and dependability of a DNN-based system by enabling fail-safe solutions.
- We conduct extensive experiments across various datasets and DNN architectures to evaluate our framework. Deep Validation consistently reports prominent results on eight different categories of corner cases with an overall ROC-AUC score of 0.9937 on MNIST, 0.9805 on CIFAR-10, and 0.9506 on SVHN, respectively. The superior performance of Deep Validation also breaks the unexplored belief that detection methods tailored for intentional attacks can also work well facing real-world corner cases.
- We investigate the efficacy of Deep Validation on defending against numerous cutting-edge white-box attacks. It also achieves impressive performance with an overall ROC-AUC score of 0.9572 and 0.9755 in two settings, respectively. Both are competitive with state-of-the-art results. We further test Deep Validation under high dynamical range working conditions. It confirms that Deep Validation is sensitive to impending dangers with consistently satisfactory detection rates.

3.2 Preliminaries

The organization of this section is as follows. In order to precisely characterize the real-world corner cases that attract substantial interest in the community and disclose their harmfulness, we first elaborate on the system model and fault model of DNN classifiers considered in this chapter [112, 113, 150]. We then present a categorization of representative techniques for adversarial image detection, which makes it convenient to determine the state-of-the-art benchmark methods for better comparisons.

3.2.1 System Model of DNN Classifiers

In the field of image classification, a customized DNN structure, convolutional neural network (CNN), is embraced as the canonical solution [56,75]. CNN classifiers often stack numerous simple components (namely, layers of neurons) with non-linear activation. These simple components collaborate to extract increasingly abstract features automatically [54, 100, 101, 174]. As a whole, they can learn the mapping between raw input images and a predefined set of labels (namely, classes of images) by mere end-to-end supervision.

We now set up some notations. We can regard a CNN as a function $f : X \to Y$. Here X denotes the input space, and Y is the output space representing a predefined categorical set $\{1, ..., N\}$. In a conventional CNN, neurons in each layer are connected to the ones in the succeeding layer by weighted edges. We can thus regard the i^{th} layer (with $i \in \{1, ..., L\}$) as a function $f_i(f_{i-1};\theta_i)$. It is controlled through parameters θ_i and takes as inputs the outputs from the former layer f_{i-1}.

Consequently, we can formulate the link between input images and label predictions delineated by a CNN as a composition of L parametric functions:

$$f(\mathbf{x};\theta) = f_L(f_{L-1}(...f_1(\mathbf{x};\theta_1);\theta_{L-1});\theta_L). \tag{3.1}$$

Here $\mathbf{x}$ represents the vectorized pixel values of raw images, which is considerably high-dimensional. The parameter set $\theta := \{\theta_1, \theta_2, ..., \theta_L\}$ hence encodes the model knowledge learned from the training data. When it is clear from the context, we may take $f_i(\mathbf{x})$ or f_i as shorthand for the output of the i^{th} layer for the input image $\mathbf{x}$.

We usually translate the final output of a CNN classifier as a probability vector $f(\mathbf{x})$, where the k^{th} entry $f(\mathbf{x})[k]$ stands for the confidence of the model on categorizing image $\mathbf{x}$ as class k. Because the last layer is a softmax layer, we can see the final output $f(\mathbf{x})$ as fitting a logistic regression on the logit outputs z_k ($k \in \{1, ..., N\}$) of the penultimate layer.

3.2.2 Fault Model of DNN Classifiers

We adopt a behavioral-level fault model to specify the fault type of DNN classifiers covered in this chapter [14, 68, 114]. It clarifies the real-world corner cases with which we are concerned. Therefore, the fault model can not only guide the selection of strategies for corner case generation but also profile the testbed for evaluating anomaly detection methods.

As introduced before, a DNN classifier merely possesses a limited capacity, and hence it often misbehaves in the presence of unfamiliar images. In contrast with the training data of a DNN classifier, these error-inducing samples bear distinct characteristics. They usually arise out of unexpected, especially dramatic changes in the working environment, like the variations in illumination, which is why they are dubbed real-world corner cases in the literature [112, 113, 138, 150, 154].

Therefore, in order to simulate variable working conditions a DNN-based system may face, we follow the idea of metamorphic testing to generate real-world corner cases [20]. Specifically, we convert available normal images into real-world corner cases by applying naturally occurring image transformations *without* destroying their original semantic meanings. For example, utilizing image rotation, we can craft images perceived by a DNN-based system when its camera deviates from the original position.

Although the defects of computing infrastructures can thwart the normal functionality of a DNN-based system, the misuse of DNN classifiers is one main culprit of their failures in practice, which remains challenging to alleviate [30, 32, 57, 102, 112, 113, 150, 176]. We believe that error-inducing corner cases are not born of the same distribution that the DNN classifier has grasped. Hence identifying these abnormal inputs beforehand can prevent the misuse of DNN classifiers.

3.3 Methodology

3.3.1 Corner Case Generation

In line with the fault model we account for, we now detail the metamorphic testing technique employed to synthesize real-world corner cases, which can simulate unexpected changes in the working conditions of a DNN-based system. In particular, we will introduce the image transformations and search strategy exploited.

Image Transformations

There is a growing body of research on DNN testing that applies metamorphic testing techniques to enrich their test cases [31,32,57,112,113,150,176]. They find that classical image transformations, like rotation, are capable of effectively mimicking changing working environments and exposing erroneous behaviors of DNNs in practice [32, 57, 112, 113, 150].

Note that as one cannot cover all types of corner cases, we employ well-recognized transformations (brightness adjustment, contrast adjustment, rotation, shear, scale, and translation) to synthesize corner cases following [150]. Compared to other transformations, our preliminary study also confirms that selected ones can generate effective real-world corner cases and reserve original labels of images through controllable parameters, avoiding introducing problematic test cases.

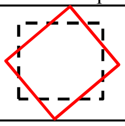
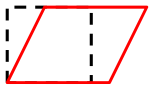
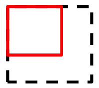
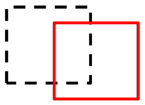
We now detail each transformation. Image brightness is a measure of color intensity. Therefore, to change the brightness of an image, we can increase or reduce all the current pixel values by a constant bias β. The contrast of an image, as its name implies, is determined by the amount of color and luminance differentiation that exists between various objects in the image. We can manipulate the contrast of an image through multiplying all the current pixel values by a constant gain α. The brightness and contrast of an image can frequently change in practice due to the variation of illumination.

Rotation, shear, scale, and translation compose the whole set of affine transformations, which is common geometric deformation that happens to captured images due to perspective irregularity. We leverage them to simulate the distorted appearance of an object resulting from varying camera positions and the shift of this object, which can frequently occur in the real world as well.

Using homogeneous coordinates can formulate affine transformations succinctly. In homogeneous coordinates, we first extend the original coordinates of a two-dimension image $\mathbf{I} = (a, b)$ into homogeneous coordinates $\mathbf{I} = (a, b, 1)$ with three dimensions. Then the coordinates of an affinely transformed image $\mathbf{I}^{'} = (a^{'}, b^{'}, 1)$ are merely the dot product of the corresponding transformation matrix $\mathbf{T}$ and $\mathbf{I}$ (*i.e.,* $\mathbf{I}^{'} = \mathbf{T} \cdot \mathbf{I}$). We list the transformation matrices of four kinds of affine transformation in Table 3.1.

Complement is a different type of image transformation that flips all the pixel values of an image. For example, in the complement of a binary image, black and white are reversed. For greyscale images, their complements are still clearly distinguishable and familiar to human observers, whereas the

Table 3.1: Transformation matrices of affine transformations covered in this chapter.

Affine Transformation	Transformation Matrix	Parameters	Example
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 1 \\ 0 & 0 & 1 \end{bmatrix}$	θ : the rotation angle	
Shear	$\begin{bmatrix} 1 & s_h & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_h : shear ratio along x axis s_v : shear ratio along y axis	
Scale	$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_x : scale ratio along x axis s_y : scale ratio along y axis	
Translation	$\begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$	T_x : shift along x axis T_y : shift along y axis	

complements of color images look peculiar and are unlikely to appear in reality. Therefore, we only apply this transformation to greyscale images. Since the model never sees the complement of an image during training, it can be regarded as a kind of corner cases as well.

Search Strategy

As introduced before, all the image transformations we covered except for complement can be parameterized with different strengths, so the first question is how to determine the most suitable parameters for them. A small degree of deformation is scarcely sufficient to reproduce real-world corner cases and disclose the vulnerability of DNNs. On the other hand, too much distortion may compromise the semantic meanings of seed images and render them unrecognizable. We resolve this issue by grid search in a trial-and-error fashion.

The grid search proceeds as follows. For the sake of exploring a variety of real-world corner cases, we start by applying single transformations to each seed image in turn. We then follow by performing the combination of different image transformations. To decide the parameter for single transformations, we apply a transformation to a fixed set of clean test images with growing distortion strength iteratively. During the search, we also monitor whether the altered images preserve their original semantic meanings. The search stops when the average accuracy of the model on the transformed image set starts to drop by a notable margin. We take it as a sign that the DNN classifier is unfamiliar with the distorted test images and working in trouble. We hold the resultant synthetic test images for the following experiments.

We also consider the combination of image transformations. Combining these transformations also contributes to discovering new corner cases. However, it is computationally prohibitive to explore all combinations exhaustively. Therefore we mainly consider the combination of two transformations. The idea of figuring out the most suitable parameters here is similar to that discussed before. Small modification according to empirical observation is explained in Section 3.4.2.

Although the most suitable parameter choices rely on subjective judgments, we also collect error-inducing samples via transformations with a broader range of parameters. We evaluate Deep Validation under this dynamic setting in Section 3.4.4.

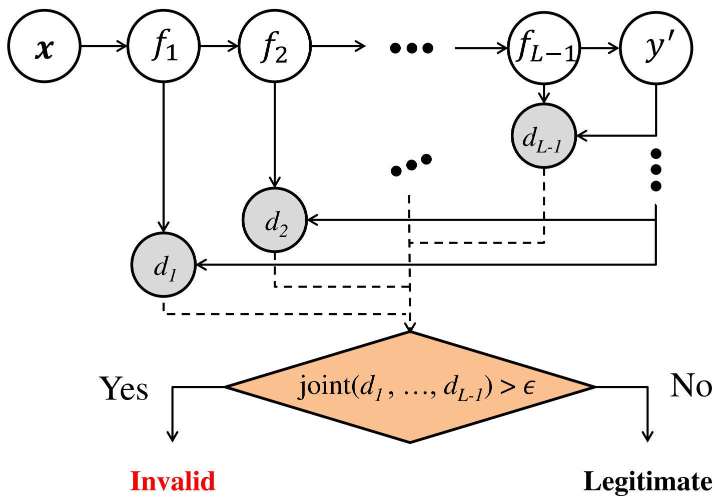


Figure 3.1: Deep Validation framework: f_i is short-hand for the output of the i^{th} hidden layer where $i \in \{1, 2, ..., L-1\}$. y' means the predicted label for input $\mathbf{x}$. d_i is the discrepancy estimation for the output of the i^{th} hidden layer.

3.3.2 Deep Validation

Figure 3.1 outlines the framework of Deep Validation. In a word, we take a trained DNN model and add probes into every layer internal to it. During inference, rather than taking the classifier as a black-box oracle and trusting its final decisions blindly, we validate the internal states of the model, namely, outputs of the layer i through discrepancy estimation d_i. We quantify the validity of an input image by its joint discrepancy. Once the total discrepancy exceeds a preset threshold ϵ, we will mark the test image as invalid input, namely, corner cases that may lead to unexpected behaviors of the system. In this way, we seek to ascertain that the model functions correctly and that there is no sign of tampering.

More specifically, we make sure that the internal states of a running DNN classifier follow consistent patterns observed from the training samples when making the same prediction. Otherwise, it is risky to accept its decisions due to rare training samples similar to those it confronts.

We present our framework as follows. We at first justify the fundamental idea of Deep Validation that validating the internal states of a model assists in alleviating its misbehavior. Then we show how to turn to the training data to quantify the validity of inputs via computing discrepancy, which estimates its distance to the region where the probability density of training data resides.

Justification

Our motivation comes from the idea of data validation in traditional software engineering. In a traditional program, complex tasks are usually divided into smaller ones and conquered by individual modules separately. We can hence assign these subtasks to different developers. In order to make these modules work together seamlessly, it is a good practice to work out a detailed specification first. The specification of a module generally elaborates on what tasks it is supposed to complete, the expected inputs and outputs, and so on. Data validation is a beneficial way to guarantee that every component within a large program abides by the respective specification and functions correctly during collaboration [6, 8, 24, 51, 106, 116].

Similarly, as introduced in Section 3.2, a DNN classifier can also be regarded as a sophisticated program. Its layers are small components that summarize the raw input image into increasingly abstract forms, which are usually called the hidden representations of the image. Recall that each layer can only perform a specific simple computation and owns relatively much limited capability. It is thus apparent that each layer has an input domain where it has learned to work well. We can regard this region as its valid input range.

Therefore, we propose to validate the input of each layer, which is favorable. Under normal circumstances, since the test images follow the same distribution of the training data, the raw images are mapped into valid input space of each inner layer sequentially, and all layers cooperate in harmony as a whole. Therefore it is enough for images to escape from the "familiar" input region of any middle layer to crack the whole system. Worse still, due to the high dimensionality of input space, small unsafe perturbations can be exacerbated when pushed forward along layers [42, 107]. As a result, the contaminated outputs of internal layers can gradually deviate from the regular input areas of succeeding layers and bring about false predictions in the end. Therefore, validating the input/state of all middle layers is conducive to spotting abnormal images that the DNN classifier has not yet learned to handle and preventing the misuse of each component layer.

Discrepancy Estimation

However, it is challenging to adapt data validation for DNNs. In traditional programs, we can develop data validation routines in line with detailed program specifications. Nevertheless, the obstacle in the context of DNN models is that the legitimate input range for every layer is ill-defined. It is because the decision functions of these layers are learned on their own rather than manually designed by the developers. Moreover, the classification rules they derive from the training data are encoded in millions of parameters, which are nearly impossible to translate.

We instead circumvent this difficulty by backtracking the training data. Since the model has learned to work well on training samples, their hidden representations are supposed to outline the valid working areas of corresponding layers. These legitimate regions, however, can still be too complicated to depict. We hence further decompose the valid input domain of each layer according to different image classes.

Our solution to capture the valid working areas of each layer is motivated by the following observations. Recall that in order to tell different categories of images apart, each layer within a DNN should carefully abandon redundant features and retain discriminative ones until only the label information survives in the end. Consequently, images of different classes can fire different patterns and follow different paths when transferred from one area into another one when going through layers.

Based on this observation, Deep Validation proceeds by validating the following claim for each intermediate output of a test image:

Whether the output stays near the region where the corresponding hidden representations of training images with the same label are concentrated.

Whenever there are considerable discrepancies, the prediction for the test image is no longer credible. Besides, this abnormal input is likely to be an error-inducing corner case, because some components within the DNN classifier are compelled to extrapolate in unknown regions, or the input has gone through a strange mapping route.

As a result, the algorithm of Deep Validation proceeds as follows. We begin by portraying the regions where training images of different categories locate layer by layer. We then mark them as reference distributions. When a new test image comes, we evaluate its divergence regarding the corresponding reference distributions to determine whether the test image is legitimate.

We build our discrepancy estimation on the method proposed by Schölkopf *et al.* [128] to efficiently model the reference distributions. Roughly speaking, their method aims to locate the separating hyperplane of training points after casting them into kernel space. We leverage this approach to capture the region where the probability density of training images resides. Specifically, for each reference distribution, we train a one-class support vector machine (SVM) on the corresponding set of hidden representations of training images. We follow their idea to learn the decision function through requiring its values to be non-negative on small input regions where most of the training data spread while negative otherwise.

After that, we approximate the validity of a given sample by calculating its signed distance to the learned supporting hyperplane in kernel space. By doing so, we circumvent the difficulty in depicting data whose underlying distribution is too elusive to express explicitly. On top of that, since we simplify the reference distribution through a careful segmentation of training samples, it helps the training of one-class SVM to scale well to

Algorithm 1 One-class SVM Training

```
Require: CNN classifier $f$, layer number $L$, class number $N$, and training
    data set $X_{train}$
 1: // obtain correctly classified images from the training dataset
 2: $X_{train} \leftarrow \{\mathbf{x}^{(t)} \in X_{train} : y^{(t)} == f(\mathbf{x}^{(t)})\}$
 3: for layer $i \in \{1, ..., L-1\}$ do
 4:     for class $k \in \{1, ..., N\}$ do
 5:         // select corresponding training data
 6:         $X^k \leftarrow \{\mathbf{x}^{(t)} \in X_{train} : y^{(t)} == k\}$
 7:         // get hidden representations in a specific layer
 8:         $X_i^k \leftarrow \{f_i(\mathbf{x}^{(t)}) : \mathbf{x}^{(t)} \in X^k\}$
 9:         // train one-class SVM$(i, k)$
10:         SVMTRAIN$(X_i^k)$
11:     end for
12: end for
```

high-dimension data.

Algorithm 1 elucidates the procedure for training these one-class SVMs. In simple terms, we begin by removing training images misclassified by the model, since they are likely to be outliers and will harm the training of SVMs. Then in each layer except for the last one, we get the hidden representations of all the training images and group them based on their original labels. Each subset of these points is applied to fit one SVM. Therefore, we conduct SVM training repeatedly for every class in every layer. The training procedure $SVMTRAIN$ is based on the implementation of the algorithm of Schölkopf *et al.* in the scikit-learn library [13, 111]. We denote the final distance function of SVM(i, k), namely, the signed distance of input to its decision hyperplane, as t_i^k. Here SVM(i, k) is the SVM trained with hidden features of images from class k in layer i.

Algorithm 2 describes the routine to estimate the discrepancy of a test image $\mathbf{x}_{test}$. In order to obtain the discrepancy

Algorithm 2 Discrepancy Estimation

```
Require: CNN classifier $f$, layer number $L$, and test image $\mathbf{x}_{test}$
1: $y^{'} \leftarrow f(\mathbf{x}_{test})$
2: for layer $i \in \{1, ..., L-1\}$ do
3:     // compute discrepancy $d_i$ for layer $i$
4:     $d_i \leftarrow DISCREPANCY(y^{'}, f_i(\mathbf{x}_{test}))$
5: end for
6: // evaluate joint discrepancy $d$ for the test image
7: $d \leftarrow joint\ (d_1, d_2, ..., d_{L-1})$
```

estimation for the intermediate output $f_i(\mathbf{x}_{test})$ of the test image in the i^{th} layer, we first obtain its label prediction $y^{'}$ to index the reference SVM $(i, y^{'})$. Next, we feed $f_i(\mathbf{x}_{test})$ to the corresponding SVM distance function and directly define the opposite as the discrepancy value.

Definition 3.3.1. (Discrepancy in the i^{th} layer). The discrepancy $DISCREPANCY(y^{'}, f_i(\mathbf{x}_{test}))$ of a test image $\mathbf{x}_{test}$ in the i^{th} layer is defined as follows.

$$DISCREPANCY(y^{'}, f_i(\mathbf{x}_{test})) = -\ t^{k}_{y'}(f_i(\mathbf{x}_{test})). \tag{3.2}$$

It is because we want to have positive discrepancy values for outliers and negative ones otherwise. Finally, we define the total discrepancy d as the unweighted sum of discrepancy estimations of all layers.

Definition 3.3.2. (Total discrepancy). The total discrepancy d of a test image $\mathbf{x}_{test}$ is:

$$d = joint(d_1, d_2, ..., d_{L-1}) = \sum_{i=1}^{L-1} d_i. \tag{3.3}$$

This simple joint function turns out to work well. Still, we can further explore it since a better combination can lead to a

Table 3.2: Model architecture for SVHN.

Layer Type	Parameters
Convolution + ReLU	64 filters (3 × 3)
Convolution + ReLU + Max Pooling(2 × 2)	64 filters (3 × 3)
Convolution + ReLU	128 filters (3 × 3)
Convolution + ReLU + Max Pooling(2 × 2)	128 filters (3 × 3)
Fully Connected + ReLU	256
Fully Connected + ReLU	256
Softmax	10

Table 3.3: Model accuracy on test data.

Dataset	Accuracy on Test Data	Mean Top-1 Prediction Confidence
MNIST	0.9943	0.9979
CIFAR-10	0.9484	0.9456
SVHN	0.9223	0.9878

more precise estimation.

3.4 Experiments

3.4.1 Experimental Setup

We consider three standard datasets for image classification: MNIST [76], CIFAR-10 [69], and SVHN [99]. We note that SVHN is a relatively "noisy" dataset without much data pre-processing effort in advance. It has two formats, of which we utilize the one made up of 32-by-32 color images with cropped digits. We engage the standard training-test partitions of all these datasets.

We now describe the models in our experiments. We utilize pre-trained models for MNIST and CIFAR-10 for the sake of fair comparisons in the following experiments. The MNIST

Table 3.4: Transformations and search space utilized when synthesizing corner cases.

Transformation	Parameter	Parameter Range and Search Step
Brightness	bias β	0 through 0.95, step 0.004
Contrast	gain α	0 through 5.0, step 0.1
Rotation	rotation angle θ	1° through 70°, step 1°
Shear	shear vector (s_h, s_v)	(0, 0) through (0.5, 0.5), step (0.1, 0.1)
Scale	scale vector (s_x, s_y)	(1, 1) through (0.4, 0.4), step (0.1, 0.1)
Translation	translation vector (T_x, T_y)	(0, 0) through (18, 18), step (1, 1)
Complement	maximum pixel value 1.0	-

model is a seven-layer CNN [171], and the CIFAR-10 model is DenseNet [59,89], which has 40 layers in total. We train a seven-layer CNN for SVHN, and Table 3.2 summarizes its architecture. We adopt an Adadelta optimizer [173] during training, with an initial learning rate of 1.0 and a decay factor of 0.95. We train the model for 60 epochs with a batch size of 128. We do not apply any data augmentation during training. Table 3.3 presents the mean accuracy and prediction confidence of these models on test datasets. Their performances are all comparable to the state-of-the-art results [9].

3.4.2 Corner Case Generation

The procedure to generate real-world corner cases is as follows. We fix a clean seed image set of 200 images for each model. They are randomly sampled from the corresponding test dataset. We make sure that all get correctly classified before any modification. These seed images are leveraged to synthesize corner cases according to the strategies introduced in Section 3.3.1. Table 3.4 lists the search range and step size for each transformation.

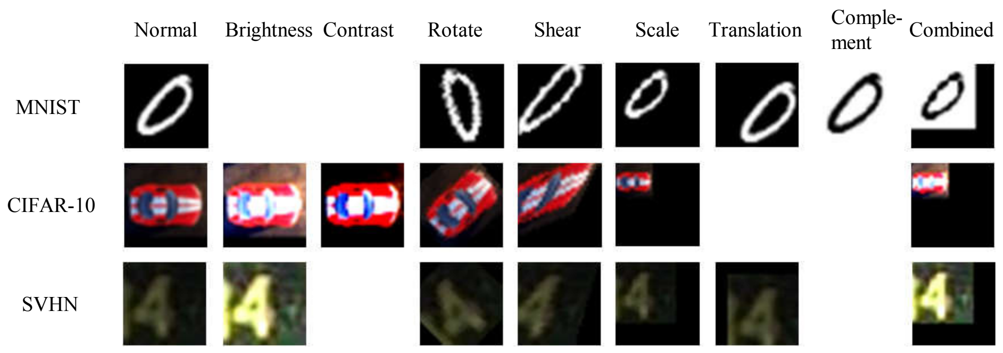


Figure 3.2: Examples of synthetic corner cases.

We note that we only alter the seed images from MNIST by complement since the other datasets are all color images.

We search for suitable transformation strength as follows. At each iteration, we evaluate the *accuracy* of the target classifier on synthetic images and define the success rate of each configuration as $1 - accuracy$. As we expect, different transformations have different destructive power in different datasets. Some transformations degrade the accuracy of target classifiers quickly with increasing distortion, while others fail to convert legitimate samples into error-inducing counterparts until they become hardly discernible. For individual transformation, the search stops when it obtains a success rate of about 60%. In the following experiments, we do not consider transformations that cannot achieve a success rate of greater than 30% in the end. Combined transformations are mainly meant to enrich the corner cases, and therefore we directly utilize the final parameters above to parametrize component transformations. We select one transformation combination for each dataset that results in the smallest deformation because it can preserve the semantic meaning of images and test the sensitivity of detectors.

Table 3.5: Success rates of different kinds of corner cases.

Dataset	Transformation	Configuration	Success Rate	Mean Top-1 Prediction Confidence
MNIST	Brightness	-	-	-
	Contrast	-	-	-
	Rotation	$\theta = 50°$	0.62	0.8854
	Shear	$(s_h, s_v) = (0.2, 0.3)$	0.64	0.8491
	Scale	$(s_x, s_y) = (0.8, 0.8)$	0.665	0.9031
	Translation	$(T_x, T_y) = (4, 3)$	0.625	0.9118
	Complement	maximum pixel value 1.0	0.53	0.8507
	Combined Transformations	complement combined with scale	0.87	0.8645
CIFAR-10	Brightness	$\beta = 0.51$	0.625	0.7213
	Contrast	$\alpha = 4$	0.585	0.7405
	Rotation	$\theta = 40°$	0.655	0.7782
	Shear	$(s_h, s_v) = (0.5, 0.4)$	0.59	0.7325
	Scale	$(s_x, s_y) = (0.6, 0.6)$	0.585	0.4608
	Translation	-	-	-
	Complement	-	-	-
	Combined Transformations	brightness adjustment combined with scale	0.855	0.3891
SVHN	Brightness	$\beta = 0.49$	0.575	0.8315
	Contrast	-	-	-
	Rotation	$\theta = 44°$	0.615	0.9396
	Shear	$(s_h, s_v) = (0.3, 0)$	0.605	0.9362
	Scale	$(s_x, s_y) = (0.7, 0.7)$	0.715	0.9524
	Translation	$(T_x, T_y) = (5, 5)$	0.63	0.9405
	Complement	-	-	-
	Combined Transformations	brightness adjustment combined with scale	0.865	0.9731

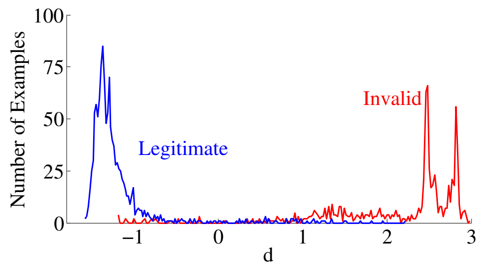
Table 3.5 lists the success rates of all settings along with the final parameters we employ. Figure 3.2 illustrates some examples of resultant corner cases. No images are given for transformations with less than a 30% success rate. Although our target models obtain exceptional accuracy on clean test data, they are susceptible to these unusual corner cases and undesirably show high confidence in their wrong predictions. It once again reveals the serious threat real-world corner cases can pose.

3.4.3 One-class SVM Training

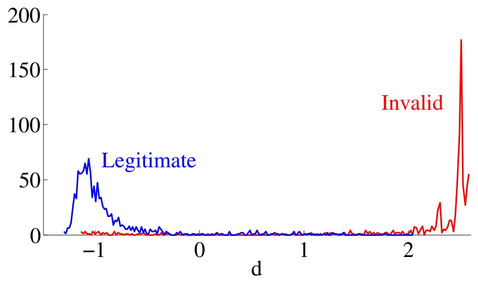
The training of one-class SVMs proceeds as follows. We note that training one-class SVMs merely utilizes the clean training data introduced in Section 3.4.1. To facilitate selecting SVM parameters, we leave out 1000 examples as validation data from each training dataset. We apply the same training parameter for all the SVMs within the same layer (*i.e.,* SVM$(i, k) : k \in \{1, ..., N\}$). Nevertheless, they vary from layer to layer since the intermediate outputs from different layers are usually of substantially different dimensions.

We note that the overhead of the proposed framework is low, because it is much cheaper to train one-class SVMs than to train DNNs. As the hidden representations of input images are already available when running DNN systems, querying SVMs also incurs negligible costs. Besides, the training and validation pipeline can be parallelized based on our design.

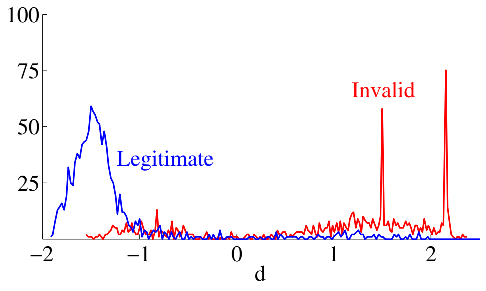
We make some adjustments on CIFAR-10 when implementing our framework. The target model for CIFAR-10, DenseNet, contains 40 layers. Therefore it takes much more time if we train SVMs for all layers. Thanks to the dense inter-connections between layers, it is convenient for the latter layers to receive the outputs from the former ones directly. Accordingly, errors that happen in the early layers can also smoothly propagate to the latter ones, which means that it may be enough to validate the inputs of the rear layers. Based on this observation, Deep Validation only works on the last six layers of DenseNet. However, we envisage that validating internal inputs of all layers can make further improvements.



(a) MNIST



(b) CIFAR-10



(c) SVHN

Figure 3.3: Discrepancy distributions of legitimate images and invalid ones (successful corner cases). Each plot is based on 200 histogram bins and fitted over discrepancy estimations for the corresponding evaluation dataset.

3.4.4 Corner Case Detection

Evaluation Dataset

We curate the evaluation dataset as follows. As shown in Table 3.5, we have six kinds of successful corner cases for each dataset. Therefore, for each target classifier, there are 1200 synthetic corner cases. We sample the same number of images from the corresponding clean test dataset. They together compose the evaluation dataset. According to whether these corner cases get misclassified or not, we further group them into successful corner cases (SCCs) and failed corner cases (FCCs).

Table 3.6: ROC-AUC scores of Deep Validation.

Configuration		Transformation Method Used to Synthesize Corner Cases								Overall ROC-AUC Score
Validator	Layer No.	Brightness	Contrast	Rotation	Shear	Scale	Translation	Complement	Combined Transformations	
Single Validator	1	-	-	0.8760	0.9987	0.8827	0.8952	**1.0000**	**1.0000**	0.9440
	2	-	-	0.9200	0.9719	0.8048	0.8893	**1.0000**	0.9996	0.9324
	3	-	-	0.9741	0.9797	0.9591	0.9728	0.9850	0.9197	0.9618
	4	-	-	0.9740	0.9823	0.9224	0.9657	0.9876	0.9670	0.9657
	5	-	-	0.9732	0.9788	0.9053	0.9602	0.9861	0.9630	0.9601
	6	-	-	0.9659	0.9889	0.9237	0.9620	0.9871	0.9786	0.9676
Best Transformation-specific Single Validator		-	-	0.9741	0.9987	0.9591	0.9728	**1.0000**	**1.0000**	0.9676
Joint Validator		-	-	**0.9891**	**0.9991**	**0.9881**	**0.9844**	**1.0000**	**1.0000**	**0.9937**

(a) MNIST

Configuration		Transformation Method Used to Synthesize Corner Cases								Overall ROC-AUC Score
Validator	Layer No.	Brightness	Contrast	Rotation	Shear	Scale	Translation	Complement	Combined Transformations	
Single Validator	34	0.7615	0.7579	0.8154	0.8828	0.9670	-	-	0.9752	0.8669
	35	0.5119	0.8749	0.9175	0.8765	0.9339	-	-	0.9099	0.8412
	36	0.7111	0.8230	0.8821	0.8557	0.9722	-	-	0.9497	0.8706
	37	0.8958	0.9664	0.9109	0.9315	0.9988	-	-	0.9993	0.9528
	38	**0.9674**	0.9788	0.9272	0.9351	0.9988	-	-	0.9992	0.9692
	39	0.9321	0.9566	0.9245	0.9353	**1.0000**	-	-	**1.0000**	0.9603
Best Transformation-specific Single Validator		**0.9674**	0.9788	0.9272	0.9353	**1.0000**	-	-	**1.0000**	0.9692
Joint Validator		0.9550	**0.9882**	**0.9561**	**0.9787**	**1.0000**	-	-	**1.0000**	**0.9805**

(b) CIFAR-10

Table 3.6: (Continued) ROC-AUC scores of Deep Validation.

Configuration		Transformation Method Used to Synthesize Corner Cases								Overall ROC-AUC Score
Validator	Layer No.	Brightness	Contrast	Rotation	Shear	Scale	Translation	Complement	Combined Transformations	
Single Validator	1	**0.9887**	-	0.8367	0.7200	0.9824	0.9307	-	0.9992	0.9168
	2	0.8880	-	0.9006	0.7988	**0.9934**	**0.9664**	-	**0.9994**	0.9317
	3	0.8186	-	0.8714	0.7385	0.9141	0.8993	-	0.9980	0.8831
	4	0.7696	-	0.8650	0.8596	0.9057	0.8958	-	0.9858	0.8887
	5	0.8923	-	0.8759	**0.8930**	0.9314	0.9124	-	0.9940	0.9220
	6	0.9602	-	0.8631	0.8075	0.7784	0.8737	-	0.9007	0.8633
Best Transformation-specific Single Validator		**0.9887**	-	0.9006	**0.8930**	**0.9934**	**0.9664**	-	**0.9994**	0.9317
Joint Validator		0.9638	-	**0.9181**	0.8729	0.9757	0.9510	-	0.9979	**0.9506**

(c) SVHN

Evaluation Metric

We adopt the ROC-AUC score as the evaluation metric. The ROC-AUC score is a widely recognized metric to assess an anomaly detection method in similar tasks [17, 33]. It takes both the false positive rate (FPR) and the true positive rate (TPR) into consideration. Since some corner cases fail to fool the target model, how to define true positives is the first problem we need to address. Although detectors should also label failed corner cases as true positives under some application-specific requirements, we follow the practice in adversarial machine learning to put aside failed corner cases first [171]. We defer further discussions to Section 3.4.4. Consequently, the true positive rate means the proportion of detected SCCs in all SCCs present. Because taking clean samples for true positives is undesired in most cases, we define the false positive rate as the percentage of normal instances mislabeled by the detectors.

Detection Results

Figure 3.3 describes the distributions of the normalized discrepancy estimation d in three datasets. As we expected, almost all legitimate images have negative discrepancy values, while the opposite holds for SCCs. One can set the center of both distribution centroids as the discrepancy threshold ϵ. It is a reasonable trade-off between achieving high true positive rates and remaining relatively low false positive rates.

We report ROC-AUC scores of Deep Validation in Table 3.6. We call the whole set of SVMs in the i^{th} layer as the i^{th} single validator. The "Single Validator" row indicates the detection result when we only exploit the discrepancy estimation from the specific single validator. We show the best result for each transformation among these single validators in the "Best Transformation-specific Single Validator" row. It depicts the best results single validators can achieve when they are allowed to adapt to different settings by choosing corresponding top performers. A joint validator represents the actual Deep Validation system we deploy, and the last row of every dataset shows its performance.

We draw the following observations. For the MNIST model, the best single validators against specific transformations all lie in the first three layers. The first and third single validators each can resist one-half transformations most effectively. Because the target model never sees the synthesized corner cases during training, these corner cases may cause great discrepancy once entering the system, which renders the former validators to be immediately aware of them. However, the last validator possesses the most balanced detection capacity, which makes it stand out in the fight against all corner cases. We suspect the reason is that SCCs are pushed so far away from the normal distribution that during inference, they cannot return to the valid input region of the last layer. Therefore, the last validator gets a chance to spot them.

As for CIFAR-10, now each of the last two validators performs best under one-half transformations respectively. The penultimate validator is the best candidate that can handle all transformations when working alone. It supports our design in Section 3.4.3 to focus on the validation of rear layers. Finally, for single validators in the SVHN model, the best players return to the first two layers except for shear transformation, where the penultimate validator outperforms the others. The accumulation of small discrepancies throughout former layers may explicate why the penultimate validator can observe larger discrepancy values.

Notably, the last single validators in the SVHN model are less capable of distinguishing SCCs from clean images, which leads to deteriorated detection performance of the joint validator, especially for scale. It may imply that some corner cases have been transferred into "safe" regions by former layers mistakenly, where normal samples may concentrate as well. As a result, the last layer thinks that these are the legitimate inputs it has seen before and is confident about its predictions. As shown in Table 3.5, the relatively high confidence of the SVHN model on its wrong predictions corroborates our reasoning.

Since different single validators are capable of coping with different transformations, it inspires us to build up a versatile detector through listening to all their opinions. Also, when dealing with invalid inputs, layers are working in unhealthy conditions, which can cause the shift of these samples towards legitimate ones of latter layers by mistake. As such, the performance of single validators can fluctuate. Therefore, combining them as a joint validator can improve and stabilize the performance.

The fact that joint validators obtain the best ROC-AUC scores under most settings evidences the idea. In MNIST and CIFAR-10, joint validators always outperform single validators except for brightness adjustment in CIFAR-10. It may result from the unsatisfactory performance of the second validator. Nevertheless, this is compensated by the latter layers as the discrepancies can propagate along inter-connections. Therefore the degradation of the joint validator in this configuration is negligible. As for SVHN, the joint validator precedes the best single validators in addressing rotated images, while in the other scenarios, it lags slightly behind the best single validators. We suppose the unstable performance of single validators is the main reason. However, it can be improved via carefully assigning different weights to different single validators when computing joint discrepancy values, rather than adopting equal importance here.

We note that joint validators achieve the best overall ROC-AUC scores across three datasets, which again corroborates their effectiveness in detecting a variety of invalid inputs (*i.e.,* SCCs). When constraining the overall false positive rate to be around 3%, 7%, and 11% on MNIST, CIFAR-10, and SVHN, respectively, joint validators can achieve a respective overall true positive rate (*i.e.,* detection rate) over 96%, 94%, and 90%.

Comparison with Adversarial Image Detection Methods

We now compare our methods with adversarial image detection approaches. The reasons are as follows. Adversarial images can fool a DNN classifier by attaching imperceptible additive perturbations to clean ones after acquiring white-box access. Methods that succeed in filtering out these samples are supposed to capture the intrinsic properties of normal images. In contrast, real-world corner cases seem to introduce much larger distortions. Hence one may think such detection mechanisms can also identify real-world corner cases with ease.

Table 3.7: Comparisons with feature squeezing and kernel density estimation in detecting real-world corner cases.

Dataset	Method	Overall ROC-AUC Score (SCCs)
MNIST	Deep Validation	**0.9937**
	Feature Squeezing	0.9784
	Kernel Density Estimation	0.1436
CIFAR-10	Deep Validation	**0.9805**
	Feature Squeezing	0.8796
	Kernel Density Estimation	0.1254
SVHN	Deep Validation	**0.9506**
	Feature Squeezing	0.6870
	Kernel Density Estimation	0.2543

In order to verify this conjecture, we examine two representative detection methods, feature squeezing [171] and kernel density estimation [33], that both report state-of-the-art detection results on a crowd of adversarial images. We directly adopt their original implementations and deploy them according to the descriptions in their work. For feature squeezing, they exploit the same MNIST and CIFAR-10 models we experiment with, so we employ the same squeezer (*i.e.,* detector) configurations as they suggested. Since they do not consider the SVHN dataset, we try out the best squeezer combination they offered. For kernel density estimation, we train and fine-tune their detectors on the same data we leverage.

The comparison between Deep Validation and these two benchmark approaches, however, provides a surprising counter-example, as shown in Table 3.7. In short, despite their tremendous success against numerous white-box attacks, both detection methods disappoint us in the face of real-world corner cases. Our Deep Validation, on the contrary, overwhelmingly dominates the competition.

We take a close look at these baselines and make the following observations. For kernel density estimation, it can hardly mitigate real-world corner cases, as its ROC-AUC scores are below 0.26 across all three datasets. We suspect the reason is that, unlike our method, they rely on only one layer and mix all the clean images from different classes together. It is therefore difficult to precisely depict the complicated distribution.

As for feature squeezing, it surpasses kernel density estimation in all experiments. However, its performance is still consistently inferior to ours. As introduced before, even the clean images in the SVHN dataset are a little noisy, which makes it hard to characterize their distribution. Consequently, it is remarkable that we prevail by a significant margin in the SVHN dataset. There is enough reason to doubt whether feature squeezing indeed captures the very nature of the normal data.

The severe degradation of the detection performance of both benchmark techniques also demonstrates that real-world corner cases may embrace distinct properties, compared to artificially generated adversarial samples. It would be an interesting problem for future study.

Use Case in Defending against White-box Attacks

Since our Deep Validation is designed to be oblivious to application scenarios, we expect that it can alleviate white-box attacks as well. We set up experiments to validate this statement as follows. We only compare Deep Validation with feature squeezing here because it has been tested under more attacks. We conduct

Table 3.8: Comparison with feature squeezing in the face of white-box attacks. We adopt the same notations in [171], where "Next" and "LL" mean the next class and least-likely class in reference to the ground-truth label, respectively. Successful adversarial samples (SAEs) refer to those that cause wrong predictions regardless of their target labels, while the others are named failed adversarial samples (FAEs). Adversarial samples (AEs) contain both of them.

Attack Method		FGSM	BIM	CW_∞		CW_2		CW_0		JSMA		Overall ROC-AUC Score
Target Label		Untargeted	Untargeted	Next	LL	Next	LL	Next	LL	Next	LL	
Success Rate		0.4300	0.9100	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.6650	0.5150	
SAEs	Deep Validation	1.0000	1.0000	0.9992	0.9965	0.9347	0.9758	0.9329	0.9651	0.9851	0.9944	0.9755
	Feature Squeezing	0.9970	0.9972	1.0000	1.0000	0.9993	0.9996	0.9920	0.9920	0.9973	0.9972	**0.9971**
AEs	Deep Validation	1.0000	1.0000	0.9992	0.9965	0.9347	0.9758	0.9329	0.9651	0.9282	0.8399	**0.9572**
	Feature Squeezing	0.9441	0.9691	1.0000	1.0000	0.9993	0.9996	0.9920	0.9920	0.8169	0.6870	0.9400

extensive experiments on the MNIST dataset following the same setting as described in [171]. We explore all the white-box attacks that were covered: fast gradient sign method (FGSM) [42], basic iterative method (BIM) [72], Jacobian-based saliency map approach (JSMA) [109], and Carlini/Wagner Attacks (CW_2, CW_∞, and CW_0) [18]. They are all representative and fierce attack methods to date. We utilize the same seed and clean images in the previous evaluation dataset for consistency.

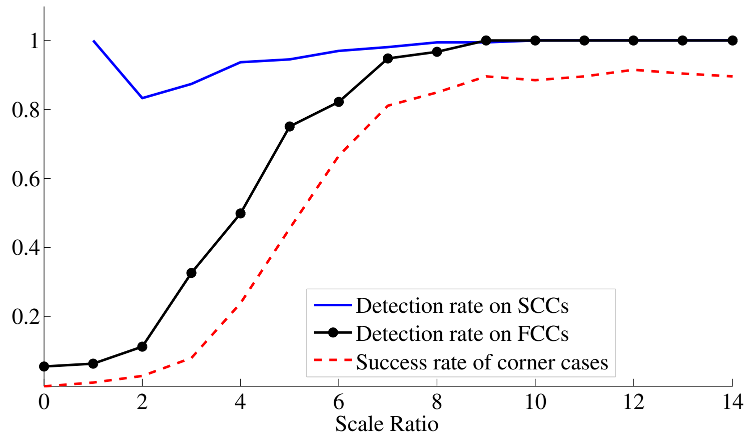
Table 3.8 enumerates all the results. We adopt the same notations in [171], where "Next" and "LL" mean the next class and least-likely class in reference to the ground-truth label, respectively. However, successful adversarial samples (SAEs) still mean those that cause wrong predictions regardless of their target labels, which is more reasonable from the perspective of defenders. The others are named failed adversarial samples (FAEs). Adversarial samples (AEs) contain both of them.

Overall, Deep Validation obtains impressive results comparable with feature squeezing. When only counting SAEs as true positives, Deep Validation slightly falls behind feature squeezing, but the result reverses when also incorporating FAEs as true positives. As adversarial samples are deliberately synthesized images that seldom happen in practice, failed attempts are an apparent sign of intrusion. It is, therefore, commendable that Deep Validation can outperform feature squeezing in spotting unsuccessful efforts, which is conducive to alerting people for upcoming attacks.

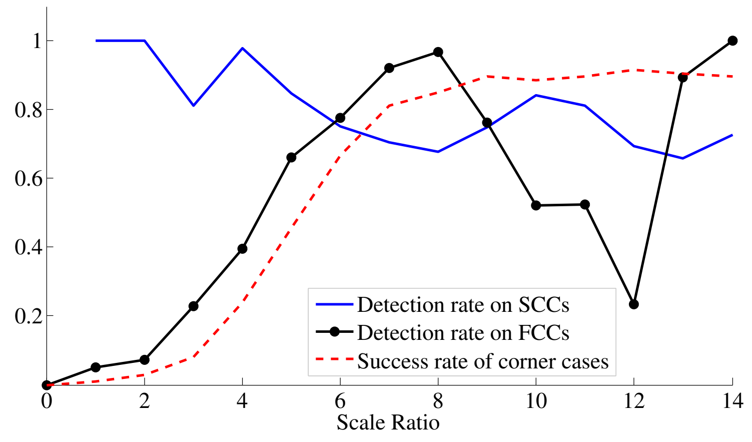
The experiments here are just to show one potential use case of Deep Validation. Since we focus this chapter on real-world corner cases, we do not conduct similar experiments on other datasets, which, however, are worthwhile for future work. We further note that the prominent performance of Deep Validation observed here is not a guarantee that Deep Validation can be immune to arbitrary attacks. Instead, the promising results confirm that it can be combined with other security methods to make the life of attackers harder.

Detection Rate Variations under Increasing Distortions

Here we propose to monitor the detection rate variations under increasing distortions. During the search for crafting real-world corner cases, increasing distortion can lead to more and more error-inducing samples in most cases. A prominent detector is supposed to identify all successful corner cases (SCCs), even those with slight perturbation. Besides, it is generally more appreciated that detectors should also pay attention to failed corner cases (FCCs) when there is a significant distortion,



(a) Deep Validation



(b) Feature Squeezing

Figure 3.4: Detection rate with regard to increasing scale ratios in MNIST. Both methods have the same false positive rate of 0.059 on clean data.

because it means that the system is working at elevated risk of fatal mistakes. On the other hand, gentle image deformation can frequently happen in practice, and we do not want to care about it as the underlying system can adapt to such changes on its own.

Consequently, we conduct similar experiments as in Section 3.4.4 to study how the detection rates of Deep Validation and feature squeezing vary with increasing distortion. We do not consider kernel density estimation here due to its poor performance reported before. Figure 3.4 shows the results for scale transformation in MNIST. The results for other settings show a similar trend and are thus omitted here.

We make the following observations. Deep Validation almost keeps 100% detection rates on SCCs with various scale ratios. The drop in scale ratio two is because there are six SCCs in total, and only one escapes from the detection. Increasing the scale ratio is likely to push normal images further away from the distribution they previously stay, which leads to growing detection rates of Deep Validation on FCCs as well. Notably, the growth is proportional to the success rate of corner cases, which is desirable, as stated before. It means that Deep Validation is aware of the imminent danger.

As for feature squeezing, its performance is unsatisfactory. Feature squeezing exhibits irregular violent oscillations and a trend of deterioration in detection rates with increasing deformation. Worse still, it hardly approaches satisfactory detection rates on SCCs, even when the degradation of model accuracy has become disastrous. This result further confirms the flaws of feature squeezing in handling real-world corner cases.

3.5 Summary

In this chapter, we propose to detect real-world corner cases to elevate the robustness of DNNs against accidental failures. The blind spots of DNN-based systems are more stealthy and challenging to fix than traditional programs, rendering them error-prone under real-world corner cases. Current efforts mostly concentrate on developing precaution strategies and neglect the importance of the fail-safe mechanism. Our Deep Validation is the first promising solution to automatically monitor and validate the intermediate input/state of running DNN classifiers. It can identify error-inducing inputs and actively call for human intervention when the system works incorrectly.

Extensive investigations of Deep Validation exhibit its exceptional performance in addressing a broad spectrum of real-world corner cases and sensitivity to approaching risks. The comparisons with well-recognized detection methods of adversarial images further showcase its superiority and promise of complementing existing approaches to make DNN-based systems more safe and dependable.

It is convenient to deploy Deep Validation in real-world situations, such as the vision systems in self-driving cars. After obtaining the video input from the car's cameras, we can uniformly sample some video frames from the video input at a pre-defined interval. Then we feed these frames into Deep Validation and acquire the final validation results by majority voting. If we observe several invalid sets of video frames, it implies that the self-driving car is running at risk, for example, under bad weather conditions. We then remind the driver to take control of the car to reduce danger.

The limitation of Deep Validation stems from the computation overhead of validating all components. Fortunately, it can be mitigated by ad hoc modifications according to network structures, as we have done for DenseNet before. It is an exciting direction for future work to offer the flexibility that allows a trade-off between ultra dependability and high efficiency.

□ **End of chapter.**

Chapter 4

Synthesizing Adversarial Samples against Undefended Deep Neural Networks

The widespread deployment of deep models necessitates the assessment of model vulnerability in practice, especially for safety- and security-sensitive domains such as autonomous driving and medical diagnosis. Transfer-based attacks against image classifiers thus attract mounting interest, where attackers are required to craft adversarial images based on local proxy models without the feedback information from remote target ones. However, under such a challenging but practical setup, the synthesized adversarial samples often achieve limited success due to overfitting to the local model employed. In this chapter, we propose a novel mechanism to alleviate the overfitting issue. It computes model attention over extracted features to regularize the search of adversarial examples, which prioritizes the corruption of critical features that are likely to be adopted by diverse architectures. Consequently, it can promote the transferability of resultant adversarial instances. Extensive experiments on

ImageNet classifiers confirm the effectiveness of our strategy and its superiority to state-of-the-art benchmarks in both white-box and black-box settings.

4.1 Problem and Motivation

Deep neural networks (DNNs) have emerged as a cutting-edge solution to a broad spectrum of real-world applications, such as object detection, speech recognition, and machine translation [117]. Despite the impressive performance of these deep learning techniques, they are surprisingly vulnerable to the so-called adversarial samples [147]. For example, by imposing human-imperceptible noises on legitimate images purposefully, the resultant adversarial input can incur erroneous predictions from state-of-the-art image classifiers. It raises growing concerns over the reliability of these high-performance black boxes and hinders the deployment of these models in practice, especially in safety- and security-sensitive domains such as autonomous driving and medical diagnosis [11].

Attacks thus play an important part in evaluating a model and revealing its blind spots before deployment. In this chapter, we center on one of the most fundamental and recognized tasks: generating adversarial images against undefended DNN image classifiers [11].

To simulate the threat a DNN image classifier may face, there are generally two kinds of threat models considered in the literature [74]. One is white-box settings, where attackers have full access to the victim model, such as the model architectures and parameters. The other one is black-box settings, where attackers only possess query access to the target model, namely, offering input images and obtaining output predictions.

Corresponding to the threat models that they are tailored for, there exist two sorts of attacks: white-box attacks and black-box ones [74]. White-box attacks can exploit the exact gradient information of the victim model to craft malicious instances [18, 42, 147]. Black-box attacks can be further divided into two categories according to the mechanism attackers adopt [29]. One is query-based, and the other one is transfer-based. Query-based black-box attacks usually require excessive queries before a successful trial [62]. On the contrary, without the feedback information from the target model, transfer-based black-box attacks devise adversarial samples with off-the-shelf local models (*i.e.,* source models) and directly harness the resultant example to fool the remote target model (*i.e.,* victim models) [29, 185].

Among these two sorts of black-box attacks, the transfer-based one has attracted ever-increasing attention recently [29]. In general, only costly query access to deployed models is available in practice. Therefore, white-box attacks hardly reflect the possible threat to a model, and query-based attacks have less practical applicability than the transfer-based counterparts due to the prohibitive query cost they may incur [29].

Thanks to the observed cross-model transferability of adversarial samples, a popular practice is to freely employ any white-box attack strategy as transfer-based black-box attacks [81]. Unfortunately, the malicious images synthesized by such a scheme are prone to overfit to the exclusive blind spots of the source model [28, 29, 167, 185]. Specifically, although the crafted

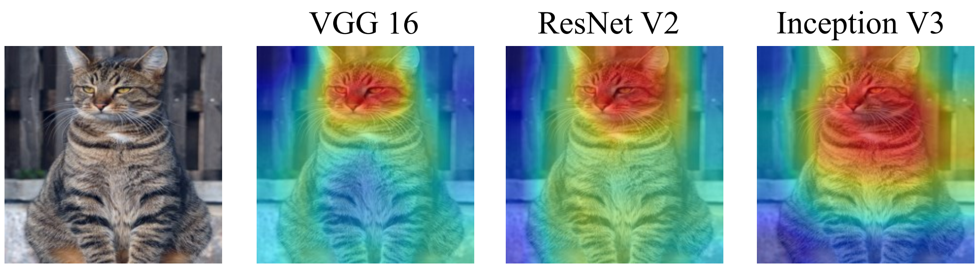


Figure 4.1: The attention heatmaps of three representative models (VGG 16 [135], ResNet V2 [48, 49], and Inception V3 [146]) for a cat prediction. The visualization is generated with the technique of [130] as detailed in Section 4.3.2. Redder regions possess higher importance to the model decision.

adversarial samples can attack the source model with near 100% success rates, they suffer from limited success against the target model.

In this chapter, we aim to promote the effectiveness of such transfer-based attacks against undefended models. It requires improving the transferability of adversarial samples crafted with white-box attack strategies. We expect that the crux is to guide the search of adversarial images towards the common vulnerable directions of both the source and the target models. Therefore, it inspires us to seek the common characteristics of diverse models and exploit such information to ameliorate the overfitting issue.

We discover that one of the similarities between different models is the essential features that they attend to. Before different models arrive at a correct decision, they should first extract various features and then weigh these features appropriately, namely, allocating suitable *attention* over extracted features[1]. Although some models may adopt exclusive feature

[1]In this chapter, we consistently employ the term "feature" to refer to the hidden

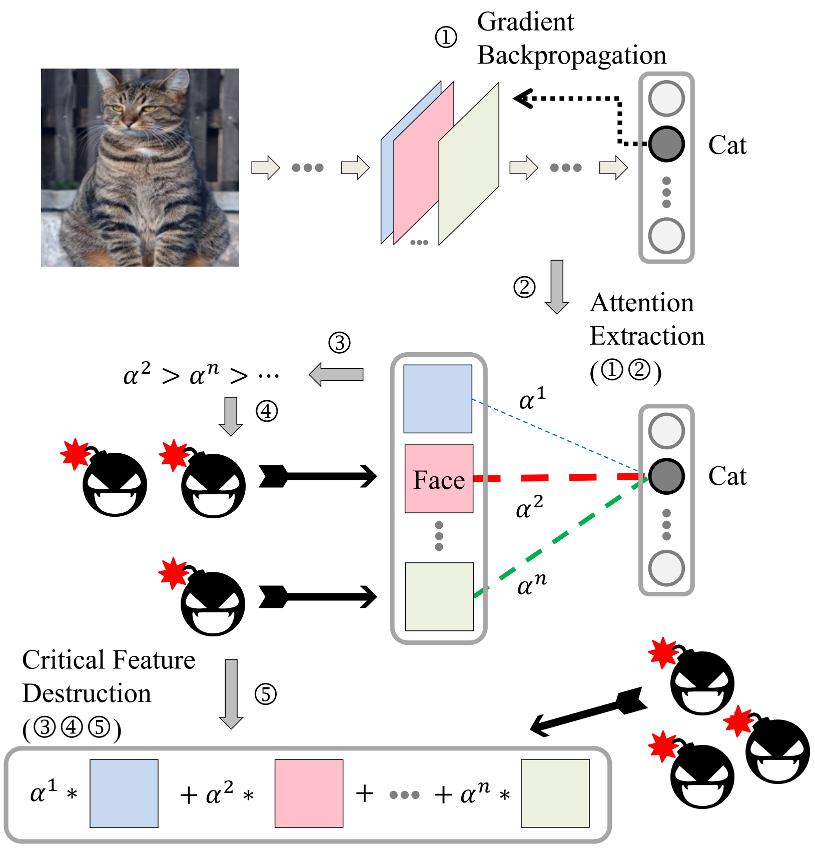

① Gradient Backpropagation
Cat
② Attention Extraction (①②)
③
$\alpha^2 > \alpha^n > \cdots$
④
α^1
Face
α^2
α^n
Cat
Critical Feature Destruction (③④⑤)
⑤
$\alpha^1 *$ $+ \alpha^2 *$ $+ \cdots + \alpha^n *$

Figure 4.2: The proposed procedure of model *attention extraction* and its application to guide the search of deceptive samples towards *critical feature destruction*.

extractors, the most critical features that diverse architectures employ tend to overlap largely in our numerous observations. For instance, as demonstrated in Figure 4.1, when different models recognize a cat image, albeit one of the models (Inception V3) also looks for features extracted from the cat neck, all of them tend to pay attention to the face-related features spontaneously.

The similarity among different models in the employed fea-

representations of images extracted by middle layers of DNNs, rather than the raw image pixels.

tures inspires us to exploit the model's attention to guide the search of adversarial perturbations. Figure 4.2 illustrates the proposed strategy. In short, we first adopt back-propagated gradients to approximate the importance of different features to model decisions (*i.e., attention extraction*). Then we require the adversarial manipulation to contaminate attention-weighed feature outputs. As a result, the synthesized malicious noise can focus on undermining the most vital image features that the local source model employs (*i.e., critical feature destruction*). Since different models strongly rely on such features, we can alleviate overfitting to a specific source model and boost the transferability of resultant adversarial samples.

In summary, we would like to highlight the following contributions of this chapter:

- We propose a novel strategy to boost the transferability of adversarial images. It features an introduction of model attention to regularize the search of deceptive noises, which mitigates overfitting to specific blind spots of the source model.
- Extensive experiments show that our attention-guided transfer attack (ATA) can severely compromise diverse top-performance image classifiers. Empirical results also witness the superior performance of our proposal to state-of-the-art benchmarks in both white-box and black-box scenarios.
- We show that our strategy is generally compatible with other transfer-based attacks and can be conveniently integrated into several state-of-the-art approaches to improve their performance.

4.2 Preliminaries

We now set up some notations. We represent a DNN image classifier as a function $f(\mathbf{x})$, which is usually a hierarchical composition of layers of neurons. It outputs the probability vector for a given image $\mathbf{x}$, where $f(\mathbf{x})[i]$ denotes the probability of the image $\mathbf{x}$ belonging to class i. We signify the hidden representation of $\mathbf{x}$ in layer k as $A_k(\mathbf{x}) = f_k(\mathbf{x})$, which consists of multiple feature maps. We will omit the input $\mathbf{x}$ when it is clear from the context. Therefore, A_k^c is the c^{th} feature map in layer k, and $A_k^c[m, n]$ is the output of the neuron with the spatial position $[m, n]$ therein.

Given a model f, an adversarial counterpart $\mathbf{x}'$ of the clean image $\mathbf{x}$ with ground truth label t should satisfy the following two conditions:

$$\text{argmax } f(\mathbf{x}') \neq t, \tag{4.1}$$

and

$$||\mathbf{x}' - \mathbf{x}||_p \leq \epsilon. \tag{4.2}$$

The first condition formulates the attack object, namely, misleading the target model to a wrong prediction with the malicious instance $\mathbf{x}'$. The second condition ensures that the induced distortion to the original image $\mathbf{x}$ is imperceptible, since ϵ is usually a fairly small number. We adopt the l_∞-norm in this chapter, as it is the most widely advocated in the community [42]. We also note that our method is generally applicable to other norm choices.

Prevailing methods to generate adversarial samples work as follows. Let $l(f(\mathbf{x}), t)$ signify the loss function to guide the training of model f. Attackers can harness the training loss function as a surrogate for the attack object in Eq. (4.1) and formulate the generation of an adversarial image $\mathbf{x}'$ as the optimization problem below:

$$\begin{aligned} \text{maximize} \quad & l(f(\mathbf{x}'), t), \\ \text{subject to} \quad & ||\mathbf{x}' - \mathbf{x}||_p \leq \epsilon. \end{aligned} \tag{4.3}$$

4.3 Methodology

Unfortunately, such white-box attacks have some disadvantages. Under the setup of transfer-based black-box attacks, attackers can only exploit off-the-shelf local models to manufacture deceptive samples. However, the solution to the above optimization problem of Eq. (4.3) usually exhibits limited transferability due to overfitting to the source model.

To overcome the pitfall, we propose to augment the vanilla training loss function with an attention-based regularization term in Eq. (4.3). It encourages the search toward harmful directions common to different deep architectures when updating the deceptive perturbations.

Our methodology proceeds as follows. As illustrated in Figure 4.2, we will first approximate model attention over extracted features with corresponding back-propagated gradients (Section 4.3.1). Then we formulate the destruction of attention-weighed combinations of feature maps as a regularization term to Eq. (4.3) (Section 4.3.3). Finally, we explain the algorithm we employ to solve the reformulated optimization problem for adversarial sample generation (Section 4.3.4).

4.3.1 Attention Extraction

We suppose that transfer-based attackers can benefit from explicitly attacking hidden feature detectors within DNN image classifiers. Unlike traditional image classification approaches that count on hand-designed features, deep learning-based image classifiers are renowned for their competence to automatically extract discriminative features from images [54]. We can thus separate a DNN image classifier into two parts: a hierarchical feature extraction module and a softmax classifier. The learned feature extractors of a DNN image classifier are often so generic that they can adapt to different domains and tasks [131]. Inspired by the fact, we expect that lots of feature descriptors are shared among different architectures for the same task, for example, the edge detector for face recognition. Therefore, if the synthesized adversarial noise can not only fool the final prediction of a target model, but also severely contaminate the extracted intermediate features, it is more likely to transfer across different models. However, polluting the intermediate features under a restricted perturbation budget may still suffer from overfitting to a specific model, since there are some feature filters exclusive to the source model.

To address the issue, we ask the deceptive noise to focus on undermining critical features for the model prediction. We assume that although different models may seek distinct feature evidence to arrive at the final decision, the most crucial features one model pays attention to are frequently shared among various architectures. For example, for a cat image, it is very likely that different models all need to exploit the face-related features when making a correct prediction (Figure 4.1).

Consequently, we need to derive the importance of diverse features to model decisions, namely, the model's attention. We regard one whole feature map as the basis feature detectors. Therefore, we approximate the importance of feature map A_k^c (*i.e.,* the c-th feature map in layer k) to class t with spatially pooled gradients as follows.

Definition 4.3.1. (The importance of feature map A_k^c). Let $\alpha_k^c[t]$ denote the importance of feature map A_k^c. Then

$$\alpha_k^c[t] = \frac{1}{Z}\sum_m\sum_n \frac{\partial f(\mathbf{x})[t]}{\partial A_k^c[m,n]}. \tag{4.4}$$

Here Z is a normalizing constant such that $\alpha_k^c[i] \in [-1, 1]$. We name $\alpha_k[t]$ the *attention weight* of the model to various features extracted in layer k regarding class t.

4.3.2 Attention Visualization

Built upon the deduced attention weights, we propose to visualize the attention maps of various models with the technique of [130]. Such visualization aims to explore what the model attention looks like and examine whether distinct models showcase similar attentions for the same correctly classified image. Therefore, it serves as a proof of concept for our idea.

We now detail how to obtain the visualization of attention maps. Specifically, we first scale different feature maps with corresponding model attention weights $\alpha_k^c[t]$. Then we perform a channel-wise summation of all feature maps in the same layer.

After that, we proceed with a ReLU operation to derive the attention map for the label prediction t as follows.

Definition 4.3.2. (The attention map for the label prediction t). Let H_k^t denote the attention map for the label prediction t. Then

$$H_k^t = \text{ReLU}(\sum_c \alpha_k^c[t] \cdot A_k^c). \tag{4.5}$$

We apply the ReLU operation here to remove negative pixels in the attention map so that we can focus on supportive features, which have a positive influence on the class of interest. Negative pixels probably stand for features from other classes. We note that H_k^t is of the same spatial resolution as the feature maps in layer k. Since the size of the feature maps varies across different layers and models, we finally bilinearly interpolate the attention map to the same resolution as the input image for better comparison.

For the same cat image, Figure 4.1 displays the attention heatmaps of various ImageNet classifiers regarding the cat prediction. We note that all these models correctly classify the cat image. It corroborates our assumption that diverse models exhibit similar attention when making a correct prediction.

4.3.3 Critical Feature Destruction

After obtaining the model attention, we can now ask the adversarial samples to not only mislead the final decision of the target model, but also destroy the vital intermediate features. We combine both goals as a novel surrogate attack object function for Eq. (4.1):

$$\begin{aligned} \text{maximize} \quad & J(\mathbf{x}, \mathbf{x}', t, f), \\ \text{where} \quad & J(\mathbf{x}, \mathbf{x}', t, f) = l(f(\mathbf{x}'), t) + \\ & \quad \lambda \sum_k ||H_k^t(\mathbf{x}') - H_k^t(\mathbf{x})||^2. \end{aligned} \tag{4.6}$$

Here the first term in J is the vanilla training loss (*i.e.,* the cross-entropy loss), and we maximize it to achieve the first goal. The second term measures the distance between attention-weighed combinations of original feature outputs and the corrupted counterparts. It corresponds to preferring great alterations to features with large attention weight and thus accounts for the second goal. λ is a tunable weight to control the regularization effect of the second term.

4.3.4 Optimization Algorithm

After substituting the proposed attack object function (Eq. (4.6)) for that in Eq. (4.3), we can now reformulate the manufacture of transferable adversarial samples as the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & J(\mathbf{x}, \mathbf{x}', t, f), \\ \text{where} \quad & J(\mathbf{x}, \mathbf{x}', t, f) = l(f(\mathbf{x}'), t) + \\ & \quad \lambda \sum_k ||H_k^t(\mathbf{x}') - H_k^t(\mathbf{x})||_2, \\ \text{subject to} \quad & ||\mathbf{x}' - \mathbf{x}||_p \leq \epsilon. \end{aligned} \tag{4.7}$$

Therefore, we can freely apply different backbone optimization algorithms to acquire a solution. For fair comparisons, the

Algorithm 3 Attention-guided Transfer Attack (ATA)

```
Require: A classifier $f$, attack object function $J$ (Eq. (4.6)), a clean image $\mathbf{x}$, and its ground-truth label $t$
Require: The perturbation budget $\epsilon$, iteration number $K$
Ensure: $||\mathbf{x}' - \mathbf{x}||_\infty \leq \epsilon$
1: $\epsilon' = \frac{\epsilon}{K}$
2: $\mathbf{x'}_0 = \mathbf{x}$
3: for $k = 0$ to $K - 1$ do
4:     $\mathbf{x'}_{k+1} = \text{clip}_{\mathbf{x},\epsilon}\{\mathbf{x'}_k + \epsilon' \text{ sign}(\frac{\partial J(\mathbf{x}, \mathbf{x'}_k, t, f)}{\partial \mathbf{x}})\}$
5: end for
6: return $\mathbf{x}' = \mathbf{x'}_K$
```

optimization strategy we apply in this chapter is the same as the white-box benchmark (BIM), which is an iterative refinement of FGSM. Concretely speaking, BIM extends FGSM into an iterative procedure with a smaller step size ϵ' in each run:

$$\mathbf{x'}_{k+1} = \text{clip}_{\mathbf{x},\epsilon}\{\mathbf{x'}_k + \epsilon' \text{ sign}(\frac{\partial l(f(\mathbf{x'}_k), t)}{\partial \mathbf{x}})\}, \tag{4.8}$$

where $\mathbf{x'}_0 = \mathbf{x}$, and $\text{clip}_{\mathbf{x},\epsilon}\{\mathbf{x}'\}$ conducts pixel-wise clipping for the resultant image $\mathbf{x}'$. Accordingly, it guarantees that $\mathbf{x}'$ stays within the l_∞ ϵ-neighborhood of the seed image $\mathbf{x}$.

Algorithm 3 summarizes our algorithm to craft transferable adversarial samples. In short, it features an introduction of the attention-based regularization term to the optimization procedure of BIM.

4.4 Experiments

This section is organized as follows. We first elucidate the experimental setup in Section 4.4.1. Then we report the results of our attack against diverse top-performance models and make comparisons with numerous state-of-the-art benchmark approaches in Section 4.4.2. Subsequently, we investigate the effect of hyper-parameters on our attack success rates in Section 4.4.3. Finally, we verify the complementing effect of our strategy on compatible algorithms in Section 4.4.4.

4.4.1 Experimental Setup

We focus on attacking image classifiers trained on ImageNet [124], which is the most broadly recognized benchmark task for transfer-based black-box attacks [16, 74]. We follow the protocol of the baseline method [185] to curate experimental datasets and target models for fair comparisons.

Dataset. We need two sorts of datasets to develop and assess our attacks, respectively. The development dataset is the ILSVRC 2012 validation dataset [124], where we fine-tune our hyper-parameters. The test data adopted to assess our technique is the ImageNet-compatible dataset released by the NeurIPS 2017 adversarial competition [74]. This test set contains 1000 images that are not included in the original ImageNet dataset. Therefore, it satisfies the requirement of evaluating the generalization capability of attack algorithms in practice.

Target model. We examine our technique with both undefended and defended models. As for undefended models, we employ numerous top-performance models with diverse architectures, including ResNet V2 [48,49], Inception V3 [146], Inception

V4 [143], and Inception-ResNet V2 [143][2]. We also attack the corresponding ensemble model (referred to as *Ensemble*), whose prediction is the average probability output of all the above models.

When it comes to the defended models, we adopt multiple state-of-the-art adversarially trained models as remote targets [73, 153], since adversarial training is arguably the most promising and effective defense to date [88]. These adversarially trained models include adversarially trained Inception V3 (Adv-Inc-v3), adversarially trained Inception-ResNet V2 (Adv-IncRes-v2), adversarially trained Inception V3 with deceptive samples from an ensemble of three models (Ens3-Adv-Inc-v3) and four models (Ens4-Adv-Inc-v3), respectively[3].

Baseline. We compare the performance of our attack with three kinds of benchmark techniques. As a naive baseline attack, we attach Gaussian noise under the same norm constraint to clean images, which is denoted as the *Random Noise* attack. More importantly, we compare our strategies with diverse state-of-the-art white-box attacks, including FGSM [42], BIM [72], C&W [18], and JSMA [109], to showcase the effectiveness of our algorithm in alleviating the overfitting issue and improving the transferability of white-box attacks. Since the original C&W implementation cannot strictly meet the l_∞ budget, we employ the modified l_∞ version of C&W as introduced by [185], which can explicitly satisfy the l_∞ norm constraint. Similar to our strategy, TAP [185] boosts adversarial transferability through

[2]These pre-trained models are all publicly available at `https://github.com/Cadene/pretrained-models.pytorch`.

[3]These models are all publicly available at `https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models`.

two regularization terms and is the state-of-the-art approach under this category. Therefore, we also include TAP in the competing benchmarks.

Metric. We compare different attacks via the top-1 accuracy of target models. Accordingly, lower accuracy of victim models on the synthesized adversarial samples represents better attack performance.

Parameter. We only include the last convolutional layer of the source model in our regularization term based on our preliminary experiments. For fair comparisons, we adopt default parameters as recommended in benchmark approaches and Foolbox [119, 185]. The random noise is sampled from a clipped normal distribution with mean 0 and variance 1. Following [185], we fix the perturbation budget ϵ to 16 for all methods. We conduct a grid search on the development dataset to settle the best hyper-parameter for our algorithm. In all our experiments, the attack iteration number K is set to 5. The regularization weight λ roughly balances the contribution of each term in the loss function J (Eq. (4.6)).

4.4.2 Transferability of Attacks

Here we study the performance of our attack against both undefended and defended models. Specifically, we first fix a source model and run our algorithm on the model to produce adversarial samples. The resultant samples are then directly fed to the source and other models to simulate the white-box and black-box setups, respectively.

We first attack undefended models, and Table 4.1 reports

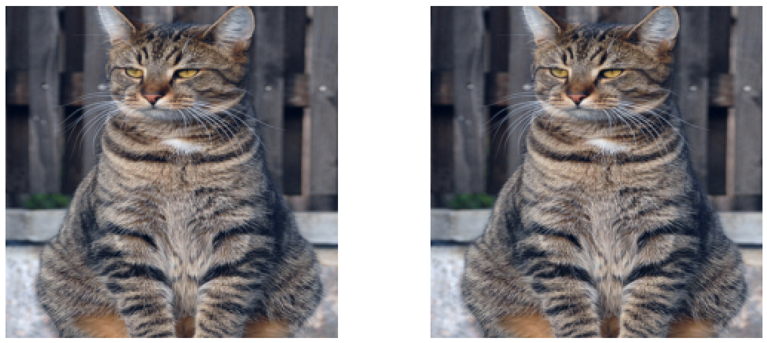
Table 4.1: Accuracy of undefended models under attacks. The first column shows the source model employed, while the first row states the remote target models.

	Attack	ResNet V2	Inception V3	Inception V4	Inception-ResNet V2	Ensemble
	No Perturbation	89.6%	96.4%	97.6%	100%	99.8%
	Random Noise	84.5%	91.7%	94.6%	97.8%	98.1%
ResNet V2	FGSM	14.6%	56.3%	64.8%	66.8%	63.1%
	BIM	**4.4%**	53.2%	62.0%	63.8%	54.3%
	C&W	37.7%	94.5%	96.4%	98.5%	98.5%
	JSMA	27.2%	59.3%	65.2%	62.1%	64.4%
	TAP	9.5%	**51.2%**	60.1%	55.5%	50.3%
	ATA	8.7%	52.9%	**58.3%**	**55.1%**	**49.4%**
Inception V3	FGSM	65.7%	27.2%	70.2%	72.9%	76.2%
	BIM	76.8%	**0.01%**	67.7%	70.2%	73.6%
	C&W	86.9%	24.5%	93.5%	96.2%	96.0%
	JSMA	66.4%	22.4%	57.2%	60.3%	68.9%
	TAP	48.2%	0.1%	24.5%	26.3%	34.2%
	ATA	**47.2%**	0.1%	**22.1%**	**25.7%**	**31.9%**
Inception V4	FGSM	68.3%	67.1%	50.3%	72.8%	76.4%
	BIM	62.1%	40.9%	**0.9%**	69.1%	55.5%
	C&W	86.7%	91.7%	49.5%	93.2%	92.9%
	JSMA	70.7%	68.9%	30.0%	65.2%	68.9%
	TAP	**58.4%**	27.3%	1.8%	24.2%	51.7%
	ATA	59.9%	**24.8%**	**0.9%**	**22.1%**	**50.3%**
Inception-ResNet V2	FGSM	71.7%	69.0%	76.5%	57.2%	78.7%
	BIM	60.4%	41.5%	51.5%	**1.2%**	54.5%
	C&W	85.6%	91.7%	92.4%	49.0%	93.5%
	JSMA	55.4%	62.7%	66.8%	50.3%	64.9%
	TAP	53.3%	25.9%	33.2%	4.8%	48.2%
	ATA	**49.8%**	**22.1%**	**30.1%**	**1.2%**	**45.3%**

Table 4.2: Accuracy of adversarially trained models under attacks. The first column shows the source model employed, while the first row states the remote target models.

	Attack	Adv-Inc-v3	Adv-IncRes-v2	Ens3-Adv-Inc-v3	Ens4-Adv-Inc-v3
ResNet V2	FGSM	62.1%	85.7%	77.4%	77.8%
	BIM	64.7%	82.6%	72.3%	74.7%
	C&W	94.0%	96.3%	92.8%	90.5%
	JSMA	58.2%	80.3%	75.2%	75.9%
	TAP	**49.2%**	66.5%	59.1%	**56.0%**
	ATA	**49.2%**	**60.3%**	**57.8%**	58.2%
Inception V3	FGSM	72.1%	93.6%	85.1%	86.4%
	BIM	82.4%	93.9%	88.2%	88.5%
	C&W	93.0%	96.4%	92.3%	90.0%
	JSMA	81.4%	93.6%	89.5%	87.4%
	TAP	55.8%	68.8%	61.3%	60.6%
	ATA	**54.1%**	**61.3%**	**60.2%**	**60.2%**
Inception V4	FGSM	74.8%	93.8%	88.1%	86.9%
	BIM	71.9%	92.9%	85.3%	85.3%
	C&W	92.8%	94.8%	91.9%	90.0%
	JSMA	70.6%	91.7%	87.9%	88.4%
	TAP	**65.3%**	90.4%	83.2%	87.3%
	ATA	69.1%	**89.8%**	**80.9%**	**82.9%**
Inception-ResNet V2	FGSM	73.9%	92.7%	86.9%	87.3%
	BIM	70.8%	92.9%	84.8%	86.9%
	C&W	91.8%	94.9%	91.9%	89.3%
	JSMA	72.1%	94.9%	83.3%	84.6%
	TAP	60.5%	87.8%	81.2%	84.3%
	ATA	**58.9%**	**85.9%**	**80.9%**	**81.4%**

the results. We make the following observations. First, all these models possess impressive clean accuracy and appear resistant to random noise. Models with higher capacity usually exhibit better performance. Second, under white-box setups, BIM is the winning attack. Our algorithm achieves matching results to BIM and significantly outperforms the others. Third, under black-box settings, our attack significantly boosts the transferability of BIM. For example, when employing Inception V3 as the source model, our attack witnesses an average gain of 40.4% on attack success rates compared to BIM.



(a) Clean (b) ATA

Figure 4.3: A clean source image and the corresponding adversarial image crafted with the proposed ATA. The target model is Inception V3. Although the perturbation is imperceptible to humans, it can successfully fool top-performance models.

Moreover, we defeat all the other benchmark methods with a significant margin, except for two cases, where we only lag a little behind TAP. We note that TAP employs two regularization terms, one for maximizing internal feature distances and the other for smoothing resultant perturbations. Contrarily, by applying only one regularization term to maximize attention-weighed internal feature distances, our method outperforms TAP in almost all cases.

As reported in Table 4.1, our average improvement of adversarial transferability over TAP is 1.6%. We note that this is a significant performance gain, since the progress of the state-of-the-art classification accuracy on the ImageNet dataset has been only 0.6% over the last three years (2019-2021) [115, 151, 152, 161, 168, 172, 175]. It means that our attack can significantly degrade the advance in ImageNet classification.

We next attack models defended by adversarial training. For fair comparisons with the baseline approach [185], we stick to employing undefended models as local source models. Therefore, we explore a more challenging black-box scenario where the source and target models possess more distinct properties.

We present the results in Table 4.2. We draw the following conclusions. First, we consistently improve the transferabilty of BIM to a great extent. For example, we increase the attack success rate of BIM by 29.3% on average, when applying Inception V3 as the source model. Second, our ATA remarkably outperforms all the other benchmarks except for two cases, where we only slightly lag behind TAP.

Figure 4.3 displays one generated adversarial image against Inception V3 with our attack. We note that the deduced manipulations to the clean image are hardly visible. It confirms that our attack is stealthy.

4.4.3 Effect of Hyper-parameters on Attack Success Rates

The regularization weight λ is the dominant hyper-parameter in our algorithm, and here we explore its effect on our attack success rates. Specifically, we vary λ while keeping the other parameters fixed to synthesize adversarial samples. Similar to previous experiments, we report the top-1 accuracy of target models on the resultant malicious examples to measure the attack success rates.

Figure 4.4 illustrates the effect of λ on attack success rates against all undefended and defended models, where the source model is Inception V3. We vary λ from 1×10^{-4} to 1 with a step size of 1 on the log scale. We observe similar trends when employing other source models and thus omit such results. We

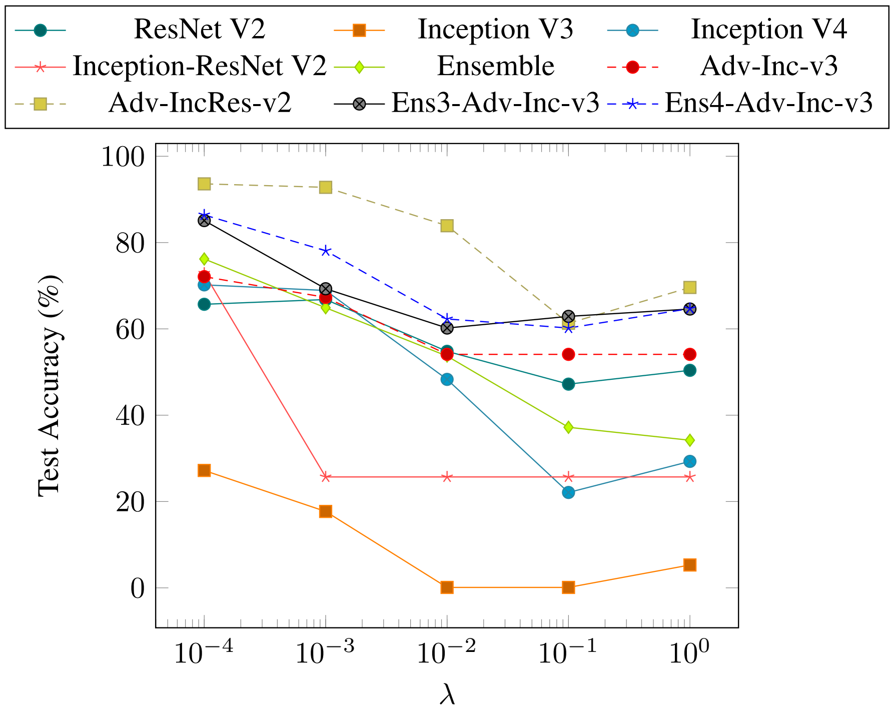


Figure 4.4: The effect of hyper-parameter λ on our attack success rates.

note that there is generally a trade-off between the two terms in J (Eq. (4.6)). Because under a restricted perturbation budget, it is crucial to balance the contribution from each term to alleviate overfitting.

4.4.4 Complementary Effect of the Proposed Strategy

In principle, our strategy is compatible with other transfer-based black-box attacks. Therefore, we can conveniently integrate the proposed technique with such algorithms. We select two sorts of cutting-edge transfer-based attacks to corroborate the complementary effect introduced by our strategy. One is the ensemble-based translation-invariant attack (TI) developed by [29], and the other is the regularization-based transferable adversarial

Table 4.3: Accuracy of models under attacks that combine the proposed ATA and compatible algorithms.

Attack	ResNet V2	Inception V3	Inception V4	Inception-ResNet V2	Ensemble	Adv-Inc-v3	Adv-IncRes-v2	Ens3-Adv-Inc-v3	Ens4-Adv-Inc-v3
TAP	58.4%	27.3%	1.8%	24.2%	51.7%	65.3%	90.4%	83.2%	87.3%
TAP+ATA	53.6%	22.7%	0.8%	19.8%	48.1%	57.9%	85.3%	73.2%	72.9%
TI	57.1%	30.9%	2.1%	26.9%	58.3%	62.7%	91.4%	81.9%	83.5%
TI+ATA	56.2%	24.9%	0.7%	24.2%	50.1%	57.9%	88.2%	76.9%	77.6%

perturbation (TAP) proposed by [185]. With the integrated attacks, we conduct experiments similar to Section 4.4.2.

We detail how to combine these attacks. Specifically, the combination of TI and ATA will only modify the update rule of Algorithm 3 as:

$$\mathbf{x'}_{k+1} = \mathrm{clip}_{\mathbf{x},\epsilon}\{\mathbf{x'}_k + \epsilon' \ \mathrm{sign}(\mathbf{W} * \frac{\partial J(\mathbf{x}, \mathbf{x'}_k, t, f)}{\partial \mathbf{x}})\}, \tag{4.9}$$

where $\mathbf{W}$ is a pre-defined kernel, and $*$ signifies convolution operation. The integration of TAP and ATA only adds the following term into the attack object function J (Eq. (4.6)):

$$\eta||\mathbf{S} * (\mathbf{x'} - \mathbf{x})||_1, \tag{4.10}$$

where $\mathbf{S}$ is a pre-defined convolution filter. We abandon the other term in TAP for simplicity because we do not have the issue of vanishing gradients.

Table 4.3 shows the results with Inception V4 as the source model. In black-box settings, our strategy promotes the average attack success rate of TAP and TI by 6.8% and 4.6%, respectively. In white-box settings, our strategy can also further improve their attack success rates. Therefore, it corroborates the complementing effect of our technique to existing efforts.

4.5 Summary

In this chapter, we introduce an attention-guided transfer attack to synthesize adversarial samples against black-box DNNs without any feedback information from the target model. The proposed strategy exploits the attention of the source model to regularize the search direction for adversarial samples. Consequently, it can focus on undermining critical features that different models count on and manifest remarkable transferability. We conduct extensive experiments to validate the effectiveness of our approach and confirm its superiority to state-of-the-art baselines. Therefore, our attack can more faithfully expose the vulnerability of deep models and serve as a strong benchmark when examining their robustness.

The pitfall of our method lies in the limited success against succeeding defended DNNs [22, 82]. We note that in our experiments, we also attack adversarially trained models to examine the general applicability of our method. However, our method is not tailored for attacking defended DNNs, as they may possess different characteristics than their normally trained counterparts [155]. Besides, except for adversarial training, many new defenses have emerged, especially the transformation-based defenses [22]. Existing attacks, including our ATA, are shown to be less effective against such defenses [22, 82]. Therefore, it calls for new attack methods to better evaluate the robustness of defended DNNs against adversarial samples.

□ **End of chapter.**

Chapter 5

Synthesizing Adversarial Samples against Defended Deep Neural Networks

With the development of adversarial attacks, more and more defenses have emerged. It thus requires us to evaluate the robustness of different defenses against adversarial attacks. In this chapter, we also consider transfer-based attacks due to their high threat in practice. However, existing transfer-based attacks frequently suffer from low success rates when defenses are present. To boost the transferability of adversarial samples against defended DNNs, we propose to improve the effectiveness of synthesized adversarial samples via adversarial transformations. Specifically, we employ an adversarial transformation network to model the most harmful distortions that can destroy adversarial noises and require the synthesized adversarial samples to become resistant to such adversarial transformations. Extensive experiments on the ImageNet benchmark showcase the superiority of our method to state-of-the-art baselines in attacking both undefended and defended models.

5.1 Problem and Motivation

There is an arms race between attacks and defenses regarding the robustness of DNNs against adversarial samples. With the development of transfer-based attacks [28, 162], more and more defenses have also emerged to counteract transfer-based attacks, especially the transformation-based defenses [22]. However, existing transfer-based attacks frequently manifest limited transferability against defended DNNs [82, 166]. Concretely, although the generated adversarial samples can fool the source model with high success rates, they can hardly remain malicious to a defended model.

Inspired by the data augmentation strategy [49,70,135], prior efforts propose to enhance the transferability of adversarial samples against defenses by training them to remain effective against common image transformations, such as resizing [167], translation [29], and scaling [78]. Unfortunately, these works explicitly model the applied image distortions by employing only individual image transformations or their simple combinations under a fixed distortion magnitude. Therefore, it makes the generated adversarial samples overfit to the applied image transformations and hardly resist unknown distortions [22], leading to inferior transferability.

To mitigate poor transferability caused by employing a fixed image transformation, a naive idea is to identify a rich collection of representative image transformations and then carefully tune a combination of them for each image. However, such a strategy can incur prohibitive computational costs. Therefore, we propose to exploit an adversarial transformation network to

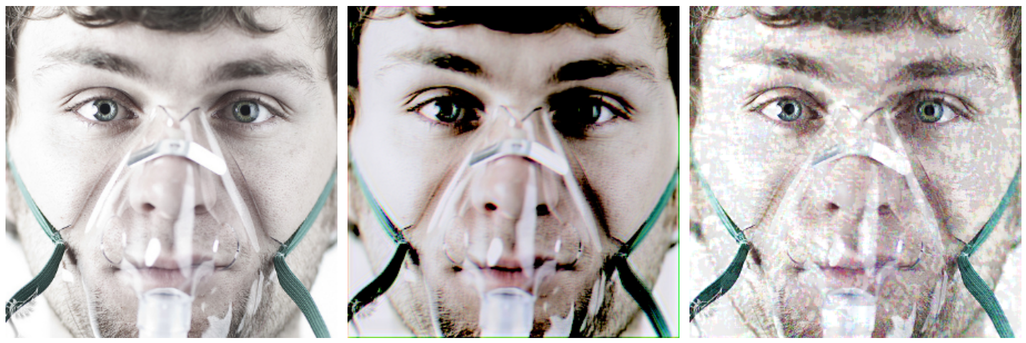


Figure 5.1: From left to right: An example of the clean image, the resultant image distorted by our adversarial transformation network, and the corresponding adversarial image generated by our method.

automate this distortion tuning process. Figure 5.1 illustrates an image manipulated by our adversarial transformation network.

Figure 5.2 depicts the diagram of our Adversarial Transformation-enhanced Transfer Attack (ATTA). Specifically, motivated by the recent advance in applying convolutional neural networks (CNNs) to conduct diverse image manipulation tasks, like digital watermarking [85,187] and style transfer [36], we propose to train a CNN as the adversarial transformation network by adversarial learning, which can capture the most harmful deformations to adversarial noises. After finishing the learning of the adversarial transformation network, we require the crafted adversarial samples to resist the distortions introduced by the adversarial transformation network. As such, we can make the generated adversarial samples more effective and improve their transferability against defended DNNs.

In summary, we would like to highlight the following contributions of this chapter:

- We propose a novel technique to improve the transferability of adversarial samples against defended DNNs with adver-

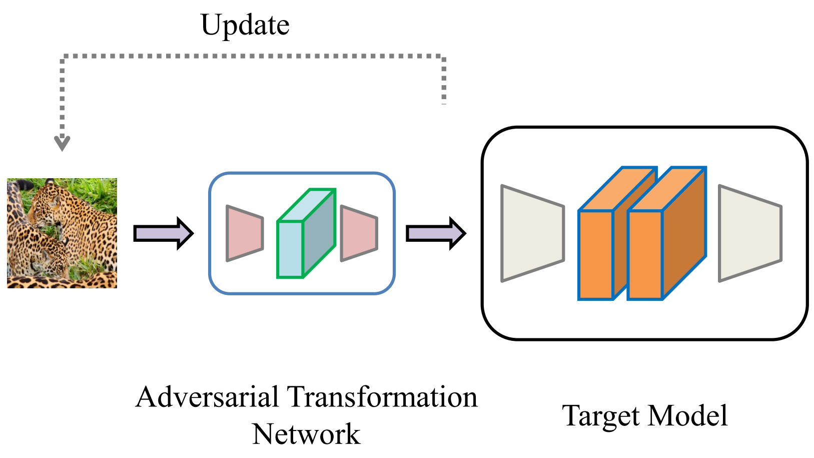


Figure 5.2: The diagram of our attack strategy. We proceed by first training an adversarial transformation network that can characterize the most harmful image transformations to adversarial noises. We then manufacture adversarial samples by additionally requiring them to remain effective against the adversarial transformation network.

sarial transformations.

- We conduct extensive experiments on the ImageNet benchmark to evaluate our approach. Experimental results confirm the superiority of our method over state-of-the-art baselines in attacking both undefended and defended models.
- We show that our technology generally complements other state-of-the-art schemes, suggesting it as a general strategy to boost adversarial transferability.

5.2 Methodology

In this section, we detail our attack technique, and the organization is as follows. We first introduce the task of crafting adversarial samples in Section 5.2.1. Then in Section 5.2.2, we elaborate on the proposed adversarial transformation network. Finally, we present our algorithm to generate adversarial samples in Section 5.2.3.

5.2.1 Problem Description

We set up some notations. Let $\mathbf{x}$ denote a clean image with ground-truth label y. We can regard a deep image classifier as a function $f(\mathbf{x})$, which returns a probability vector, indicating the probabilities of the input belonging to each class.

Given a target model f and a clean image $\mathbf{x}$, the task of attackers is to find an adversarial counterpart $\mathbf{x}^{adv}$, which satisfies the following two conditions:

$$\arg\max\ f(\mathbf{x}^{adv}) \neq y, \tag{5.1}$$

and

$$||\mathbf{x}^{adv} - \mathbf{x}||_p \leq \epsilon. \tag{5.2}$$

Here the first requirement reflects the attacker's goal of misleading the target model into wrong predictions. The second condition constrains the admissible perturbation budget for the attacker. In practice, the perturbation budget ϵ is usually a fairly small number, which ensures that the alteration to the clean image is human-imperceptible. In this chapter, we exploit the l_∞ norm to define the visibility of adversarial perturbations, since it is the most widely advocated measurement in the community [42, 78]. Nevertheless, our approach is generally applicable to other norm choices with simple modifications.

Prevailing attacks usually work as follows. We employ

$J(f(\mathbf{x}), y)$ to signify the training loss function of the classifier f. Then attackers can reformulate the task of generating an adversarial sample $\mathbf{x}^{adv}$ as the following optimization problem:

$$\begin{aligned} \max_{\mathbf{x}^{adv}} \quad & J(f(\mathbf{x}^{adv}), y), \\ \text{s.t.} \quad & ||\mathbf{x}^{adv} - \mathbf{x}||_\infty \leq \epsilon. \end{aligned} \tag{5.3}$$

Here the attackers apply the training loss function $J(f(\mathbf{x}), y)$ as a surrogate for the original attack object function (Eq. (5.1)).

In this chapter, we endeavor to develop a transfer-based attack, which works by attacking a local white-box model and harnessing the crafted adversarial samples to fool the black-box victim. In other words, we aim to improve the transferability of adversarial samples. By escalating the transferability of adversarial samples, we can attack the target black-box model with high success rates.

5.2.2 Adversarial Transformation Network

We attempt to improve the transferability of adversarial samples by the data augmentation methodology [135]. It works by asking the adversarial samples to remain effective against various image transformations, which may eliminate adversarial noises while still preserve the semantic meaning of the image [167]. Since only adopting a fixed transformation may lead to poor generalization to unknown ones, we endeavor to address the issue of explicitly modeling the applied image transformations by figuring out the most harmful image transformations to each adversarial image. We expect that if the generated adversarial samples can resist the toughest image deformations, they can also survive under other weaker distortions [88].

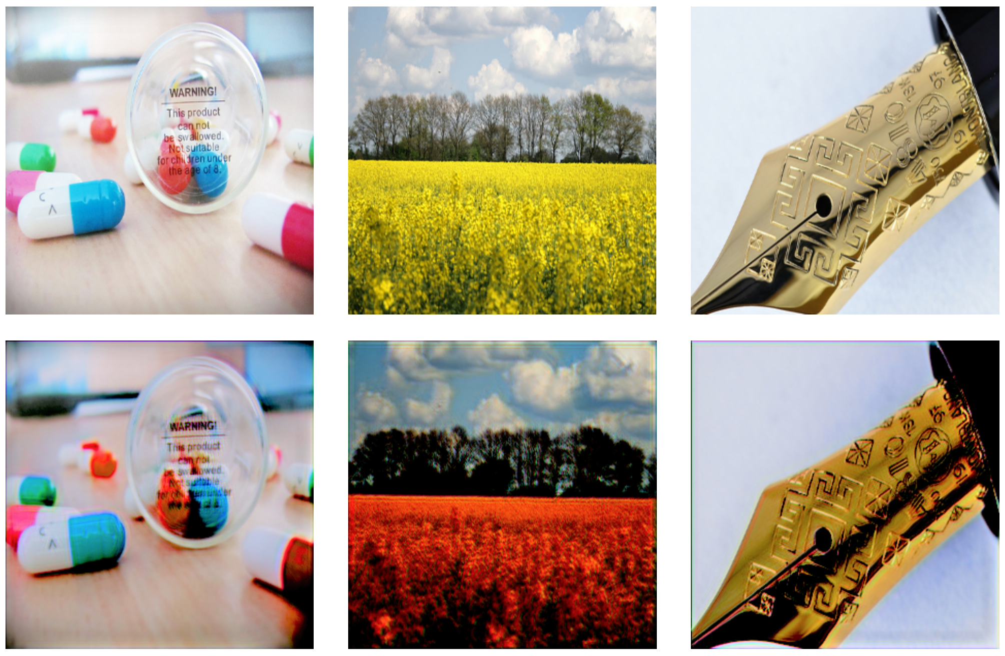
We formulate our idea as follows. Specifically, let H signify an image transformation function with parameter θ_H, which can be a composition of multiple simple image transformations, such as blurring and coloring. $H(\mathbf{x})$ thus denotes the transformed image given an input sample $\mathbf{x}$. As per Eq. (5.3), we can formulate the task of searching for the most harmful image transformations to an adversarial image $\mathbf{x}^{adv}$ as the following min-max problem:

$$\begin{aligned} \min_{\theta_H} \max_{\mathbf{x}^{adv}} \quad & J(f(H(\mathbf{x}^{adv})), y), \\ \text{s.t.} \quad & ||\mathbf{x}^{adv} - \mathbf{x}||_\infty \leq \epsilon, \\ & \arg\max \; f(H(\mathbf{x})) = y. \end{aligned} \tag{5.4}$$

Recall that y is the ground-truth label of the legitimate image $\mathbf{x}$. Here the inner maximization problem corresponds to finding an adversarial image $\mathbf{x}^{adv}$. In contrast, the outer minimization problem accounts for optimizing the transformation parameters to rectify the adversarial image, so that they become no longer malicious. The second constraint ensures that the learned image transformations can maintain the content of the clean image.

Unfortunately, it is non-trivial to solve the above optimization problem. A straightforward way to solve the optimization problem of Eq. (5.4) involves first spotting all candidate image transformations, and then tuning their combinations and distortion strengths for each adversarial image. However, such a process can incur prohibitive computational costs.

Motivated by the recent success of deep learning-based image manipulation techniques [85, 187], we propose to train a CNN-based adversarial transformation network to automate the


WARNING!
This product can not be swallowed. Not suitable for children under the age of 8.

(a) blur (b) color & blur (c) sharpen

Figure 5.3: Illustrations of the output images from our adversarial transformation network T. The top row shows the clean input images, while the bottom row enumerates the corresponding images transformed by T. We discover that the learned adversarial transformation network can perform a diverse set of image manipulations, such as blurring and a combination of multiple simple transformations. Best viewed zoomed-in on-screen.

process of tuning the most harmful image transformations to each adversarial image. Specifically, we approximately solve the optimization problem of Eq. (5.4) by restricting the hypothesis space of the transformation function H to be some class of convolutional neural networks $T(\mathbf{x};\theta_T)$ parameterized with θ_T. Therefore, the optimization problem of Eq. (5.4) now reduces to the task as follows.

$$\begin{aligned} \min_{\theta_T} \max_{\mathbf{x}^{adv}} \quad & J(f(T(\mathbf{x}^{adv})), y), \\ \text{s.t.} \quad & ||\mathbf{x}^{adv} - \mathbf{x}||_\infty \leq \epsilon, \\ & \arg\max \; f(T(\mathbf{x})) = y. \end{aligned} \tag{5.5}$$

Employing CNN to model the applied transformations affords two-fold merits. The first one is that CNNs possess the capacity to generate a cornucopia of image distortions, as demonstrated in Figure 5.3. It ensures that although we have reduced the hypothesis space of the transformation function H to be some class of convolutional neural networks, the constrained hypothesis space of the transformation function H is still large enough. Therefore, the solution to the optimization problem of Eq. (5.5) is fairly close to the optimal of the original task of Eq. (5.4). The second virtue is that we can learn the CNN function in an end-to-end fashion, which automates the tuning of the exploited transformations for each adversarial image. Therefore, it is faster and more convenient by circumventing the prohibitive overhead of manually tuning.

To train the CNN-based adversarial transformation network, we resort to the adversarial learning scheme [41, 42] to solve the optimization problem of Eq. (5.5). Specifically, we first define the outer training loss function of Eq. (5.5) as follows.

Definition 5.2.1. (The outer training loss function of Eq. (5.5)). Let L_T denote the outer training loss function of Eq. (5.5), which is the training loss function of the adversarial transformation network T. Then

$$\begin{aligned} L_T =& J(f(T(\mathbf{x}^{adv})), y) + \alpha_1 J(f(T(\mathbf{x})), y) \\ &+\alpha_2 ||\mathbf{x}^{adv} - T(\mathbf{x}^{adv})||^2. \end{aligned} \tag{5.6}$$

Specifically, the first term of L_T reflects the adversarial transformation network's pursuit of counteracting the adversarial noises, namely, rendering the adversarial sample no longer destructive to the target image classifier after the pre-processing of the adversarial transformation network. In contrast, the second term requires the adversarial transformation network to retain the content of the clean image, so that it will not incur misclassification of the target model on distorted legitimate images. The last term constrains the distortion strength introduced by the adversarial transformation network. It serves as a regularizer to alleviate the overfitting issue during the training of the adversarial transformation network. In this chapter, we employ the l_2 norm to formulate the transformation magnitude for simplicity. Nonetheless, we can also adopt other semantic measurements, like the distance calculated on the feature space of a pre-trained deep model [135]. α_1 and α_2 are the scalar weights to balance the contributions of each term in Eq. (5.6).

For the inner maximization problem of Eq. (5.5), we define the inner training loss function as follows.

Definition 5.2.2. (The inner training loss function of Eq. (5.5)).

Algorithm 4 Adversarial Transformation Network Training

```
Require: The fooling object function $L_{fool}$, the training loss function $L_T$ of
    the adversarial transformation network, and a clean image $\mathbf{x}$
Require: The perturbation budget $\epsilon$, the iteration numbers $K_{outer}$ and
    $K_{inner}$
 1: Initialize $\mathbf{x}^{adv} = \mathbf{x}$
 2: Randomly initialize $\theta_T$
 3: for $k_{outer} = 1$ to $K_{outer}$ do
 4:     for $k_{inner} = 1$ to $K_{inner}$ do
 5:         Update $\mathbf{x}^{adv} = \mathbf{x}^{adv} - \mathrm{Adam}(L_{fool})$
 6:         Clip $\mathbf{x}^{adv} = \mathrm{Clip}_{\mathbf{x}}^{\epsilon}\{\mathbf{x}^{adv}\}$
 7:     end for
 8:     Update $\theta_T = \theta_T - \mathrm{Adam}(L_T)$
 9: end for
10: return the parameter $\theta_T$ of the learned adversarial transformation
    network
```

Let L_{fool} signify the inner training loss function of Eq. (5.5), which is the fooling object function to search for an adversarial instance $\mathbf{x}^{adv}$. Then

$$L_{fool} = -J(f(T(\mathbf{x}^{adv})), y) - \beta J(f(\mathbf{x}^{adv}), y). \qquad (5.7)$$

Specifically, the second term of L_{fool} exploits the training loss function of the target model as the surrogate to seek an adversarial example $\mathbf{x}^{adv}$. Moreover, the first term takes into account the deformation induced by the adversarial transformation network, and endeavors to make the adversarial example remain malicious under the adversarial transformation network. β is the scalar weight to control the strength of each term in Eq. (5.7).

The above definitions of the outer and inner training loss functions lead us to an end-to-end training algorithm of the adversarial transformation network, which is detailed in Al-

Algorithm 5 Adversarial Sample Generation

```
Require: A classifier $f$, the attack object function $L_{attack}$, the adversarial
    transformation network $T$, a clean image $\mathbf{x}$, and its ground-truth label $y$
Require: The perturbation budget $\epsilon$ and iteration number $K$
Ensure: $||\mathbf{x}^{adv} - \mathbf{x}||_\infty \leq \epsilon$
1: $\epsilon' = \frac{\epsilon}{K}$
2: $\mathbf{x}_0^{adv} = \mathbf{x}$
3: for $k = 0$ to $K - 1$ do
4:     $\mathbf{x}_{k+1}^{adv} = \text{Clip}_{\mathbf{x}}^{\epsilon}\{\mathbf{x}_k^{adv} + \epsilon' \text{ sign}(\frac{\partial L_{attack}}{\partial \mathbf{x}})\}$
5: end for
6: return $\mathbf{x}^{adv} = \mathbf{x}_K^{adv}$
```

gorithm 4. In short, we alternate the searching for the adversarial example and the training of the adversarial transformation network, which amount to the optimization of the inner maximization problem and the outer minimization task of Eq. (5.5), respectively. Here we employ an Adam optimizer [67] to compute the updating value (Adam($\cdot$)) in each iteration. Additionally, we apply the function $\text{Clip}_{\mathbf{x}}^{\epsilon}$ to clip the resultant adversarial sample to be within the ϵ-neighborhood of the source image $\mathbf{x}$ in the l_∞ space. Therefore, we can satisfy the norm constraint for the adversarial sample in Eq. (5.5).

5.2.3 Adversarial Sample Generation

We employ the trained adversarial transformation network to generate adversarial samples. Specifically, after finishing the training of the adversarial transformation network, we can view the learned adversarial transformation network as a preprocessing module, and attach it to the target image classification model, as depicted in Figure 5.2. As a result, we can regard the cascaded adversarial transformation network and image classifier as another victim model to attack. We define the following attack object function for the attackers:

$$L_{attack} = J(f(\mathbf{x}^{adv}), y) + \gamma J(f(T(\mathbf{x}^{adv})), y), \tag{5.8}$$

where γ is the scalar weight to trade-off the contributions of each term in Eq. (5.8).

To resolve the optimization problem of Eq. (5.8), we can now turn to any backbone optimization algorithm to find an approximate solution. In this chapter, we apply the basic iterative method [72], since it is simple and efficient. Algorithm 5 elaborates on our procedure to synthesize adversarial samples.

5.3 Experiments

In this section, we conduct experiments to evaluate the effectiveness of our approach, and the organization is as follows. We first state the experimental setup in Section 5.3.1. Then in Section 5.3.2, we offer the results of our attacks against both cutting-edge undefended and defended models. We follow by an in-depth investigation of our approach in Section 5.3.3. We finally verify the complementary effect of our strategy on other compatible state-of-the-art approaches in Section 5.3.4.

5.3.1 Experimental Setup

We center on attacking image classifiers trained on the ImageNet dataset [124], which is the most widely recognized benchmark task for transfer-based attacks [16, 29]. We follow the protocol of the state-of-the-art baseline [78] to set up the experiments, which we detail as follows.

Dataset. We employ the ILSVRC 2012 training partition [124] as the development set to develop our attack, where we train the adversarial transformation network and fine-tune the hyper-parameters. For the test data adopted to evaluate our method, we randomly sample 1000 images of different categories from the ILSVRC 2012 validation set [124]. We also ensure that nearly all selected test images can be correctly classified by every model exploited in this chapter.

Target model. We attack both undefended and defended models. For undefended models, we consider multiple top-performance models with diversified architectures, incorporating ResNet v2 (Res-v2) [48, 49], Inception v3 (Inc-v3) [146], Inception v4 (Inc-v4) [143], and Inception-ResNet v2 (IncRes-v2) [143].

For defended models, we focus on several cutting-edge adversarially trained models, since adversarial training is arguably the most effective and promising defense to date [88]. Specifically, we explore adversarially trained Inception-ResNet v2 (IncRes-v2$_{\text{adv}}$), adversarially trained Inception v3 with deceptive samples from an ensemble of three models (Inc-v3$_{\text{ens3}}$) and four models (Inc-v3$_{\text{ens4}}$), respectively [73, 153].

Furthermore, we study another line of state-of-the-art defenses that aims to rectify adversarial samples. These defenses cover high-level representation guided denoiser (HGD) [77], random resizing and padding (R&P) [166], NIPS-r3[1], feature distillation (FD) [82], compression defense (ComDefend) [63],

[1]`https://github.com/anlthms/nips-2017/tree/master/mmd`

and randomized smoothing (RS) [22].

Baseline. We compare our approach with two sorts of baselines. The first one represents top-performance white-box attacks that manifest greater transferability than the other white-box techniques [29], including Fast Gradient Sign Method (FGSM) [42] and Basic Iterative Method (BIM) [72]. The second category incorporates state-of-the-art transfer-based attacks, embracing Diverse Input Method (DIM) [167], Translation-Invariant Method (TIM) [29], Scale-Invariant Method (SIM) [78], Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [28], and Nesterov Iterative Fast Gradient Sign Method (NI-FGSM) [78]. Similar to us, they also seek to boost the transferability of adversarial samples from the perspective of optimization and generalization, either by employing more advanced optimizers or by data augmentation.

Parameter. For the adversarial transformation network, we adopt a two-layer CNN: $T(\mathbf{x}) = \mathrm{Conv}_{3\times3} \circ \text{Leaky ReLU} \circ \mathrm{Conv}_{16\times3}(\mathbf{x})$, where Conv indicates a convolutional layer with the denotation of $\mathrm{Conv}_{\text{kernel size}\times\text{number}}$. For benchmark attacks, we employ the recommended parameters in their original implementation for fair comparisons. Following [28, 78], we set the perturbation budget $\epsilon = 16$ for all attacks. The iteration numbers K, K_{outer}, and K_{inner} are set to 10. We determine the best hyper-parameters of our algorithm with grid search on the development set. The weight parameters are 1.0, 10, 1.0, and 1.0 for α_1, α_2, β, and γ, respectively.

Table 5.1: Success rates (%) of different attacks against seven models. The first column lists the source model adopted to craft adversarial samples, while the first row shows the target model.

	Attack	Res-v2	Inc-v3	Inc-v4	IncRes-v2	Inc-v3$_{\text{ens3}}$	Inc-v3$_{\text{ens4}}$	IncRes-v2$_{\text{adv}}$
Res-v2	FGSM	85.4	43.7	35.2	33.2	22.6	22.2	14.3
	BIM	95.6	46.8	38.0	36.2	27.6	25.3	17.4
	DIM	97.9	66.3	57.2	55.6	30.5	29.6	20.8
	TIM	98.8	65.2	59.8	57.4	35.6	31.7	25.8
	SIM	98.8	**67.3**	57.4	57.4	38.1	30.1	26.7
	MI-FGSM	98.2	57.9	53.9	49.4	33.0	29.2	21.8
	NI-FGSM	98.6	62.2	55.5	53.3	33.1	28.9	21.1
	ATTA (Ours)	**99.8**	64.3	**61.8**	**59.2**	**42.1**	**38.9**	**29.1**
Inc-v3	FGSM	34.3	72.8	29.8	27.1	14.9	13.6	17.9
	BIM	33.2	99.9	32.3	29.8	11.8	11.5	17.6
	DIM	39.2	**100**	39.2	37.6	23.2	24.3	14.0
	TIM	39.2	**100**	44.3	45.8	23.2	24.9	16.4
	SIM	40.1	**100**	42.9	46.4	22.8	24.3	16.9
	MI-FGSM	36.2	**100**	44.4	42.7	22.5	22.4	16.5
	NI-FGSM	38.0	**100**	47.4	46.4	23.2	22.4	16.4
	ATTA (Ours)	**44.8**	**100**	**52.9**	**53.2**	**25.1**	**27.9**	**18.8**
Inc-v4	FGSM	31.7	32.9	49.7	28.2	11.9	13.1	6.2
	BIM	37.9	59.1	99.1	30.9	14.7	14.7	7.1
	DIM	40.8	64.3	**99.6**	39.4	24.6	24.8	15.2
	TIM	41.4	64.3	**99.6**	48.2	25.7	25.2	16.9
	SIM	41.4	61.9	**99.6**	49.7	27.9	25.2	17.4
	MI-FGSM	40.1	58.8	**99.6**	44.4	27.0	25.1	18.1
	NI-FGSM	42.9	62.4	**99.6**	51.8	25.4	24.1	17.6
	ATTA (Ours)	**43.8**	**66.8**	**99.6**	**59.2**	**32.1**	**29.2**	**20.8**
IncRes-v2	FGSM	29.3	31.0	23.5	42.8	13.1	12.7	7.3
	BIM	39.6	58.5	23.5	42.8	15.2	13.1	7.1
	DIM	41.3	63.4	58.3	97.7	30.7	29.2	19.8
	TIM	43.1	62.9	55.4	**98.9**	31.8	29.2	20.6
	SIM	42.1	60.9	52.7	**98.9**	29.6	29.2	20.9
	MI-FGSM	39.9	56.8	48.6	97.7	19.6	26.0	21.7
	NI-FGSM	39.7	59.1	51.2	**98.9**	25.6	25.2	20.6
	ATTA (Ours)	**44.8**	**68.9**	**65.2**	**98.9**	**33.0**	**31.9**	**24.3**

5.3.2 Attacking Results

Here we assess the performance of our attacks against both undefended and defended models. Specifically, for a given source model, we mount attacks on it and directly apply the result adversarial samples to fool the other different models, which amounts to the black-box setting. We also test the attacking results on the source model itself, which corresponds to the white-box setting.

Table 5.2: Success rates (%) of different attacks against advanced defense methods.

Attack	HGD	R&P	NIPS-r3	FD	ComDefend	RS	Average
FGSM	8.9	16.8	23.1	19.2	13.4	6.8	14.7
BIM	12.1	19.3	23.8	21.8	17.2	8.9	17.2
DIM	79.5	74.7	81.9	76.4	72.3	42.3	71.2
TIM	73.3	69.8	79.4	78.2	69.2	36.2	67.7
SIM	76.2	77.7	84.2	79.8	75.4	39.3	72.1
MI-FGSM	33.4	27.2	42.1	47.3	42.8	29.9	37.1
NI-FGSM	35.2	30.3	40.8	49.2	44.9	32.3	38.8
ATTA (Ours)	**85.9**	**83.2**	**89.5**	**84.4**	**79.9**	**47.4**	**78.4**

Table 5.1 reports the attacking performance of different methods against both undefended and adversarially trained models. Our attack achieves nearly **100%** success rates under the white-box scenarios. More importantly, we can see that under the black-box settings, our technique can drastically improve the transferability of BIM. For instance, when applying Inc-v3 as the source model, our attacking performance exceeds BIM by over **14.4%** on average. Besides, our attack consistently outperforms all state-of-the-art baselines by a significant margin under the black-box settings, which further corroborates the superiority of our strategy on synthesizing transferable adversarial samples.

We also evaluate the success rates of different attacks against other advanced defenses. Table 5.2 shows the results when adopting Inc-v3 as the source model to attack other models defended with different mechanisms. Our attacks achieve an average success rate of **78.4%**, defeating all state-of-the-art attacks by a significant margin of over **6.3%**. It further evidences the effectiveness of our attacks against both top-performance undefended and defended models, and raises a new security concern for developing more robust defenses.

Table 5.3: Success rates (%) of our attack when varying the complexity of the adversarial transformation network. The first row shows the target model.

Structure	Res-v2	Inc-v3	Inc-v4	IncRes-v2	Inc-v3$_{\text{ens3}}$	Inc-v3$_{\text{ens4}}$	IncRes-v2$_{\text{adv}}$
Conv (4, 3)	39.7	100	42.8	44.9	19.3	19.5	16.2
Conv (16, 3)	**44.8**	**100**	**52.9**	**53.2**	**25.1**	**27.9**	**18.8**
Conv (32, 3)	34.8	100	31.6	32.2	15.9	13.6	16.6
Conv (32, 32, 3)	33.8	100	34.1	31.3	12.3	11.9	17.9

5.3.3 Further Analysis

We first analyze the contribution of the proposed adversarial transformation network. As shown in Algorithm 5, our attack is built upon BIM by augmenting an adversarial transformation network. Therefore, comparing the performance of BIM and our method in Table 5.1 and Table 5.2 constitutes an ablation study. The remarkable advance of our attack over BIM verifies the contribution of the proposed adversarial transformation network.

We then analyze the effect of the complexity of the adversarial transformation network. Specifically, we adjust the structures of the adversarial transformation network and perform attacks as in Section 5.3.2. We present the results when exploiting Inc-v3 as the source model in Table 5.3. We indicate the architecture of the adversarial transformation network in the format of Conv $(a, b, ...)$, where we specify the kernel size of each convolutional layer in parentheses. The number of kernels is three across all convolutional layers. From Table 5.3, we can observe that over simple or sophisticated structures can deteriorate our attack performance, since the former hardly owns enough representation capacity, while the latter can make the adversarial transformation network overfit to the backbone attack algorithm.

5.3.4 Complementary Effect of Our Technique

In principle, our strategy is compatible with other state-of-the-art transfer-based attacks. Therefore, we can conveniently combine our technique with these attacks.

To validate the complementary effect of our technology, we experiment with the state-of-the-art integrated transfer-based attack (SI-NI-TI-DIM) [78], which is a composition of SIM, NI-FGSM, TIM, and DIM. Specifically, to integrate our strategy with SI-NI-TI-DIM, we just need to first regard the cascaded adversarial transformation network and image classifier as another victim model. Then we attack both the cascaded network and the original classifier with SI-NI-TI-DIM. We denote the combination of our ATTA and SI-NI-TI-DIM as AT-SI-NI-TI-DIM.

We conduct similar experiments as in Section 5.3.2, and Table 5.4 states the results. We make the following observations. First, our attack (AT-SI-NI-TI-DIM) can attain almost **100%** success rates under the white-box context. Second, our method can consistently promote the success rates of the state-of-the-art baseline by a considerable margin, under all black-box cases. Therefore, it affirms the complementary effect of our technique.

5.4 Summary

In this chapter, we introduce a novel technique, Adversarial Transformation-enhanced Transfer Attack (ATTA), to boost the transferability of adversarial samples against defended models. Inspired by the data augmentation methodology, it features

Table 5.4: Attack success rates (%) when combining our strategy with compatible algorithms. The first column lists the source model adopted to craft adversarial samples, while the first row shows the target model.

	Attack	Res-v2	Inc-v3	Inc-v4	IncRes-v2	Inc-v3$_{\mathrm{ens3}}$	Inc-v3$_{\mathrm{ens4}}$	IncRes-v2$_{\mathrm{adv}}$
Res-v2	SI-NI-TI-DIM	**99.8**	78.3	70.2	71.8	34.9	35.9	30.2
	AT-SI-NI-TI-DIM (Ours)	**99.8**	**80.1**	**74.9**	**74.9**	**36.8**	**37.3**	**33.2**
Inc-v3	SI-NI-TI-DIM	48.3	**100**	54.3	56.2	**27.8**	28.1	24.5
	AT-SI-NI-TI-DIM (Ours)	**49.1**	**100**	**55.9**	**57.1**	**27.8**	**28.6**	**24.9**
Inc-v4	SI-NI-TI-DIM	49.5	72.1	**99.6**	60.3	33.2	31.8	26.9
	AT-SI-NI-TI-DIM (Ours)	**50.4**	**75.2**	**99.6**	**62.8**	**33.9**	**32.3**	**27.6**
IncRes-v2	SI-NI-TI-DIM	50.1	72.9	69.6	**98.9**	34.5	32.7	27.4
	AT-SI-NI-TI-DIM (Ours)	**55.3**	**77.8**	**74.2**	**98.9**	**36.5**	**34.9**	**29.1**

training a CNN-based adversarial transformation network by adversarial learning, and requiring the generated adversarial samples to withstand the adversarial transformation network. Moreover, our strategy can be conveniently combined with other transfer-based attacks to further promote their performance. Extensive experiments corroborate the superiority of our approach on synthesizing transferable adversarial samples against both state-of-the-art undefended and defended models. Therefore, our attack can serve as a strong benchmark to evaluate future defenses.

□ **End of chapter.**

Chapter 6

Global Explanations of Deep Neural Networks with Concept Attribution

With the growing prevalence of convolutional neural networks (CNNs), there is an urgent demand to explain their behaviors. Global explanations contribute to understanding model predictions on a whole category of samples, and thus have attracted increasing interest recently. However, existing methods overwhelmingly conduct separate input attribution or rely on local approximations of models, making them fail to offer faithful global explanations of CNNs. To overcome such drawbacks, we propose a novel two-stage framework, Attacking for Interpretability (AfI), which explains model decisions in terms of the importance of user-defined concepts. AfI first conducts a feature occlusion analysis, which resembles a process of attacking models to derive the category-wide importance of different features. We then map the feature importance to concept importance through ad-hoc semantic tasks. Experimental results confirm the effectiveness of AfI and its superiority in providing more accurate estimations of concept importance than existing proposals.

6.1 Problem and Motivation

Convolutional neural networks (CNNs) have emerged as a cutting-edge solution to a broad spectrum of real-world applications, such as object recognition [70], audio processing [53], and natural language analysis [180]. Despite the startling advance of these powerful computational architectures, their inner decision operations remain a mystery. Interpreting and understanding the behaviors of CNNs have become an increasingly crucial topic of research. It can not only justify the decisions of CNNs to promote model trustworthiness, but also spot their latent defects to inspire the development of better models [37, 43, 50, 165].

Among diverse explanation techniques, attribution endeavors to succinctly summarize how CNNs arrive at their final decisions [38,105]. Under the context of image classification, the convention is to measure the importance of human-understandable units to model predictions, such as pixels (*i.e.,* input attribution) and concepts (*i.e.,* concept attribution) [66]. Concept attribution can overcome the ambiguity of input attribution and thus has attracted growing attention recently [38, 66, 184].

There are two explanation interfaces of concept attribution studied in the literature: local explanations [184] and global ones [66]. We focus on the latter in this chapter, which is imperative but under-explored. Local explanations investigate the rationale of model predictions on individual data points, which are helpful when we only care about a specific instance. In contrast, global explanations center on mining generic decision modes that apply to an entire class of examples. For instance, global explanations can answer to what extent the banded texture is related to a zebra class in model cognition. Therefore, such global explanations are conducive to summarize the model knowledge succinctly and understand the model as a whole [66].

In general, existing concept attribution methods implicitly follow a two-stage procedure [38, 66, 184]. First, since the model decisions are built upon a cornucopia of feature detectors, they conduct *feature attribution* to quantify the importance of individual feature detectors to model predictions[1]. In this step, current attempts simply employ backpropagated gradients as the estimation of feature importance. Second, they achieve *concept attribution* by translating feature importance into concept importance. Most first settle the embedding of a concept in the model feature space (*i.e.,* the concept vector), and then measure the alignment between this concept vector and the vector of feature importance. As for works that focus on global explanations, they just analyze individual predictions in isolation with the above procedure and then return summary statistics [38, 66].

It is doubtful whether such a strategy to obtain global explanations indeed sees "globally". The deficiency primarily originates from the process of feature attribution with backpropagated gradients, which implicitly builds upon a local linear approximation of CNNs. Unfortunately, such an approximation holds merely when we deal with the proximity of individual

[1]To avoid confusion, we consistently use the term "feature" to refer to the visual patterns detected by feature filters of CNNs (*e.g.,* the banded texture), rather than the input pixels.

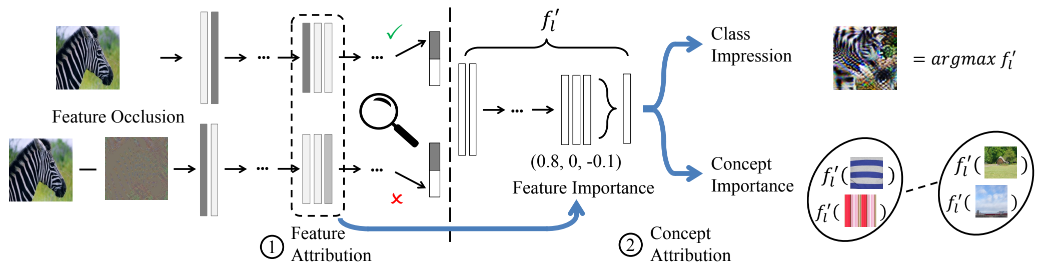


Figure 6.1: The workflow of our framework: Attacking for Interpretability (AfI).

instances or the last linear layer of CNNs. Worse still, inspecting individual predictions separately with respective gradients ignores the connections among examples of the same class. It may not be able to capture the generic properties of the class embedded in model knowledge.

To surmount the pitfalls of existing proposals, we propose a novel concept attribution framework for global explanations of CNNs. It explicitly builds upon the two-stage prototype of prior efforts. As such, we systematize the process to model explanations in that we make each step grounded and propose to evaluate intermediate results. More crucially, we thoughtfully extend the methodology of input occlusion to feature occlusion, which enables learning a global explanation and delving into model internals for layer-wise inspections (Section 6.3.4).

Figure 6.1 outlines the workflow of our framework: Attacking for Interpretability (AfI). In the first stage, we conduct feature attribution through a thoughtful feature occlusion analysis. Based on an opposite view of attribution, and the fact that feature detectors in CNNs can be depressed by structured patterns [126], we proceed by learning such a feature occluder in the input space for an entire category of images. The feature occluder is applied to undermine critical feature filters of the class so that models will deviate from their original predictions. Such a feature occlusion procedure coincides with that of attacking CNNs to fool their decisions (attacking). We then record the resultant activation alterations of feature detectors, and score the importance of different features accordingly.

In the second stage, we accomplish concept attribution via directly anchoring feature importance to concept importance (interpretability). We first directly combine feature detectors as per their importance scores to obtain a class-specific meta-detector, and then run semantic tests for a concept of interest. As such, higher performance of the meta-detector in the semantic test implies greater importance of the investigated concept to the class.

In summary, the main contributions of this chapter are:

- We propose a novel concept attribution framework for global explanations of CNNs. Our framework explicitly builds upon a two-stage procedure and employs a novel feature occlusion methodology to learn a global interpretation. As such, we systematize the process to model explanations, and overcome the deficiencies of most existing global explanation techniques that bank on a local approximation of CNNs.
- We conduct extensive experiments on the ImageNet dataset with three representative models. Experimental results validate the effectiveness of our approach and showcase its superiority to previous efforts.
- With the global explanations our framework affords, we

demonstrate its use cases in providing insights into CNNs, like grounding model decisions and revealing biases in model cognition.

6.2 Methodology

In this section, we will detail the design of our framework. As illustrated in Figure 6.1, our two-stage approach proceeds by tackling the following tasks sequentially: (a) how to learn a feature occluder to perform feature occlusion (Section 6.2.1), (b) how to complete feature attribution with feature occluders (Section 6.2.1), and (c) how to achieve concept attribution via aligning feature importance with concept importance (Section 6.2.2).

We first set up some notations. We regard an input image as a vector $\mathbf{x} \in \mathbb{R}^n$ with label prediction $y \in \mathbb{Y}$, where $\mathbb{Y} := \{1, ..., K\}$ is a categorical set of interest. By convention, images will be normalized such that $\mathbf{x}$ stays within the range of $[-1, 1]^n$ with zero mean before feeding into models. In a CNN classifier with L layers, the l^{th} layer with m neurons learns a mapping from inputs to hidden representations $f_l : \mathbb{R}^n \to \mathbb{R}^m$. In particular, the final layer computes a logit vector $Z(\mathbf{x}) \in \mathbb{R}^K$ and then yields a probability vector $f_L(\mathbf{x})$ after softmax normalization. The y^{th} entry $f_L(\mathbf{x})[y]$ corresponds to the probability of $\mathbf{x}$ belonging to class y. A CNN classifier will output label predictions in the end, and thus its decision function is $f : \mathbb{R}^n \to \mathbb{Y}$.

6.2.1 Feature Attribution

For feature attribution, we propose to extend the methodology of input occlusion to feature occlusion. The general procedure of input occlusion is to occlude some input pixels and regard the resultant alterations of model output as their importance score [174]. Unfortunately, a straightforward adaptation scarcely applies to feature occlusion. In modern CNN architectures, there are innumerous neurons that work in close collaboration [34]. Therefore, separately occluding individual neurons ignores their intensive interconnections, while exhausting all possible combinations is prohibitively expensive.

We circumvent this difficulty via an opposite view of attribution via occlusion. Given an image $\mathbf{x}$ and its prediction y, the fundamental problem in attribution is to explain how a model discriminates class y from all the others. Furthermore, in the form of feature attribution, we can summarize the reasoning process of a model in this binary classification task as:

$$\begin{aligned} &\textit{the features of class } y \textit{ in image } \mathbf{x} \textit{ are more prominent} \\ \Longleftrightarrow\ &\textit{the label prediction for image } \mathbf{x} \textit{ is } y. \end{aligned} \tag{6.1}$$

Consequently, it reduces to spot supporting features for model decisions. To this end, we first transform the forward reasoning of (6.1) into its logic equivalence:

$$\begin{aligned} &\textit{the label prediction for image } \mathbf{x} \textit{ is } \text{not } y \longrightarrow \\ &\textit{the features of class } y \textit{ in image } \mathbf{x} \textit{ are } \text{less } \textit{prominent.} \end{aligned} \tag{6.2}$$

Then by combining with the backward reasoning of (6.1):

$$\begin{aligned}&\textit{the label prediction for image } \mathbf{x} \text{ is } y \longrightarrow \\ &\textit{the features of class } y \textit{ in image } \mathbf{x} \textit{ are} \text{ more } \textit{prominent}, \end{aligned} \qquad (6.3)$$

it leads us to an opposite procedure for attribution with occlusion. Specifically, we can conservatively undermine the feature filters of a model until it is forced to abandon its original decisions. As such, the resultant variations of neuron activations represent their importance to model predictions.

Moreover, since feature filters of CNNs are susceptible to structured noise [126], such an opposite view empowers us to perform feature occlusion from the input space. Specifically, we can first learn such a malicious perturbation to "subtract" minimal image features, which suffice to flip model predictions. We name such perturbations *feature occluders*, which effectively work by disturbing responsible feature detectors [5,126]. Therefore, feature occluders need not destroy images in a human-recognizable manner, or align with actual regions where filters extract features. Then we examine the change of neuron outputs to rate their importance.

Global Feature Occluder

As we seek a global explanation of samples under the same category, we start by crafting a *global feature occluder* for them. Formally, let D denote a distance function. t signifies an image transformation function, like random noising. Given a collection of images $\{\mathbf{x}_i : i = 1, \ldots, N\}$ with identical classification y, we define their global feature occluder δ^* as follows.

Definition 6.2.1. (Global feature occluder). The global feature occluder δ^* for a set of images with the same label y solves the following optimization problem:

$$\begin{aligned}
\delta^* = \ & \text{argmin } D(\delta) \\
\text{such that} \ & f(\mathbf{x}_i - \delta) \neq y \\
& f(t(\mathbf{x}_i - \delta)) \neq y \qquad i = 1, \ \ldots, N \\
& f(t(\mathbf{x}_i)) = f(\mathbf{x}_i) = y \\
& \mathbf{x}_i - \delta \in [-1, 1]^n.
\end{aligned} \tag{6.4}$$

We elucidate the definition as follows. In the object function of (6.4), distance function D measures the magnitude of δ. As such, we aim to search for minimal perturbations, which reflects the appeal of disturbing minimal feature filters so that we can identify the most critical features of the class. In light of the sliding-window scheme in CNNs [40], we implement D via l_1 distance.

The first condition of (6.4) further requires that a global feature occluder is the minimal noise needed to flip the model predictions on all the given instances simultaneously. Therefore, it will prefer to impede decisive feature detectors common to images of the same class, which takes into account the relations among samples embedded in model memory. In this sense, our approach conducts a sort of reverse engineering of the model training process, which is conducive to expose a more global picture of model logic.

The second condition of (6.4) conducts regularization. We suppose that purely learning deceptive distortion may end up spoiling some fragile filters less relevant to essential image features. To eliminate such artifacts, we additionally require that a global feature occluder should remain effective when applied to the transformed versions of original images. We expect that the outputs of supporting feature filters can maintain relatively unchanged compared to the others when inputting transformed images. Consequently, such a requirement can make feature occluders focus on dimming critical features rather than arrive at the cheapest structure.

As for the last two conditions, the third condition of (6.4) constrains t to constitute an effective regularization. Specifically, we ensure that t will not harm the judgment of the model on clean images. The last condition of (6.4) guarantees that occluded images are still valid inputs for models.

We now need to solve the optimization problem of (6.4). As CNNs are involved, directly solving (6.4) is intractable. We instead obtain an approximation by employing Adam optimizer [67] to minimize the following object function iteratively:

$$\frac{1}{N}\Sigma_{i=1}^{N}(Z(\mathbf{x}_i - \delta)[y]+Z(t(\mathbf{x}_i - \delta))[y]) + \lambda \cdot D(\delta). \tag{6.5}$$

Our algorithm terminates once the occluder satisfies all the constraints in (6.4), or when we exceed preset maximum iterations.

Feature Importance Score

Now we can calculate feature importance scores with the obtained global feature occluder for class y. Specifically, the importance score of the feature that the j^{th} neuron in the l^{th}

layer detects is:

$$s_l^j = \frac{1}{N}\Sigma_{i=1}^{N}(f_l(\mathbf{x}_i)[j] - f_l(\mathbf{x}_i - \delta^*)[j]). \tag{6.6}$$

The sign of the importance score differentiates two sorts of features related to model decisions. Neurons with positive scores account for supporting features, while those with negative scores vote for antagonistic counterparts [130, 182].

Similar to conventional practice, we focus on features and concepts that have positive contributions to model decisions [130, 184]. Therefore, we zero out negative importance scores in s_l^j to obtain the final feature importance score (FIS) we adopt:

$$s_l^{'j} = \max(s_l^j, 0). \tag{6.7}$$

6.2.2 Concept Attribution

Some prior proposals communicate feature importance in terms of the importance of semantic notions readily accessible to humans by the following procedure. They first examine CNN units separately to work out their concept labels. They then read the importance scores of these concepts from the feature importance scores of corresponding units [105, 178]. However, such strategies overlook concepts with entangled encodings in CNNs [7, 34].

To surmount this defect, we propose a two-step procedure. In a word, we first combine CNN units as per their importance scores, which leads to a class-specific meta-detector. Then we estimate the representation capacity of the meta-detector for a concept of interest through carefully designed semantic tasks, where higher representation power signifies greater importance of the concept to the investigated class.

We now elaborate on our concept attribution technique. Specifically, in the first step, to acquire a class-specific meta-detector, we also regard feature maps as basis CNN units like the prior art [7,34]. We denote the c^{th} feature map in layer l as A_l^c. Therefore, for class y, we normalize the total importance scores of neurons within A_l^c as its channel importance score (CIS):

$$w_l^c = \frac{1}{B}\Sigma_{j \in P_l^c} s_l^{'j}. \tag{6.8}$$

Here P_l^c is the index set of neurons in A_l^c, and B is a normalizing constant such that $w_l^c \in [0,1]$. We view the fully connected layers with C neurons as C feature maps with a spatial resolution of 1×1. Subsequently, we combine feature maps in layer l with CIS to get the meta-detector:

$$f_l^{'} = \Sigma_c w_l^c \cdot A_l^c. \tag{6.9}$$

It encodes the relevance of various concepts to class y in model cognition.

In the second step, inspired by the work [7,34,66,97,184], we propose two kinds of semantic tasks to evaluate the representation power of the meta-detector. They are tailored for qualitative and quantitative concept attribution, respectively.

For qualitative concept attribution, we devise a generation task. Specifically, we adapt the technology of model visualization [97] to synthesize images, which can maximize the total activation of the meta-detector for class y. The crafted image corresponds to a class impression. It qualitatively depicts the most distinct characteristics of the class concept y in the memory of the model.

For quantitative concept attribution, we reify it as a concept classification task, where we gauge the capability of the meta-detector to distinguish different concepts, and rank the importance of these concepts accordingly. Specifically, we resort to probe datasets with concept labels as in [38, 66]. For each probe image, we first obtain the outputs from the meta-detector as its new representation. Then for a concept of interest, we compute the discrepancy of its samples to the benchmark ones with irrelevant concept tags. The discrepancy quantifies the discriminative power of the meta-detector regarding this concept. We adopt the Maximum Mean Discrepancy (MMD) as the discrepancy metric [45]. Therefore, we sum the MMD values calculated in all the middle layers, and view the normalized results as the importance score of the corresponding concept.

6.3 Experiments

The organization of this section is as follows. We first report the intermediate attacking results in Section 6.3.1. Then we evaluate our feature and concept attribution results in Section 6.3.2 and Section 6.3.3, respectively. Finally, we present some qualitative and quantitative explanations we obtain in Section 6.3.4 and Section 6.3.5, respectively, which showcase the use cases of our framework.

We demonstrate the effectiveness of our framework with three CNNs trained for ImageNet (ILSVRC2012) classification: ResNet-50, GoogLeNet, and VGG-16 [48, 125, 135, 145]. These

Table 6.1: Average top-1 accuracy of different models on clean images and the counterparts perturbed with corresponding global feature occluders.

Model	Clean	Perturbed
ResNet-50	0.8771	0.0973
GoogLeNet	0.8115	0.0907
VGG-16	0.8095	0.1001

models cover representative sorts of models for image classification and have wide application in practice [120]. Therefore, such a model choice can confirm the general applicability of our approach.

We focus on the ImageNet dataset. It is a widely recognized dataset for evaluating explanation techniques [38, 134]. Besides, diverse pre-trained models for ImageNet classification are publicly available. Accordingly, such a dataset choice facilitates fair comparisons with the existing efforts [38, 66]. We adopt the training set of ImageNet to learn global feature occluders so that we can work on the same page as models.

Parameters are settled experimentally. The transformation function t is a composition of: (1) applying uniform random noise within $[-0.04, 0.04]^n$ and (2) random rotation within $[-5°, 5°]$. λ is set to balance the contribution of each term in (6.5).

6.3.1 Attacking Results

We now learn global feature occluders. As experimental demonstrations, we first randomly select 100 classes from all the 1000 classes in the ImageNet dataset [125], and fix these classes for our experiments. We then learn one global feature occluder for each class.

To examine the attack success rates of the resultant global feature occluders, we perturb the images with the corresponding global feature occluders and calculate the average top-1 accuracy of the model on these samples. Table 6.1 reports the results. We can see that our global feature occluders can severely undermine the model performance on perturbed images. Therefore, it is feasible to learn global feature occluders with our approach.

Based on our preliminary experiments, we note that we can obtain fairly accurate global attribution results as long as the attack success rates are high enough (not necessarily 100%). It may be because that global concept attribution should spot concepts that are frequently important for a class in model cognition (*e.g.,* leaves for trees though some trees may not have leaves at present), and have to pay less attention to unrepresentative samples. On the other hand, if occluders fail to achieve high success rates, the performance of our global explanation approach will degenerate. Consequently, we mitigate it by class-specific fine-tuning in our experiments.

6.3.2 Evaluation of the Feature Attribution Results

To examine our feature attribution results—feature importance scores, we propose a distillation test similar to that in [71, 148]. We regard a model we aim to explain as a teacher model. If, for class y, the teacher model owns outstanding accuracy, and our feature importance scores are correct, the derived meta-detector should also possess high discriminative competence for the class concept y. In other words, given the activation of the meta-detector as inputs, a compact student model can differentiate class y from the others. Higher performance of the student model indicates that the feature attribution results are more precise.

Therefore, we implement the distillation test as binary classification tasks in ImageNet. For each class, we first randomly sample a balanced dataset, which consists of the same number of instances from the class and complement ones. We also make sure that the teacher model can correctly recognize all the included images. Then for each sample, we compute the outputs from the meta-detectors of the teacher model, which are flattened as the representation of the image. Finally, we train student models to conduct binary classification on the resultant data as per the original training-validation partition of ImageNet.

For comparison, we also conduct the same distillation test based on the feature attribution results from the state-of-the-art baseline (TCAV) [38, 66]. Specifically, TCAV proposes to perform feature attribution for individual samples with back-propagated gradients. Since TCAV does not acquire a global feature importance score (FIS) for a class, we average its feature attribution results over the whole class of examples as the FIS to test.

Table 6.2 reports the average accuracy of student models over 100 classes. All the student models we exploit are neural networks with three fully connected layers, where there are 32, 16, and 2 neurons, respectively. Student models derived from our method (AfI) can obtain remarkable accuracy, exceeding the gradient-based baseline (TCAV) by a significant margin. It

Table 6.2: The average accuracy of student models derived from different approaches.

Teacher Model	Gradient-based	AfI (Without t)	AfI
ResNet-50	0.8899	0.8918	**0.9592**
GoogLeNet	0.8383	0.8896	**0.9826**
VGG-16	0.8531	0.8679	**0.9468**

validates the effectiveness of our feature attribution mechanism and its superiority to the state-of-the-art benchmark. Moreover, under our method, student models of GoogLeNet manifest the best performance compared to the other teacher models. Since we obtain student models via global explanations of model decisions, it may indicate that GoogLeNet relies on more consistent combinations of features to identify samples from the same class, and thus adopts more category-generic decision modes than the other models.

We run an ablation study to verify the contribution of the transformation function t, where we remove it from (6.5) when learning feature occluders. The performance degradation of the resultant student models confirms the regularization efficacy of t.

6.3.3 Evaluation of the Concept Attribution Results

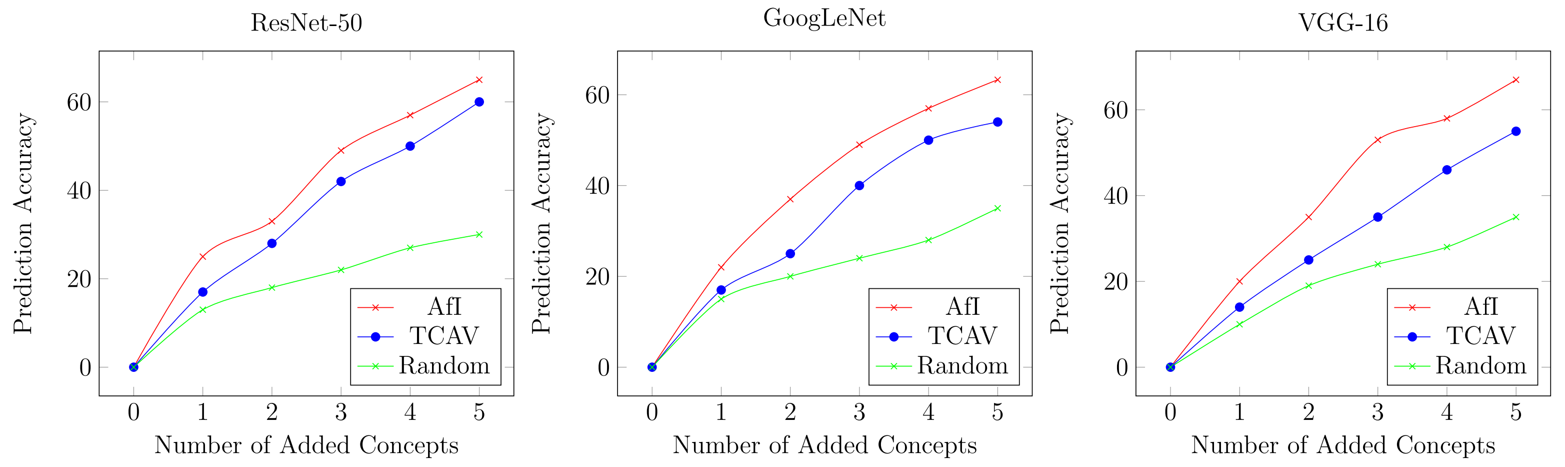
We follow [38] to evaluate our concept attribution results—concept importance scores, since [38] can conduct extensive quantitative assessments with high efficiency. Specifically, [38] regards semantic image segments as concept data. It leads to two metrics: the smallest sufficient concepts (SSCs) and the smallest destroying concepts (SDCs). SSCs are the smallest set of concepts sufficing for models to predict the target class, while SDCs are the smallest concept collections whose absence will incur wrong predictions. More accurate concept importance scores can lead to a more precise estimation of SSCs and SDCs.

We detail the applied evaluation procedure. Given a class, we first segment images of the class and cluster similar segments. Each cluster represents examples for one concept. With these concept data, we then calculate the importance score of each concept, and curate the most important concepts as SSCs and SDCs. Finally, we sequentially add SSCs to a blank image or remove SDCs from the source image as per their importance order. We record the change of model accuracy to examine the concept importance scores we derive. We also test the state-of-the-art baseline (TCAV) under the same setup for comparison [38, 66].

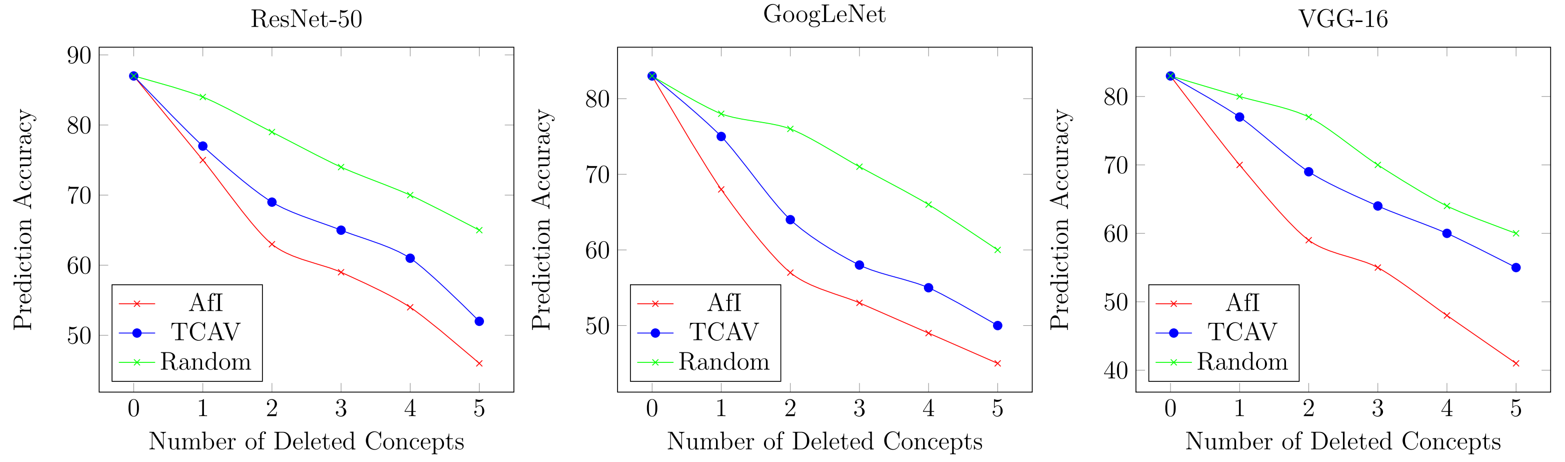
Figure 6.2 exhibits the average result over 100 classes. It confirms that our estimation of SSCs and SDCs is remarkably more accurate than TCAV, as the change of model accuracy during concept adding/removing is more drastic. Therefore, our estimated concept importance scores are more precise than the state-of-the-art benchmark.

6.3.4 Class Concept Visualization

With our qualitative concept attribution strategy, we visualize class concepts captured by a model. Specifically, for a random class, we first separately generate images that can highly activate the meta-detector in each middle layer. We then spot the first layer where class concepts emerge through visual



(a) SSC



(b) SDC

Figure 6.2: Model accuracy variation when we start editing the most important SSCs/SDCs estimated by different approaches. For our method (AfI), the top-5 SSCs are enough to recover over 74% of the original accuracy across all models, while removing the top-5 SDCs can result in a degradation of over 45% of the original accuracy across all models. We also plot the effect of editing concepts in random order for comparison. The concept importance scores derived by our method (AfI) are consistently more accurate than the benchmark (TCAV), since the change of model accuracy is more drastic for our approach.

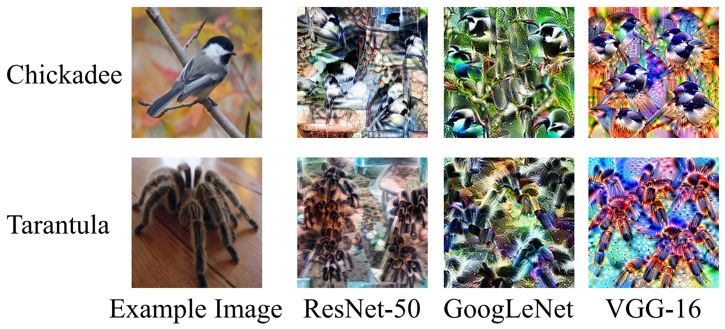


Figure 6.3: Class concepts captured by different models. Example images of the corresponding class are exhibited for better comparison.

Table 6.3: The layer selected to craft class impressions and its original output shape (spatial resolution × channel number).

Model	Layer Name	Original Output Shape
ResNet-50	ResBlock_4c	7 x 7 x 2048
GoogLeNet	Mixed_5b	7 x 7 x 832
VGG-16	Fc_6	1 x 1 x 4096

investigation. The visualization of class concepts from this layer is regarded as class impressions. During the generation of class impressions, except for the total variation penalization, we do not resort to any other natural image priors, such as a generative network [100]. Accordingly, it ensures that the class impressions are only born of the knowledge of the model under inspection.

Figure 6.3 displays some class impressions we obtain, along with example images of the corresponding classes for better comparison. It illustrates that CNNs can capture the most prominent characteristics of image classes, for example, the texture for the tarantula class. Additionally, ResNet-50 appears to better capture and exploit the color property of images than the other models, because the class impressions of ResNet-50 are more similar to raw images of the corresponding classes in terms of their color.

Table 6.3 reports the layer we choose to craft class impressions for each model. We note that in the middle layers, it is non-trivial to infer the links of copious neurons to image categories. Because, unlike the last logit layer, their mappings are not specified during training. Consequently, the competence to uncover class concept embeddings in the middle layers of CNNs further verifies the effectiveness of our framework.

6.3.5 User-defined Concept Attribution

With our quantitative concept attribution scheme, we measure the importance of user-defined concepts to classification. We center on explaining widely-used ResNet-50, which has been less covered in the literature. As experimental examples, we gauge the importance of concepts from three representative groups (*i.e.,* texture, gender, and race) to three classes, respectively. We follow [66] to curate probe concept data [7, 60, 125]. Concretely, for each pair of the concept type and image class, we first randomly select the same number of images as the concept data for each concept. Then we fix a random benchmark set of the same size. We finally compute concept importance scores with the probe data.

Figure 6.4 reports the average result over 100 runs. It validates that CNNs can extract rational grounds for their decisions, like the banded texture for the zebra. However, consistent with the findings of [139], we discover that they also sometimes learn undesirable stereotypes about some classes, such as the

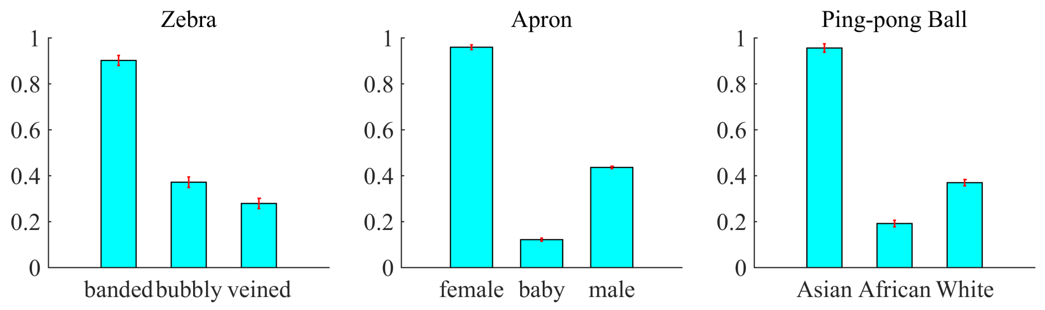


Figure 6.4: Importance scores of different concepts to classification results. Error bars indicate the standard deviation.

relatively stronger positive connections of women to the apron and Asians to the ping-pong ball. Therefore, it demonstrates the use case of our framework in model confirmation and bias revelation.

6.4 Summary

In this chapter, we propose a novel two-step framework for global explanations of CNNs. It first derives feature importance via a novel feature occlusion analysis, and then communicates such information in terms of the importance of human-comprehensible concepts. Empirical results corroborate the effectiveness and superiority of our technique in explaining model behaviors. More crucially, we demonstrate that we can achieve concept attribution via two semantic tasks. It showcases the exciting opportunity to integrate prior feature visualization efforts into our framework, which is a promising direction for future work.

□ **End of chapter.**

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we examine several facets of the robustness and interpretability of deep learning models. In terms of the robustness of DNNs, we cover the detection of real-world corner cases for DNNs and the generation of adversarial samples against undefended and defended DNNs. In terms of the interpretability of DNNs, we work on providing global concept attribution for DNNs.

In Chapter 3, to promote the robustness of DNNs against accidental failures, we develop Deep Validation, the first framework to automatically validate the input/state of DNNs during runtime. It can be employed to detect real-world corner cases for DNNs to enable fail-safe mechanisms. Moreover, we evaluate the performance of our framework with extensive experiments, achieving state-of-the-art detection results under various scenarios.

In Chapter 4, we endeavor to test the robustness of undefended DNNs when adversaries are present. We present a novel transfer-based attack to search for adversarial samples that can fool multiple models simultaneously. Therefore, we can improve the transferability of the crafted adversarial samples and better evaluate the robustness of DNNs against various attacks. We also offer extensive experimental evidence to corroborate the superiority of our method over state-of-the-art attacks.

In Chapter 5, we turn to test the robustness of defended DNNs against adversaries. To this end, we propose a novel solution to enhance the effectiveness of adversarial samples against defended DNNs via adversarial transformations. We conduct extensive experiments to compare the performance of our method with state-of-the-art baselines. Experimental results confirm that our method can consistently outperform state-of-the-art baselines by a considerable margin.

In Chapter 6, we work on improving the interpretability of DNNs with concept attribution. We introduce a novel concept attribution framework that can provide global explanations of DNNs. We systematically evaluate our framework both qualitatively and quantitatively, which shows that our framework can provide markedly more accurate explanations of DNNs than the prior art.

7.2 Future Work

With the deep learning model's growing prevalence in a broad spectrum of real-world applications, especially in safety- and security-sensitive domains, recent years have witnessed an exploding interest in researching the robustness and interpretability of deep learning models. Nevertheless, several emergent research directions are of paramount importance but still under-explored in the literature, which we detail as follows.

- **Synthesizing diverse real-world corner cases**

  In order to test the robustness of DNNs against accidental failures, we need to generate diverse real-world corner cases that can achieve high test adequacy. Unfortunately, to workaround the test-oracle problem [58], existing proposals heavily hinge on the metamorphic testing technique [20, 90]. Specifically, they frequently apply semantic-preserving transformations to seed samples when crafting test cases, which cannot cover different sorts of real-world scenarios.

  Therefore, it is imperative to overcome the excessive reliance on the metamorphic testing technique so that we can generate more diverse test cases. Motivated by the recent advance in generative models [64, 92], we can employ the conditional Generative Adversarial Network (GAN) to synthesize abundant test cases by optimizing a well-designed coverage criterion, which is a promising research direction to explore.

- **Generating adversarial samples without off-the-shelf local models**

  Manufacturing adversarial samples serves as a crucial surrogate to evaluate the robustness of DNNs against adversaries, and transfer-based attacks are one exemplar solution. Transfer-based attacks proceed by synthesizing adversarial samples with off-the-shelf local models as the substitute target and directly utilizing the resultant adversarial samples to attack the victim model. Although such attacks are more practical than query-based attacks that require excessive queries [28, 29], they rely on the availability of off-the-shelf local models that perform similar tasks as the target model. Unfortunately, such pre-trained models may not be accessible to the attackers, for example, the medical diagnosis models for cataracts [118].

Therefore, under this circumstance, attackers have to generate adversarial samples without off-the-shelf local models. Exploring this challenge can identify new security vulnerabilities for a vast body of DNN-based systems, and spur better defenses.

- **Defending against adaptive adversaries**

  As revealed by recent studies [121,170], defenders still seem to lag far behind in the arms race against attackers due to the reliance on reactive defense methodology. As such, although some defenses can perform well against existing attacks, they are often quickly defeated by unforeseen attacks [2].

  Therefore, when we design defense mechanisms, we should instead consider an adaptive adversary, who is aware of the defenses that we deploy [23]. If we can develop a defense approach that functions well in this setup, we can ensure the robustness of DNNs against novel attacks. Unfortunately, defending against adaptive adversaries remains an open problem, and is an urgent mission for future works.

- **Explainable DNNs by design**

  An overwhelming majority of efforts have been devoted to producing interpretations of DNNs. However, it is painful to fix the discovered pitfalls of DNNs when the explanations show that the DNNs' predictions are unreasonable, such as the racial bias in face recognition [19].

It thus calls for attention to develop DNNs that are explainable by design [177]. The reasoning process of explainable DNNs is explicitly encoded. Therefore, at inference, the model affords both the decision and the corresponding logic behind its decision. As such, we can more conveniently comprehend the logic of DNNs and incorporate human knowledge into models when we identify irrational logic. These prominent virtues make the development of explainable DNNs an imperative and intriguing topic to study in future work.

□ **End of chapter.**

Appendix

The publications during my Ph.D. study are listed as follows.

1. **Weibin Wu**, Yuxin Su, Michael R. Lyu, and Irwin King. "Improving the Transferability of Adversarial Samples with Adversarial Transformations." IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

2. **Weibin Wu**, Yuxin Su, Xixian Chen, Shenglin Zhao, Irwin King, Michael R. Lyu, and Yu-Wing Tai. "Boosting the Transferability of Adversarial Samples via Attention." IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

3. **Weibin Wu**, Yuxin Su, Xixian Chen, Shenglin Zhao, Irwin King, Michael R. Lyu, and Yu-Wing Tai. "Towards Global Explanations of Convolutional Neural Networks with Concept Attribution (**Oral Presentation**)." IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

4. **Weibin Wu**, Hui Xu, Sanqiang Zhong, Michael R. Lyu, and Irwin King. "Deep Validation: Toward Detecting Real-world Corner Cases for Deep Neural Networks." 49th Annual IEEE/IFIP International Conference on Dependable

Systems and Networks (DSN), 2019.

5. Hui Xu, Zhuangbin Chen, **Weibin Wu**, Zhi Jin, Sy-Yen Kuo, and Michael R. Lyu. "NV-DNN: Towards Fault-Tolerant DNN Systems with N-Version Programming." 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2019.

We note that the first four papers are partially involved in this thesis.

Bibliography

[1] Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84, 1990.

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

[3] A. Avizienis, J.-C. Laprie, B. Randell, et al. *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[4] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[5] R. Balestriero et al. A spline theory of deep networks. In *International Conference on Machine Learning (ICML)*, pages 383–392, 2018.

[6] V. R. Basili and D. M. Weiss. A methodology for collecting valid software engineering data. Technical report, Naval Research Lab Washington DC, 1983.

[7] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network Dissection: Quantifying interpretability of deep visual representations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6541–6549, 2017.

[8] K. Beaver. *Hacking for dummies.* John Wiley & Sons, 2007.

[9] R. Benenson. Classification datasets results. Accessed: July 2018.

[10] A. N. Bhagoji, W. He, B. Li, and D. Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *The European Conference on Computer Vision (ECCV)*, pages 158–174. Springer, 2018.

[11] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[12] A. Boopathy, S. Liu, G. Zhang, C. Liu, P.-Y. Chen, S. Chang, and L. Daniel. Proper network interpretability helps adversarial robustness in classification. In *International Conference on Machine Learning*, pages 1014–1023. PMLR, 2020.

[13] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[14] M. L. M. L. Bushnell. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Frontiers in electronic testing. Kluwer Academic, Boston ; London, 2000.

[15] J. Campos, A. Riboira, A. Perez, and R. Abreu. Gzoltar: an eclipse plug-in for testing and debugging. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pages 378–381, 2012.

[16] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, and A. Madry. On evaluating adversarial robustness. *arXiv:1902.06705*, 2019.

[17] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[18] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

[19] J. G. Cavazos, P. J. Phillips, C. D. Castillo, and A. J. O'Toole. Accuracy comparison across face recognition algorithms: Where are we on measuring race bias? *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2020.

[20] T. Y. Chen, S. C. Cheung, and S. M. Yiu. Metamorphic testing: a new approach for generating next test cases. Technical report, Technical Report HKUST-CS98-01, De-

partment of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.

[21] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[22] J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320, 2019.

[23] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 311–326. Springer, 1999.

[24] M. Curphey, D. Endler, W. Hau, S. Taylor, T. Smith, A. Russell, G. McKenna, R. Parke, K. McLaughlin, N. Tranter, et al. A guide to building secure web applications. *The Open Web Application Security Project*, 1(1), 2002.

[25] P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6967–6976, 2017.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[27] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the

missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 592–603, 2018.

[28] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[29] Y. Dong, T. Pang, H. Su, and J. Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] F. F. dos Santos, L. Draghetti, L. Weigel, L. Carro, P. Navaux, and P. Rech. Evaluation and mitigation of soft-errors in neural network-based object detection in three gpu architectures. In *Dependable Systems and Networks Workshop (DSN-W), 2017 47th Annual IEEE/IFIP International Conference on*, pages 169–176. IEEE, 2017.

[31] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. Bose, N. Dubash, and S. Podder. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 118–128. ACM, 2018.

[32] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. In *NIPS Machine Learning and Computer Security Workshop*, 2018.

[33] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[34] R. Fong and A. Vedaldi. Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8730–8738, 2018.

[35] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *International Conference on Computer Vision (ICCV)*, pages 3449–3457. IEEE, 2017.

[36] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[37] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019.

[38] A. Gohorbani, J. Wexler, J. Zou, and B. Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[39] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[40] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[41] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[42] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[43] B. Goodman and S. Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[44] Y. Goyal, U. Shalit, and B. Kim. Explaining classifiers with Causal Concept Effect (CaCE). *arXiv preprint arXiv:1907.07165*, 2019.

[45] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[46] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning (ICML)*, 2019.

[47] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.

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

[49] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *The European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

[50] L. A. Hendricks, K. Burns, K. Saenko, T. Darrell, and A. Rohrbach. Women also snowboard: Overcoming bias in captioning models. In *The European Conference on Computer Vision (ECCV)*, pages 793–811. Springer, 2018.

[51] K. L. Heninger. Specifying software requirements for complex systems: New techniques and their application. *IEEE Transactions on Software Engineering*, (1):2–13, 1980.

[52] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[53] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al. CNN architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135. IEEE, 2017.

[54] G. E. Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

[55] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[56] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[57] H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran. On the limitation of convolutional neural networks in recognizing negative images. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*, pages 352–358. IEEE, 2017.

[58] W. E. Howden. Theoretical and empirical studies of program testing. In *Proceedings of the 3rd international conference on Software engineering*, pages 305–311. IEEE Press, 1978.

[59] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[60] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled Faces in the Wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[61] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.

[62] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning (ICML)*, pages 2142–2151, 2018.

[63] X. Jia, X. Wei, X. Cao, and H. Foroosh. ComDefend: An efficient image compression model to defend adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[64] M. Kang and J. Park. ContraGAN: Contrastive Learning for Conditional Image Generation. 2020.

[65] A. Khakzar, S. Albarqouni, and N. Navab. Learning interpretable features via adversarially robust optimization. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 793–800. Springer, 2019.

[66] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *International Conference on Machine Learning (ICML)*, pages 2673–2682. PMLR, 2018.

[67] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[68] V. B. Krishna, M. Rausch, B. E. Ujcich, I. Gupta, and W. H. Sanders. Remax: Reachability-maximizing p2p detection of erroneous readings in wireless sensor networks.

In *Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on*, pages 321–332. IEEE, 2017.

[69] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[70] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. Curran Associates, Inc., 2012.

[71] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *International Conference on Computer Vision (ICCV)*, pages 365–372. IEEE, 2009.

[72] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *ICLR Workshop*, 2017.

[73] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2017.

[74] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, et al. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, 2018.

[75] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[76] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[77] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[78] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2020.

[79] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[80] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 1, 2017.

[81] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations (ICLR)*, 2017.

[82] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, and W. Wen. Feature Distillation: DNN-Oriented JPEG compression against adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[83] S.-C. Lo, S.-L. Lou, J.-S. Lin, M. T. Freedman, M. V. Chien, and S. K. Mun. Artificial convolution neural network techniques and applications for lung nodule detection. *IEEE Transactions on Medical Imaging*, 14(4):711–718, 1995.

[84] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4765–4774, 2017.

[85] X. Luo, R. Zhan, H. Chang, F. Yang, and P. Milanfar. Distortion agnostic deep watermarking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13548–13557, 2020.

[86] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

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

[88] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[89] S. Majumdar. Densenet, 2017. Accessed: March 2018.

[90] W. M. McKeeman. Differential testing for software. *Digital Technical Journal*, 10(1):100–107, 1998.

[91] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshop Track*, 2013.

[92] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[93] D. Mishkin and J. Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[94] C. Molnar. *Interpretable machine learning.* Lulu. com, 2020.

[95] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[96] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1765–1773, 2017.

[97] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog*, 2015. Retrieved: October 2018.

[98] M. Naseer, S. Khan, M. Hayat, F. S. Khan, and F. Porikli. On generating transferable targeted perturbations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[99] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with

unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[100] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3387–3395. Curran Associates, Inc., 2016.

[101] A. Nguyen, J. Yosinski, and J. Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *ICML Visualization for Deep Learning Workshop*, 2016.

[102] B. Nie, J. Xue, S. Gupta, T. Patel, C. Engelmann, E. Smirni, and D. Tiwari. Machine learning models for gpu error prediction in a large scale hpc system. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 95–106. IEEE, 2018.

[103] M. A. Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

[104] O. Nuriel, S. Benaim, and L. Wolf. Permuted adain: Reducing the bias towards global statistics in image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9482–9491, June 2021.

[105] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The

building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.

[106] OWASP Foundation. Open web application security project: Data validation, 2013. Accessed: September 2018.

[107] N. Papernot and P. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[108] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[109] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *The IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387, 2016.

[110] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016.

[111] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[112] K. Pei, Y. Cao, J. Yang, and S. Jana. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18. ACM, 2017.

[113] K. Pei, Y. Cao, J. Yang, and S. Jana. Towards practical verification of machine learning: The case of computer vision systems. *arXiv preprint arXiv:1712.01785*, 2017.

[114] C. Pham, D. Chen, Z. Kalbarczyk, and R. K. Iyer. Cloudval: A framework for validation of virtualization environment in cloud infrastructure. In *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 189–196. IEEE, 2011.

[115] H. Pham, Z. Dai, Q. Xie, and Q. V. Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11557–11568, June 2021.

[116] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[117] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 2018.

[118] T. Pratap and P. Kokil. Computer-aided diagnosis of cataract using deep transfer learning. *Biomedical Signal Processing and Control*, 53:101533, 2019.

[119] J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *ICML Workshop*, 2017.

[120] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[121] K. Ren, T. Zheng, Z. Qin, and X. Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[122] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144. ACM, 2016.

[123] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.

[124] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.

[125] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[126] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. In *International Conference on Learning Representations (ICLR)*, 2016.

[127] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa. Triplet probabilistic embedding for face verification and clustering. In *The International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8. IEEE, 2016.

[128] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[129] J. Schrouff, S. Baur, S. Hou, D. Mincu, E. Loreaux, R. Blanes, J. Wexler, A. Karthikesalingam, and B. Kim. Best of both worlds: local and global explanations with human-understandable concepts. *arXiv preprint arXiv:2106.08641*, 2021.

[130] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE, 2017.

[131] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshop*, 2014.

[132] Y. Sharma, T.-D. Le, and M. Alzantot. CAAD 2018: Generating transferable adversarial examples. *arXiv preprint arXiv:1810.01268*, 2018.

[133] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning (ICML)*, pages 3145–3153. PMLR, 2017.

[134] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations (ICLR)*, 2014.

[135] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[136] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. SmoothGrad: Removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[137] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR Workshop*, 2015.

[138] J. Stewart. Tesla's autopilot was involved in another deadly car crash, March 2018.

[139] P. Stock and M. Cisse. ConvNets and ImageNet beyond accuracy: Understanding mistakes and uncovering biases. In *The European Conference on Computer Vision (ECCV)*, pages 498–512, 2018.

[140] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore. Deepconcolic: Testing and debugging deep neural networks. In *2019 IEEE/ACM 41st International*

Conference on Software Engineering: Companion Proceedings (ICSE-Companion), pages 111–114. IEEE, 2019.

[141] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, pages 3319–3328. PMLR, 2017.

[142] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[143] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[144] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[145] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[146] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE International Conference on Computer Vision (ICCV)*, 2016.

[147] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[148] G. Tao, S. Ma, Y. Liu, and X. Zhang. Attacks meet interpretability: Attribute-steered detection of adversarial samples. In *Advances in Neural Information Processing Systems (NIPS)*, pages 7717–7728, 2018.

[149] Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5087, 2015.

[150] Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, pages 303–314. ACM, 2018.

[151] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021.

[152] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

[153] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018.

[154] T.S. Why uber's self-driving car killed a pedestrian, May 2018.

[155] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[156] V. N. Vapnik. *Statistical learning theory.* Adaptive and learning systems for signal processing, communications, and control. John Wiley & Sons, New York, 1998.

[157] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[158] J. Wang, H. Ding, F. A. Bidgoli, B. Zhou, C. Iribarren, S. Molloi, and P. Baldi. Detecting cardiovascular disease from mammograms with deep learning. *IEEE Trans. Med. Imaging*, 36(5):1172–1181, 2017.

[159] L. Wang, E. Wong, and D. Xu. A threat model driven approach for security testing. In *Third International Workshop on Software Engineering for Secure Systems (SESS'07: ICSE Workshops 2007)*, pages 10–10. IEEE, 2007.

[160] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[161] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[162] W. Wu, Y. Su, X. Chen, S. Zhao, I. King, M. R. Lyu, and Y.-W. Tai. Boosting the transferability of adversarial samples via attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1161–1170, 2020.

[163] W. Wu, Y. Su, X. Chen, S. Zhao, I. King, M. R. Lyu, and Y.-W. Tai. Towards global explanations of convolutional neural networks with concept attribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8652–8661, 2020.

[164] W. Wu, Y. Su, M. R. Lyu, and I. King. Improving the transferability of adversarial samples with adversarial transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9024–9033, 2021.

[165] W. Wu, H. Xu, S. Zhong, M. R. Lyu, and I. King. Deep Validation: Toward detecting real-world corner cases for deep neural networks. In *International Conference on Dependable Systems and Networks (DSN)*, pages 125–137. IEEE, 2019.

[166] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.

[167] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille. Improving transferability of adversarial examples with input diversity. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[168] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[169] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2174–2182, 2017.

[170] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

[171] W. Xu, D. Evans, and Y. Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of the 2018 Network and Distributed Systems Security Symposium (NDSS)*, 2018.

[172] L. Yuan, Q. Hou, Z. Jiang, J. Feng, and S. Yan. Volo: Vision outlooker for visual recognition. *arXiv preprint arXiv:2106.13112*, 2021.

[173] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[174] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *The European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[175] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. *arXiv preprint arXiv:2106.04560*, 2021.

[176] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 132–142. ACM, 2018.

[177] Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018.

[178] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu. Interpreting CNNs via decision trees. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6261–6270, 2019.

[179] X. Zhang, A. Solar-Lezama, and R. Singh. Interpreting neural network judgments via minimal, stable, and symbolic corrections. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4874–4885, 2018.

[180] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems (NIPS)*, pages 649–657, 2015.

[181] Z. Zhang, P. Wang, H. Guo, Z. Wang, Y. Zhou, and Z. Huang. Deepbackground: Metamorphic testing for deep-learning-driven image recognition systems accompanied by background-relevance. *Information and Software Technology*, page 106701, 2021.

[182] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene CNNs. In *International Conference on Learning Representations (ICLR)*, 2015.

[183] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016.

[184] B. Zhou, Y. Sun, D. Bau, and A. Torralba. Interpretable basis decomposition for visual explanation. In *The European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

[185] W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, and Y. Yang. Transferable adversarial perturbations. In *The European Conference on Computer Vision (ECCV)*, 2018.

[186] H. Zhu, P. A. Hall, and J. H. May. Software unit test coverage and adequacy. *Acm computing surveys (csur)*, 29(4):366–427, 1997.

[187] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018.

[188] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *International Conference on Learning Representations (ICLR)*, 2017.