Analyzing Adversarial Examples: A Framework to Study Adversary Knowledge

by

Lucas Fenaux

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Adversarial examples are malicious inputs to trained machine learning models supplied to trigger a misclassification. This type of attack has been studied for close to a decade, and we find that there is a lack of study and formalization of adversary knowledge when mounting attacks. This has yielded a complex space of attack research with hard-to-compare threat models and attacks. We solve this in the image classification domain by providing a theoretical framework to study adversary knowledge inspired by work in order theory. We present an adversarial example game, based on cryptographic games, to standardize attack procedures. We survey recent attacks in the image classification domain that showcase the current state of adversarial example research. Together with our formalization, we compile results that both confirm existing beliefs about adversary knowledge, such as the potency of information about the attacked model as well as allow us to derive new conclusions on the difficulty associated with the white-box and transferable threat models, for example, transferable attacks might not be as difficult as previously thought.

Acknowledgements

I would like to thank all the people who made this thesis possible. My family for their unending support. My friends, from the CrySP lab and others, for helping me retain my sanity through this arduous process. My supervisor for his valuable advice, help, feedback, and guidance.

Dedication

I dedicate this work to my family.

Table of Contents

List of Tables x

List of Figures xii

1 Introduction 1

2 Background 3

2.1 Grammar . . . 3

2.2 Machine Learning . . . 3

2.3 Security . . . 4

2.4 Adversarial Examples . . . 4

3 Methodology 6

4 Survey 8

4.1 Paper Summaries . . . 8

4.1.1 Object-based Diverse Input Attack . . . 8

4.1.2 Geometry-Aware Attack . . . 9

4.1.3 Large Geometric Vicinity Attack . . . 10

4.1.4 Pixle Attack . . . 10

4.1.5 MASSA Attack . . . 11

4.1.6 AI-FGTM Attack . . . 12

4.1.7 SSAH Attack . . . 14
4.1.8 F-Attack . . . 16
4.1.9 BIA Attack . . . 18
4.1.10 ACG Attack . . . 19
4.1.11 Admix Attack . . . 20
4.1.12 ATA Attack . . . 21
4.1.13 Shadow Attack . . . 22
4.1.14 DAPatch Attack . . . 23
4.1.15 S^2I Attack . . . 25
4.1.16 AI-GAN Attack . . . 26
4.1.17 AEG Attack . . . 27
4.1.18 ACA Attack . . . 29
4.1.19 DiffAttack . . . 30
4.1.20 A^3 Attack . . . 31

5 Formalization **34**
5.1 Introduction . . . 34
5.2 A formal approach to understanding and modeling adversary action space . . . 36
5.2.1 Adversary action space . . . 37
5.2.2 Bounding the adversary action space: the distinguisher . . . 39
5.2.3 Definitions . . . 41
5.3 Categorization of threat models . . . 42
5.3.1 Information Extraction Oracles . . . 42
5.3.2 Information Categories . . . 44
5.3.3 Information Hasse Diagram . . . 46
5.4 Game . . . 55
5.4.1 Definitions . . . 55
5.4.2 Game Diagram . . . 57
5.4.3 Measuring Success . . . 59
5.4.4 Showcase . . . 60

6 Conversion **65**

6.1 Papers . . . 65

6.1.1 Object-based Diverse Input Attack . . . 65

6.1.2 Geometry-Aware Attack . . . 66

6.1.3 Large Geometric Vicinity Attack . . . 66

6.1.4 Pixle Attack . . . 67

6.1.5 MASSA Attack . . . 67

6.1.6 AI-FGTM Attack . . . 67

6.1.7 SSAH Attack . . . 68

6.1.8 F-Attack . . . 68

6.1.9 BIA Attack . . . 69

6.1.10 ACG Attack . . . 69

6.1.11 Admix Attack . . . 69

6.1.12 ATA Attack . . . 70

6.1.13 Shadow Attack . . . 70

6.1.14 DAPatch Attack . . . 71

6.1.15 S^2I Attack . . . 71

6.1.16 AI-GAN Attack . . . 71

6.1.17 AEG Attack . . . 71

6.1.18 ACA Attack . . . 72

6.1.19 DiffAttack . . . 72

6.1.20 A^3 Attack . . . 73

6.2 Compiled tables . . . 73

7 Results **86**

7.1 Content-Preserving untargeted attacks . . . 87

7.1.1 Attacks on the ILSVRC2012 Dataset . . . 87

7.1.2 Attacks on the GTSRB dataset . . . 89

7.1.3 Attacks on the NIPS 2017 dataset . . . 91
7.2 Indistinguishable untargeted attacks . . . 93
7.2.1 Attacks on the NIPS 2017 dataset . . . 93
7.2.2 Attacks on the CIFAR10 dataset . . . 95
7.2.3 Attacks on the CIFAR100 dataset . . . 100
7.2.4 Attacks on the ImageNet dataset . . . 104
7.3 Takeaways . . . 111

8 Conclusion **120**

References **122**

List of Tables

4.1 List of attacked models by A^3 on CIFAR10 33
4.2 List of attacked models by A^3 on CIFAR100 33

5.1 PGD attacks knowledge table 63

6.1 Attack Information Table part 1 74
6.2 Attack Information Table part 2 75
6.3 Attack properties part 1 76
6.4 Attack properties part 2 77
6.5 List of Datasets used to evaluate attacks part 1 78
6.6 List of Datasets used to evaluate attacks part 2 79
6.7 List of attacked models by more than two papers 80
6.8 List of attacked models by two papers 81
6.9 List of attacked defended models by more than one paper 82
6.10 List of attacked models by only a single paper part 1 83
6.11 List of attacked models by only a single paper part 2 84
6.12 List of attacked defended models by one paper 85

7.1 Salient situation-dataset pairs with more than one attack 86
7.2 ILSVRC2012 dataset content-preserving results 88
7.3 GTSRB dataset content-preserving undefended results 90
7.4 GTSRB dataset content-preserving defended results 91

7.5 NIPS 2017 dataset content-preserving transferable undefended results . . . 94
7.6 NIPS 2017 dataset transferable defended results . . . 95
7.7 NIPS 2017 dataset content-preserving white-box undefended results . . . 96
7.8 NIPS 2017 dataset content-preserving transferable and white-box ASR comparison . . . 97
7.9 NIPS 2017 dataset content-preserving transferable and white-box score comparison . . . 98
7.10 NIPS 2017 dataset indistinguishable transferable defended results . . . 101
7.11 NIPS 2017 dataset indistinguishable transferable and white-box undefended comparison . . . 102
7.12 CIFAR10 dataset indistinguishable BIA (A) undefended results . . . 102
7.13 CIFAR10 dataset indistinguishable AEG (B) undefended results . . . 105
7.14 CIFAR10 dataset indistinguishable AEG (B) defended results . . . 106
7.15 CIFAR10 dataset indistinguishable SSAH (A) and A^3 comparison defended 106
7.16 CIFAR10 dataset indistinguishable A^3 and ACG defended comparison. . . . 107
7.17 CIFAR10 dataset indistinguishable AEG (B) & Pixle comparison defended 108
7.18 CIFAR100 dataset indistinguishable SSAH (A) defended results . . . 108
7.19 CIFAR100 dataset indistinguishable A^3 defended results . . . 109
7.20 CIFAR100 dataset indistinguishable A^3 and ACG defended comparison . . . 110
7.21 CIFAR100 dataset indistinguishable BIA (A) undefended results . . . 110
7.22 ImageNet dataset indistinguishable Blue ASR comparison undefended . . . 111
7.23 ImageNet dataset indistinguishable Blue score comparison undefended . . . 112
7.24 ImageNet dataset indistinguishable LGV (B) results undefended . . . 112
7.25 ImageNet dataset indistinguishable ATA (B) results undefended . . . 113
7.26 ImageNet dataset indistinguishable ACG results defended . . . 113
7.27 ImageNet dataset indistinguishable LGV (B) & SSAH (A) comparison undefended . . . 114
7.28 ImageNet dataset indistinguishable BIA (C) results undefended . . . 114
7.29 ImageNet dataset indistinguishable MASSA & Pixle comparison undefended 115

List of Figures

5.1 Model Oracle Hasse Diagram 48
5.2 Data Oracle Hasse Diagram 52
5.3 Train Oracle Hasse Diagram 53
5.4 Defense Hasse Diagram 55
5.5 Relative Performance Score for various benign and adversary ESRs 61

7.1 ILSVRC 2012 dataset content-preserving attacks 87
7.2 ILSVRC2012 dataset content-preserving results. 89
7.3 GTSRB dataset content-preserving attacks 89
7.4 GTSRB dataset content-preserving results. 92
7.5 NIPS 2017 dataset content-preserving attacks 93
7.6 NIPS 2017 dataset content-preserving results. 99
7.7 NIPS 2017 dataset indistinguishable attacks 100
7.8 NIPS 2017 dataset indistinguishable results. 103
7.9 CIFAR10 dataset indistinguishable attacks 104
7.10 CIFAR10 dataset indistinguishable attack results. 116
7.11 CIFAR100 dataset indistinguishable attacks 117
7.12 CIFAR100 dataset indistinguishable attack results. 117
7.13 ImageNet dataset indistinguishable attacks 118
7.14 CIFAR10 dataset indistinguishable attack results. 119

Chapter 1

Introduction

Currently, an estimated 46% of companies worldwide have deployed customized ML machine learning (ML) models for business purposes [1]. Many are deployed in critical environments like healthcare [2], self-driving cars [3][4], aerospace and aviation [5], and much more. Model failures in these settings can directly incur grave consequences on human life. While there exists a large body of literature on the safe and secure deployment of these models, the proposed solutions remain mainly academic, difficult to implement and replicate, and fall short of the security standards we would typically expect from critical systems. Despite all of this, the deployment of ML models continues to grow. This surge in deployment, while promising, introduces a pressing concern: the vulnerability of these models to security flaws, both known and yet to be uncovered. The potential consequences of such vulnerabilities extend far beyond the limits of academia, with implications economic and societal implications.

Adversarial example attacks are one such attack, originally discovered by Szegedy et al. ([6]) in 2014. They have shown the existence of easily craftable imperceptible vulnerabilities in image classifiers. These vulnerabilities have taken the form of perturbations, imperceptible to the human eye, that can significantly alter a model's prediction on an otherwise benign input. Additionally, the low requirements for mounting adversarial attacks make it a prime vulnerability. Prior work in the domain of adversarial examples suggests a notable advantage for attackers, with the majority of attacks capable of, to some degree, impeding model performance. Among the existing defenses, few techniques have demonstrated a degree of robustness when it comes to safeguarding machine learning models, for example, adversarial training methods [7] and ensemble approaches [8] (and their combination). However, even in their strongest forms and under strict threat models that greatly limit the attacker's capabilities, current defenses still fail to provide adequate protection.

To remedy this, a branch of research on provably robust defenses emerged [9]. However, in practice, this is not yet a feasible approach since it increases the computational cost of model inference by several orders of magnitude. Since provable defenses are impractical, a majority of the field has focused on improving the quality of empirical defenses. Nevertheless, the absence of a reliable evaluation standard or assessment method for overall performance has resulted in an ongoing competitive cycle between attackers and defenders based on empirical experiments. Similarly, the field of encryption went through the same process last century. Establishing a systematic and well-defined theoretical framework was essential in stabilizing the space and yielded security protocols that endured. In security, information and knowledge are key. A complete theoretical framework is crucial to creating this necessary evaluation standard and shifting this competitive cycle into a manageable challenge for adversarial example research.

In our work, we concentrate on the image classification domain and aim to address a significant challenge: the current lack of emphasis on rigorous assessment of adversary knowledge in threat models. Adversary knowledge is one of the key components of a well-defined threat model and has long been overlooked and neglected by both attack and defense works. We study the current lack of formalization of adversary knowledge in adversarial example research and how it affects attack comparability and performance. Defending against attackers with ill-defined capabilities places an additional burden on the defender. Whereas attacking with ill-defined knowledge hinders reproducibility and comparability. We develop this formalization as a theoretical framework to model various adversarial example attacks. We conduct a thorough review of recent attacks in the image classification domain, offering an up-to-date overview of the current landscape of adversaries in this specific context. This formalization allows us to define and categorize attacks and their associated threat models, in turn, laying a foundation for defenses. Additionally, combined with our survey of attacks, we measure the effect of the information available to an attacker on an attack's performance. We compare various threat models and confirm, with falsifiable results, the generally held belief that certain categories of information, like information related to the attacked classification model, have a disproportionate effect on attack performance. However, we also find that a lack of information in that category, in the case of transferable attacks, can be compensated by the use of additional information such as information related to the training data or the training process, to yield attacks that are almost just as potent.

Chapter 2

Background

2.1 Grammar

Our entire structure is based on the following primitives:

- Real numbers: $\mathbb{R}$
- The $\perp$ symbol.

and the following objects built on our primitives:

- Sets of primitives :$\{\}$, $\mathcal{A}, \mathcal{B}$
- Functions over primitives: f, g

2.2 Machine Learning

Using our primitives, we can define data and the rest of the machine learning pipeline:

Definition 1 (Data)**.** *We define data as the set of all possible inputs to a machine learning model:* $\mathcal{I} \subset \mathbb{R}^*$ *and the set of all possible labels* $\mathcal{L} \subset \mathbb{R}^* \cup \{\perp\}$.

Definition 2 (Ground-truth function)**.** *We define the ground-truth function* $gt : \mathcal{I} \to \mathcal{L}$ *as the function that assigns its true label to an input sample. (In the case where the label does not have a true label, it returns* $\perp$*).*

Definition 3 (Machine Learning Model)**.** *We define a machine learning (ML) model as a parametrized function* $M : \mathcal{I} \rightarrow \mathcal{L}$*.*

Definition 4 (ML Model Parameters)**.** *The parameters of an ML model (a parametrized function as defined above) are* $\theta_M \in \mathbb{R}^*$

Definition 5 (Set of all ML Models)**.** *The set of all ML models is* $\mathcal{M}$ *which is a set of parametrized functions.*

If further terms describing model processes are used and not defined in this work, one can refer to the National Institute of Standards and Technology (NISTIR) [10] for their definition.

2.3 Security

Definition 6 (Threat Model)**.** *NISTIR [10] defined a threat model as adversarial goals, knowledge, and capabilities that a system is designed to defend against.*

Definition 7 (White-box)**.** *We define white-box threat model as a threat model where the attacker has access to the attacked (also called target) model's parameters.*

Definition 8 (Query-based)**.** *We define query-based threat model as a threat model where the attacker can query the target model with samples.*

2.4 Adversarial Examples

The National Institute of Standards and Technology (NISTIR) [10] defines adversarial examples as:

Definition 9 (Adversarial Example)**.** *ML input sample formed by applying a small but intentionally worst-case perturbation (see adversarial perturbation) to a clean (benign) example, such that the perturbed input causes a learned model to output an incorrect answer.*

Depending on the adversary's goal, an incorrect answer will have a different meaning. The two main attack goals that can be found in the field are untargeted and targeted attacks.

Definition 10 (Targeted Adversarial Example)**.** *An input sample $i \in \mathcal{I}$ and its associated label $gt(i) \in \mathcal{L}$ is said to be a targeted adversarial example if for a fixed chosen label $c \in \mathcal{L}, c \neq gt(i)$, it triggers a learned model to output label c.*

Definition 11 (Untargeted Adversarial Example)**.** *An input sample $i \in \mathcal{I}$ and its associated label $gt(i) \in \mathcal{L}$ is said to be an untargeted adversarial example if it triggers a learned model to output a label $l \neq gt(i), l \in \mathcal{L}$*

Definition 12 (Grounded Adversarial Example)**.** *An input sample $i \in \mathcal{I}$ and its associated label $gt(i) \in \mathcal{L}$ is said to be a grounded adversarial example if it was created using a benign sample $x \in \mathcal{I}$ and its associated label $gt(x) \in \mathcal{L}$.*

Defining the adversarial nature of an input sample whose origin is unknown is difficult, as it is context-dependent. When a model misclassifies an input sample, determining whether it was due to the adversarial nature of said sample or whether it was a shortcoming of the model itself on a benign sample is a complicated task. Adversarial examples exploit a model's shortcomings in modeling the ground-truth function gt. Such shortcomings can be model-specific or sometimes shared amongst many if not most models. In this class, we can see two kinds of adversarial examples arise: universal adversarial perturbations and transferable adversarial examples.

Definition 13 (Universal Adversarial Perturbation)**.** *A universal adversarial perturbation is an input perturbation vector $p \in \mathbb{R}^*$ that when applied to any input sample $i \in \mathcal{I}$, will cause the input sample to become a (targeted or untargeted) adversarial example.*

Definition 14 (Transferable Adversarial Example)**.** *A transferable adversarial example is an adversarial example that is crafted to successfully trigger a learned model M_1 to output an incorrect answer and also successfully triggers a set of other learned models $\{M_2, M_3, ...\}$ to output an incorrect answer.*

Chapter 3

Methodology

To survey the field of adversarial examples in the image classification domain, we first use Google Scholar for the initial search and search "machine learning adversarial examples." We focus on recent papers, papers published in 2022 or after to narrow the field of the search and allow for an up-to-date perspective of the field of image classification adversarial research. We limit the search to the first ten pages of results, considering that the initial pages typically contain papers of greater significance and the field of adversarial example research is quite vast. From there, we identify relevant adversarial example attack papers/defenses from the discovered papers. This yields seventy-three papers. ([11–83])

We then used our expertise in the field to read the papers and identify the ones of high quality. To narrow down our search, we select papers that provide us with a diverse set of studied threat models, to better showcase our framework. This process provides us with thirteen papers from 2022 and later ([11, 14, 15, 37–40, 45, 54, 65, 66, 82, 83]). To avoid redundancy between papers whose objectives and methodology are similar, we pick a representative paper from each such group of papers.

Furthermore, we identify attack papers presented at conferences with a distinguished reputation within the machine learning community. The selection of conferences was based on their standing and recognition, ensuring that the surveyed papers were from reputable sources. To avoid redundancy, only papers that had not been previously surveyed are considered. This yields two more papers that we included in our survey ([64, 84]). Furthermore, we extend our search to include papers investigating the combination of adversarial examples with stable diffusion. This is motivated by the contemporary nature of stable diffusion-based attack papers, which represent a new approach to leveraging image domain information for conducting attacks. The objective is to assess whether these newly

developed attacks would provide valuable insights into the evolving field of adversarial example research on image classification. We selected another two papers ([85, 86]) out of seven ([85–90]). Three of them were not on classification tasks, one was a defense paper, and one other was of lesser quality. Finally, we include three papers from 2020 ([91]) and 2021 ([92, 93]) that had not been surveyed by other work and that present an interesting approach from our perspective, for a total of twenty papers.

For each paper we select, we gather the paper's results. The focus was specifically on the results presented by the authors themselves or results referenced from external sources, such as RobustBench [94] or defense papers. This ensures the accuracy of the results we report.

The inclusion and exclusion criteria are defined to ensure the relevance of the selected papers to our research objectives. Papers were included if they provided insights into novel adversarial attack techniques or presented unique perspectives on the vulnerabilities of machine learning models.

For our formalization, we were inspired by cryptographic security games and their associated proofs. For example, the game-based proofs of CPA and CCA security. We also took inspiration from Hasse Diagrams, from order theory, to create, order and prove our information categories.

Chapter 4

Survey

4.1 Paper Summaries

4.1.1 Object-based Diverse Input Attack

The Object-based Diverse Input (ODI) Attack [45] is a transferable attack focused on input transformation. It is an attack used to boost the transferability of targeted adversarial examples generated using other attacks. They project the generated adversarial image onto carefully crafted 3D-rendered objects. Taking both the selected 3D mesh and the adversarial image, they use a differentiable renderer to project the image onto the 3D mesh and apply a random viewpoint and lighting. Finally, they add a randomly generated background to the image as the selected 3D mesh might not fill the entire image.

Since the rendered is differentiable, it can also be directly incorporated into the adversarial generation process of iterative attacks by adding it to the gradient computation step as follows:

$$\hat{g}_{t+1} = \nabla_{x_t^{adv}} \mathcal{L}(f(ODI(x_t^{adv})), y_t) \tag{4.1}$$

Where $\hat{g}_{t+1}$ is the gradient at step $t+1$, x_t^{adv} is the crafted input at time step t ($x_0^{adv} = x$ where x is the original clean sample), $\mathcal{L}$ is the loss function used for optimization, y_t is the target class for the targeted attack and f is a classifier. In practice, since their attack requires gradients, the classifier f that is used to compute the gradients is a source model separate from the model they are attacking (as otherwise, their attack would be a white-box attack).

Their attack is tested using the following source models (that they also attack): ResNet-50 [95], Inception-v3 [96], DenseNet-121 [97], and VGG-16_bn [98]. And the following additional target models: ResNet-18 [95], Inception-v4 [99], MobileNet-v2 [100], Inception ResNet-v2 [99], adversarially trained Inception-v3 [8] and ensemble adversarially trained Inception ResNet-v2 [8].

4.1.2 Geometry-Aware Attack

The Geometry-Aware (GA) attack [40] is a transferable attack focused on minimizing the necessary perturbation budget to achieve misclassification. It is a grounded and untargeted attack that uses a classification-validation split of pre-trained classifiers to generate adversarial examples with the smallest adaptive budget possible while retaining a high likelihood of transferability to the target model. Their attack is an attack aimed at boosting transferability (similar to ODI [45]) of existing attack methods. In particular, they explore the effectiveness of their meta-attack in the content-preserving salient situation. To do so, they use a set of pre-trained models (on ImageNet-1k's training set [101]) that they split into a training (f) and validation (h) set. They iteratively generate an adversarial example by using an existing attack with a fraction of their maximum perturbation size against the training set of models f. They then compute a confidence score that represents the probability of the true class on the validation models:

$$\text{conf} = \frac{\exp(h_y(x_k, \theta))}{\sum_j \exp(h_j(x_k, \theta))} \tag{4.2}$$

Where y is the original image x's label, θ are the parameters of the models and k is the current step in the iterative process. If the confidence score is lower than a threshold η, they early-stop their algorithm to save budget (they save at least $(K-k)\frac{k\epsilon}{K}$ where ϵ is their maximum perturbation budget and K is the number of iterations). If the confidence score never reaches lower than the threshold η, they return the generated image x_K after the K steps occur.

To go with their meta-attack procedure, they combine the Feature Space Attack (FSA) [102] with transfer-based l_∞-norm attacks [103], [104] to create a content-preserving attack. Using a pre-trained encoder ϕ, they compute the mean $\mu(\phi(x))$ and standard deviation $\sigma(\phi(x))$ of the output of the encoder on the original image x. They then adversarially perturb $\mu(\phi(x))$ and $\sigma(\phi(x))$ before projecting them back onto the input space using a pre-trained decoder ϕ^{-1} to generate x'. These adversarial perturbations on $\mu(\phi(x))$ and $\sigma(\phi(x))$ can be enhanced using input diversity [104] and momentum-based methods [103] to yield their full attack.

Multiple variations of their attack are tested on various defended models like Inception-ResNet-v2$_{\text{Ens-adv}}$ [8]. They also competed in the *CVPR'21 Security AI Challenger: Unrestricted Adversarial Attacks on ImageNet* [105] where their entry ranked 1st out of the 1,559 teams competing.

4.1.3 Large Geometric Vicinity Attack

This is another attack to improve the transferability of adversarial examples [54]. They use a pre-trained surrogate model and apply a two-phase algorithm. The first phase consists of continuing the training of the pre-trained surrogate model for a few epochs with an increased learning rate. During those epochs of training, they save the weight-state of the surrogate model at regular intervals (4 times per epoch). This technique allows them to sample a wide array of well-trained models with varying loss landscapes. They claim it improves over simply adding random noise to the weights. Once all those models are collected, they proceed with the second phase of the attack algorithm: they iteratively attack the collected models. They use the I-FGSM [106] as a baseline attack where for an input, at every step of the iterative process, they sample a different model from their set of collected models (and cycle through if they run out of models) and optimize against it. This is an untargeted and grounded attack.

They compare their augmentation method against four other test-time transformations applied on top of I-FGSM. The surrogates they use for their attack are pre-trained ResNet-50 [107]. They attack eight pre-trained models provided by PyTorch [108]: ResNet-50 [95], ResNet-152 [95], ResNeXt-50 [109], WideResNet-50 [110], DenseNet-201 [97], VGG-19 [98], Inception-v1 [111], and Inception-v3 [96]. Their experiments are performed using a random subset of 2000 images from the test set of the ImageNet dataset [101] that all eight models classify correctly.

4.1.4 Pixle Attack

This is a black-box query-based adversarial example attack [37] whose goal is to generate potent adversarial examples by re-arranging a few pixels within the attacked image based on random search. They claim they do so with few queries and negligible perturbations between the original and adversarial images. This attack is therefore grounded. They provide both a targeted and untargeted attack algorithm for their attack. They follow the indistinguishable-perturbation salient situation under the l_0-norm. To do so, they select pixels from a randomly sampled small patch of neighboring pixels P in the image.

These pixels are then mapped to other pixels in the image using a mapping function $m : (i, j) \in P \rightarrow \mathbb{N}_{+}^{2} \in [1, h] \times [1, w]$ where h and w are height and width of the image respectively. The adversarial image's pixels can be defined as the following:

$$\hat{x_{i,j}} = \begin{cases} m(i,j) & \text{if } (i,j) \in P \\ x_{i,j} & \text{Otherwise} \end{cases} \tag{4.3}$$

They use three different mapping functions: random mapping (map to a random pixel that is not itself); mapping based on pixel similarity, either the most similar pixel (in pixel value) that is not itself or the least similar. Using this mapping technique, they iteratively sample patches of the image, apply the mapping transformation and then query the model on the adversarial sample. If the score of the true label class is lower than the previously measured, they save the patch as the current best patch. They repeat this procedure a fixed number of times (hyperparameter) and then return the best adversarial example they found.

They evaluate their attack on three datasets: CIFAR10 [112], TinyImageNet [113] and ImageNet [101]. Furthermore, they train and evaluate ResNet-20 [95] and VGG-11 [98] on CIFAR10; ResNet-50 [95] and VGG-16 [98] for Tiny ImageNet. However, for ImageNet, they use pre-trained ResNet-50 and VGG-16 models without any fine-tuning. They sample randomly for each dataset 1000 images from the test set (100, 5 and 1 image per-class respectively for CIFAR10, TinyImageNet and ImageNet). They also evaluate the targeted version of their attack on a 200-image subset of CIFAR10 (20 images per class).

4.1.5 MASSA Attack

MASSA [66] is also query-based black-box attack that adds noise to an initial benign image in the frequency domain and optimizes that noise by a binary search through each of the frequency components of the frequency domains to minimize the perturbation created while maintaining the misclassification. This frequency domain representation of the image is obtained using the Fast Fourier Transform (FFT). They rely on the observation that for images, most of the information is contained in the low-frequency part of the frequency domain. They present their attack as a grounded, untargeted decision-based black-box attack and operate under the indistinguishable perturbation salient situation with the l_2-norm as their metric. Furthermore, they summarize their attack methodology as follows: Firstly, they sample an initial noise in the frequency domain. To do so, they first use statistical analysis to separate the frequency domain into the low/mid/high-frequency bands. Once this is done, they generate a histogram of the values in the frequency domain and associate

a matching normal distribution N^* to it with mean μ and standard deviation σ (they work under the assumption that any image's frequency domain histogram will follow some form of normal distribution). They also define t_l and t_r as the left and right tail of the histogram respectively:

$$t_l = \mu - \alpha_l \sigma \text{ and } t_r = \mu + \alpha_r \sigma \tag{4.4}$$

for some scaling factors α_l, α_r. They use those tails to delimit the three low/mid/high-frequency bands. Then for each band, they replace the values in the frequency domain with values sampled from N^* with a restricted range depending on the band: $[F_{min}, t_l]$ for the low band, $[t_l, t_r]$ for the mid band and $[t_r, F_{max}]$ for the high band, where F_{min} and F_{max} are the lowest and largest value in the frequency spectrum respectively.

Once this is done, they query the model to ensure that the generated image is adversarial (untargeted). However, this method of noise injection in the frequency domain ends up generating quite a significant amount of noise when projected back into the image space. Therefore, they employ a second step that reduces the perturbation magnitude in each frequency band while preserving the adversarial nature of the image. They do so using a method they call *Frequency Spectrum Binary Search*. They use this binary search method to minimize the difference between the original image and the adversarial image band by band (whilst querying the target model to make sure the image remains adversarial), starting with the low-frequency band as it is the one carrying the most important features. Finally, they perform a final step to remove redundant noise. This step is a method that recursively divides the noise image (difference between adversarial image and original image) into four patches and removes a patch before querying the model. If this removal does not change the prediction (still adversarial), it permanently deletes it. Otherwise, it keeps that patch. They repeat this step while progressively using finer and finer patch sizes.

They evaluate their attack using a set of 500 correctly classified random images from the validation set of the ImageNet [101] dataset. They attack both undefended and defended ResNet-50 [95] (adversarially trained [**?**]), VGG-16 [98] and VGG-19 [98] models. The defenses they attack are JPEG compression [114] and adversarial training [8].

4.1.6 AI-FGTM Attack

In their paper [15], the authors propose a new gradient-based optimization technique to generate adversarial examples. They integrate this new technique with existing transfer-based attacks to deliver a transfer-based attack that achieves state-of-the-art performance

for transfer-based attacks. It is a grounded and untargeted attack in the indistinguishable perturbation salient situation with the l_∞-norm metric. This attack claims to improve on the shortcomings of gradient sign-based attacks by preserving more gradient information. This can be summarized as the following improvements to different parts of the algorithm:

- Replacing the sign function with the tanh function to generate smaller perturbations
- Using Adam [115] instead of a momentum method and gradient normalization to get larger model losses in fewer iterations
- Using smaller kernels in Gaussian blur to avoid the loss of gradient information
- Gradually increasing the step size

Adam [115] limits the updates' reliance on only the past few gradients. They do so by using a first moment vector m_t, a second moment vector v_t. However, in this paper, the author modify how these vectors are computed a bit to tailor them to the dual optimization problem of adversarial examples. They define them as follows:

$$m_{t+1} = m_t + \mu_1 \cdot \nabla_{x_t^{adv}} J(x_t^{adv}, y^{true}) \tag{4.5}$$

$$v_{t+1} = v_t + \mu_2 \cdot (\nabla_{x_t^{adv}} J(x_t^{adv}, y^{true}))^2 \tag{4.6}$$

Where $x_0^{adv} = x$ (x is a starting benign sample and y^{true} its label), $m_0 = 0$, $v_0 = 0$, J is the loss function of a given classifier, μ_1 and μ_2 denote the first and second moment factors, respectively.

They then update x_{t+1}^{adv} as follows:

$$\alpha_t = \frac{\varepsilon}{\sum_{t=0}^{T-1} \frac{1-\beta_1^{t+1}}{\sqrt{(1-\beta_2^{t+1})}}} \frac{1-\beta_1^{t+1}}{\sqrt{(1-\beta_2^{t+1})}} \tag{4.7}$$

$$x_{t+1}^{adv} = \text{Clip}_{\varepsilon}^{X} \left\{ x_t^{adv} + \alpha_t \cdot \tanh(\lambda \frac{m_{t+1}}{\sqrt{v_{t+1} + \delta}}) \right\} \tag{4.8}$$

Where β_1 and β_2 are exponential decay rates, λ denotes the scale factor, ε is the parameter for the l_∞-norm and α_t is the increase step size with $\sum_{t=0}^{T-1} \alpha_t = \varepsilon$.

They then combine this aforementioned adversarial example generation process with the Nesterov Iterative Method (NIM [116]), the Translation-Invariant Method (TIM [117]) and the Diverse Input Method (DIM [104]) to yield their complete transfer-based attack.

They evaluate their attack on an ImageNet-compatible dataset (1000 images from the NIPS 2017 adversarial competition [118]) using four non-defended pre-trained models(on ImageNet [101]) as source models: Inception-v3 [96], Inception-v4 [99], Inception-ResNet-v2 [99], and ResNet-101-v2 [119]. They also use the ensemble of these four models to generate adversarial examples. They attack a total of nine defended pre-trained models (on ImageNet [101]): Inception-v3$_{\text{Ens3}}$, Inception-v3$_{\text{Ens4}}$, Inception-ResNet-v2$_{\text{ens}}$ [8], high-level representation guided denoiser (HGD) [120], input transformation through random resizing and padding (R&P) [121], rank-3 submission 2 in the NIPS 2017 adversarial competition (NIPS-r3) [118]. Out of those nine models, they also attack three models with what they consider "advanced" defenses. The first one: Feature Distillation (FD) [122] uses the ImageNet-trained [101] MobileNet architecture [123] from its original paper [124]. The second: Comdefend [125] trains their own ResNet-50 [95]. They evaluate against more than just this model but since it's the only model they train specifically, we have to assume this is the one they mean as the authors do not specify it in their paper. Finally, the last model: Randomized Smoothing (RS) [9] also uses a ResNet-50 [95] as a base.

4.1.7 SSAH Attack

In this paper [65], the authors present an attack that addresses two existing problems with other approaches: inherent limitation in cross-dataset generalization and poor imperceptibility to the human visual system (HVS). To tackle these problems, they offer a grounded attack with both a targeted and untargeted variant. They also work in the indistinguishable perturbation setting but instead of working with l_p-norms, they instead use semantic similarity metrics. However, instead of using existing metrics such as SSIM [126] or LPIPS [127], they provide a frequency-domain-based metric. Their attack works as follows: Given a batch $X = [x_1, \ldots, x_N]$, they optimize (untargeted):

$$x_i^{adv} = \arg_{x_i'} \min[s'_{i,i} - \min\left\{s'_{i,j} | j \neq i\right\}]_+ \tag{4.9}$$

Where $[\cdot]_+$ denotes $\max(\cdot, 0)$, x_i' is the optimization variable initialized as x_i and $s'_{i,j} = \text{sim}(f(x_i'), f(x_j))$ is a similarity score. They define $s'_{i,j}$ as:

$$s'_{i,j} = \frac{f(x_i')^T f(x_j)}{\|f(x_i')\|_2 \|f(x_j)\|_2} \tag{4.10}$$

In the targeted attack setting, they instead optimize:

$$x_i^{adv} = \arg_{x_i'} \min[s'_{i,i} - s'_i, t]_+ \tag{4.11}$$

Where t is the index of the target image in the batch. Since they attack semantic similarity, their target is a particular image of the target class rather than the target class as a whole. Now that they have the function they want to optimize, they design a self-paced weighting mechanism to avoid redundant perturbations. It modifies the function to optimize by adding two parameters α_i and β_i:

$$x_i^{adv} = \arg_{x_i'} \min[\alpha_i s'_{i,i} - \beta_i \min\left\{s'_{i,j} | j \neq i\right\}]_+ \tag{4.12}$$

that self-adjust based on the similarity of the generated adversarial example to its original benign image and the other images in the batch. This completes the first part of their algorithm that they call L_{SSA}. The second part is called the low-frequency constraint. Using the observation that HVS is (according to them) more sensitive to changes in the low-frequency components than the high-frequency components, they employ the Discrete Wavelet Transform (DWT) and its inverse (IDWT) to deconstruct and then reconstruct the image without its high-frequency components as:

$$\phi(x) = L^T(LxL^T)L \tag{4.13}$$

where L are the low-pass filters of an orthogonal wavelet. Using $\phi(x)$ they define a new perceptual similarity constraint:

$$D_{lf}(x, x') = \|\phi(x) - \phi(x')\|_1 \tag{4.14}$$

This yields the final attack:

$$x_i^{adv} = \lambda D_{lf}(x_i, x_i') + L_{SSA}(x_i, x_i') \tag{4.15}$$

They evaluate their attack in both the whitebox and transferable settings on the CIFAR10 [112], CIFAR100 [112] and ImageNet [101] datasets. For CIFAR10/100, they use a ResNet-20 [95] that they train themselves (92.6% and 69.6% accuracy on CIFAR10/100 respectively) and a ResNet-50 [95] for ImageNet that they also train themselves (76.15% accuracy). They evaluate their performance against defended models by attacking models

(WideResNet-34-10 [110]) defended by FSAT [128] and TRADES [129]. In the transferable setting, they attack the online models from Microsoft Azure [130], Tencent Cloud [131], and Baidu AI Cloud [132]. They evaluate the quality of their attack using four different perception metrics: l_2-norm, l_∞-norm, the Fréchet Inception Distance (FID) [133] and their own metric they call the average distortion of low-frequency components (LF) that can be computed as such:

$$LF = \frac{1}{N}\sum_{i=1}^{N} \|\phi(x_i) - \phi(x_i^{adv})\|_2 \tag{4.16}$$

4.1.8 F-Attack

This attack [38] is a decision-based grounded adversarial attack that similarly to MASSA [66] or SSAh [65] utilizes the frequency domain to mount its attack. Their attack is built upon an algorithm they build called *f-mixup* that can also be used to improve the query efficiency of other decision-based attacks when combined with a frequency binary search algorithm. They heavily focus on query efficiency. They do not specify the salient situation they are in but since they do not use distance metrics or image quality/perceptional metrics, we have to assume they are within the content-preserving setting.

f-mixup starts with a set of benign images and crafts adversarial examples by replacing the mid-high frequency components of the original benign sample with the corresponding part of a reference image (another benign image in the set). This is again based on the observation that most of the information in the frequency domain is carried by the low frequencies and the human visual system (HVS) is less sensitive to changes in the high frequencies. They use the Discrete Fourier Transform (DFT) to compute the frequency domain of the input samples and its inverse (IDFT) to recover the image from the frequency domain. Their algorithm $f - mixup$ can be summarized as follows:

1. Given two benign samples x_0 and x_i, compute f_0 and f_i using the DFT.
2. Using the band thresholds, r_l and r_h (low and high respectively). They apply a band-stop filter (bs) to f_0 to remove all the frequencies within the band (r_h, r_l) and a band-pass filter (bp) to f_i to get only the frequencies within the band (r_h, r_l). $f_0^{bs} = bs(f_0, r_h, r_l)$, $f_i^{bp} = bp(f_i, r_h, r_l)$.
3. Then, they add the extracted bands together to get the adversarial examples' frequency domain representation: $\hat{f_0 i} = f_o^{bs} + f_i^{bp}$.

4. Finally, they recover the adversarial example $\hat{x_o i}$ using the IDFT.

Using $f - mixup$, they build their attack $f - attack$ as follows:

1. Given a set of n benign images $X = \{x_1, \ldots, x_n\}$, they first compute r_l and r_h. They set the upper cut-off r_h arbitrarily and use it to compute r_l. Using the clean example x_0 and a distortion threshold ρ, they get r_l by performing a binary search on $[0, r_h]$ until $\|x_0^{bp}\| \sim 0.7\rho$.
2. Use *f-mixup* on x_0 for each $x_i \in X$. This yields n candidates.
3. Sequentially query the target model with the generated candidates until it misclassifies, or the attacker uses up the query budget.

One of their core idea to bypass detection is that since *f-attack* discards model feedback that is normally used in conventional decision-based attacks, it is not as detectable. They emphasize that their attack is tailored to small query budgets as according to them its performance does not scale well when the query budget is increased. In this other setting, they offer a *Frequency Binary Search* algorithm to improve the query efficiency of other existing decision-based algorithms. Given an adversarial example and its clean counterpart, using *f-mixup* they iteratively update the adversarial sample while performing a binary search for the optimal frequency band (r_h, r_l).

They test their attack on two datasets:

- CIFAR10 [112], where they attack the following models (that they train themselves): VGG-16 [98] and ResNet-32 [95].
- ImageNet [101], where they attack the following models pre-trained by Pytorch [108]: ResNet-50 [95] and MobileNet-v2 [100].

For ImageNet, they attack 500 randomly sampled correctly classified examples. They additionally take another 500 examples to construct their reference set that all clean examples will share. They only perform an untargeted attack on ImageNet. As for CIFAR10, they perform both a targeted and untargeted attack using a reference set of 100 clean samples.

4.1.9 BIA Attack

This paper [11] is one of the first papers to consider untargeted attacks where the task of the target model is partially unknown (they call it black-box domain). While they remain within the image domain, their attack (BIA) explores the threat model where the adversary has neither knowledge of the training data nor the target model's specific goal. They operate under the indistinguishable perturbation salient situation with the l_∞-norm metric and train a generative model G_θ to destroy the low-level features (features that can be found within the last few layers of the feature extractor component of a model) of the surrogate models they attack. This in turn would allow the attack to be domain-agnostic to the extent that most image classification tasks will require a similar set of baseline features to be extracted.

Therefore, they teach their generator G_θ to minimize the cosine similarity [134] between a benign image and a different (adversarial) example to make the features featureless:

$$\theta^* = \arg_\theta \min \mathcal{L}_{cos}(f_s^L(x'_s), f_s^L(x_s)) \tag{4.17}$$

where θ^* are the learned generator's parameters, $\mathcal{L}_{cos}$ is the cosine similarity loss, x_s is a benign image sampled from the surrogate domain X_S, and L is the specific layer of the surrogate model they are trying to attack. This allows their generator in the inference phase to directly craft adversarial examples from given input images x_t from the target domain:

$$x'_t = \min(x_t + \epsilon, \max(G_{\theta^*}(x_t), x_t - \epsilon)) \tag{4.18}$$

where ϵ is the l_∞-norm parameter (for the indistinguishable perturbation salient situation).

Additionally, to enhance the transferability of their attack, they perform a random normalization-based input transformation (RN) to simulate having access to different data distributions in the training phase. This is furthermore combined with a domain-agnostic attention module A^L by applying a cross-channel average pooling to the feature maps at layer L. This is done to narrow the domain gap from the model's perspective between the source and target domain.

This yields the optimization function to generate the learned parameters for G_{θ^*}:

$$\theta^* = \arg_\theta \min \mathcal{L}_{cos}(A^L \odot f_s^L(RN(x'_s)), A^L \odot f_s^L(RN(x_s))) \tag{4.19}$$

where $\odot$ is the Hadamard product.

They work with the ImageNet dataset [101] and attack seven other datasets: CIFAR10, CIFAR100 [112], STL-10 [135], SVHN [136], CUB-200-2011 [137], Stanford Cars [138], and

FGVC Aircraft [139]. They use four ImageNet [101] pre-trained models to train their attack generators: VGG-16, VGG-19 [98], ResNet-152 [95], and DenseNet-169 [97]. They then attack each of the target datasets (other than ImageNet):

- CIFAR10, CIFAR100, STL-10, and SVHN: custom models.
- CUB-200-2011, Stanford Cars, and FGVC Aircraft: DCL framework [140] with ResNet-50 [95], SENet-154 and SE-ResNet-101 [141] backbones.

Additionally, they study the transferability of their attack against other models trained on ImageNet in two other settings: white-box and transferable. For the whitebox setting evaluation, they evaluate pre-trained VGG-16 [98] and DenseNet-169 [97] models. Whereas for the transferable setting evaluation, they use the pre-trained VGG-16 and DenseNet-169 as source models (they also use one to attack the other and vice-versa) to attack the following models: VGG-19 [98], ResNet-50, ResNet-121 [95], DenseNet-121 [97], and Inception-v3 [96].

4.1.10 ACG Attack

In their paper [83], the authors present a whitebox untargeted and grounded attack based on the conjugate gradient (CG) method that they call the Auto Conjugate Method (ACG). The motivation behind using the conjugate method is that the commonly used steepest descent method used by most attacks can suffer against ill-conditioned problems. Additionally, they also define a measure of diversity called the Diversity Index (DI) and argue that the more diverse the search employed by an attack, the higher its success rate. They operate under the indistinguishable perturbation salient situation with an l_∞-norm metric.

Similarly to PGD [7], ACG is an iterative algorithm that repeatedly iterates over a sample within an l_p-norm restriction (in this case l_∞). The ACG step can be mathematically summarized as follows:

$$x^{k+1} = P_S(x^k + \eta^k \cdot (s^k)) \tag{4.20}$$

Where P_S is the projection onto the feasible region S, $\eta^k = (\text{argmin}\left\{f(x^k + \eta s^k)|\eta \geq 0\right\})$ is the step size, and σ is a type of normalization. s^k is iteratively computed as follows

$$s^k = \nabla f(x^k) + \beta^k_{HS} s^{k-1} \tag{4.21}$$

$$\beta_{HS}^{k} = \frac{\langle -\nabla f(x^k), \nabla f(x^{k-1}) - \nabla f(x^k) \rangle}{\langle s^{k-1}, \nabla f(x^{k-1}) - \nabla f(x^k) \rangle} \tag{4.22}$$

They use the same method as Auto PGD [142] to select the step size. They summarized the process as follows: The initial step size η^0 is set to 2ϵ, and when the number of iterations reaches the precomputed checkpoint w_j, the step size η is halved if either of the following two conditions are satisfied:

- $N_{\text{inc}} < \rho \cdot (w_j - w_{j-1})$,
- $\eta^{w_{j-1}} = \eta^{w_j}$ and $f_{\max}^{w_{j-1}} = f_{\max}^{w_j}$,

where $N_{\text{inc}} := \#\{i = w_{j-1}, \ldots, w_j - 1 | f(x^{i+1}) > f(x^i)\}$ and $f_{\max}^{k} := \max\{f(x^i) | i = 1, \ldots, k\}$.

They evaluate their attack on CIFAR10, CIFAR100 [112] and ImageNet [101] against 42, 17 and 5 models, respectively. They use all 10000 images of the validation set for CIFAR10/100 and 5000 images of the validation set for ImageNet. For CIFAR10 and CIFAR100, they use an epsilon of $\epsilon = 8/255$ for the l_∞-norm, and they use $\epsilon = 4/255$ for ImageNet. The models they use are models listed in RobustBench [94]. These are defended models collected from various previous papers that use the following network architectures for each dataset:

- CIFAR10: PreActResNet-18 [119], ResNet-18, ResNet-50 [95], WideResNet-106-16, WideResNet-28-10, WideResNet-28-4, WideResNet-34-10, WideResNet-34-15, WideResNet-34-20, WideResNet-34-R, WideResNet-70-16 [110].
- CIFAR100: PreActResNet-18 [119], WideResNet-28-10, WideResNet-34-10, WideResNet-34-20, WideResNet-70-16 [110].
- ImageNet: ResNet-18, ResNet-50, [95], WideResNet-50-2 [110].

4.1.11 Admix Attack

Admix [92] is a transferable attack that is similar to SSAH [65] and F-Attack [38] in using additional data from the same distribution as the given image (and the training data) to generate transferable adversarial examples using surrogate models. To use these additional images, they employ a modified version of the *mixup* operation that has been previously used to improve the generalization of standard training [143]. *Mixup* interpolates two

randomly sampled examples and their labels. *Admix*, however, unlike *mixup*, does treat the images being interpolated equally and does not interpolate the labels (keeps the original image's label). Instead, they bias the interpolation towards the original image rather than the additional image to preserve as much information about the original image as possible.

$$\hat{x} = \gamma \cdot x + \eta' \cdot x' = \gamma \cdot (x + \eta \cdot x') \tag{4.23}$$

where x is the original image, x' is the additional image, $\eta = \eta'/\gamma$, $\gamma \in [0, 1]$ and $\eta' \in [0, \gamma)$ control the proportion of the original image and the additional image are admixed respectively. Then, taking a set of admixed images, they can compute the average gradient w.r.t to the original image across the admixed images against a surrogate model and use this gradient for any traditional gradient-based attack. Therefore, their attack can be used for both targeted and untargeted attacks.

$$\bar{g}_{t+1} = \frac{1}{m_1 \cdot m_2} \sum_{x' \in X'} \sum_{i=0}^{m_1 - 1} \nabla_{x_t^{adv}} J(\gamma_i \cdot (x_t^{adv} + \eta \cdot x'), y; \theta) \tag{4.24}$$

where m_1 is the number of admixed images for each x' and X' denotes the set of m_2 randomly sampled images from other classes. J is the model's loss function and θ are the surrogate model's parameters.

They evaluate their attack using 1000 images from the 1000 different classes that are randomly sampled from the ILSVRC 2012 ImageNet-Compatible validation set [144] provided by another paper [116].

They use four models as source models: Inception-v3 [96], Inception-v4 [99], Inception-ResNet-v2 [99], ResNet-101 [95]. They also attack those four models (they treat the scenario where source and target models are the same as white-box, meaning they share their parameters) as well as three ensemble adversarially trained models: Inception-v3$_{\text{Ens3}}$, Inception-v3$_{\text{Ens4}}$, Inception-ResNet-v2$_{\text{ens}}$ [8]. They also attack nine additional defended models: HGD [120], R&P [121], NIPS-r3 [118], Bit-Red [145], MobileNet$_{\text{FD}}$ [122], JPEG [146], ResNet-50$_{\text{RS}}$ [9], ARS [147] and NRP [148].

4.1.12 ATA Attack

In their paper [82], the authors present a grounded and untargeted transferable adversarial attack tailored to Vision Transformer models (ViTs). This is motivated by the unique architectural challenges that ViTs pose to transferable adversarial examples due to their use of self-attention and image-embedding layers. Feature extraction in ViTs is done

through these image embedding and self-attention layers, hence the authors posit that confusing these layers could result in effective and transferable adversarial examples. They do so in two steps:

Uncertain Attention Activation: Self-attention is a key component of the transformer architecture and allows a model to identify and focus on the critical tokens within the input. They first obtain the class activated matrix C of the b-th block of a ViT $\mathbb{F}$ following [149] where $C_{q,p}$ represents the contribution of token p to token q. They then calculate the uncertainty weight w_p of each x_{pt}^p (from the sequence of flattened patches returned by the image embedding layer with indices $p = 1, \ldots, N$) using entropy:

$$w_p = \frac{\sum_q C_{q,p} \cdot \log C_{q,p}}{\sum_p \sum_q C_{q,p} \cdot \log C_{q,p}} \tag{4.25}$$

Which they recombine into $W = w_1, \ldots, w_N$. Using those weights, they can find the patch-wise regions to activate the uncertain attention within the pixel number (l_0-norm ϵ) constraint by calculating the patch-wise region map $M_c = \text{floor}(m \cdot W)$ where m is the pixel number.

Sensitive Embedding Perturbation: Given the patch-wise attentional regions, they leverage the shared use of image embedding layers amongst ViTs to find the sensitive pixels that most influence the image embedding layers. Concisely, given the patch-wise region map M_c, the first find the non-zero element M_c^p and its corresponding flattened patch x_{pt}^p and then search the most sensitive pixels with full consideration of the embedding strategy.

They evaluate their attack on the ImageNet [101] dataset, taking 2000 images from the validation set. They attack Pytorch [108] pre-trained CNNs and ViTs:

- For CNNs: ResNet-50 [95], DenseNet-121 [97], AlexNet [150], and VGG-16 [98].
- For transformers (come in tiny "-T", small "-S" and base "-B"): ViT family [151], DeiT family [152], and ConViT family [153].

4.1.13 Shadow Attack

The authors of the paper [14] propose a black-box score query-based attack using natural phenomenons, in particular shadows, to generate content-constrained, grounded and

untargeted adversarial examples. They argue that using shadows rather than artificial adversarial patterns can lead to less detectable and more practical (both digitally and physically) adversarial examples. Their algorithm works as follows: Given a mask M to locate a target object in an image, they attempt to optimize the location of a set of vertices $V = \{(m_1, n_1), \ldots, (m_s, n_s)\}$ that represents a polygon where the shadow will be drawn. In practice, they find that a triangle works best. Whereas, for the magnitude of the shadow, they use Expectation Over Transformation (EOT) [154] and convert the input images from RGB space to LAB space to use the L channel which corresponds to lightness. So, they first convert the clean input image x in RGB color space to LAB color space:

$$LAB(x) = LAB([R_x, G_x, B_x]) = [L_x, A_x, B_x] \tag{4.26}$$

Then, given a Polygon P_V and a mask M, they compute the value of each pixel (i, j) in the adversarial image x_{adv} as follows:

$$LAB^{i,j}(x_{adv}) = \left[L^{i,j}_{x_{adv}}, A^{i,j}_{x_{adv}}, B_{x_{adv}}\right] \tag{4.27}$$

$$= \begin{cases} LAB^{i,j}(x) \cdot [k, 1, 1]^T & (i, j) \in P_V \cap M \\ LAB^{i,j}(x) \cdot [1, 1, 1]^T & (i, j) \notin P_V \cap M \end{cases} \tag{4.28}$$

where k is the shadow coefficient. This coefficient can be computed from a reference benign dataset or modified arbitrarily. Then they convert it back to the RGB color space to yield x_{adv} and their complete attack:

$$x_{adv} = S(x, P_V, M, k) \tag{4.29}$$

Since M is given, they offer two possible optimization strategies to optimize M: the first is a particle swarm optimization (PSO) strategy [155] and the other is EOT [154].

They evaluate their attack on the LISA [156] dataset, which consists of 47 different US road sign classes and the GTSRB [157] dataset, which consists of 43 different German road sign classes. They attack the following models: LISA-CNN [158] and GTSRB-CNN [158]. They also adversarially train robust versions of these models (LISA-CNN_{rob}, GTSRB-CNN_{rob}) against adversarial examples crafted with their attack and evaluate their performance. They use the SBU Shadow dataset [159] to set their shadow parameter k (setting it to the mean of the dataset).

4.1.14 DAPatch Attack

This paper [84] presents a white-box, grounded and untargeted patch attack that focuses on optimizing the shape of the patch on top of the content of the patch itself. This yields

an attack that operates under the content-preserving salient situation (since the patch is visible to the human eye). They do so by iteratively and differentiably deforming the patch shape. Representing their patch as a composition of triangles, they can differentiably verify if a point is inside or outside the contour and the shape model can be mapped into a binary mask. This allows them to jointly optimize the shape and the texture of adversarial patches. This triangle composition is represented as one center point O and a set of R rays of various lengths $r = r_1, r_2, \ldots, r_R$. Then the entire patch can be represented as triangles crafted from pairs of rays and the center point O. Then given O and r, they describe a Deformable Patch Representation (DPR) algorithm to differentiably obtain the mask M. This algorithm works as follows:

Given a triangle AOB and any pixel $C \in x$ where x is an input image, they compute $\frac{|CO|}{|DO|}$ where D is the intersection point of AB and CO (or its extended line). If $\frac{|CO|}{|DO|} < 1$ then C is in AOB and vice versa. They then use a function $\phi(x)$ to map it to the binary space differentiably to get the final mask computation:

$$M(C, r) = \phi(\frac{|CO|}{|DO|}) \in \{0, 1\} \tag{4.30}$$

To allow for more complex contour modeling, they introduce a multi-anchor mechanism:

$$r^0 = \{r_1^0, r_2^0, \ldots, r_R^0\}, \tag{4.31}$$

$$e^i = \{e_1^i, \ldots, e_R^i\}, i = 0, 1, \ldots, R-1, \tag{4.32}$$

$$r^{i+1} = r^i + e^i, i = 0, 1, \ldots, R-1 \tag{4.33}$$

where r^i is the length of the ray in the i-th anchor and e^i denotes the margin between r^i and r^{i+1}.

Using this process, representation, they construct their Deformable Adversarial Patch (DAPatch) attack. This attack jointly optimizes the shape and texture of the patch represented by DPR. Given a maximum area ps that the patch can take up on the image, they define the loss function as follows to control the area of the patch:

$$L = \begin{cases} L_{adv} & area \leq ps \\ L_{adv} + \beta \cdot L_{shape} & area > ps \end{cases} \tag{4.34}$$

where $area$ is the area covered by the patch and β is a hyper-parameter. They define L_{shape} as follows:

$$L_{shape} = \text{mean}(M^k) \tag{4.35}$$

Then the adversarial example at step k x_{adv}^k can be expressed as:

$$x_{adv}^k = \theta^{k-1} \odot M^{k-1} + x_{adv}^{k-1} \odot (I - M^{k-1}) \tag{4.36}$$

where θ represents the texture. The texture and shape update process can be computed as follows:

$$\theta^k = \theta^{k-1} + \alpha \cdot \text{sign}(\nabla_{x_{adv}^k} L) \tag{4.37}$$

$$r^k = r^{k-1} + \gamma \cdot \text{sign}(\nabla_{r^{k-1}} L) \tag{4.38}$$

They evaluate their attack on the GTSRB [157] dataset (500 images from the validation set) and the ImageNet-Compatible ILSVRC2012 dataset [144] (1000 images from the validation set). They attack the following models: VGG-19 [98], ResNet-152 [95], DenseNet-161 [97], MobileNet-v2 [100], ViT-B [151], Swin-B [160], and EfficientNet-b7 [161]. They also attack defended models: ResNet-50_{SIN}, ResNet-$50_{\text{SIN+IN}}$, ResNet-$50_{\text{SIN+IN-IN}}$ [162], ResNet-50_{Debiased}, ResNet-152_{Debiased} [163], FastAT [164], ResNet-152_{Denoise}, ResNeXt-101_{Denoise}, and ResNet-152_{adv} [165].

4.1.15 S^{2}I Attack

In their paper [64], the authors propose a transferable grounded adversarial example attack combining diversity and the frequency domain. However, unlike the ACG attack [83] (summary 4.1.10) that looks into the diversity of the search algorithm used by an attack, the S^{2}I attack looks into the diversity of the surrogate models used to improve the transferability of existing white-box adversarial example attacks. To enhance the model output diversity, they decided to transform the inputs given to each of the surrogate models in the ensemble used to generate the transferable example. However, as they state, existing transformations for the spatial domain do not translate to significant diversification of model outputs. Hence, they apply a spectrum transformation based on the discrete cosine transform (DCT) and its inverse to perform the model augmentation in the frequency domain instead. Additionally, to measure diversity, they use the spectrum saliency map:

$$S_\phi = \frac{\partial J(D_I(D(x)), y; \phi)}{\partial D(x)} \tag{4.39}$$

Where D is the DCT, D_I is the inverse DCT (IDCT), J is the loss function, x is the input image, y is its class label, and ϕ are the model parameters.

Their attack can be described as a random spectrum transformation $\tau(\cdot)$:

$$\tau(x) = D_I((D(x) + D(\xi)) \odot M) \tag{4.40}$$
$$= D_I(D(x + \xi) \odot M) \tag{4.41}$$

where $\odot$ denotes the Hadamard product, $\xi \sim \mathcal{N}(0, \sigma^2, I)$ and each element of $M \sim \mathcal{U}(1 - \rho, 1 + \rho)$ are random variants sampled from a Gaussian Distribution and a Uniform distribution, respectively. Then their algorithm can be summed up as follows:

- transform x into $\tau(x)$
- compute the gradient $g = \nabla_x J(\tau(x), y; \phi)$ on the surrogate model(s)
- use the gradient for any gradient-based white-box attack. In their paper, they use FGSM

They evaluate their attack on the ImageNet-Compatible dataset (NIPS 2017 adversarial competition [118]) of 1000 images. They also happen to compare against one of the papers we also include in our survey (Admix [92], summary **??**). They use 4 models as surrogate source models: Inception-v3 [96], Inception-v4 [99], Inception-ResNet-v2 [99], ResNet-152 [95] that they also attack in the white-box setting. In the transferable setting, they attack undefended ResNet-50 and ResNet-101 [95] models. They also attack some defended models: Inception-v3$_{\text{Ens3}}$, Inception-v3$_{\text{Ens4}}$, Inception-ResNet-v2$_{\text{ens}}$ [8], HGD [120], R&P [121], NIPS-r3 [118], JPEG [146], ResNet-50$_{\text{RS}}$ [9], NRP [148].

4.1.16 AI-GAN Attack

AI-GAN [93] is an attack that is similar to the BIA attack [11] (summary 4.1.9) trains and uses a generator to generate adversarial examples. AI-GAN is a work that follows from and improves on AdvGAN [166]. The generators for both of these works are based on the Generative Adversarial Network architecture [167] that they adapt to generate adversarial noise that is then added to the original image. AI-GAN is grounded and targeted. It can be summarized as follows:

Given the target (or surrogate) model, the authors train a noise generator that takes an input sample x and a target t to generate noise that is added to x to generate x'. x' is then given to the local classification model. This allows them to compute a loss on the performance of the generator at fooling the local model $L_{target_{pert}}$:

$$L_{G_{pert}} = \mathbb{E}[\log P(c = t|x')] \tag{4.42}$$

where c is the predicted class by the local model. x' is also given to the discriminator D (part of the GAN architecture) that they enhance to make it predict both whether the image is the original (x) or the perturbed (x') image (L_S) and to make it predict the actual class of the image ($L_{D_{pert}}$). The total loss for the generator can be then described as:

$$L_G = L_{G_{pert}} + L_{D_{pert}} - L_S \tag{4.43}$$

$$L_{GD_{pert}} = \mathbb{E}[\log P(c_d = t|x')] \tag{4.44}$$

where c_d is the prediction of the class of x' by the discriminator.

$$L_S = \mathbb{E}[\log P(S = \text{real}|x)] + \mathbb{E}[\log(S = \text{pert}|x')] \tag{4.45}$$

where S is whether the image is real or perturbed. The discriminator also needs its own loss to be trained alongside the generator. An improvement they bring over AdvGAN is an additional loss-term to help stabilize the learning of the discriminator as GANs are well-known for their unstable training ([168]). Following previous work on using robustness to stabilize GAN training [168], they incorporate an attacker into their framework (separate from the generator). Using another attack like Fast-Gradient-Sign-Method [169], they attack the classification part of the discriminator with an adversarial example x''. The loss of the discriminator can be written as follows:

$$L_D = L_S + L_{D_{adv}} + L_{DG_{pert}} \tag{4.46}$$

$$L_{D_{adv}} = \mathbb{E}[\log P(c_d = y|x'')] \tag{4.47}$$

$$L_{DG_{pert}} = \mathbb{E}[\log P(c_d = y|x')] \tag{4.48}$$

They train and attack models on the MNIST [170], CIFAR10 and CIFAR100 [112] datasets. They attack custom models on MNIST and ResNet-32 [95], WideResNet-34 [110] on CIFAR10/100.

4.1.17 AEG Attack

This paper [91] proposes a grounded transferable attack that uses a min-max game of adversarial example generation to generate adversarial examples given what they call a

hypothesis class (e.g. architecture). This paper is a precursor to this work as it introduces and takes some steps toward formalizing adversary knowledge. Using this, they argue that the Nash equilibrium reached by their attack in their min-max game yields a distribution of adversarial examples that are the most effective against any classifier. To train this generator, they are similar to the existing GAN [167] approach in using the divergence minimization approach [171]. However, where GANs look at the divergence between two distributions, they instead look at a notion of entropy: the $\mathcal{F} - entropy$ of a distribution p_g given a hypothesis class $\mathcal{F}$ (the class of models with a specific architecture) is defined as:

$$H_{\mathcal{F}}(p_g) := \min_{f_c \in \mathcal{F}} \mathbb{E}_{(x,y)\sim p_g}[l(f_c(x), y)] \tag{4.49}$$

where l is the cross-entropy loss. It aims to quantify the amount of "classification information" available in the given distribution p_g using the class of classifiers $\mathcal{F}$. f_c is a classifier sampled from $\mathcal{F}$ using the reference dataset $\mathcal{D}_{ref}$. Therefore, they aim to train a generator g (that takes in both a benign input x and its label y and outputs adversarial noise) that finds a regularized adversarial distribution of maximal $\mathcal{F}$-entropy:

$$\max_{g \in G_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, g) = (1+\lambda) \max_{g \in G_\epsilon} H_{\mathcal{F}}\left(\frac{1}{1+\lambda} p_g + \frac{\lambda}{1+\lambda} \mathcal{D}_{ref}\right) \tag{4.50}$$

where the distribution $\frac{1}{1+\lambda} p_g + \frac{\lambda}{1+\lambda} \mathcal{D}_{ref}$ is the mixture of the generated distribution p_g and the empirical distribution over a reference dataset $\mathcal{D}_{ref}$.
$\varphi_\lambda(f, g)$ is the regularized version of the AEG game. By regularized, they mean regularized w.r.t. to the empirical distribution of the reference dataset $\mathcal{D}_{ref}$.

They evaluate their attack on the MNIST [170] and CIFAR10 [112] datasets. They train their reference (surrogate) model using a different optimizer than the target model they are using (both are based on SGD and cross-entropy loss). The way they do so is that for a given dataset, they split the dataset into 5 distinct subsets (10000 images each) and use one subset as the reference set to attack four models each trained on one of the other subsets. This makes direct comparison against other works that train on the whole training set tricky. Then in another part of the evaluation, they relax their assumption and assume access to the whole training set. They use the following architectures:

- CIFAFR10: ResNet-18 [95], Wide-ResNet-34-10 [110], DenseNet-121 [97], VGG-16 [98], Inception-v3 [96]. They also attack the ens3 [8] defended versions of all of those models as well as Madry-Adv [7].
- MNIST: AlexNet [150], A_{ens4}, B_{ens4}, C_{ens4}, D_{ens4} [8], Madry-Adv [7].

4.1.18 ACA Attack

This paper [86] presents a grounded and untargeted, content-preserving attack using the latent space of pre-trained stable diffusion models. Their algorithm operates in two steps: first, it uses Image Latent Mapping to map images into the latent space. Then, it uses Adversarial Latent Optimization to generate adversarial examples. Specifically, they work with text prompted Stable Diffusion models.

The Image Latent Mapping process can be summarized as follows: they use the inverse process of DDIM sampling [172] with an associated condition embedding $C = \phi(P)$ from prompts P to map the images to the latent space. For an input image z_0, the image at step t z_t of the denoising process, we get the image at time step $t+1$ as follows:

$$z_{t+1} = \sqrt{\frac{\alpha_{t+1}}{\alpha_t}} z_t + (\sqrt{\frac{1}{\alpha_{t+1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1}) \cdot \epsilon_\theta(z_t, t, C) \tag{4.51}$$

where $\alpha_t = \Pi_i^t(1-\beta_i)$ for a schedule $\{\beta_0, \ldots, \beta_T\} \in (0,1)$ and ϵ_θ is the noise prediction model from the DDIM sampling. They encode the image prompt using an image caption model (BLIP-v2 [173]). They then get the classifier-free guidance prediction using ϵ_θ. However, this process can create noise that accumulates and disrupts the reconstruction process. To mitigate this, they optimize a null text embedding $\varnothing_t$ for each time step t [174] such that:

$$\min_{\varnothing_t} \|z^*_{t-1} - z_{t-1}(\bar{z}_t, t, C, \varnothing_t)\|_2^2 \tag{4.52}$$

where z_t^* are the latents output by DDIM sampling, $\bar{z_T} = z_t$ during N iterations for the time steps $t = \{T, \ldots, 1\}$. They update z_{t-1}^- to $z_{t-1}(\bar{z}_t, t, C, \varnothing_t)$ at the end of each step and $z_{t-1}(\bar{z}_t, t, C, \varnothing_t)$ is the output of equation 4.51 conditioned on $\varnothing_t$.

This process yields them the latent of the given image, consisting of the noise $\bar{z_T}$, the null text embedding $\varnothing_T$ (to ensure the quality of the reconstructed image) and the text embedding C (to ensure the preservation of the semantic information of the image). Then given the latent, they perform Adversarial Latent Optimization as follows:

$$\Omega(z_T, T, C, \{\varnothing_t\}_{t=1}^T) = z_0(z_1(\ldots, (z_{T-1}, T-1, C, \varnothing_{T-1}), \ldots, 1, C, \varnothing_1), 0, C, \varnothing_0) \tag{4.53}$$

Ω is the denoising process. Then, they define their optimization objective to generate the adversarial example as follows:

$$\max_\theta L(F_\theta(\bar{z_o}), y), \qquad s.t. \|\delta\|_\infty \leq \kappa, \bar{z_0} = \Omega(z_T + \delta, T, C, \{\varnothing_t\}) \tag{4.54}$$

where L is the loss function composed of cross-entropy loss and mean-squared error loss added together, the model F_θ with parameters θ.

They add differentiable boundary processing to ensure the value range of $\bar{z_0}$ is bounded between $[0, 1]$ to make it a valid input. Additionally, they use momentum in their gradient optimization to improve convergence.

They evaluate their attack on the ImageNet-Compatible dataset (1000 images from ImageNet's [101] validation set). They attack the following models: MobileNet-v2 [100], Inception-v3 [96], ResNet-50 [95], ResNet-152 [95], DenseNet-161 [97], EfficienNet-b7 [161], MobileViT [175], ViT-B [151], Swin-B [160], PVT-v2 [176]. They also attack defended models: HDG [120], R&P [121], NIPS-r3 [118], JPEG [146], Bit-Red [145], DiffPure [90], Inception-v3$_{\text{Ens3}}$, Inception-v3$_{\text{Ens4}}$, and Inception-ResNet-v2$_{\text{ens}}$ [8], ResNet-50$_{\text{Debiased}}$ [163], and Shape-ResNet [177].

4.1.19 DiffAttack

This paper[85] proposes another attack (DiffAttack) that similarly to ACA [86] (summary 4.1.18) uses the latent space of a diffusion model to generate potent and transferable adversarial examples. They also leverage the discriminative capabilities of diffusion models trained on large-scale datasets to use them as surrogate models for their attack. However, they take a different approach in their attack compared to ACA [86] by focusing on the cross-attention maps between the text and image pixels while implementing control measures to limit the distortion from the initial semantics by constraining the self-attention and controlling the inversion strength (when inverting images to noise). The particular diffusion model that they use is Stable Diffusion [178], and they use DDIM Inversion [179] to convert the original input image into the latent space. This process can be described as:

$$x_t = \text{Inverse}(x_{t-1}) \circ \cdots \circ \text{Inverse}(x_0) \tag{4.55}$$

where Inverse is the DDIM Inversion operation where x_t is the latent at time step t and x_0 is the original clean image. Then, they directly perturb the latent x_t:

$$\arg_{x_t}\min L_{attack} = -J(x', y; G_\phi), \qquad \text{where } x' = x'_0 = \text{Denoise} \circ \cdots \circ \text{Denoise}(x_t) \tag{4.56}$$

where J is the cross-entropy loss, Denoise is the diffusion denoising process, and G_ϕ is a surrogate model. They introduce a second loss component to enhance the transferability of the attack:

$$\arg_{x_t}\min L_{transfer} = \text{VarAverageCross}(x_t, t, C; \text{SDM}) \tag{4.57}$$

where Var calculates the input's variance, Cross denotes the accumulation of all the cross-attention maps in the denoising process and SDM is the Stable Diffusion model. They do this to distract the diffusion model's attention from the labeled objects. To control the distortion from the initial semantics, they use an additional (and final) loss component to control the self-attention and ensure structure retention. They take a clean copy of the original inversed latent $x_{t(fix)}$ as a reference point for the attacked latent x_t. They then calculate the self attention-maps $S_{t(fix)}$ and S_t of $x_{t(fix)}$ and x_t respectively and incentivize them to remain close by minimizing their l_2-distance:

$$\arg_{x_t}\min L_{structure} = \|S_t - S_{t(fix)}\|_2^2 \tag{4.58}$$

This yields a final objective function for their attack as follows:

$$\arg_{x_t}\min L = \alpha L_{attack} + \beta L_{transfer} + \gamma L_{structure} \tag{4.59}$$

where α, β, and γ are hyperparameters.

They evaluate their attack on the development (DEV) set of the ImageNet-Compatible dataset (same as the one from the NIPS 2017 adversarial competition [118]). They attack the following models: ConvNeXt [180], ResNet-50 [95], VGG-19 [98], Inception-v3 [96], MobileNet-v2 [100], Vit-B [151], Swin-B [160], DeiT-B and DeiT-S [152], Mixer-B and Mixer-L [181]. They also attack defended models: DiffPure [90], ResNet-50_{RS} [9], R&P [121], HGD [120], NIPS-r3 [118], NRP [148], Inception-$v3_{\text{adv}}$ [106], Inception-$v3_{\text{Ens3}}$, Inception-$v3_{\text{Ens4}}$, and Inception-ResNet-$v2_{\text{ens}}$ [8].

4.1.20 A^3 Attack

This paper [39] proposes a parameter-free, efficient, adaptive white-box attack to be used as a robustness test for defenses. This white-box attack is grounded, and untargeted. The authors argue it should be used to evaluate the lower bound of performance for a defense. They developed this adaptive attack by combining two strategies: Adaptive Direction Initialization (ADI) and Online Statistics-based Discarding (OSD). ADI studies the first restart in the restart-iterative attack algorithm and based on the directions of diversification of adversarial examples, it improves the sampling of starting points for future restarts. OSD discarding improves on naively treating all images equally regarding the number of iterations to be allocated. It avoids making unnecessary efforts to perturb hard-to-attack images.

ADI works as follows:

- First randomly sample directions of diversification.
- It then uses ODI [182] (not the same ODI as the one we review in this survey) to get starting points, initialize PGD attacks from them and obtain the crafted adversarial examples (set $W = w_d$ of all found adversarial examples).
- It then adopts the sign of the summed w_d as prior knowledge to generate the adaptive direction w_a.
- Using this prior knowledge, it generates each component of w_a by biasing the random sampling depending on the sign of the same component of the prior (uniform in $(-0.5, 0.1)$ if the prior is negative and $(-0.1, 0.5)$ if it is positive).
- Finally, it generates a label following the sign of the prior on the label. (-0.8 if negative, 0.8 if positive).
- It uniformly samples the labels for the other classes at uniformly random between -1 and 1. This helps improve the diversity of starting points.

OSD works as follows:

- Using the loss values of each image in the test set to attack, it sorts them in descending order by corresponding loss value at each restart and then discards the hard-to-attack images (ones with small loss values).
- They define the discarding rate at the r-th restart as follows:

$$\varsigma^r = \phi + r \times \iota \tag{4.60}$$

  where ϕ is the initial discarding rate, and ι is the discarding increment.
- It then computes how many iterations should be allocated to images that were not discarded as follows:

$$N^r_{\text{atk}} = \gamma + r \times \nu \tag{4.61}$$

  where γ is the number of iterations for attacks and ν is an iteration increment.

With their attack, they won the CPR 2021 White-box Adversarial Attacks on Defense Models competition. In their paper, they evaluate their attack against almost 50 defended models using a limited per-image number of iterations (100) on CIFAR10 and CIFAR100

[112] (entire validation set). They also participated in the CVPR 2021 White-box Adversarial Attacks on Defense Models competition (ImageNet [101], 1000 images from the validation set) and finished first. The model architectures (with the associated defenses) that were evaluated can be found in Tables 4.1 and 4.2, most if not all of these models can be found on RobustBench [94]:

Table 4.1: List of attacked models by A^3 on CIFAR10

Model	**Defense**
WideResNet-70-16 [110]	ULAT [183], Fixing Data [184]
WideResNet-28-10 [110]	ULAT [183], Fixing Data [184], AWP [185], RLPE [186], Geometry [187], RST [188], HYDRA [189], MART [190], FS [128], Interpolation [191]
WideResNet-34-15 [110]	RLPE [186]
WideResNet-34-10 [110]	Proxy [192], OOAT [193], Pre-training [194], SAT [195], YOPO [196], Sensible [197], FAT [198], Self-adaptive [199], TRADES [129], LBGAT [200]
WideResNet-34-20 [110]	ULAT [183], AT_HE [201], LBGAT [200], Overfitting [202]
ResNet-18 [95]	OOAT [193], DNR [203], CNL [204], Regularization [205], Proxy [192]
ResNet-50 [95]	Robustness [206]
WideResNet-28-4 [110]	MMA [207]

Table 4.2: List of attacked models by A^3 on CIFAR100

Model	**Defense**
WideResNet-70-16 [110]	ULAT [183], Fixing Data [184]
WideResNet-28-10 [110]	Fixing Data [184], Pre-training [194]
WideResNet-34-10 [110]	OOAT [193], LBGAT [200], AWP [185], SAT [195]
WideResNet-34-20 [110]	LBGAT [200]
ResNet-18 [95]	OOAT [193]
PreActResNet-18 [119]	Overfitting [202]

Chapter 5

Formalization

5.1 Introduction

In this chapter, we formalize adversarial example attacks with an emphasis on the information and the capabilities the adversary has access to to perform their attack. Another concern we tackle later in this chapter is how the problem of adversarial examples is treated. This machine learning security problem is mostly studied as a machine learning problem rather than a security problem. This is a problem that has led to a lack of a proper theoretical framework that is based on proper security principles. In turn, this has been reflected in work whose impact is limited due to the inherent flaws in their security assumptions.

Previous systematization of knowledge works ([208] or [209]) have looked at the adversarial example problem from various angles. Papernot et al. ([208]) took a holistic approach and proposed a thorough overview of the possible threat models as well as the training and inference process in an adversarial setting. They emphasized a clear distinction between the different domains (physical vs. digital) while describing the overall ML attack surface. This was reiterated more recently by Lin et al. ([62]) where the authors present an attack on the physical domain specifically. In contrast, Byun et al. ([45]) took advantage of the properties of the physical domain by projecting adversarial examples onto virtually 3D-rendered physical objects (like a mug) to improve the transferability of their attack.

Another approach was the one by Carlini & Wagner ([209]). In their paper, they present generating adversarial examples as a constrained optimization problem. They first

cover a set of existing attacks at the time and from their formalization, they introduced three new attacks adapted each for a specific norm distance metric (L_0-norm, L_1-norm, and L_2-norm). However, we found that these works fail to properly cover adversarial information and capabilities. This was not a significant problem back in 2017 and 2018 when most of the literature focused on white-box attacks, attacks where the adversary has immense knowledge and capabilities and therefore covering the minutia is not as important.

Recently, however, the field has experienced a shift toward more practical, realistic and usable attacks. This is reflected in the emergence of unrestricted adversarial examples ([86], [105], [85]) as well as threat models like the black-box and no-box (transferable) threat models ([11], [45], [54], [64], [92], [82]). A work that took the first step toward rationalizing this shift is the work of Gilmer et al. ([210]). One of the core messages of their work is that, at the time, research on adversarial examples was predominantly focused on abstract, hypothetical scenarios that lacked direct relevance to any specific security issues. This still partially holds up to this day. They did mention the question of the attacker's knowledge but did not go into depth beyond the already common concepts of white-box and black-box. Papers on defense techniques have also not provided a comprehensive account of attackers' capabilities and constraints that would be applicable in real-world security scenarios. However, they presented a wide-reaching list of salient situations that represent at their core the action space of the attacker. The action space of an attacker can be defined as the set of potential actions or strategies at the disposal of an adversary. This encompasses the various methods and choices available to the attacker when attempting to compromise a system, exploit vulnerabilities, or achieve their malicious goals during an attack.

In this work, we propose to delve deeper into the knowledge that an attacker has when performing an attack. This goes beyond just the knowledge of the machine learning system that it is trying to attack but also includes access to data and computing resources, as well as knowledge of the code and hyperparameters used to set up and train said machine learning system. To this end, we categorize adversary knowledge into four orthogonal information categories. They are the following: model information (white-box, black-box...), training information, data information, and defense information.

Gilmer et al. ([210]) were also among the first to present the problem of adversarial examples as a security game. This contrasts with Carlini & Wagner ([209]) and their optimization-based representation. This brought it more in line with other security issues and away from a pure machine learning problem. To emphasize this, they also address in their paper the notion of game sequence. This is the player order and whether the

game is repeated. This was not particularly addressed by [209] but is key to a practical understanding of the problem. This relates the problem of adversarial examples to those of standard cryptographic security games. Later, Bose et al. ([91]) expanded upon the game nature of adversarial examples while tying it in with its underlying min-max optimization nature. They use this expansion to develop an attack in what they call the NoBox (also called transferable) setting and to bridge the gap between the theory and the practice in more demanding and realistic threat models.

Previous work has taken valuable steps toward bridging the gap between theory and experimental, but we believe that much remains to be done. To that effect, we present an updated, concise and comprehensive security game for adversarial examples that conforms with the security and machine learning communities. It draws heavy inspiration from cryptographic security games. Therefore, we propose a novel theoretical framework to study adversarial examples. It contains:

- A formal approach to understanding and modeling adversary action space and revised definitions to adjust for it;
- A categorization of adversary knowledge into four information categories;
- An updated, concise and comprehensive security game for adversarial examples that conforms with the security and machine learning communities.

5.2 A formal approach to understanding and modeling adversary action space

We introduce the building blocks for a new theoretical framework to study adversarial examples. This is why in the next sections we:

- Review and expand on previous work.
- Revise the definitions and theoretical elements that prevent the construction of a unified theoretical framework.
- Introduce a new theoretical object: the detector, which fully captures realistic restrictions to the action space of the attacker into our adversarial example security game.

5.2.1 Adversary action space

A key element to differentiate between inference time attacks is the action space that is available to the attacker. Real-world practical restrictions are translated into mathematical constraints on the generated adversarial examples. [210] first attempted to properly define the action space of the adversary for adversarial example attacks. In their paper, they identify five salient situations that we will re-iterate and connect to other work in the context of the image classification domain. Researchers usually represent the restriction to the action space of the attacker under the form of a distance function between a benign input (or distribution of inputs) and an adversarial input $|x, x'|_c$. These distance functions are also usually accompanied by an associated per-input $\epsilon(x, x')$ budget that describes the maximum distance allowed under the constraint. These functions have become core components of current threat models, but they are not always built on a solid foundation reflecting reality. Threat models and the implied restrictions on the action space of the attacker need to be directly built from (potentially yet to be deployed) usable machine learning systems. Threat models not built on such foundations can yield meaningless attacks in terms of actual security. To that purpose, we complement previous work ([91]) by linking existing work in the image classification domain to each of the different salient situations when applicable.

- *Indistinguishable perturbation*: The distance metric here is not optional. This setting is where most of the research efforts have been focused since the discovery of adversarial examples by [6]. Most of the literature uses either the l_2 or l_∞ norm with a set maximum epsilon ϵ budget per image. Some works have argued for the use of the l_0 norm ([37], [211], [212]). As the dimensionality of the inputs increases, so does the indistinguishability of l_0-perturbed inputs, as each pixel occupies a smaller and smaller fraction of the overall image.

- *Content-preserving perturbation*: Here, the distance metric can be optional depending on the process used to create the adversarial examples. For example, in [45], the process takes in an input and modifies it in a way that is known and sure to produce a new input that will preserve the content of the original input They do so by first applying an indistinguishable perturbation (which is inherently content preserving for meaningful ϵ values) and then applying it as the texture to a 3D rendered mesh. They finally apply transformations when rendering the image (changing the orientation, the lighting etc.) but none of these significantly alter the content of the input. Other methods for content-preserving (mistakenly sometimes called unrestricted in the literature) include methods based on empirical assumptions. [213] attempt to

discover stylized content-preserving adversarial examples by assuming that images belonging to the same class exhibit consistent content while primarily varying in their styles. Likewise, [62] use the CycleGAN framework to generate adversarial makeup that is constrained such that it applies to a person's face. Other such attacks would be color-based attacks ([214], [215], [216], [217], [218], [219]) which aim to conserve content, granted the content is not inherently dependent on its color (coloring the ocean red vs. changing the color of a piece of clothing).

- *Non-suspicious input*: The notion of distance here is changed from a notion of distance between pairs of inputs (a benign and an adversarial input) to a distance between an (adversarial) input and the distribution of benign inputs of a class. To paraphrase [210], the attacker has the freedom to create any input example they desire, with the condition that it must convincingly pass as a genuine input to a human. Attacks that fit such a description would be generative methods [220], such as Generative Adversarial Network (GAN) based examples [221]. These generative methods do not necessarily need to use an existing benign sample as input to generate adversarial examples. However, some ([166], [93]) can instead be used to generate an indistinguishable noise perturbation that is added to a benign image.

- *Content-constrained input*: In this case, the restriction is around the content (similarly to a content-preserving perturbation). The input is required to contain certain content but human perception might not be a limiting factor. In most cases, it has limited relevance or suitability for the image classification domain beyond maybe fooling detection frameworks (object detection for example).

- *Unconstrained input*: by definition, there is no distance metric since the attacker is allowed to submit anything. It is not very applicable to the image classification domain beyond maybe attacking visual biometrics locks.

Only three out of the five salient situations currently have a significant amount of related work while one is still in its infancy (content-constrained inputs). However, even when ruling out the two irrelevant salient situations, there is a lack of a unified theoretical framework that can fit all the situations simultaneously. The indistinguishable perturbation situation relies on l_p-norm-based restrictions while the content-preserving perturbation situation relies more on image similarity metrics and non-suspicious situations on image quality metrics. Each of these metrics has inherent differences that make it hard to unify them under a singular theoretical framework. For example, an indistinguishable perturbation implies the existence of an originally unperturbed image that is used to craft said

perturbation. A non-suspicious or content-constrained input does not have such an inherent requirement.

5.2.2 Bounding the adversary action space: the distinguisher

To propose a unified theoretical framework that can accommodate every salient situation in 5.2.1, we introduce an object that is already well-defined in the field of cryptography: the distinguisher. In the realm of cryptography, a distinguishing attack encompasses various cryptanalysis techniques applied to data encrypted using a cipher, to enable an attacker to differentiate the encrypted data from randomly generated data. In essence, we have the same situation appearing in adversarial examples, where the salient situation can be seen as trying to differentiate between benign data and malicious data within setting-related restrictions. The attacker in this setting is called a distinguisher and in cryptography, they can be used in three distinct classes of settings: the information-theoretically secure setting, the statistically secure setting, and the computationally secure setting. We re-use the definitions from [222] to describe all three settings as follows:

In the **information theoretic** setting, the adversary is computationally unbounded. Therefore, security in this scenario doesn't depend on unverified complexity assumptions. It is directly associated with the notion of *perfect security*. Perfect security, in more straightforward terms, entails that the result of a real protocol execution with a legitimate adversary should be indistinguishable from the result of an ideal execution involving a trusted party and an ideal adversary/simulator.
The **statistical** setting also assumes that the adversary is computationally unbounded, however, the outcome of the actual protocol execution should be statistically similar to the outcome of an ideal execution, with no requirement for an exact match.
Finally, the **computational** setting does not assume a computationally unbounded adversary. Instead, it is assumed that the adversary operates within a probabilistic polynomial-time framework, and the security of protocols usually depends on the presumed difficulty of solving certain problems, such as factoring a large composite number into its prime factors (RSA).

In the context of adversarial machine learning and even more so inference time attacks, these settings are incongruent. The computationally unbounded settings conflict with our aim to impose practical limitations on both attackers and defenders. In the context of adversarial examples, it would overly constrain the attacker, potentially making more realistic

attacks more effective than reported within practical threat models. Lastly, our scenario closely resembles the computational setting. Nevertheless, our fundamental challenge is rooted in simulating human-like image perception, a problem lacking a well-established mathematical framework. Consequently, we must define our own setting, complete with its corresponding distinguisher. The context we are operating within assumes human-like capabilities for the distinguisher, for example, for indistinguishable perturbations, it is so human reviewers would not be able to distinguish that the image is adversarial. Therefore, we need to introduce a new class of distinguishers, the **human-like distinguisher** that we define as a distinguisher whose capabilities match that of a human or any human-made real-world system. Current existing metrics (l_p-norm, similarity, quality...) all fall under this category and are used as an approximation of human-like capabilities for their salient situation.

By combining the goal of the adversary, the level of involvement of the human-like distinguisher, and the information accessible to the adversary, we can represent any existing real-world threat model. When we refer to "the level of involvement," we are alluding to the extent of human engagement in the validation process of input images as they transition from the physical realm to the digital domain before being input into the machine learning model. For example, in the case where every input is individually vetted by a human directly, then an indistinguishable perturbation is necessary. Whereas, in scenarios such as copyright detection algorithms, a perturbation that preserves the content is satisfactory. In this case, the primary objective of the adversary is to maintain the quality of the copyrighted content while avoiding detection by the copyright algorithm. However, it becomes clear that once a human is introduced into the process, the content is likely to be identified, given the nature of human discernment. Nonetheless, this is not a hindrance, as we can create a distinguisher with human-like capabilities to emulate the detection system's capabilities and develop our attack while adhering to such constraints. This example showcases that the distinguisher, like the adversary in a cryptographic game, is limited to the underlying capacity assumptions laid out by the salient situation. Our definition places an upper bound on the capabilities of the distinguisher (only as good as a human or a human-made real world system). Informally, it encompasses the *least amount of work necessary* to avoid detection before considering whether the attack is successful or not. As detection capabilities evolve, these underlying assumptions can be revised, much like in the computational setting when the hardness of a problem is disproven (like factoring a large composite number into its prime factors after the invention of quantum computers).

5.2.3 Definitions

We can represent a salient situation as a distinguisher $D : \mathcal{I} \to \{0, 1\}$ who attempts to output 0 on benign samples and 1 on adversarial samples, it is meant to capture the detection capabilities in place in the system under attack. We can use this formalization of salient situations and provide a proper formalization of the notion of *indistinguishability* for any setting, not just the indistinguishable perturbation setting.

Definition 15 (Indistinguishability)**.** *Assuming a salient situation with distinguisher $D : \mathcal{I} \to \{0, 1\}$, let $A, B \subset \mathcal{I}$ where A is the set of all adversarial examples and B is the set of all benign samples we say an input $x \in \mathcal{I}$ is indistinguishable if we have:*

$$\zeta(n) = |\Pr_{x \leftarrow A}[D(x) = 1] - \Pr_{x \leftarrow B}[D(x) = 1]| \tag{5.1}$$

where $\zeta(n)$ is negligible in terms of $|A \cup B| = n$.

However, this definition is too restrictive as in practice n is too large to obtain $\zeta(n) = \text{negl}(n)$ and should therefore serve as a target for the community to strive for rather than a strict requirement. Instead, for practical purposes, we can use a loosened notion of *stealth* to bind adversarial examples.

Definition 16 (Stealth)**.** *Assuming a salient situation with distinguisher $D : \mathcal{I} \to \{0, 1\}$, let $A, B \subset \mathcal{I}$ where A is the set of all adversarial examples and B is the set of all benign samples we say an input $x \in \mathcal{I}$ is indistinguishable if we have:*

$$\zeta(n) = |\Pr_{x \leftarrow A}[D(x) = 1] - \Pr_{x \leftarrow B}[D(x) = 1]| \tag{5.2}$$

where $\zeta(n) \in \mathcal{O}(log(n))$, $|A \cup B| = n$.

Even then, stealth is still very hard to achieve. If we discretize the space of images, assuming image height h, image width w and RGB coloring (256 values per color, 3 colors), we get $(h * w * 3)^{256}$ possible images.

With our new definitions of indistinguishability and stealth, we converted an optimization constraint into a security game constraint.

Recall [10]'s definition of adversarial examples, definition 9. While this is a strong definition, it misses a key element of adversarial examples, namely input-free generative

methods. These methods do not actually need an input sample to construct another input that itself is adversarial. "It is unclear how such (malicious perturbations on correctly handled examples) misclassifications represent a different kind of security problem than other errors, or even other attacker-produced examples that have no specific relationship to an uncorrupted input." ([210]). We therefore extend the definition to the following (a similar definition to [223]):

Definition 17 (Adversarial Example)**.** *An input sample $i \in \mathcal{I}$ and its associated label $gt(i) \in \mathcal{L}$ is said to be adversarial if it was specifically crafted to successfully trigger a learned model to output an incorrect answer.*

This new definition can now seamlessly accommodate all the relevant salient situations that an attacker might use to construct a threat model. Additionally, we now call adversarial examples crafted according to the previous definition (9) *grounded* adversarial examples (as they require an existing benign image to craft).

5.3 Categorization of threat models

Using section 5.2, we can decompose a threat model into three crucial components:

- A distinguisher (salient situation)
- The information the adversary has access to
- An adversary goal

We have formalized the initial component and will proceed to formalize the second component.

5.3.1 Information Extraction Oracles

To formalize the information an adversary has access to when mounting and performing an attack, we need a generic theoretical representation of information. Since precisely quantifying and defining information is quite difficult, we avoid it altogether and instead present a generic method for representing information given/acquired by the adversary. To that effect, we introduce Information Extraction Oracles (IEOs). IEOs serve as a

translation tool that allows an attacker to transform a threat model into actionable and properly defined information. They can be used to represent the real-world access to information available to the attacker (through any of its real-world attack capabilities). The threat model usually describes the attacker's capabilities. We now introduce an improved method for presenting threat models in a standardized and formalized fashion.

Definition 18 (Information Extraction Oracle)**.** *An Information Extraction Oracle* $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ *is a stateful oracle machine that can take in a query and outputs information in a binary format.*

For convenience, we define the set of all information extraction oracles as $\mathbb{O}$. Instead of describing attacker capabilities with vague terminology, we can now use information extraction oracles to precisely and accurately capture the attacker's knowledge and capabilities. As mentioned in definition 18, information extraction oracles are stateful. The adversary can extract the oracle's state using the *State* function.

Definition 19. *We define the function State* $: \mathbb{O} \to \mathbb{R}^*$ *that takes in as input an Information Extraction Oracle and outputs its current state as a set of real numbers. If the input oracle does not have a state, it returns* $\emptyset$*.*

To accurately represent relationships between various threat models (white-box vs. black-box for example), we need to introduce a relational structure to information extraction oracles. To do so, we define the domination $\sqsubset$ operator on IEOs.

But first, we need to define the different symbols we will use:

- $\{\}$ represents an unordered set (nothing crazy)
- $[]$ represents an ordered set
- $\cdot$ represents what we call an element, which we define as anything that is not a set.

Definition 20 (Information Extraction Oracle domination operator)**.** *We define the operator* $\sqsubset$ *for information extraction oracles over their outputs in the following way:*

Let $\mathcal{O}_1(a) = x$ *and* $\mathcal{O}_2(a) = y$*. We have three base cases:*

1. *(a)* x *and* y *are both sets. Then* $\mathcal{O}_2 \sqsubset \mathcal{O}_1$ *if* $\forall a \in \mathcal{I}$, $x \subset y$.
   (b) x *is an element and* y *is an unordered set. Then* $\mathcal{O}_2 \sqsubset \mathcal{O}_1$ *if* $\forall a \in \mathcal{I}$, $x \in y$.

(c) *x is either an unordered set or an element and y is an element. Then $\mathcal{O}_2 \sqsubset \mathcal{O}_1$ if there exists a probabilistic polynomial-time (PPT) function f s.t. $\forall a \in \mathcal{I}$, $f(x) = y$.*

Using these base cases, we can expand to the three ordered set cases.

2. (a) *$x = [x_1, \ldots, x_n]$ is an ordered set and y is either an unordered set or an element (n is a positive integer). Then $\mathcal{O}_2 \sqsubset \mathcal{O}_1$ if $(y \sqsubset x_1) \vee \cdots \vee (y \sqsubset x_n)$.*

   (b) *x is an unordered set or an element and $y = [y_1, \ldots, y_k]$ is an ordered set (k is a positive integer). Then $\mathcal{O}_2 \sqsubset \mathcal{O}_1$ if $(y_1 \sqsubset x) \wedge \cdots \wedge (y_k \sqsubset x)$.*

   (c) *$x = [x_1, \ldots, x_n]$ is an ordered set and $y = [y_1, \ldots, y_k]$ is an ordered set (n and k are positive integers). Then $\mathcal{O}_2 \sqsubset \mathcal{O}_1$ if $\forall i \in \{1, \ldots, k\}, \exists j \in \{1, \ldots, n\}$ s.t. $y_i \sqsubset x_j$.*

We let $\not\sqsubset$ be the not operator for $\sqsubset$. Meaning when not $A \sqsubset B$, then $A \not\sqsubset B$.

When an attacker has access to multiple oracles $\mathcal{O}_A, \mathcal{O}_B$ where a domination relation cannot be established, we describe the resulting combined oracle as follows: $\mathcal{O}_{A\&B}(x) = [\mathcal{O}_A(x), \mathcal{O}_B(x)]$.

Definition 21 (Information Extraction Oracle Combination)**.** *Given two IEOs $\mathcal{O}_A, \mathcal{O}_B$ for which neither $\mathcal{O}_A \sqsubset \mathcal{O}_B$ nor $\mathcal{O}_B \sqsubset \mathcal{O}_A$ is true. Then we define the combined oracle $\mathcal{O}_{A\&B}$ as follows: $\mathcal{O}_{A\&B}(x) = [\mathcal{O}_A(x), \mathcal{O}_B(x)]$*

5.3.2 Information Categories

An IEO provides an interface for the attacker to both the knowledge of the defender (for example, information about the target model) and any additional external knowledge possessed by the attacker (for example, an additional dataset that is disjoint from the defender's training set but is drawn from the same distribution). We can then categorize the different aspects of a threat model into different IEOs that allow the attacker access to the associated information and nothing more. We used all the distinct threat models we observed in the literature to build this categorization to have a complete representation of the threat models in the field and a meaningful categorization.

We can classify the information involved in an adversarial example attack into three distinct types:

- Information used by the defender.

- Information generated by the defender (for example model parameters θ_M).
- Additional public information (data, pre-trained models ...).

Within those classes, we identified the salient categories that combined can be used to reconstruct any threat model:

- Model information.
- Data information.
- Training information.
- Defense information.

We informally introduce those categories, note that this is a non-exhaustive list and that if any additional information were to be used by attacks in the future, they should be added to this list.

Model information: Model information encompasses all model-related information once it has been trained and all the information that can be gained by the attacker from querying said model. It represents what people usually describe as a "white-box", "gray-box", "black-box" and "no-box" (or "transferable") setting.

Data information: Data information can be described as all the information relating to model inputs $\mathcal{I}$ and their labels $\mathcal{L}$. This can be the training data $\mathcal{D} \subset \mathcal{I} \times \mathcal{L}$ used to train the model under attack, any additional data $\mathcal{D}' \subset \mathcal{I} \times \mathcal{L}$ drawn from the same distribution as $\mathcal{D}$, and any other data that is not from the same distribution as the training data $\mathcal{E} \subset \mathcal{I} \times \mathcal{L}$.

Training information: Training information includes any information describing the training process that was used to obtain the model parameters θ_M. We may divide it into algorithmic information and the associated hyperparameters. Algorithmic information represents the properties of the training algorithm, such as the loss function used to train the model, the optimization algorithm (SGD, Adam), the learning rate scheduler, any data pre-processing, ...

Defense information Defense information pertains to defense mechanisms employed by the defender on top of the distinguisher D. This information can range from being fully aware of any defensive precautions to not being aware of any. We can also include knowledge and implementation of current existing state-of-the-art defenses as proxies for the unknown actual defense as possible knowledge at the hands of the adversary.

5.3.3 Information Hasse Diagram

Now that we have defined broad information categories for attackers, we can introduce a structure that will allow for meaningful comparisons between sets of assumptions within each category.

Unfortunately, direct comparisons within each category are not always possible because there is currently no theoretical method to compute the relative capabilities provided by each information extraction oracle. As a result, it is not feasible to define a meaningful "$<$" operator within this context. To emphasize my point, how could one provably demonstrate that having access to $\mathcal{D}$ is necessarily better (or worse) than having access to $\mathcal{D}' \cup \mathcal{E}$ for every possible D, D' and E?

Therefore, we instead make use of the $\sqsubset$ operator that we defined in definition 20 to build Hasse diagrams. These diagrams are usually used to visually represent sets ordered by inclusion. However, it is not restricted to just that. We design our Hasse Diagrams to visually represent our oracles ordered by $\sqsubset$. For each category, we provide the associated Hasse diagram. In order theory, a Hasse diagram is supposed to represent finite partially ordered sets, however, we extend it to infinite partially ordered sets by allowing formulaic set descriptions (which can expand to generate infinite but partially ordered sets). For simplicity's sake, when we describe information extraction oracles, we assume that the inputs they are queried with are in $\mathcal{I}$, otherwise, they return [].

Model information Hasse diagram:

For model-related information, it can either be *static* or *query-based*. For static information, we have:

- Model parameters θ_M (white-box). Let $\mathcal{O}_M$ be its information extraction oracle. We define $\mathcal{O}_M$ as $\mathcal{O}_M(x) = [\theta_M, x]$ and $State(\mathcal{O}_M) = []$.
- Model architecture: either the exact model architecture or a set of possible architectures.

– Exact model architecture ϕ_M. Let $\mathcal{O}_A$ be its information extraction oracle. We define $\mathcal{O}_A$ as $\mathcal{O}_A(x) = [\phi_M, x]$ and $State(\mathcal{O}_A) = []$.

– Set of possible architectures: Let $\mathcal{O}_{SPA}$ be its information extraction oracle. We define $\mathcal{O}_{SPA}$ as $\mathcal{O}_{SPA}(x) = [\{\phi_0, \phi_1, \ldots, \phi_k\}, x]$ for some positive integer k and where for a uniformly randomly sampled $i \in [0, 1, \ldots, k]$, $\phi_i = \phi_M$. We have $State(\mathcal{O}_{SPA}) = []$.

For query-based information, we have:

- Query-access to model scores (model logit probabilities or values on queried inputs). Let $\mathcal{O}_S$ be its information extraction oracle. We define $\mathcal{O}_S$ as $\mathcal{O}_S(x) = [M(x), x]$
- Query-access to model labels (predicted class on queried inputs). Let $\mathcal{O}_L$ be its information extraction oracle. We define $\mathcal{O}_L$ as $\mathcal{O}_L(x) = [argmax(M(x)), x]$. We let $argmax$ be the function that returns the label from the model predictions by using the highest score value as an indicator.

For both, in the case where the attacker has a limited query budget, we let $State(\mathcal{O}) = [k]$ where $k \in \mathbb{Z}^+$ is the number of queries left for the attacker for this attack (0 when he runs out). Otherwise, it returns []. Finally, there is the case where no information model information is available, the no-box setting. We can then order their associated information extraction oracles in the following way:

Theorem 1. *Figure 5.1 holds under the $\sqsubset$ ordering.*

We can prove the ordering of this Hasse diagram using definition 20. We will first state the following lemmas and use them to prove Theorem 1. We will prove them afterwards.

Lemma 1. *$\mathcal{O}_{S\&A} \sqsubset \mathcal{O}_M$ and $\mathcal{O}_M \not\sqsubset \mathcal{O}_{S\&A}$*

Lemma 2. *$\mathcal{O}_L \sqsubset \mathcal{O}_S$ and $\mathcal{O}_S \not\sqsubset \mathcal{O}_L$*

Lemma 3. *$\mathcal{O}_{SPA} \sqsubset \mathcal{O}_A$ and $\mathcal{O}_A \not\sqsubset \mathcal{O}_{SPA}$*

Proof: Theorem: 1. First, $\mathcal{O}_{SPA} \sqsubset \mathcal{O}_{L\&SPA}$, $\mathcal{O}_S \sqsubset \mathcal{O}_{S\&SPA}$, $\mathcal{O}_S \sqsubset \mathcal{O}_{S\&A}$, $\mathcal{O}_A \sqsubset \mathcal{O}_{L\&A}$, and $\mathcal{O}_L \sqsubset \mathcal{O}_{L\&A}$ follow directly from definition 21 and part 2(a) of definition 20.
Likewise, $\mathcal{O}_{L\&SPA} \not\sqsubset \mathcal{O}_{SPA}$, $\mathcal{O}_{S\&SPA} \not\sqsubset \mathcal{O}_S$, $\mathcal{O}_{S\&A} \not\sqsubset \mathcal{O}_S$, $\mathcal{O}_{L\&A} \not\sqsubset \mathcal{O}_A$, and $\mathcal{O}_{L\&A} \not\sqsubset \mathcal{O}_L$ follow directly from definition 21 and part 2(b) of definition 20.
Using Lemma 2 and part 2(c) of definition 20, we yield that $\mathcal{O}_{L\&A} \sqsubset \mathcal{O}_{S\&A}$, $\mathcal{O}_{S\&A} \not\sqsubset \mathcal{O}_{L\&A}$,

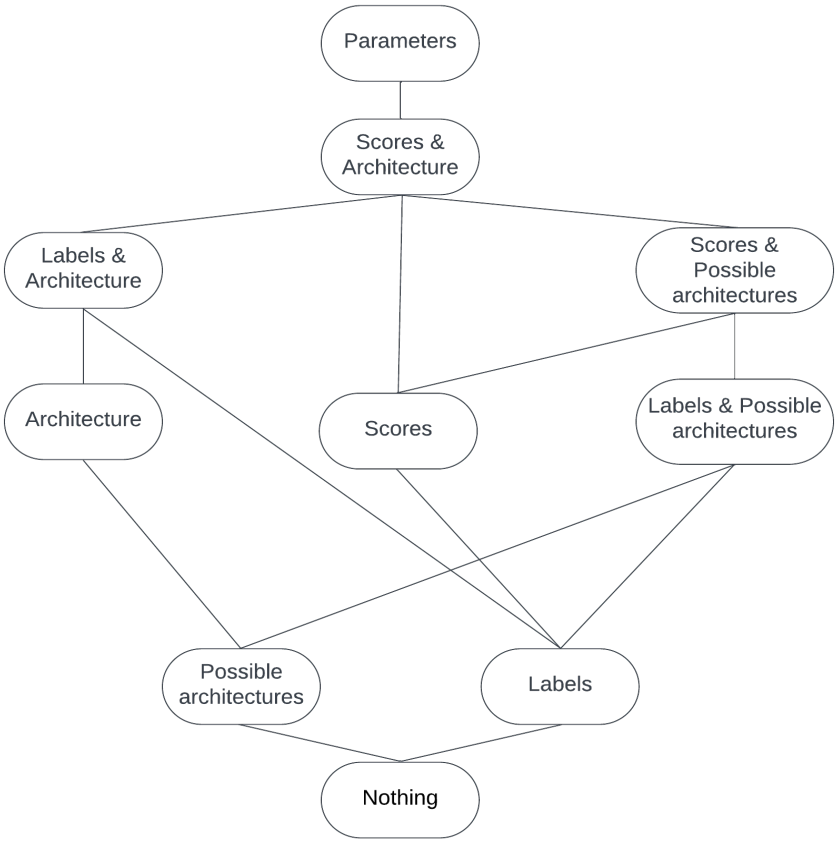

Figure 5.1: Model Oracle Hasse Diagram

$\mathcal{O}_{L\&SPA} \sqsubset \mathcal{O}_{S\&SPAA}$, and $\mathcal{O}_{S\&SPA} \not\sqsubset \mathcal{O}_{L\&SPA}$.
Using Lemma 3 and part 2(c) of definition 20, we get that $\mathcal{O}_{S\&SPA} \sqsubset \mathcal{O}_{S\&A}$ and $\mathcal{O}_{S\&A} \not\sqsubset \mathcal{O}_{S\&SPA}$.
Using all of this in addition to Lemma 1, we get that Theorem 1 holds under the $\sqsubset$ ordering. □

We now move on to proving the various Lemmas we used to prove Theorem 1.

Proof: Lemma 1. To show $\mathcal{O}_{S\&A} \sqsubset \mathcal{O}_M$, we need to find PPT f such that, $\forall a \in \{0,1\}^*$, $\mathcal{O}_{S\&A}(a) = f(\mathcal{O}_M(a))$. Incidentally, we also need that $State(\mathcal{O}_{S\&A}) = f(\mathcal{O}_M(a))$ when both oracles are queried the same number of times. We will have two cases, Case 1 where

the attacker has unlimited queries, and Case 2 where the attacker is limited.
For the sake of simplicity, we assume that $a \in \mathcal{I}$ (as otherwise they both return [], and we can set f to return [] when it receives [] anyway and $\mathcal{O}_{S\&A}(a) = f(\mathcal{O}_M(a))$).

Case 1: Let f be the following:

- On input $x \in \mathcal{I}$, f receives $[\theta_M, x] = \mathcal{O}_M(x)$. From θ_M, we can construct M and ϕ_M (by definition of θ_M).
- We can then compute and return $[M(x), \phi_M, x] = \mathcal{O}_{S\&A}(x)$.

Case 2: Let f be the following and k the current allowed number of queries left:

- On input $x \in \mathcal{I}$, f receives $[\theta_M, x] = \mathcal{O}_M(x)$. From θ_M, we can construct M and ϕ_M (by definition of θ_M).
- If $k = 0$, return $[[], \phi_M, x] = \mathcal{O}_{S\&A}(x)$.
- Else, update $State(\mathcal{O}_M(x))$ to be the function that returns $[k-1]$, return $[M(x), \phi_M, x] = \mathcal{O}_{S\&A}(x)$.

So we have that $\forall a \in \{0,1\}^*$, $\mathcal{O}_{S\&A}(a) = f(\mathcal{O}_M(a))$. Hence, $\mathcal{O}_{S\&A} \sqsubset \mathcal{O}_M$.

Now for the opposite, $\mathcal{O}_M \not\sqsubset \mathcal{O}_{S\&A}$. For simplicity's sake, we'll cover only the first case, but the proof also applies to the second case by performing the update to the *State* function as we did above. We'll prove this by contradiction. Given $\mathcal{O}_{S\&A}$, an attacker can receive ϕ_M and $M(x)$ for any $x \in \mathcal{I}$. We are trying to reconstruct θ_M. In the case where M is linear, with enough queries, we can solve the system of linear equations and reconstruct θ_M. However, in the case where M is non-linear, there can be an infinite number of possible viable solutions for any given finite amount of queries, therefore it is impossible to always determine with certainty the exact θ_M (although it is possible to approximate it, as model stealing attacks demonstrate). Hence, it is not always possible to reconstruct θ_M from ϕ_M and score query-access, meaning that there is no PPT f s.t. $\forall a \in \{0,1\}^*, f(\mathcal{O}_{S\&A}(a)) = \mathcal{O}_M(a)$. □

Proof: Lemma 2. We will prove Lemma 2 in the same way we proved Lemma 1, i.e. $\exists f$ such that $\forall a \in \mathcal{I}$, $f(\mathcal{O}_S) = \mathcal{O}_L$.

Case 1. Unlimited queries: Let f be the following:

- On input $x \in \{0,1\}^*$, if $x \notin \mathcal{I}$, then f receives [] and return [].
- Else, on input $x \in \mathcal{I}$, f receives $[M(x), x]$ and returns $[argmax(M(x)), x] = \mathcal{O}_L$.

Case 2. Limited queries: Let f be the following and k the current allowed number of queries left:

- On input $x \in \{0,1\}^*$, if $x \notin \mathcal{I}$, then f receives [] and return [].
- Else, on input $x \in \mathcal{I}$, f receives $[M(x), x]$ if $k > 0$ else it receives $[[], x]$.
- If $k = 0$, return $[[], x] = \mathcal{O}_L$.
- Else, return $[argmax(M(x)), x] = \mathcal{O}_L$.

For the other direction, similarly to Lemma 1, we will prove only the first case as the proof can trivially be extended to the second case. Given $\mathcal{O}_L$, we want to find a PPT function f s.t. $\forall a \in \mathcal{I}$, $f(\mathcal{O}_L(a)) = \mathcal{O}_S(a)$. We will show that such a function cannot exist. For $f(\mathcal{O}_L(a)) = \mathcal{O}_S(a)$ to happen, it means that f needs to always be able to at least compute the score value of the label itself. It is given no information beyond the label itself for any input $x \in \mathcal{I}$. Assume the following setup. One party holds two models M_1 and M_2 that make identical score predictions except for one specific input x^* where $M_1(x^*) = u$ and $M_2(x^*) = v$ and $u \neq v$, we will also assume that while the score changes when inferring on x^*, the predicted label remains the same for both. We argue that it would be impossible for the other party that receives only the predicted label to distinguish between both models. This means that there exists at least one instance where there is no PPT function f s.t. $\forall a \in \mathcal{I}$, $f(\mathcal{O}_L(a)) = \mathcal{O}_S(a)$. Therefore, we showed the other direction also holds.
Hence, Lemma 2 holds. □

In this case, we do not need to modify $State(\mathcal{O}_S)$ as we query $\mathcal{O}_S$ exactly once whenever we query $\mathcal{O}_L$.

Proof: Lemma 3. $\mathcal{O}_{SPA} \sqsubset \mathcal{O}_A$ follows from part 1(b) of definition 20 and $\mathcal{O}_{SPA}$'s definition. For the other direction, since $\mathcal{O}_{SPA}(x) = [\{\phi_0, \phi_1, \ldots, \phi_k\}, x]$ for some positive integer k and where for a uniformly randomly sampled $i \in [0, 1, \ldots, k]$, $\phi_i = \phi_M$, a PPT function

f would be able to always determine which of the k elements in the unordered set is the correct architecture. Given that the function is only given the unordered set, it would be akin to being able to always correctly guess a random number, which is not possible. Therefore, $\mathcal{O}_A \not\sqsubset \mathcal{O}_{SPA}$ and Lemma 3 holds. □

Data Information Hasse Diagram Regarding data-related information, we are concerned with three sets of data. The training data $\mathcal{D} \subset \mathcal{I} \times \mathcal{L}$ that was used to train the target model M and its information extraction oracle $\mathcal{O}_{\mathcal{D}}$. We define $\mathcal{O}_{\mathcal{D}}$ as $\mathcal{O}_{\mathcal{D}}(x) = [\mathcal{D}, x]$ for any $x \in \{0,1\}^*$ and $\mathcal{O}_{\mathcal{D}}(x) = []$ otherwise. We have $State(\mathcal{O}_{\mathcal{D}}) = []$. Any other data samples $\mathcal{D}' \subset \mathcal{I} \times \mathcal{L}$ that were drawn from the same distribution as $\mathcal{D}$ but were not in the training data. Likewise, we define its information extraction oracle $\mathcal{O}_{\mathcal{D}'}$ as follows: $\mathcal{O}_{\mathcal{D}'}(x) = [\mathcal{D}', x]$ for any $x \in \{0,1\}^*$ and $\mathcal{O}_{\mathcal{D}'}(x) = []$ otherwise. We have $State(\mathcal{O}_{\mathcal{D}'}) = []$. Finally, any other data samples $\mathcal{E} \subset \mathcal{I} \times \mathcal{L}$ that were not drawn from the same distribution as $\mathcal{D}$ and $\mathcal{D}'$. We define its information extraction oracle $\mathcal{O}_{\mathcal{E}}$ as follows: $\mathcal{O}_{\mathcal{E}}(x) = [\mathcal{E}, x]$ for any $x \in \{0,1\}^*$ and $\mathcal{O}_{\mathcal{E}}(x) = []$ otherwise. We have $State(\mathcal{O}_{\mathcal{E}}) = []$. In this case, creating the Hasse diagram is relatively straightforward because we only need to contemplate all feasible combinations of oracles using definition 21.

Theorem 2. *Figure 5.2 holds under the $\sqsubset$ ordering.*

Proof: Theorem 2. Theorem 2 follows directly from definitions 20 and 21. □

Training Information Hasse Diagram We obviously cannot explicitly list every possible property that may describe a training algorithm, but we can describe a systemized method for identifying its known elements. First, we must define exactly what we mean by training algorithm or function.

Definition 22 (Train function)**.** *We define a training function $Train : \{0,1\}^* \rightarrow \mathcal{M}$ as the function representing the algorithm used by the defender to generate the target model M.*

The $Train$ function can take in the algorithm's hyperparameters as well as any training data or environment (optional) needed to train the model. We can articulate known information about the training algorithm as one of the following training information sets: $T = \{Train' | g_{Train}(Train') = True\}$ where g_{Train} is a function that returns $True$ if a particular condition in the training algorithm with respect to the original $Train$ function is satisfied (e.g. the training algorithm uses cross-entropy loss) and $False$ otherwise. We

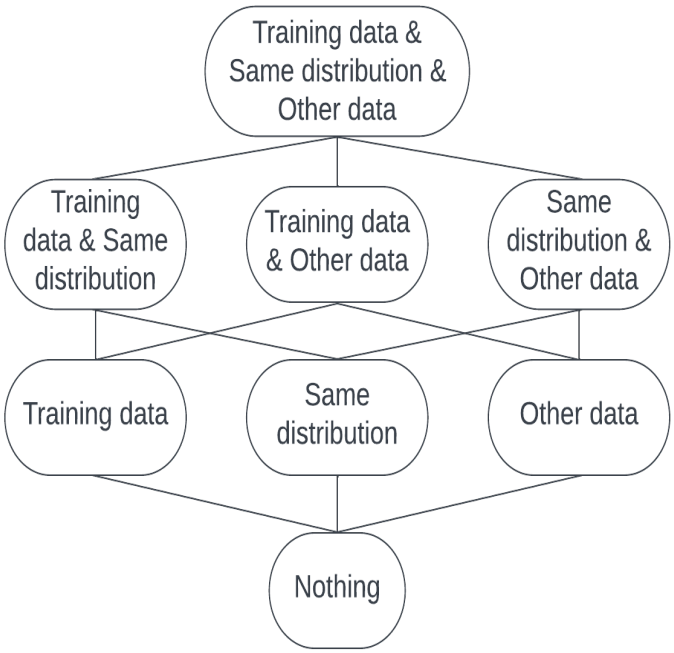

Figure 5.2: Data Oracle Hasse Diagram

can then create two kinds of information extraction oracles. The first one is $\mathcal{O}_{Train}$ which is associated with the original $Train$ function itself. It can be understood as the oracle representing having access to the actual complete training algorithm used by the defender (stolen or public). We define $\mathcal{O}_{Train}$ as $\mathcal{O}_{Train}(x) = [Train, x]$ for any $x \in \{0,1\}^*$. This is the information that can be used (combined with data) to train/assume access to pre-trained models. The other kind is when the attacker has access to one of the sets T. There are infinitely many of them therefore we will just define a generic oracle for them $\mathcal{O}_{T_i}$ for any integer i where is any one of the sets $T_i = \{Train' | g^i_{Train}(Train') = True\}$ we previously described. We then have $\mathcal{O}_{T_i}(x) = [T_i, x]$ for any $x \in \{0,1\}^*$. The oracles generated from this generic oracle can then be ordered using $\sqsubset$ following the usual rules defined in definition 20.

Theorem 3. *Figure 5.3 holds under the $\sqsubset$ ordering.*

Proof: Theorem 3. By definition, we have the oracles defined by $\mathcal{O}_T$ that are properly defined and ordered under $\sqsubset$. It remains to show that $\mathcal{O}_{Train}$ dominates any one of them

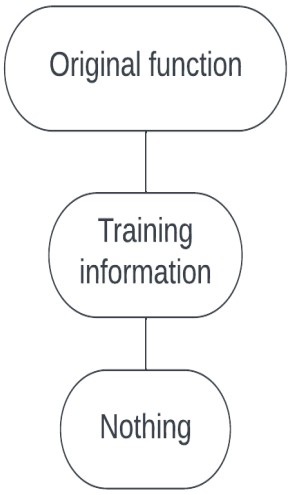

Figure 5.3: Train Oracle Hasse Diagram

and none of them dominates $\mathcal{O}_{Train}$. By how any T_i is defined, we have that $Train \in T_i$ for any integer i. Since T_i is an unordered set, we can apply part 1(b) of definition 20 to yield that $\forall i \in \mathbb{Z}$, $\mathcal{O}_{T_i} \sqsubset \mathcal{O}_{Train}$. We'll show that $\forall i \in \mathbb{Z}$, $\mathcal{O}_{Train} \not\sqsubset \mathcal{O}_{T_i}$ by contradiction. By part 1(c) of definition 20, for any integer i, there needs to be a PPT function f such that $f([T_i, x]) = [Train, x]$ for any $x \in \{0, 1\}$. Hence, we need some PPT function that can extract $Train$ from T_i (as we know it is in T_i by definition of T_i). However, T_i is an unordered set and f only has access to T_i (and the query itself x but in this case it should not provide additional information about the ordering of T_i). So, for f to be able to distinguish $Train$ from any other of the other $Train'$ functions in T_i would be equivalent to being able to perfectly guess truly random numbers, which is impossible. Therefore, we have a contradiction and so $\forall i \in \mathbb{Z}$, $\mathcal{O}_{Train} \not\sqsubset \mathcal{O}_{T_i}$. □

Defense Information Hasse Diagram As mentioned in section 5.3.2, defense information relates to any defense mechanisms put in place by the defender on top of the distinguisher D. We can partition it as follows:

- Full awareness of the defense and its parameters (similar to white-box for model information). We represent this as an algorithm ρ and its parameters ϱ. Let $\mathcal{O}_{FA}$

be its information extraction oracle. We define $\mathcal{O}_{FA}$ as $\mathcal{O}_{FA}(x) = [\rho, \varrho, x]$ for any $x \in \{0,1\}^*$.

- Partial awareness of the defense and its parameters: knowledge of the defense but not the specific parameters of the instance. Let $\mathcal{O}_{PA}$ be its information extraction oracle. We define $\mathcal{O}_{PA}$ as $\mathcal{O}_{PA}(x) = [\rho, \{\varrho_1, \ldots, \varrho_k\}, x]$ for any $x \in \{0,1\}^*$, where k is a positive integer. We also have that for some uniformly randomly sampled $i \in \{1, \ldots, k\}$, $\varrho_i = \varrho$.
- A set of potential defenses that can be obtained by having partial insider information. Let $\mathcal{O}_{SPD}$ be its information extraction oracle. We define $\mathcal{O}_{SPD}$ as $\mathcal{O}_{SPD}(x) = [\{\rho_0, \ldots, \rho_k\}, \{\varrho_1, \ldots, \varrho_k\}, x]$ for any $x \in \{0,1\}^*$, where k is a positive integer. We also have that for some uniformly randomly sampled $i \in \{1, \ldots, k\}$, $\rho_i = \rho$ and $\varrho_i = \rho$.

All of the defense information extraction oracles do not have any state and therefore the *State* function returns [] for all of them. This partition was identified from a combination of threat models used in current literature and filling the logical gaps that have not yet been explored. As the field transitions toward real-world attack scenarios, we suggest that the inclusion of partial awareness and domain knowledge introduces unexplored aspects in the domain of adaptive attack threat models. The majority, if not all, of the attacks described in attack papers, as well as the adaptive attacks discussed in defense papers, either presume complete awareness or a complete lack of awareness regarding defensive capabilities. However, in practice, actual adversaries would most likely have at least some domain knowledge and insider information when performing an attack. Assuming that an attacker does not know defensive capabilities allows for a lower-bound on the attack's performance, it might, however, unnecessarily reduce attack performance and provide a false sense of security for the defender.

Theorem 4. *Figure 5.4 holds under the* $\sqsubset$ *ordering.*

Proof: Theorem 4. $\mathcal{O}_{PA} \sqsubset \mathcal{O}_{FA}$ follows directly from part 2(c) and 1(b) of definition 20 as well as $\mathcal{O}_{FA}$'s definition since $\varrho \in \{\varrho_1, \ldots, \varrho_k\}$. $\mathcal{O}_{FA} \not\sqsubset \mathcal{O}_{PA}$ follows from the impossibility of perfectly guessing truly random numbers (since i s.t. $\varrho_i = \varrho$ is uniformly randomly sampled) and part 1(c) of definition 20. $\mathcal{O}_{SPD} \sqsubset \mathcal{O}_{PA}$ also follows directly from part 2(c) and 1(b) of definition 20 as well as $\mathcal{O}_{SPD}$'s definition since $\rho \in \{\rho_1, \ldots, \rho_k\}$. $\mathcal{O}_{PA} \not\sqsubset \mathcal{O}_{SPD}$ also follows from the impossibility of perfectly guessing truly random numbers and part 1(c) of definition 20.

□

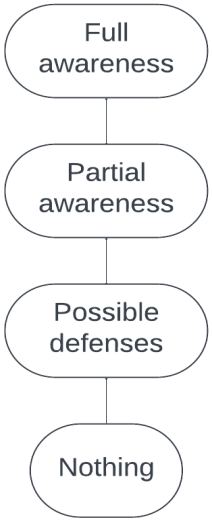

Figure 5.4: Defense Hasse Diagram

5.4 Game

Ultimately, we can reduce any adversarial attack to a very simple adversarial machine learning game. This game, involving the defender and the attacker, can be used to theoretically reason about both attacks and defenses in the field and provides a solid foundation upon which we can base our evaluations. A game can also allow us to reduce the attack process to the essence of what it is attempting to do and therefore improve knowledge transfer in the field by having a consistent theoretical frame across multiple works. However, to be able to define such a game, we are missing a couple of essential components. Therefore, in the following section, we will define those missing components. Afterwards, we will describe the game itself and finally, since games have victors and losers, we will establish and solidify the notions of success for this game.

5.4.1 Definitions

Before being able to properly define the remaining necessary functions for our game, I need to introduce a symbol and some new information extraction oracles that will appear when defining them. They are the following:

- $\varnothing$ is the termination symbol. This will typically be used by the defender when a distinguisher succeeds on an input provided by the attacker and represents the defender refusing to answer the attacker's query as it detected that it is adversarial.
- $\mathcal{O}_X^{a,b}$ is an input information extraction oracle about the data granted to the adversary for an attack. To be more precise, for example, it is the oracle that one would query to get the starting input sample when performing an attack in an indistinguishable perturbation salient situation. The construction of the attack is dependent on the attacker's goal, hence, we superscript $\mathcal{O}_X^{a,b}$ with a bit a that represents whether the attack is grounded 5.2.3 or not (0 for grounded and 1 for not-grounded). We also superscript it with a bit $b \in \{0, 1\}$ where if the attack is untargeted then $b = 0$ and if it is targeted then $b = 1$. We enumerate below the different things it can return:
  - The start input sample $x \in \mathcal{I}$ and its associated ground-truth label $y \in \mathcal{L}$ if the attack is grounded.
  - In the case of a targeted attack, the target label $y_t \in \mathcal{L}$.
- $\mathcal{O}_{Dist}$ is an information extraction oracle that returns a distinguisher D' that the attacker can utilize to verify that the generated adversarial examples are within the restrictions of the salient situation. By default, $D' = D$ where D is the actual distinguisher used by the defender. However, in situations where only partial information is known about the distinguisher D or the associated salient situation, D' can be used to capture the attacker's knowledge of D.

We can now define the key functions that we will use in our security game:

Definition 23 (*AdvGen*)**.** $AdvGen : \mathbb{O} \to \mathcal{I} \times \mathcal{L}$.
$AdvGen(O)$ on input set of information oracles $O \subset \mathbb{O}$ (including the input information oracle $\mathcal{O}_X^{a,b}$), outputs the following:

$$AdvGen(O) = [x', y'] \tag{5.3}$$

where $x' \in \mathcal{I}$ is the adversarial example and $y' \in \mathcal{L}$ is its ground-truth label.

Additionally, we also define two other functions that are vital to our security game: $Evaluate$ and $Classify$. As mentioned in Gilmer et al.'s work ([210]), an attacker can have different goals, therefore we use the $Evaluate$ function to represent an attack's success with respect to the attacker's goal. (Either a targeted (definition 12) or an untargeted (definition 11) attack or any other goal an attacker might have).

Definition 24 (*Classify*)**.** *Classify*(M, x) *on input model* M *and input sample* $x \in \mathcal{I}$ *returns the predicted label of* x *using the classifier* M *while executing any inference-time pre-processing or defense mechanisms the defender has in place.*

Definition 25 (*Evaluate*)**.** *We have two evaluation functions, one for untargeted attacks* $Evaluate_0$ *and one for targeted attacks* $Evaluate_1$ *(same as* $\mathcal{O}_X^{a,0}$ *and* $\mathcal{O}_X^{a,1}$*). They are defined as the following:*

$$Evaluate_0(y, r) = \mathbb{I}[r \neq y] \tag{5.4}$$

and

$$Evaluate_1(y_t, r) = \mathbb{I}[r = y_t] \tag{5.5}$$

Where y *is the ground-truth label,* r *is the predicted label and* y_t *is the target label for a targeted attack.*

5.4.2 Game Diagram

Now that we have all the components of the game ready, we can define it. We take strong inspiration from cryptographic games that are used to prove the security of protocols and schemes (such as encryption/decryption schemes). This game has two participants, the attacker and the defender. We assume that the attacker has access to its own distinguisher D' that would represent the attacker's assumed knowledge of the defender's distinguisher D. In theory, $D = D'$, however, in practice, a mismatch between D and D' can occur when the attacker improperly evaluates the threat model. For that reason, D' is also provided with y' and $\mathcal{O}_X^{a,b}$ on top of x' unlike D.

	Attacker		**Defender**
	$(O, AdvGen, Evaluate_b)$		$(D, Train, Classify)$
			$M \leftarrow Train(\cdot)$
	$l \leftarrow \{\}$		
	$i \leftarrow 0$		
	$\mathcal{O}_{Dist} \leftarrow O[\cdot]$		
	$D' \leftarrow \mathcal{O}_{Dist}$		
	$x', d \leftarrow \emptyset, 1$		
	$x'_i \leftarrow x'$		
	While $(d = 1) \wedge (x'_i \notin l)$		
	Do		
	$x'_i, y'_i \leftarrow AdvGen(O)$		
	$l \leftarrow l \cup \{x'_i\}$		
	$d \leftarrow D'(x'_i, y'_i, \mathcal{O}_X^{a,b})$		
	$i \leftarrow i + 1$		
	Done		
	$x' \leftarrow x'_i$		
	If $d = 1$ **Then Return** 0		
	Else	$\xrightarrow{x'}$	
			$d' \leftarrow D(x')$
			If $d' = 1$ **Then** $r \leftarrow \varnothing$
			Else $r \leftarrow Classify(M, x')$
		$\xleftarrow{r}$	
	If $r = \varnothing$ **Then Return** 0		
	If $b = 0$ **Then Return** $Evaluate_0(y, r)$		
	Else		
	$\cdot, \cdot, y_t \leftarrow \mathcal{O}_X^{a,1}$		
	Return $Evaluate_1(y_t, r)$		

$\leftarrow$ is the assignment operator, it assigns a value to a variable

$=$ is the equality comparison operator, it compares two objects and evaluates their equality

$\cdot$ is used to represent whichever argument or index required to get what is provided by the associated $\leftarrow$

5.4.3 Measuring Success

First, we have to define success before being able to measure it.

Definition 26 (Success)**.** *We say an attacker is successful at our adversarial example security game if the game returns 1. The attacker is unsuccessful if the game returns 0.*

This definition guarantees that the attacker succeeds if and only if he manages to remain undetected by the defender while completing his objective (usually misclassification). Let $G(O, D', Evaluate_b, D, Train, Classify) \rightarrow \{0, 1\}$ be an instance of the game. A straightforward option to define a proper measure of success is the following:

Definition 27 (Expected success rate)**.** *We define the expected success rate (ESR) as:*

$$\xi_G = \mathbb{E}[G(O, D', Evaluate_b, D, Train, Classify)] \tag{5.6}$$

While in theory, this simple metric should encompass well the performance of an attacker, since it is also dependent on the specific defender. Hence, it is in practice hard to use as a comparison tool to compare different attacks' performance. This problem also exists in defense literature where defense performance metrics are dependent on the specific attacks they are evaluated against. Other possible metrics we present to attempt to bridge that gap are the following, but we are aware that they are not perfect. The expected success rate metric can fail to properly evaluate the attack's quality when the defender's model's performance against benign samples is already poor (which can imply that it was improperly trained and therefore easier to attack). Let $G_b(\mathcal{O}_X, Evaluate_b, D, Train, Classify)$ be the version of our game where instead of being given adversarial examples, the defender is provided with benign samples. We use the expected success rate on this game Υ_{G_b} as a lower-bound for attack performance (meaning that by definition no attack can perform worse than this).

Another score that can temper this failure to properly evaluate an attack's quality is one that instead looks at the relative performance of the attack compared to how the attack

performs on the benign data. For simplicity's sake, we write
$G(O, D', Evaluate_b, D, Train, Classify)$ as G and $G_b(\mathcal{O}_X, Evaluate_b, D, Train, Classify)$ as G_b.

Definition 28 (Relative performance score)**.** *We define the relative performance score as:*

$$\Upsilon_{G,G_b} = \xi_G^2 - \xi_{G_b}^2 \tag{5.7}$$

This score is contained between -1 and 1 (since G and G_b are real numbers between 0 and 1). This relative score alleviates the aforementioned problem, and it can also be used to measure defense effectiveness. The score is negative when the attack performs worse than just using benign samples and is positive otherwise. We believe it captures the various trade-offs when mounting attacks and defenses more fairly than just looking at the expected success rate or even a shifted expected success rate by the lower-bound (G_b). Benign performance is an important metric when looking at overall model performance as benign users will represent most of the users (raises whether it makes sense to provide a service if all the users are malicious and aren't trying to utilize said service but instead are only trying to attack your model). We want to reward defenders for training models that have both good benign and adversarial performance. Additionally, we want to capture the relative performance change when attacking a model rather than the absolute change. When comparing attacks/defenses across various research works, the authors might be using models with the same/similar architectures, data and training, but due to the nature of gradient descent-based optimization, they might end up with models with varying performance. Hence, to alleviate this, a relative score is more appropriate and can offer a stronger basis for comparison. Figure 5.5 provides a visual understanding of the behavior of this score when varying both ξ_G and ξ_{G_b}.

5.4.4 Showcase

We will now showcase an application of our formalization to an existing well-known and well-performing attack: Projected Gradient Descent (PGD) from [7]. This should function as a template for the practical application of our formalization and the associated game. In their paper, they showcase multiple kinds of attacks, for each, we provide a table describing the utilized information. We will first identify, for each of our information categories, which of the oracle is the most appropriate to represent the knowledge used by the attacker. Then, we will proceed with defining an example initialization of our game with one of the attacks. As a reminder, the multistep PGD attack can be defined as the following:

$$x^{t+1} = \Pi_{x+\mathcal{S}}(x^t + \alpha \ \text{sgn}(\nabla_x L(\theta, x, y))) \tag{5.8}$$

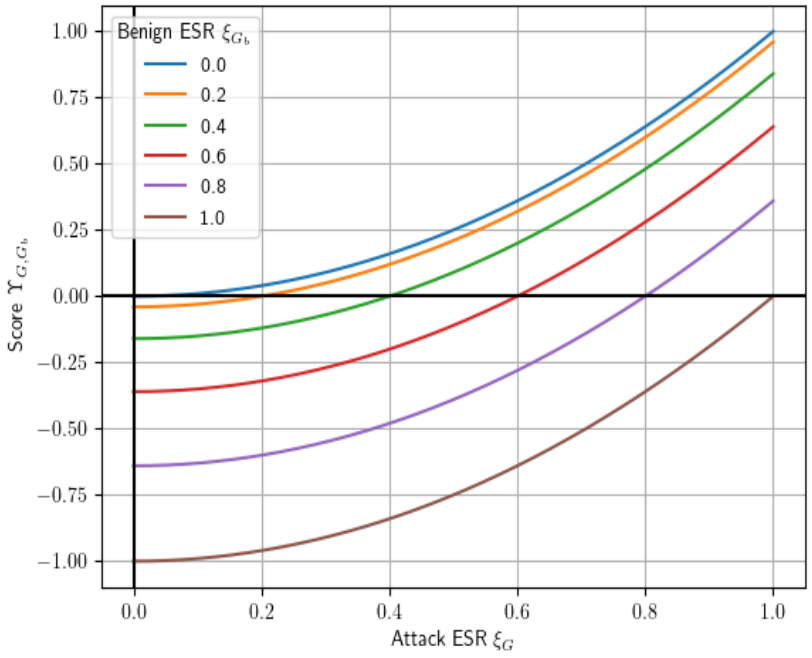

Figure 5.5: Relative Performance Score for various benign and adversary ESRs

where x is the original benign sample with its label y, x^t is the input at step t of the attack, $\mathcal{S} \subseteq \mathbb{R}^d$ is the set of allowed perturbations (d is the input dimension), θ are the model parameters, L is a loss function, sgn is the sign function (-1 if input is negative, 1 otherwise) and α is the step size.

In their paper, they consider the following adversaries:

1. "White-box attacks with PGD for a different number of iterations and restarts, denoted by source A".
2. "White-box attacks with PGD using the Carlini-Wagner (CW) loss function (directly optimizing the difference between correct and incorrect logits)".
3. "Black-box attacks from an independently trained copy of the network, denoted A'".
4. "Black-box attacks from a version of the same network trained only on natural examples, denoted A_{nat}".
5. "Black-box attacks from a different convolution architecture, denoted B".

For each, we construct the following knowledge table:

Attack	Model	Data	Train	Defense
1	Parameters	$\emptyset$	Training Information (loss function)	$\emptyset$
2	Parameters	$\emptyset$	$\emptyset$	Full Awareness
3	Scores & Architecture	Training Data	Original Function	Full Awareness
4	Scores & Architecture	Training Data	Original Function	$\emptyset$
5	Possible Architectures	Training Data	Original Function	Full Awareness

Table 5.1: PGD attacks knowledge table

Our knowledge tables allow for a clear and concise representation of the adversary's knowledge of the defender's information. We will then describe the components of the game for the first adversary (Attack #1). This is meant as a template to showcase how one would describe their attack using our game. To do this, we need to define the following components: O, D, D', $Evaluate_b$, $Classify$, $Train$, and $AdvGen$.

As shown in Table 5.1, we have the associated oracles used by the attacker in Attack #1 that are the following: $O = \{\mathcal{O}_M, \mathcal{O}_{T_1}, \mathcal{O}_{FA}, \mathcal{O}_X^{0,0}, \mathcal{O}_{Dist}\}$. For the training information (loss function) of Attack # 1, we construct $\mathcal{O}_{T_1}$, we let
$T_1 = \{Train' | g^1_{Train}(Train') = True\}$ where g^1_{Train} is the function that returns True when $Train'$ uses the same loss function as $Train$.

D varies depending on the dataset used, for example, on MNIST [170], they use an *indistinguishable-perturbation* distinguisher with the l_∞-norm metric and a maximum allowed perturbation of $\epsilon = 0.3$ (for pixel values between 0 and 1). However, for CIFAR10 [112], they also use an *indistinguishable-perturbation* distinguisher with the l_∞-norm metric, but they use a maximum allowed perturbation of $\epsilon = 8$ (where this time pixel values are RGB values between 0 and 255). They also explore a variant of their attack in the case of an l_2-bounded adversary but to keep this concise we do not analyze it. $\mathcal{O}_{Disc}(x) = D' = D$ for any $x \in \{0, 1\}$ as it is assumed that the attacker knows exactly the setting it is in (like in most other research works).

Their defense is part of the training process (adversarial training), therefore $Classify(M, x) = M(x)$. The adversary's goal is to modify the attacked model's accuracy,

meaning it is an untargeted attack (definition 11). Hence, $b = 0$ and $Evaluate_0(y, r) = \mathbb{I}[(r \neq y)]$ (definition 25). $Train$ (as defined) is the algorithm/code used to train the attacked model, due to the sheer complexity of describing the entire algorithm, we will instead point out that they link their code in their paper ([7]). We can summarize it as a standard Stochastic Gradient Descent (SGD) based model training algorithm that incorporates adversarial training as a defense. Finally, for the adversarial example generation, they use a grounded process, hence $a = 0$ and $AdvGen$ can be defined as the following (they have three Hyperparameters t, α, ϵ):

1 : $AdvGen(O)$ Hyperparameters: t, α, ϵ

2 : $\mathcal{O}_M, \mathcal{O}_{Train}, \mathcal{O}_{FA}, \mathcal{O}_X^{0,0}, \mathcal{O}_{Dist} \leftarrow O$

3 : $i \leftarrow 0$

4 : $x, y \leftarrow \mathcal{O}_X^{0,0}(0)$

5 : $\theta_M \leftarrow \mathcal{O}_M(0)$

6 : $\cdots, L, \cdots \leftarrow \mathcal{O}_{Train}(0)$

7 : $x^i \leftarrow x$

8 : **While** $i < t$

9 : **Do**

10 : $x^{i+1} \leftarrow \Pi_{x+\mathcal{S}}(x^i + \alpha \ \text{sgn}(\nabla_x L(\theta_M, x, y)))$

11 : $x^{i+1} \leftarrow \text{clip}(x^{i+1}, x - \epsilon, x + \epsilon)$

12 : $x^{i+1} \leftarrow \text{clip}(x^{i+1}, 0, 1)$

13 : $i \leftarrow i + 1$

14 : **Done**

15 : **Return** x^i

Where the clip function is the vectorized version of the following function (b and c can either be vectors or real numbers but are usually real numbers):

$$\text{clip}(a, b, c) = \begin{cases} b, & \text{if } a < b \\ c, & \text{if } a > c \\ a, & \text{otherwise} \end{cases}$$

Chapter 6

Conversion

In this chapter, we go over the papers we surveyed in section 4.1 and apply our formalization to extract the information used by each attack for each of the information categories we identified. We also identify the salient situation, whether the attack is targeted and grounded and the metrics used to limit the generated perturbations. Tables 6.3 and 6.4 condense those results. Whereas, Tables 6.1 and 6.2 condense the results from applying our formalization to each of the surveyed papers.

6.1 Papers

6.1.1 Object-based Diverse Input Attack

In their paper [45] (summary 4.1.1), the authors do not specify how the models they use as both source and target models are trained. They use the models as if they are pre-trained. This is confirmed by the code implementation that they make public in their paper. They use pre-trained models made available by either PyTorch [108] or Huggingface [224]. They avoid attacking a target model that shares the same architecture as the source model, leading to their model knowledge being akin to a set of potential architectures ($\mathcal{O}_{SPA}$). These models are all trained on the ImageNet-1k dataset [101]. This means that the attacker using the source model has access to the same training data as the data used by the defender to train its model ($\mathcal{O}_{\mathcal{D}}$). Their *AdvGen* (grounded and targeted) uses both the loss function that was used to optimize the target model to compute gradients and pre-trained models that were all trained with the same training algorithm (pre-trained by PyTorch).

Therefore, the information they used to construct their attacks in their evaluation section is that of the actual training function $Train$ ($\mathcal{O}_{Train}$). As for the salient situation they are placing themselves in, the authors opted for an indistinguishable-perturbation salient situation with the l_∞-norm as their metric an epsilon $\epsilon = 16/255$.

6.1.2 Geometry-Aware Attack

We summarize the paper [40] in summary 4.1.2. The authors use pre-trained models on the ImageNet dataset [101] to attack models that were also trained on ImageNet. Hence, it is straightforward that they have access to the training data ($\mathcal{O}_{\mathcal{D}}$). As mentioned in the summary, their technique is based on using these ensembles of pre-trained models (that are not necessarily the same as the attacked model) to generate the adversarial examples without ever querying the attacked model directly. Therefore, they have access to at least a set of possible model architectures ($\mathcal{O}_{SPA}$). While sometimes, the model architecture they attack is also in the set of surrogate models they use to construct the attack, since they use the set as a whole indiscriminately, $\mathcal{O}_{SPA}$ still applies. Their *AdvGen* (grounded and untargeted) uses the loss function that was used to optimize the model, meaning that they have access to at least the training function ($\mathcal{O}_{T_1}$, $T_1 = \{Train' | g^1_{Train}(Train') = True\}$ where g^1_{Train} is the function that returns True when $Train'$ uses the same loss function as $Train$). Additionally, it is not specified whether all those models were trained using the same training function. However, since the models are pre-trained models that sometimes come from the same source, we cannot discard that it is highly likely that at least some of those models have been trained using the same training algorithm ($\mathcal{O}_{Train}$). Finally, while some of the models they use for their training f and validation h have been trained using defenses, for the results we will use to compare against other papers (Table 3 of their paper), they do not use those models and instead only use undefended models (models 1,2,3,4,5,6 from their paper's Table 1). However, nothing prevents the addition of defense information to their attack by including models trained using said defense information.

Additionally, while their attack combines an l_∞-norm attack with an unrestricted style transfer attack, and it requires a maximum allowed perturbation parameter ϵ, they never specify what values are used for ϵ when evaluating their attack.

6.1.3 Large Geometric Vicinity Attack

We summarize the paper [54] in summary 4.1.3. Their attack uses information very similarly to 6.1.2. Through their evaluation, they have two variants of their attack (when

looking at information categories). Again, the authors use pre-trained models on the ImageNet dataset [101] by PyTorch [108] to attack models that were also trained on ImageNet. Hence, it is straightforward that they have access to the training data ($\mathcal{O}_{\mathcal{D}}$). As for the surrogate model they use for their attack (ResNet-50 [107]), in their work ([107]), the authors of the architecture specify that they used the same architecture as PyTorch's, as well as the same training function and hyperparameters and their model achieves similar performance to PyTorch's model (23.81 ± 0.15 top-1 error for their model, 23.85 for PyTorch's). Hence, in variant A, we can assume that they have access to at least a set of possible architectures ($\mathcal{O}_{SPA}$) and the training function ($\mathcal{O}_{Train}$). Unlike 6.1.2, in the case (variant B) where they attack ResNet-50 with ResNet-50 in which case they have access to the parameters ($\mathcal{O}_{M}$) since they use the same models. Finally, they do not consider defenses for either the attacker or the defender, so there is no defense-related information in this paper.

6.1.4 Pixle Attack

As was discussed in the summary (4.1.4) of the paper [37], this is a black-box query-based attack that uses a random search heuristic to find adversarial examples that lower the true class' score (untargeted attack) or increase the target class' score (targeted attack). Their attack needs no additional information beyond query access to the scores ($\mathcal{O}_{S}$), therefore for all the other information categories, they use nothing ($\emptyset$).

6.1.5 MASSA Attack

Similarly to Pixle (conversion 6.1.4), MASSA [66] (summary 4.1.5) is a query-based black-box attack that uses no additional information about the defender beyond query-access to the model. However, in their case, they only need the label ($\mathcal{O}_{L}$), not the scores for the model information. Therefore, for all the other information categories, they use nothing ($\emptyset$).

6.1.6 AI-FGTM Attack

Similarly to the other transferable attacks we summarized, this attack [15] (summary 4.1.6) uses pre-trained models that were trained on the same dataset (ImageNet [101]) as the models that they attack ($\mathcal{O}_{\mathcal{D}}$). They use a set of architectures that sometimes contain the attacked model architecture ($\mathcal{O}_{SPA}$). However, their attack algorithm makes no distinction between each model in the set and uses them indiscriminately to generate

adversarial examples. Their *AdvGen* (grounded and untargeted) uses the loss function that was used to optimize the model and pre-trained models, meaning the information they used to construct their attacks in their evaluation section is that of the actual training function ($\mathcal{O}_{Train}$). Additionally, it is not specified whether all those models were trained using the same training function. Finally, while they attack defended models, none of the pre-trained source models they use contain any defended models.

6.1.7 SSAH Attack

In their paper [65] (summary 4.1.7), the authors present the SSAH attack, a grounded attack with both a targeted and untargeted variant. They evaluate their attack in two different settings, the whitebox and the transferable setting. For ease of comprehension, we will separate them as SSAH (A) for the whitebox variant and SSAH (B) for the transferable variant.

For the whitebox variant, SSAH (A), they, as the name suggests, use the model parameters in their adversarial generation process ($\mathcal{O}_M$). They, however, do not use any training information or defense-related information. Furthermore, their *AdvGen* process requires additional data from the same distribution beyond the single sample it is provided by O_X (for equation 4.9). Hence, they require additional data from the same distribution as the training data ($\mathcal{O}_{\mathcal{D}'}$).

For the transferable variant, SSAH (B), they train surrogate models with similar but different architecture (ResNet-20 surrogate for ResNet-18 [95] and VGG-11 surrogate for VGG-16 [98]), this is akin to sampling from a set of possible architecture ($\mathcal{O}_{SPA}$). Those models are trained on different datasets from the same distribution (CIFAR10/100 [112] for the surrogates, ImageNet [101] for the target, $\mathcal{O}_{\mathcal{D}'}$) with the same training functions (they train them themselves, hence $\mathcal{O}_{Train}$). Finally, they again do not use any defense-related information in this variant.

6.1.8 F-Attack

As specified in summary 4.1.8, this paper [38] proposes a query-access decision-based (Labels $\mathcal{O}_L$) grounded attack with both targeted and untargeted variants. For their attack to function, they need access to additional data from the same distribution as the training data ($\mathcal{O}_{D'}$) to construct their reference set. They do not train nor use any training information ($\emptyset$) and likewise, they do not use any defense-related information ($\emptyset$).

6.1.9 BIA Attack

As mentioned in the summary (summary 4.1.9), this paper [11] proposes three variants of its attack. While the attack algorithm remains the same across all three variants, they evaluate it in three different scenarios where the information available to the attacker varies greatly. Therefore, we split it into three different variants:

- Variant A: BIA (A) is the original attack that attacks models trained on another dataset than the one they have access to ($\mathcal{O}_{\mathcal{E}}$). They also use a set of possible architectures to attack the other models ($\mathcal{O}_{SPA}$).
- Variant B: BIA (B) is the version that transfers between models trained on ImageNet. They use the same training data as the attacked model ($\mathcal{O}_{\mathcal{D}}$). Otherwise, it uses the same information as BIA (A).
- Variant C: BIA (C) is the whitebox variant where they train their generator directly on the target model. Therefore, they use the model parameters ($\mathcal{O}_M$) while keeping everything else the same.

Variants A and B use the training function ($\mathcal{O}_{Train}$) since all the pre-trained models come from the same source (Pytorch [108] and their GitHub repository). Additionally, all variants do not use any defense information ($\emptyset$).

6.1.10 ACG Attack

Their paper [83] (summary 4.1.10) presents a whitebox attack that uses the model parameters ($\mathcal{O}_M$) and gradients with the conjugate gradient method to generate adversarial examples. To compute the gradients, they require a function to compute them over. The authors do not specify what function they use and since the most common method (and weakest assumption) is to use the training loss function, we have to assume that they could have used it ($\mathcal{O}_{T_1}$ same as the one from conversion 6.1.6). They do not use any data ($\emptyset$) or defense ($\emptyset$) information.

6.1.11 Admix Attack

Admix [92] (summary 4.1.11) indirectly presents three underlying variants of its attack (when looking at information categories) in its evaluation section. The first one is the

one originally presented in the paper and the others are the additional variants that were introduced in the evaluation. However, all variants share the following common requirements for information: All source and target models used are trained on ImageNet [101], therefore they use the same training data ($\mathcal{O}_{\mathcal{D}}$). Additionally, they also use data from the same distribution in their attack (as mentioned in the summary 4.1.11). Their attack uses the model's loss function J as shown in equation 4.24 on top of pre-trained models, hence the information used is that of the training function ($\mathcal{O}_{Train}$). Finally, they use no defense-related information ($\emptyset$).

- Variant A: Admix (A) is the original attack. In this variant, they have only access to a set of possible model architectures through pre-trained models ($\mathcal{O}_{SPA}$).
- Variant B: Admix (B) is the case where they attack a defended version of the source model. While the model parameters are different, it is akin to training a surrogate undefended model with the same architecture ($\mathcal{O}_{A}$) using the same training data.
- Variant C: Admix (C) is the case where the model they attack is the source model. They assume access to the model parameters ($\mathcal{O}_{M}$) and treat this case as a white-box attack.

6.1.12 ATA Attack

Like many of the other papers we've summarized, this paper uses pre-trained source models to attack other models trained on the same dataset. Having access to these pre-trained models is therefore equivalent to having access to the training data ($\mathcal{O}_{\mathcal{D}}$) and the training function ($\mathcal{O}_{Train}$). They do not consider defense-related information when attacking the models ($\emptyset$).

In their evaluation, they consider two scenarios, a transferable scenario (variant A) where they do not have access to the models parameters nor the exact architecture ($\mathcal{O}_{SPA}$) and a white-box setting where they have direct access to the model parameters (variant B, $\mathcal{O}_{M}$).

6.1.13 Shadow Attack

As mentioned in the summary (4.1.13), this is a score query-based adversarial attack ($\mathcal{O}_{S}$). Additionally, the only other information they use is an additional dataset from another

distribution (SBU Shadow dataset [159], $\mathcal{O}_{\mathcal{E}}$) to tune the shadow parameter k of their attack. They do not use any training nor defense-related information even though they also attack adversarially-trained versions of their models.

6.1.14 DAPatch Attack

The conversion for this paper [84] is very straightforward. As mentioned in the summary (4.1.14), this attack is a white-box attack that uses nothing more than the model parameters.

6.1.15 S^{2}I Attack

This is a transferable attack [64] (summary 4.1.15) that uses pre-trained surrogate models trained on ImageNet [101] and the model loss function to optimize the adversarial examples. Similarly to other transferable attacks using surrogate models all pre-trained on the same dataset, the authors evaluate two variants for their attack. One is the original intended variant (A) with the surrogate pre-trained models requiring: training function $\mathcal{O}_{Train}$, training data $\mathcal{O}_{\mathcal{D}}$, and set of possible architectures $\mathcal{O}_{S}PA$ in order to train the surrogates. The other variant (B), is the white-box scenario when the surrogate model is the same as the attacked model, and they assume having access to the model parameters (rather than training second different model with the same architecture). This requires the model parameters $\mathcal{O}_M$ and for this particular attack at least the loss function used for training as their attack requires it ($\mathcal{O}_{\mathcal{T}\infty}$). Neither of the variants takes into consideration the defense information.

6.1.16 AI-GAN Attack

This attack [93] (summary 4.1.16) assumes white-box access to the target model parameters ($\mathcal{O}_M$) and needs the target model's training data to train the generator ($\mathcal{O}_{\mathcal{D}}$). It, however, needs no training or defense-related information.

6.1.17 AEG Attack

As stated in the paper [91] (summary 4.1.17), they present two variants of their attack, both transferable. Variant A is the variant that assumes knowledge of the architecture

of the target model ($\mathcal{O}_A$) as well as data from the same distribution as the training data ($\mathcal{O}_{\mathcal{D}'}$).
For both variants, they describe the way they train their surrogate models, which conveys that they have access to the loss function and some information about the optimizer. Additionally, they do not use defense-related information ($\emptyset$).
Variant B differs from variant A in that it uses a set of possible architectures ($\mathcal{O}_{SPA}$) and the actual training data ($\mathcal{O}_{\mathcal{D}}$).

6.1.18 ACA Attack

Similarly to many of the other transferable attacks we've surveyed, this attack uses standard transferable adversarial attack information to mount their attack (they use pre-trained surrogates). In their evaluation, they also consider the case where the surrogate is the same as the target model in which case they assume the standard white-box scenario. This yields two variants:

- Variant A: the transferable case, uses a set of possible architecture ($\mathcal{O}_{SPA}$), the training data ($\mathcal{O}_{\mathcal{D}}$), and the training function.
- Variant B: the white-box case, uses the model parameters ($\mathcal{O}_M$) and the loss function ($\mathcal{O}_{T_1}$) as it is required for their optimization.

Both variants also indirectly use other additional data ($\mathcal{O}_{\mathcal{E}}$) as they require a pre-trained Stable Diffusion model [178] (they use version 1.4). They also do not use any defense-related information ($\emptyset$).

6.1.19 DiffAttack

Similarly to the other Stable Diffusion-based attack [86] (summary 4.1.18), this attack [85] (summary 4.1.19) has two variants. Their transferable attack uses pre-trained surrogates and in their evaluation, they consider the case where the surrogate model is the same as the target model (white-box).

- Variant A: the transferable case, they use a set of possible architectures ($\mathcal{O}_{SPA}$), the training data ($\mathcal{O}_{\mathcal{D}}$) and the training function (for training the surrogates).
- Variant B: the white-box case, they use the model parameters ($\mathcal{O}_M$) and the loss function ($\mathcal{O}_{T_1}$) as they require it for their optimization (L_{attack}).

Additionally, since both variants assume access to a pre-trained Stable Diffusion model [178], they indirectly require additional other data ($\mathcal{O}_{\mathcal{E}}$). They also do not use any defense-related information ($\emptyset$).

6.1.20 A^3 Attack

In their paper [39] (summary 4.1.20), the authors follow a slightly different game definition compared to the base game we propose in our formalization. Instead of being given the images to attack one at a time, they assume access to the entire set of test images to attack at once. This is a valid assumption to make (as we can just redefine $\mathcal{O}_X$ to do just that). However, since the other papers do not do so, w.r.t. them, in the original base game, it would be equivalent to having access to data from the same distribution ($\mathcal{O}_{\mathcal{D}'}$). While the attack is an adaptive white-box attack, it requires access to the model parameters $\mathcal{O}_M$ but it does not use any information about the defense algorithm used. Finally, it uses the model's loss function to evaluate the hardness-to-attack of a given input image in its OSD step, therefore it requires at least knowledge of the model's loss function ($\mathcal{O}_{\mathcal{T}_\infty}$).

6.2 Compiled tables

We also include tables compiling the datasets and target model used in the evaluations of each of the attacks in their original papers. This can also serve as a general guideline future work on the datasets and target models to use to improve the comparability of their work to the rest of the work in the field.

Table 6.1: Attack Information Table part 1

Attack	**Model**	**Data**	**Train**	**Defense**
ODI [45]	Possible Architectures	Training Data	Training Function	∅
GA [40]	Possible Architectures	Training Data	Training Function	∅
LGV (A) [54]	Possible Architectures	Training Data	Training Function	∅
LGV (B) [54]	Parameters	∅	∅	∅
Pixle [37]	Scores	∅	∅	∅
MASSA [66]	Labels	∅	∅	∅
AI-FGTM (A)[15]	Possible Architectures	Training Data	Training Function	∅
SSAH (A) [65]	Parameters	Same Distribution	∅	∅
SSAH (B) [65]	Possible Architectures	Other Data	Training Function	∅
BIA (A) [11]	Possible Architectures	Other Data	Training Function	∅
BIA (B) [11]	Possible Architectures	Training Data	Training Function	∅
BIA (C) [11]	Parameters	Training Data	∅	∅
F-Attack [38]	Labels	Same Distribution	∅	∅
ACG [83]	Parameters	∅	Loss Function	∅

Table 6.2: Attack Information Table part 2

Attack	Model	Data	Train	Defense
Admix (A) [92]	Possible Architectures	Training Data & Same Distribution	Training Function	∅
Admix (B) [92]	Architecture	Training Data & Same Distribution	Training Function	∅
Admix (C) [92]	Parameters	Same Distribution	Loss Function	∅
ATA (A) [82]	Possible Architectures	Training Data	Training Function	∅
ATA (B) [82]	Parameters	∅	∅	∅
Shadow [14]	Scores	Other Data	∅	∅
DAPatch [84]	Parameters	∅	∅	∅
S^2I (A) [64]	Possible Architectures	Training Data	Training Function	∅
S^2I (B) [64]	Parameters	∅	Loss Function	∅
AI-GAN [93]	Parameters	Training Data	∅	∅
AEG (A) [91]	Architecture	Same Distribution	Loss Function & Optimizer	∅
AEG (B) [91]	Possible Architectures	Training Data	Loss Function & Optimizer	∅
ACA (A) [86]	Possible Architectures	Training Data & Other Data	Training Function	∅
ACA (B) [86]	Parameters	Other Data	Loss Function	∅
DiffAttack (A) [85]	Possible Architectures	Training Data & Other Data	Training Function	∅
DiffAttack (B) [85]	Parameters	Other Data	Loss Function	∅
A^3 [39]	Parameters	Same Distribution	Loss Function	∅

Table 6.3: Attack properties part 1

Attack	Targeted	Grounded	Salient Situation	Metric	Parameter
ODI [45]	✓	✓	Indistinguishable	l_∞	$\epsilon = 16/256$
GA [40]	✗	✓	Content-preserving	l_∞	adaptive
LGV [54]	✗	✓	Indistinguishable	l_∞	not specified
Pixle [37]	Both	✓	Indistinguishable	l_0	variable
MASSA [66]	✗	✓	Indistinguishable	l_2	variable
AI-FGTM [15]	✗	✓	Indistinguishable	l_∞	$\epsilon = 16/256$
SSAH [65]	Both	✓	Indistinguishable	l_∞	$\epsilon = 8/256$
BIA [11]	✗	✓	Indistinguishable	l_∞	$\epsilon = 10/256$
F-Attack [38]	Both	✓	Content-preserving	None	None
ACG [83]	✗	✓	Indistinguishable	l_∞	CIFAR $\epsilon = 8/256$ ImageNet $\epsilon = 4/256$
Admix [92]	Both	✓	Any (Indistinguishable)	None	None
ATA [82]	✗	✓	Indistinguishable	l_0-norm	$\epsilon = 1024/(224 \times 224)$
Shadow [14]	✗	✓	Content-preserving	None	None
DAPatch [84]	✗	✓	Content-preserving	l_0-norm	$0.5\% - 3\%$ (pixels)

Table 6.4: Attack properties part 2

Attack	**Targeted**	**Grounded**	**Salient Situation**	**Metric**	**Parameter**
S^2I [64]	Both	✓	Any (Indistinguishable)	l_∞	$\epsilon = 16/256$
AI-GAN [93]	✓	✓	Indistinguishable	l_∞-norm	CIFAR $\epsilon = 8/256$ MNIST $\epsilon = 0.3$
AEG [91]	✗	✓	Indistinguishable	l_∞-norm	CIFAR $\epsilon = 8/256$ MNIST $\epsilon = 0.3$
ACA [86]	✗	✓	Content-preserving	None	None
DiffAttack [85]	✗	✓	Content-preserving	FID [133]	variable
A^3 [39]	✗	✓	Indistinguishable	l_∞-norm	CIFAR $\epsilon = 8/255$ ImageNet $\epsilon = 4/256$

Table 6.5: List of Datasets used to evaluate attacks part 1

Attack	Datasets
ODI [45]	ImageNet-Compatible dataset (1000 images from the NIPS 2017 adversarial competition [118])
GA [40]	ILSVRC 2012 ImageNet-Compatible [144], [116] (1000 images from validation set)
LGV [54]	ImageNet [101] (2000 images from validation set)
Pixle [37]	CIFAR10 [112], ImageNet [101], TinyImageNet [113] (1000 images from validation set each)
MASSA [66]	ImageNet [101] (500 images from validation set)
AI-FGTM [15]	ImageNet-Compatible dataset (1000 images from the NIPS 2017 adversarial competition [118])
SSAH [65]	CIFAR10 [112], CIFAR100 [112], ImageNet [101]
BIA [11]	CIFAR10 [112], CIFAR100 [112], ImageNet [101], STL-10 [135], SVHN [136], CUB-200-2011 [137], Stanford Cars [138], FGVC Aircraft [139] (whole validation sets)
F-Attack [38]	CIFAR10 [112], ImageNet [101] (500 images from validation set)
ACG [83]	CIFAR10 [112], CIFAR100 [112] (entire validation set), ImageNet [101] (5000 images from validation set)
Admix [92]	ILSVRC 2012 ImageNet-Compatible [144], [116] (1000 images from validation set)

Table 6.6: List of Datasets used to evaluate attacks part 2

Attack	**Datasets**
ATA [82]	ImageNet [101] (2000 images from validation set)
Shadow [14]	LISA [156], GTSRB [157] (entire validation set)
DAPatch [84]	GTSRB [157] (500 images from validation set) ILSVRC 2012 ImageNet-Compatible [144] (1000 images from validation set)
S^2I [64]	ImageNet-Compatible dataset (1000 images from the NIPS 2017 adversarial competition [118])
AI-GAN [93]	CIFAR10 [112], CIFAR100 [112], MNIST [170]
AEG [91]	MNIST [170], CIFAR10 [112]
ACA [86]	ImageNet-Compatible dataset (1000 images from the NIPS 2017 adversarial competition [118])
DiffAttack [85]	ImageNet-Compatible dataset (1000 images from the NIPS 2017 adversarial competition [118])
A^3 [39]	CIFAR10 [112], CIFAR100[112] (entire validation set), ImageNet [101] (1000 images from validation set, CVPR 2021 competition)

Table 6.7: List of attacked models by more than two papers

Model	Attack
ResNet-50 [95]	ODI [45], LGV [54], Pixle [37], MASSA [66], AI-FGTM [15], SSAH [65], BIA [11], F-Attack [38], ACG [83], S^2I [64], A^3 [39] Admix [92], DiffAttack [85], ATA [82], DAPatch [84], ACA [86]
Inception-v3 [96]	ODI [45], GA [40], LGV [54], AI-FGTM [15], BIA [11], S^2I [64], Admix [92], AEG [91], ACA [86], DiffAttack [85]
Inception-ResNet-v2 [99]	ODI [45], GA [40], AI-FGTM [15], Admix [92], S^2I [64], ACA [86], DiffAttack [85]
VGG-16 [98]	Pixle [37], MASSA [66], BIA [11], F-Attack [38], ATA [82], AEG [91]
VGG-19 [98]	LGV [54], MASSA [66], BIA [11], DAPatch [84], DiffAttack [85]
DenseNet-121 [97]	ODI [45], BIA [11], ATA [82], AEG [91]
ResNet-152 [95]	GA [40], BIA [11], DAPatch [84], S^2I [64], ACA [86]
MobileNet-v2 [100]	ODI [45], F-Attack [38], DAPatch [84], ACA [86], DiffAttack [85]
ResNet-18 [95]	ODI [45], ACG [83], AEG [91], A^3 [39]
WideResNet-34-10 [110]	SSAH [65], ACG [83], AI-GAN [93], AEG [91], A^3 [39]
Inception-v4 [99]	ODI [45], Admix [92], S^2I [64]
ViT-B [151]	ATA [82], DAPatch [84], ACA [86], DiffAttack [85]
Swin-B [160]	DAPatch [84], ACA [86], DiffAttack [85]

Table 6.8: List of attacked models by two papers

Model	**Attack**
ResNet-20 [95]	Pixle [37], SSAH [65]
MobileNet [123]	AI-FGTM [15], Admix [92]
DenseNet-161 [97]	DAPatch [84], ACA [86]
EfficientNet-b7 [161]	DAPatch [84], ACA [86]
ResNeXt-101 [109]	GA [40], DAPatch [84]
DeiT-B [152]	ATA [82], DiffAttack [85]
DeiT-S [152]	ATA [82], DiffAttack [85]
ResNet-101 [95]	Admix [92], S^2I [64]
ResNet-32 [95]	F-Attack [38], AI-GAN [93]
A [8]	AI-GAN [93], AEG [91]
AlexNet [150]	ATA [82], AEG [91]
PreActResNet-18 [119]	ACG [83], A^3 [39]
WideResNet-28-4 [110]	ACG [83], A^3 [39]
WideResNet-28-10 [110]	ACG [83], A^3 [39]
WideResNet-34-15 [110]	ACG [83], A^3 [39]
WideResNet-34-20 [110]	ACG [83], A^3 [39]
WideResNet-70-16 [110]	ACG [83], A^3 [39]

Table 6.9: List of attacked defended models by more than one paper

Model	Attack
Inception-ResNet-v2$_{\text{ens}}$ [8]	ODI [45], GA [40], AI-FGTM [15], Admix [92], S^{2}I [64], ACA [86], DiffAttack [85]
Inception-v3$_{\text{Ens3}}$ [8]	AI-FGTM [15], Admix [92], S^{2}I [64], AEG [91], ACA [86], DiffAttack [85]
Inception-v3$_{\text{Ens4}}$ [8]	AI-FGTM [15], Admix [92], S^{2}I [64], ACA [86], DiffAttack [85]
HGD [120]	AI-FGTM [15], Admix [92], S^{2}I [64], ACA [86], DiffAttack [85]
R&P [121]	AI-FGTM [15], Admix [92], S^{2}I [64], ACA [86], DiffAttack [85]
NIPS-r3 [118]	AI-FGTM [15], Admix [92], S^{2}I [64], ACA [86], DiffAttack [85]
ResNet-50$_{\text{RS}}$ [9], [95]	AI-FGTM [15], Admix [92], S^{2}I [64], DiffAttack [85]
MobileNet$_{\text{FD}}$ [122], [124]	AI-FGTM [15], Admix [92]
ResNeXt-101$_{\text{Denoise}}$ [165]	GA [40], DAPatch [84]
ResNet-152$_{\text{adv}}$ [165]	GA [40], DAPatch [84]
ResNet-152$_{\text{Denoise}}$ [165]	GA [40], DAPatch [84]
Fast$_{AT}$ [164]	GA [40], DAPatch [84]
A$_{\text{Ens4}}$ [8]	AEG [91], AI-GAN [93]
Bit-Red [145]	Admix [92], ACA [86]
JPEG [146]	Admix [92], S^{2}I [64], ACA [86]
DiffPure [90]	ACA [86], DiffAttack [85]
WideResNet-34-10$_{\text{TRADES}}$ [129]	SSAH [65], ACG [83]
NRP [148]	Admix [92], S^{2}I [64]
ResNet-50$_{\text{Debiased}}$ [163]	DAPatch [84], ACA [86]

Table 6.10: List of attacked models by only a single paper part 1

Model	Attack
ResNeXt-50 [109]	LGV [54]
WideResNet-34-R [110]	ACG [83]
WideResNet-50 [110]	LGV [54]
WideResNet-106-16 [110]	ACG [83]
Inception-v1 [111]	LGV [54]
DenseNet-169 [97]	BIA [11]
DenseNet-201 [97]	LGV [54]
VGG-11 [98]	Pixle [37]
VGG-16_bn [98]	ODI [45]
ViT-T [151]	ATA [82]
ViT-S [151]	ATA [82]
DeiT-T [152]	ATA [82]
Mixer-B [181]	DiffAttack [85]
Mixer-L [181]	DiffAttack [85]
ConViT (T, S, & B) [153]	ATA [82]
Swin-L [160]	GA [40]
MobileViT [175]	ACA [86]
PVT-v2 [176]	ACA [86]

Table 6.11: List of attacked models by only a single paper part 2

Model	Attack
SENet-154 [141]	BIA [11]
SE-ResNet-101 [141]	BIA [11]
LISA-CNN [158]	Shadow [14]
GTSRB-CNN [158]	Shadow [14]
B [8]	AEG [91]
C [8]	AEG [91]
D [8]	AEG [91]
Microsoft Azure [130]	SSAH [65]
Tencent Cloud [131]	SSAH [65]
Baidu AI Cloud [132]	SSAH [65]
RVT-Tiny [225]	GA [40]
DeepAugment_AugMix [226]	GA [40]
EfficientNet-l2-ns [227] [161]	GA [40]

Table 6.12: List of attacked defended models by one paper

Model	**Attack**
RobustBench models [94]	ACG [83] A^3 [39]
ResNet-$50_{\text{Comdefend}}$ [125], [95]	AI-FGTM [15]
WideResNet-34-10_{FSAT} [128]	SSAH [65]
Inception-ResNet-$v2_{\text{adv}}$ [8]	ODI [45]
Inception-$v3_{\text{adv}}$ [106]	DiffAttack [85]
Free_{AT} [228]	GA [40]
Shape-ResNet [177]	ACA [86]
ARS [147]	Admix [92]
LISA-CNN_{rob} [14]	Shadow [14]
GTSRB-CNN_{rob} [14]	Shadow [14]
ResNet-50_{SIN} [162]	DAPatch [84]
ResNet-$50_{\text{SIN+IN}}$ [162]	DAPatch [84]
ResNet-$50_{\text{SIN+IN-IN}}$ [162]	DAPatch [84]
ResNet-152_{Debiased} [163]	DAPatch [84]
ResNet-18_{Ens3} [8]	AEG [91]
WideResNet-34-10_{Ens3} [8]	AEG [91]
DenseNet-121_{Ens3} [8]	AEG [91]
VGG-16_{Ens3} [8]	AEG [91]
Madry-$\text{CIFAR10}_{\text{adv}}$ [7]	AEG [91]
Madry-$\text{MNIST}_{\text{adv}}$ [7]	AEG [91]
B_{Ens4} [8]	AEG [91]
C_{Ens4} [8]	AEG [91]
D_{Ens4} [8]	AEG [91]

Chapter 7

Results

In this chapter, we will use the information compiled in Chapter 6 to perform a comparative analysis of the attacks we surveyed with respect to the information they use to mount their attack. To get meaningful comparisons, we will first separate the attacks by the salient situation they adopt, whether they are targeted or untargeted, and then the datasets they attack. We then prune all the situation/dataset pairs that have only one attack or less so we can compare within each pair. This yields Table 7.1.

Table 7.1: Salient situation-dataset pairs with more than one attack

Salient Situation	**Targeted**	**Datasets**
Content-Preserving	✗	NIPS 2017 [118], ILSVRC 2012 [144], GTSRB [157]
Indistinguishable	✗	NIPS 2017 [118], CIFAR10, CIFAR100 [112], ImageNet [101]

We only include untargeted content-preserving and indistinguishable attacks. We, unfortunately, do not have enough targeted content-preserving attacks and targeted indistinguishable attacks that share evaluation datasets to allow for comparative analysis for each. The only possible comparative analysis we could have conducted would have been on the CIFAR10/100 [112] datasets between the SSAH [65] and AI-GAN [93] attacks. However, those two papers do not have any target model architectures in common and AI-GAN does not provide benign accuracies for its models. This means we would not have been able to provide any kind of meaningful attack success rate or relative performance score comparison.

For each dataset and salient situation pair, we provide a visual tree-based representation of the information oracles used by each attack and how they differ from one another. We also color-code the different kinds of nodes as follows:

- Green: Dataset.
- Blue: Model Information.
- Yellow: Data Information.
- Pink: Training Information.
- Grey: Defense Information.
- Orange (triangle-shaped): leaf node that represents the attack.

We also showcase the comparisons that we will make by using colored rectangular borders around the attacks that will be compared. When comparing, 'score' will stand for the relative performance score (Definition 28). Additionally, if result numbers are presented in italics, it means that they were computed using an additional assumption (commonly, it is when the benign accuracy of the model is taken from another paper and therefore is not guaranteed to be that value).

7.1 Content-Preserving untargeted attacks

7.1.1 Attacks on the ILSVRC2012 Dataset

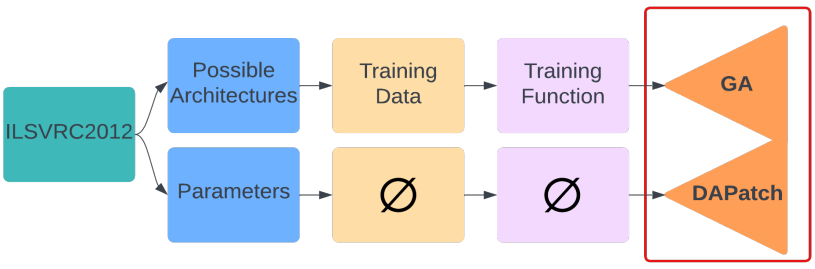

Figure 7.1: ILSVRC 2012 dataset content-preserving attacks

For the ILSVRC 2012 dataset [144], we compare the GA attack [40] (summary 4.1.2) with the DAPatch attack [84] (summary 4.1.14). DAPatch follows the traditional white-box threat model with access to the model parameters while GA follows a transferable threat model where they can train surrogates using possible architectures, the original training data and the training function. In their evaluation, they attack three of the same defended models. While only GA provides clean accuracies for these defended target models, we assume that DAPatch trained their target model in good faith, or they used the trained models provided by the original defense papers. If we do not make this assumption we cannot establish any comparative results. We provide in Table 7.2 the overview of the results. ASR stands for attack success rate (the ratio of misclassified samples to correctly classified samples).

Table 7.2: ILSVRC2012 dataset content-preserving results

Model	**Benign ASR**	**GA ASR**	**DAPatch ASR**	**GA Score**	**DAPatch Score**
Fast_{AT} [164]	33.1 (GA)	**71.5**	*51.3*	**0.402**	*0.154*
$\text{ResNeXt-101}_{\text{Denoise}}$ [165]	19.7 (GA)	52.2	***52.9***	0.234	***0.241***
$\text{ResNet-151}_{\text{Denoise}}$ [165]	27.8 (GA)	59.7	***62.3***	0.279	***0.311***
Average	26.9	**61.1**	*55.5*	**0.305**	*0.235*

Surprisingly, the transferable attack (GA) seems to perform slightly better on average. This, however, can be explained by the inherent nature of the DAPatch attack. It is a patch attack, therefore they heavily restrict the kind of attacks they can perform compared to the GA attack. We provide a summary of the results within the tree structure in Figure 7.2.

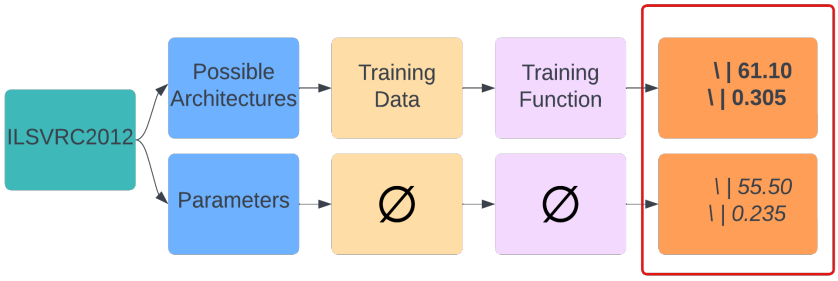

Figure 7.2: ILSVRC2012 dataset content-preserving results. The first row of numbers is the average ASR while the second is the average relative performance score. The first column of numbers is the performance on undefended models whereas the second is the performance on defended models (we write missing results as \). We bold the best performer for each row.

7.1.2 Attacks on the GTSRB dataset

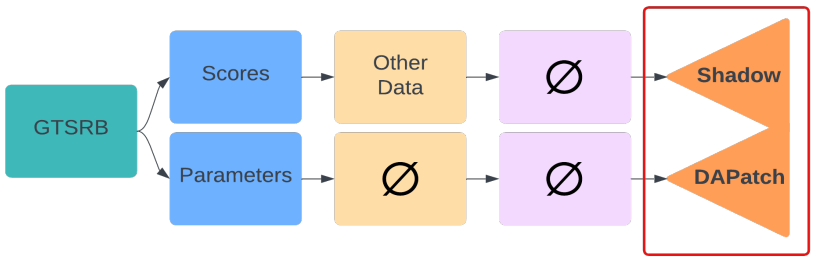

Figure 7.3: GTSRB dataset content-preserving attacks

For the GTSRB dataset [157], we compare the Shadow attack [14] (summary 4.1.13) with the DAPatch attack [84] ((summary 4.1.14)). The Shadow attack employs a score query-based threat model while GA uses a transferable threat model. Comparing these attacks in a fair setting is difficult as Shadow attacks custom models with custom architectures whereas DAPatch attacks existing architectures. Both papers attack undefended and defended models. We first look at the results for the undefended models in Table 7.3.

When looking at the averages, we can see that while DAPatch performs better, they both perform very well and render the models they attack practically unusable. Again, it is difficult to identify whether one is better as they are compared against different

Table 7.3: GTSRB dataset content-preserving undefended results

Model	**Benign ASR**	**Shadow ASR**	**DAPatch ASR**	**Shadow Score**	**DAPatch Score**
GTSRB-CNN [14]	1.0 (Shadow)	**90.47**	N/A	**0.818**	N/A
ResNet-152 [95]	0. (DAPatch)	N/A	**93.1**	N/A	**0.867**
ViT-B [151]	0. (DAPatch)	N/A	**95.0**	N/A	**0.903**
Average	1. (Shadow) 0. (DAPatch)	90.47	**94.05**	0.818	**0.885**

architectures. We refine our comparison by looking at their performance on defended models. On one hand, in their paper, Shadow develops an adaptive defense against their attack GTSRB-CNN$_{\text{rob}}$. On the other hand, DAPatch attacks models defended by existing anti-patch attack defenses.

Both attacks observe a loss in performance when attacking defended models, as expected. It is difficult to say if one is better than the other as we only have a singular result for Shadow, and it's from their adaptive defense. However, if we take the results as they are, Shadow does outperform DAPatch slightly. We provide a summary of the results within the tree structure in Figure 7.4.

Table 7.4: GTSRB dataset content-preserving defended results

Model	Benign ASR	Shadow ASR	DAPatch ASR	Shadow Score	DAPatch Score
GTSRB-CNN$_{\text{rob}}$ [14]	1.09 (Shadow)	**74.43**	N/A	**0.554**	N/A
ResNet-152$_{\text{LGS}}$ [95] [229]	3.7 (DAPatch)	N/A	**53.9**	N/A	**0.289**
ResNet-152$_{\text{DW}}$ [95] [230]	10.2 (DAPatch)	N/A	**69.5**	N/A	**0.473**
ViT-B$_{\text{LGS}}$ [151] [229]	3.0 (DAPatch)	N/A	**58.1**	N/A	**0.337**
ViT-B$_{\text{DW}}$ [151] [230]	10.3 (DAPatch)	N/A	**69.8**	N/A	**0.477**
Average	1.09 (Shadow) 6.8 (DAPatch)	**74.43**	62.83	**0.554**	0.394

7.1.3 Attacks on the NIPS 2017 dataset

For the NIPS 2017 dataset [118], we compare the ACA attack [86] (summary 4.1.18) and DiffAttack [85] (summary 4.1.19). They are both attacks that use the latent space of diffusion models to craft their attack. As for threat models, they both evaluate the same variants of the white-box and transferable threat models. We first compare them within each of the threat models and then compare them across threat models.

We first look at the transferable case. They both assume access to possible architectures, the original training data, additional data samples (from another distribution), and the training function. They evaluate their performance against both undefended and defended models. We first examine the undefended results, where both papers provide model performance on benign samples. The results are presented in Table 7.5.

For most of the undefended models, ACA and DiffAttack have similar benign accuracies except for Inception-v3 where we can see significant discrepancy. In general, DiffAttack seems to significantly outperform ACA in the transferable undefended scenario with an average ASR that is 18.9% higher and an average relative performance score that is 0.25 higher. We examine the defended model results to see if this trend is consolidated, the

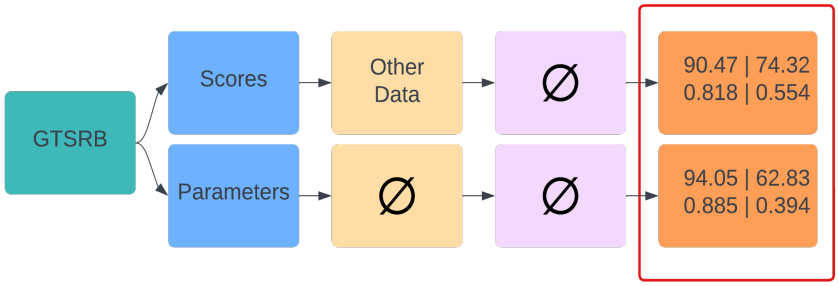

Figure 7.4: GTSRB dataset content-preserving results. The first row of numbers is the average ASR while the second is the average relative performance score. The first column of numbers is the performance on undefended models whereas the second is the performance on defended models (we write missing results as \). We bold the best performer for each row.

results are presented in Table 7.6. For defended models, DiffAttack does not provide clean accuracies therefore, similar to what we did with subsection 7.1.1, we use the clean accuracies from ACA for the defended models under the assumption that DiffAttack trained their defended models.

It appears that on defended models, unlike undefended models, ACA outperforms DiffAttack in the average and for every defended model except DiffPure.

We can compare these attacks in the traditional white-box setting. They only attack undefended models in the white-box setting. The results are presented in Table 7.7.

DiffAttack is also better than ACA against undefended models in the white-box setting, although not by much and both attacks are extremely potent.

Finally, we perform a comparison between the best results in the transferable threat model compared to the white-box threat model. We look at both the ASR differences (Table 7.8) and the relative performance score (Table 7.9).

We notice an average increase of **12.5%** ASR, or **0.21** score when going from non-white-box to white-box. While this is a significant increase, it shows that the attacks in the transferable setting can efficiently make use of the model, data and training information at hand to deliver potent attacks. We provide a summary of the results within the tree structure in Figure 7.6.

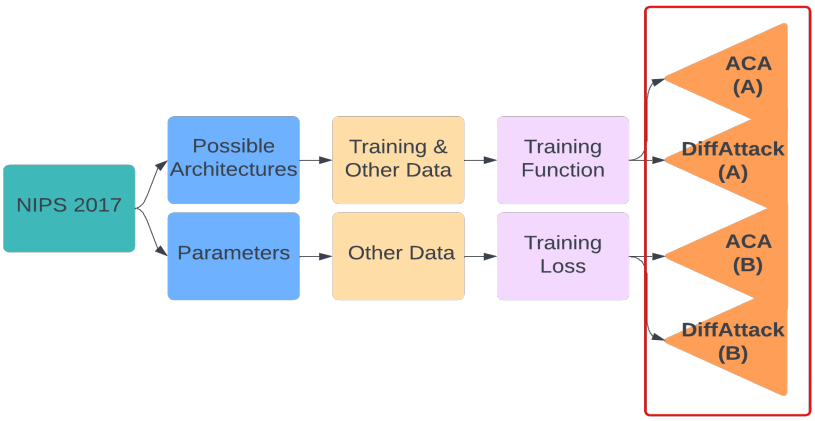

Figure 7.5: NIPS 2017 dataset content-preserving attacks

7.2 Indistinguishable untargeted attacks

7.2.1 Attacks on the NIPS 2017 dataset

For the NIPS 2017 dataset [118], we compare the AI-FGTM attack [15] (summary 4.1.6) with both variants from the S^2I attack [64] (summary 4.1.15). AI-FGTM and variant S^2I (A) are transferable attacks both using possible architectures, the original training data and the training function to mount their attack. While S^2I (B) uses the model parameters and the training loss in a white-box setting to mount their attack. Both attacks use the l_∞-norm to bind their adversarial perturbations. AI-FGTM uses an epsilon of $\epsilon = 16/256$ whereas S^2I estimates an empirical upper bound on their epsilon to be $\epsilon = 8/256$. Therefore, AI-FGTM has a significant advantage, however, we still perform a comparison as we are able to extract meaningful results.

We first look at the results in the transferable case by comparing AI-FGTM to S^2I (A). Then, we compare the best-performing transferable results to the best-performing white-box results (S^2I (B)) to derive a numerical difference between both threat models. Both papers' evaluations overlap only against defended models, so we only investigate defended models. Neither of the papers provides benign accuracies but since they cite the same defense papers and those defense papers provide their trained models we assume they attack the same models. Therefore, we also use the benign accuracies from subsection 7.1.3 as they also attack the same defended models. The results are compiled in Table 7.10.

So despite AI-FGTM's epsilon advantage, it still underperforms compared to S^2I (A)

Table 7.5: NIPS 2017 dataset content-preserving transferable undefended results

Model	Benign ASR	ACA (A) ASR	DiffAttack (A) ASR	ACA (A) Score	DiffAttack (A) Score
MobileNet-v2[100]	12.1 (ACA) 13.1 (DiffAttack)	69.3	**80.6**	0.466	**0.632**
Inception-v3[96]	4.8 (ACA) 19.5 (DiffAttack)	61.6	**74.2**	0.377	**0.513**
ResNet-50 [95]	7.0 (ACA) 7.3 (DiffAttack)	62.6	**79.1**	0.387	**0.620**
ViT-B [151]	8.9 (ACA) 6.3 (DiffAttack)	52.9	**73.3**	0.272	**0.533**
Swin-B [160]	3.5 (ACA) 4.1 (DiffAttack)	55.5	**88.6**	0.307	**0.783**
Average	7.26 (ACA) 10.06 (DiffAttack)	60.4	**79.2**	0.362	**0.616**

against all the models evaluated with an average performance discrepancy of **9.5%** ASR (**0.127** score). However, both attacks are quite potent and practically render the attacked models unusable.

We now compare the best performances of the non-white-box and white-box settings (S^2I (A) and S^2I (B)). Unfortunately, this is only possible on undefended models as this is the only models that have an overlap in their paper. The results are compiled in Table 7.11.

Unsurprisingly, the white-box version of the attack outperforms the transferable version for every undefended model with an average **11.7%** ASR increase. However, the surprising result is the fact that S^2I (A) attacking undefended models significantly underperforms S^2I (A) attacking defended models. This is not something that usually happens in adversarial example research. While they use a slightly different version of their algorithm when performing the white-box evaluation compared to the transferable evaluation, it still is a surprising result. For an attack that has such strong performance against defended models in the transferable setting, this is an empirical oddity. We provide a summary of the results within the tree structure in Figure 7.6.

Table 7.6: NIPS 2017 dataset transferable defended results

Model	Benign ASR	ACA (A) ASR	DiffAttack (A) ASR	ACA (A) Score	DiffAttack (A) Score
Inception-ResNet-v2$_{ens}$ [8]	2.6 (ACA)	**43.6**	*41.7*	**0.189**	*0.173*
Inception-v3$_{ens3}$ [8]	6.8 (ACA)	**59.8**	*56.2*	**0.353**	*0.311*
Inception-v3$_{ens4}$ [8]	8.9 (ACA)	**62.2**	*56.9*	**0.379**	*0.316*
HGD [120]	1.2 (ACA)	**52.2**	*38.0*	**0.272**	*0.144*
R&P [121]	1.8 (ACA)	**53.6**	*34.5*	**0.287**	*0.119*
NIPS-r3 [118]	3.2 (ACA)	**53.9**	*30.0*	**0.289**	*0.089*
DiffPure [90]	15.4 (ACA)	63.7	***72.2***	0.382	***0.498***
Average	5.7 (ACA)	**55.6**	*47.1*	**0.307**	*0.236*

7.2.2 Attacks on the CIFAR10 dataset

The CIFAR10 dataset [112] is a very popular dataset to train and evaluate image classification models and therefore attack them. In our case, we can perform four comparisons of a total of seven attacks or variants of attacks on the CIFAR10 dataset. We hope that we be able some relative performance indicators for the different information categories that appear.

We first start with the comparison we label as green that compares BIA (A) with AEG (B).

Green: BIA (A) & AEG (B) BIA (A) [11] (summary 4.1.9) is a transferable attack we've already encountered in subsection 7.2.3. Similarly to CIFAR100, the authors only attack a custom model, and therefore we again treat their results as unreliable. AEG (B) [91] (summary 4.1.17) on the other hand is also a transferable attack whose data information requirements (training data) dominates ($\sqsubset$) BIA (A)'s data information requirement (other data), but its training information requirement (training loss function and some information about the optimizer) is dominated by BIA (A)'s training requirement (training function).

We first summarize BIA (A)'s best performance against said custom model in Table

Table 7.7: NIPS 2017 dataset content-preserving white-box undefended results

Model	Benign ASR	ACA (B) ASR	DiffAttack (B) ASR	ACA (B) Score	DiffAttack (B) Score
MobileNet-v2[100]	12.1 (ACA) 13.1 (DiffAttack)	93.1	**98.2**	0.852	**0.947**
Inception-v3[96]	19.5 (DiffAttack)	N/A	**86.1**	N/A	**0.703**
ResNet-50 [95]	7.0 (ACA) 7.3 (DiffAttack)	88.3	**96.3**	0.775	**0.922**
ViT-B [151]	8.9 (ACA)	**87.7**	N/A	**0.761**	N/A
Swin-B [160]	4.1 (DiffAttack)	N/A	**90.1**	N/A	**0.810**
Average	9.33 (ACA) 11 (DiffAttack)	89.7	**92.7**	0.761	**0.846**

7.12.

Since BIA (A) attacks a custom model, we have to perform an aggregate analysis of AEG (B)'s performance on both undefended and defended models to be able to compare. We first report the performance on undefended models in Table 7.13.

If we treat BIA (A)'s model as undefended, then AEG (B) significantly outperforms BIA (A). This could either be due to the difference in the information required by both attacks or that BIA (A) does not extract attack performance as well out of the information it uses as AEG (B). In the transferable setting, not having access to the training access (BIA (A)'s case) can be a huge disadvantage compared to having access to it (AEG (B)'s case), this is reflected in the empirical results. To complete this comparison, we look at the performance of AEG (B) against defended models in Table 7.14.

Unsurprisingly, attacking well-defended models in the transferable setting is quite a difficult task. The results in Table 7.14 strongly imply that BIA (A)'s model is indeed undefended as it achieves benign accuracies (1-ASR) much closer to the undefended models of Table 7.13. Additionally, if BIA (A)'s model was indeed strongly defended, it would then imply that it would be as beneficial for an attacker to use data from another distribution with the training function rather than the actual training data and slightly less training information as BIA (A) obtains similar results. While this is not an impossibility, we posit it to be quite unlikely, especially due to the unreliability of BIA (A)'s results. We move on to the second comparison which we label as purple.

Table 7.8: NIPS 2017 dataset content-preserving transferable and white-box ASR comparison

Model	Best (A) ASR	Best (B) ASR	Diff
MobileNet-v2[100]	80.6	98.2	**17.6**
Inception-v3[96]	74.2	86.1	**11.9**
ResNet-50 [95]	79.1	96.3	**17.2**
ViT-B [151]	73.3	87.7	**14.4**
Swin-B [160]	88.6	90.1	**1.5**
Average	79.2	91.7	**12.5**

Purple: A^3 & SSAH (A) This is a comparison we already performed for CIFAR100 (subsection 7.2.3), however this time, we can directly compare them as they both attack a WideResNet-34-10$_{\text{TRADES}}$ [128]. We can confidently confirm that these are likely to be the same models as both papers report the same benign accuracy. We present the results in Table 7.15.

We can see that although, as stated by the authors, A^3 is an attack that focuses both on ASR and runtime, it still significantly outperforms SSAH (A) when attacking a well-defended model. It is doubtful that having access to the training loss function is a strong enough difference between the two attacks to justify the performance gap. It is more likely that SSAH (A) is inefficient at extracting the information it has access to to mount a potent attack against defended models. We move on to the next comparison (blue) where we compare the attacks in purple to ACG as we have done for CIFAR100 in subsection 7.2.3. In our final tree, we also include SSAH (A)'s performance on undefended models (they attack a ResNet-20 [95]).

Blue: ACG & A^3 & SSAH (A) We again compare ACG [83] (summary 4.1.10) and A^3 on defended models. They both attack the WideResNet-34-10$_{\text{TRADES}}$ [128] from the previous comparison, so we also compare how ACG performs against SSAH (A). On CIFAR10, ACG and A^3 attack 11 models in common from RobustBench [94]. The results are summarized in Table 7.16.

ACG outperforms both A^3 and SSAH (A) on WideResNet-34-10$_{\text{TRADES}}$ but otherwise loses to A^3 on the other models and overall. We observe again a similar, small, discrepancy

Table 7.9: NIPS 2017 dataset content-preserving transferable and white-box score comparison

Model	Best (A) Score	Best (B) Score	Diff
MobileNet-v2[100]	0.632	0.947	**0.315**
Inception-v3[96]	0.513	0.703	**0.190**
ResNet-50 [95]	0.620	0.922	**0.302**
ViT-B [151]	0.533	0.761	**0.228**
Swin-B [160]	0.783	0.810	**0.027**
Average	0.616	0.829	**0.213**

between ACG and A^3 in terms of performance which reinforces our previous conclusion that it could either be due to the information used or the inherent algorithmic differences. Finally, we are left with comparing all seven attacks against one another for our final comparison (red).

Red: AEG (A) & Pixle & Blue & Green Unfortunately, due to the amount of papers to compare, we reach the limits of what we can do while keeping the comparisons fair. We go through each of the members of the comparison to see if we have any ground for a comparison. On one hand, the attacks in Blue only attack defended models. On the other hand, for Green, BIA (A) only attacks a custom architecture and AEG (B)'s defended models do not overlap with the defended models in Blue. AEG (A) can be compared with AEG (B), however, due to the nature of their attack where they train their surrogate on a subset of the dataset and train the target models on the other subsets, a direct benign accuracy/ASR comparison would be unfair. Finally, Pixle uses the l_0-norm to bind its attack, unlike the other papers. Getting meaningful results in this case is difficult.

We first look into the straightforward comparison: AEG (A) and AEG (B). They only evaluate AEG (A) against a ResNet-18 [95]. They do not provide benign accuracies for AEG (A) which is problematic due to our previous description of how their attack is evaluated. We therefore cannot compute a relative score to attempt a comparison. We still report the ASR they report in our final tree, however, we italicize the result to indicate that it is unreliable.

Our other option would be to compare AEG (B) with Pixle, as AEG (B) provides benign

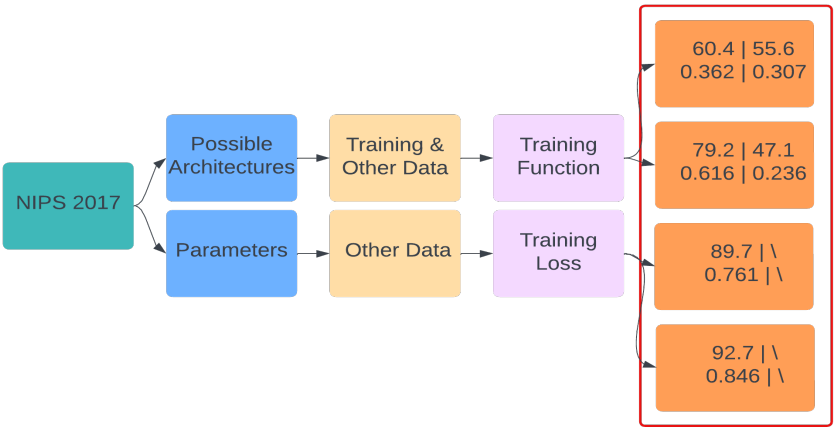

Figure 7.6: NIPS 2017 dataset content-preserving results. The first row of numbers is the average ASR while the second is the average relative performance score. The first column of numbers is the performance on undefended models whereas the second is the performance on defended models (we write missing results as \). We bold the best performer for each row.

accuracies, and they share a target model architecture (ResNet-18). Unfortunately, Pixle does not share its benign accuracy, but we can assume the benign accuracy obtained to AEG (B) would be similar to what Pixle obtained and give them the benefit of the doubt, for the sake of being able to establish some form of comparison. We get the following results in Table 7.17.

We summarize our results in Figure 7.10. While some entries are missing, we can still draw some interesting conclusions. Unsurprisingly, it requires very little model information to get effective attacks if the attacker has access to additional information like data and training information. Additionally, while it could be argued that the models attacked by A^3 and ACG might be of higher defensive quality, we still observe a strong transferable performance against defended models by AEG (B) where they obtain a score (0.167) very similar to white-box attacks (0.163 and 0.160). While this is still much worse than against undefended models, it would still in practice severely decrease the usability of the target model.

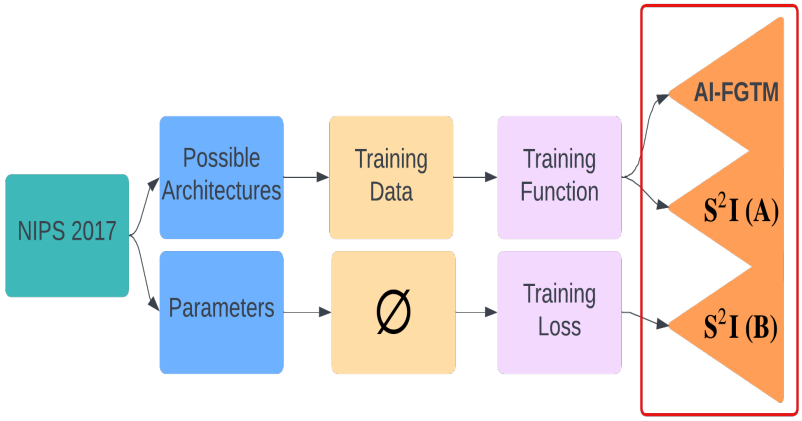

Figure 7.7: NIPS 2017 dataset indistinguishable attacks

7.2.3 Attacks on the CIFAR100 dataset

For the CIFAR100 dataset [112], as can be seen in Figure 7.11, we now have three different comparisons across 4 attacks that we need to perform to complete the tree. We colored them purple, blue and red to reference them more easily. The first comparison, purple: A^3 [39] (summary 4.1.20) and SSAH (A) [65] (summary 4.1.7).

Purple: A^3 & SSHA (A) Both these attacks are attacks that use the model parameters and data drawn from the same distribution as the training data. The only difference is that A^3 uses the training loss whereas SSAH (A) uses no training information. Unfortunately, neither of those attacks have target models in common. A^3 attacks only defended models (12 to be exact), while SSAH (A) attacks two defended models ($\text{WideResNet-34-10}_{\text{TRADES}}$ [129] and $\text{WideResNet-34-10}_{\text{FSAT}}$ [128]). Therefore, the best we can hope to do is to perform an aggregate analysis of the models and see if we can extract any meaningful results. First, we will examine SSAH (A), and we summarize the results in Table 7.18.

SSAH (A) performs well against one defense (FSAT) and very poorly against another (TRADES). This yields a very high standard deviation for both the ASR and the score, meaning that either the attack does not necessarily have a generalizable performance or the FSAT defense severely underperforms TRADES on this dataset and architecture pair. We unfortunately do not have enough data to be able to conclusively decide. Now, we look at the results from the A^3 paper. They are summarized in Table 7.19. To simplify the presentation our the results, we numbered the defended models in the order that they

Table 7.10: NIPS 2017 dataset indistinguishable transferable defended results

Model	Benign ASR	AI-FGTM ASR	S^2I (A) ASR	AI-FGTM Score	S^2I (A) Score
Inception-v3$_{ens3}$ [8]	6.8	91.8	**96.7**	0.838	**0.930**
Inception-v3$_{ens4}$ [8]	8.9	90.3	**96.7**	0.807	**0.927**
Inception-ResNet-v2$_{ens}$ [8]	2.6	85.8	**95.2**	0.735	**0.906**
HGD [120]	1.2	89.4	**96.3**	0.799	**0.927**
R&P [121]	1.8	88.6	**95.7**	0.785	**0.916**
NIPS-r3 [118]	3.2	90.1	**96.5**	0.811	**0.930**
RS [9]	N/A	66.4	**92.2**	N/A	N/A
Average	4.083	86.1	**95.6**	0.796	**0.923**

appear in the table where the authors of A^3 report their results.

A^3 shows some success against all models. Its average ASR and score are fairly close to SSAH (A)'s, however, the standard deviations are much smaller (and computed from many more models). So while A^3's results are more reliable than SSAH (A), it does not necessarily mean that SSAH (A) is worse. However, it does show that A^3 is more likely to be a more reliable attack against a wide array of defenses.

The second comparison we will look at is the comparison of the models above (A^3 and SSAH (A)) with ACG [83] (summary 4.1.10), which is also a model parameter-based attack that uses the same information as A^3 except that it does not use any data-related information. We will label this comparison as the color blue.

Blue: ACG & A^3 & SSHA (A) Similarly to the previous comparison, SSAH (A) does not attack any of the models that either ACG or A^3 attacks. However, ACG and A^3 attack a swath of defended models in common. These models were retrieved from RobustBench [94] which allows us to extract the benign accuracies (which allows us to confirm that they are indeed the same models). In total, nine models are attacked by both papers. Since SSAH (A) shares no models with the other papers, we will only focus on comparing ACG and A^3 as we can always retrieve SSAH (A)'s performance from Table 7.18. The comparison between A^3 and ACG is summarized in Table 7.20.

Table 7.11: NIPS 2017 dataset indistinguishable transferable and white-box undefended comparison

Model	S^2I (A) ASR	S^2I (B) ASR	Diff
Inception-v3 [96]	90.3	**99.7**	9.4
Inception-v4 [99]	89	**96.6**	7.6
Inception-ResNet-v2 [99]	86.5	**98.4**	11.9
ResNet-152 [95]	84.9	**99.7**	14.8
Average	87.7	**99.4**	11.7

Table 7.12: CIFAR10 dataset indistinguishable BIA (A) undefended results

Model	Benign ASR	BIA (A) ASR	BIA (A) Score
Custom model [11]	6.22	47.19	0.219

A^3 appears to be getting better results than ACG across all models. However, the difference is very small every time, therefore we can argue that when attackers already have access to the target model parameters and the training loss, having access to data from the same distribution could maybe help, but only provides very little improvement. This difference could also be attributed to the inherent algorithmic differences between A^3 and ACG rather than the information they use.

ACG also seems to be quite a consistent attack when it comes to attacking defended models which reinforces the unsuitability of SSAH (A)'s high-variance results. Finally, we can move to the last comparison with the only attack that does not use model parameters on CIFAR100: BIA (A) [11] (summary 4.1.9). We color the overall comparison on CIFAR100 red.

Red: BIA (A) & ACG & A^3 & SSHA (A) Unfortunately, BIA (A) attacks only a single custom model (it is not specified whether it is defended or not). This greatly limits the comparisons that can be made. The results gathered from this model can be found in

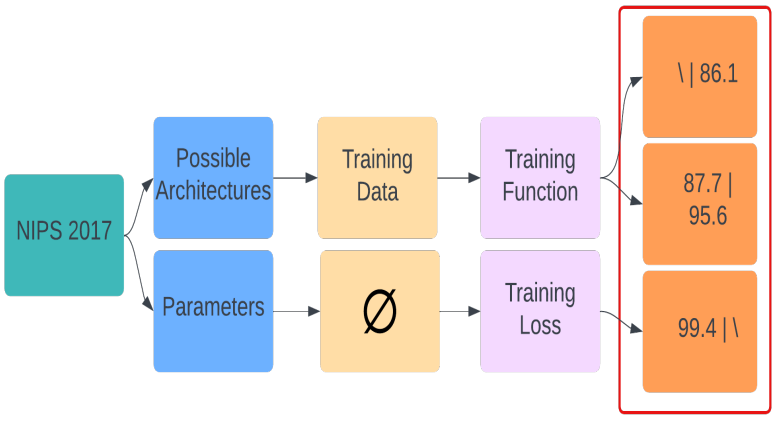

Figure 7.8: NIPS 2017 dataset indistinguishable results. The first number is the average ASR on undefended models whereas the second is the average ASR on defended models (we write missing results as \). We bold the best performer.

Table 7.21.

While this table showcases a pretty strong ASR (and associated score), for fairness purposes, we have to assume the worst case for the performance of the attack and assume that the model they are undefended as they have not stated otherwise. This results in a pretty significant drop in attack performance when going from using model parameters to a transferable attack setting as usually, most attacks achieve very high ASRs against undefended models. We present a summary of the results within the tree structure in Figure 7.12.

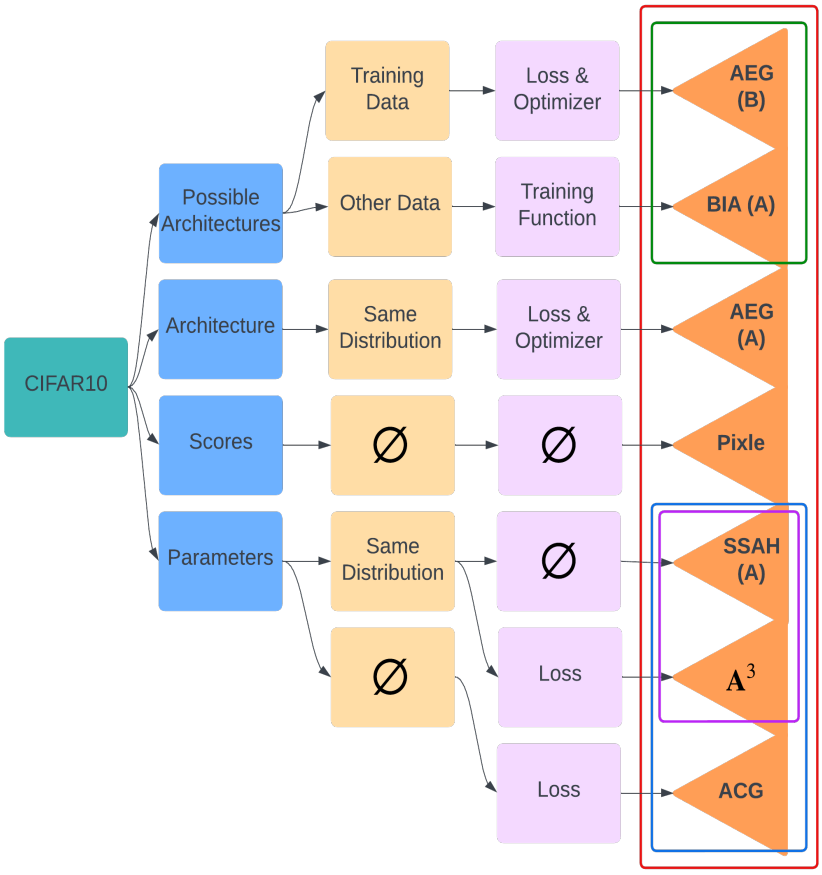

Figure 7.9: CIFAR10 dataset indistinguishable attacks

7.2.4 Attacks on the ImageNet dataset

Finally, we tackle the last dataset where comparisons are possible, ImageNet [101]. It is by far the most evaluated dataset that we study in our work, with eleven variants of attacks evaluated against it. As such, we perform four comparisons, starting by comparing the different transferable attacks: LGV (A) [54] (summary 4.1.3), BIA (B) [11] (summary 4.1.9), ATA (A) [82] (summary 4.1.12), SSAH (B) [65] (summary 4.1.7).

Blue: LGV (A) & BIA (B) & ATA (A) & SSAH (B) On one hand, LGV (A), BIA (B) and ATA (A) all use the same information: possible architectures, the training data and the training function. On the other hand, SSAH (B) uses the same information

Table 7.13: CIFAR10 dataset indistinguishable AEG (B) undefended results

Model	**Benign ASR**	**AEG (B) ASR**	**AEG (B) Score**
VGG-16 [98]	11.2	93.8	0.867
ResNet-18 [95]	13.1	97.3	0.930
WideResNet [110]	6.8	85.2	0.721
DenseNet-121 [97]	11.2	94.1	0.873
Inception-v3 [96]	9.9	92.7	0.850
Average	10.44	92.62	0.848
Standard deviation	2.33	4.49	0.077

except for the data information where it uses data from the same distribution on top of the rest. All these attacks attack undefended papers. Some share architectures, however, not all of them provide benign accuracies for the models they evaluate. When possible, we use the provided accuracies or extrapolate from the accuracies provided in other papers under the assumption that well-trained undefended models on the same dataset with the same architectures tend to have similar benign accuracies. In the case where papers do not share benign accuracies, and we compare them against another that does, we italicize the unreliable results. We compile the results in Tables 7.22 and 7.23. BIA (B), LGV(A), and SSAH (B) both use the l_∞-norm but BIA (B) uses a perturbation budget of 10/256 whereas SSAH (B) uses one of 8/256, LGV (A) does not specify the budget used. On the other hand, ATA (A) uses the l_0-norm with a perturbation budget of 1024/(224*224), meaning they are allowed to fully modify 1024 pixels. Unfortunately, this means that the following comparison is disparate.

BIA (B) outperforms the other attacks. Due to the lack of results for SSAH (B) and the fact that the few results are worse than the other attacks even though SSAH (B) has more information, we ignore it in what follows. All the other three attacks use the same information, there are three possible reasons for this disparity in results:

1. BIA (B) is a better algorithm for extracting information and building a potent attack from it.

2. LGV (A) and ATA (A) attacked undefended models that were somehow inherently more robust (unlikely).

Table 7.14: CIFAR10 dataset indistinguishable AEG (B) defended results

Model	Benign ASR	AEG (B) ASR	AEG (B) Score
ResNet-18$_{\text{ens3}}$ [8]	16.8	52.2	0.244
WideResNet$_{\text{ens3}}$ [8]	12.8	49.9	0.232
DenseNet-121$_{\text{ens3}}$ [8]	21.5	41.4	0.125
Inception-v3$_{\text{ens3}}$ [8]	14.8	47.5	0.204
Madry-Adv [7]	12.9	21.6	0.030
Average	15.76	47.52	0.167
Standard deviation	3.60	12.37	0.090

Table 7.15: CIFAR10 dataset indistinguishable SSAH (A) and A^3 comparison defended

Model	Benign ASR	SSAH (A) ASR	A^3 ASR	SSAH (A) Score	A^3 Score
WideResNet-34-10$_{\text{TRADES}}$ [128]	15.08	21.32	**46.99**	0.023	**0.198**

3. The disparity of results is caused by the different perturbation budgets/l_p-norms used.

We suspect a combination of 1. and 3. to be the reason explaining the discrepancy in the results. Additionally, ATA's attack is originally meant to attack vision transformer models, hence that could partially explain a slightly worse performance in an equitable evaluation setting.

Green: LGV (B) & ATA (B) & ACG LGV (B) and ATA (B) are the white-box versions of the attacks. All three of the attacks use the model parameters and no data information. LGV (B) and ATA (B) both also do not use any training information while ACG uses the loss function. This comparison is difficult to perform, as LGV (B) and ATA (B) both attack different undefended models while not providing any benign accuracies. While we can replicate what we did for the previous comparison of LGV (A) with LGV (B) and use the benign accuracy of the BIA paper, we, unfortunately, cannot do that for

Table 7.16: CIFAR10 dataset indistinguishable A^3 and ACG defended comparison. * WideResNet-28-$10_{\text{pretraining}}$ misreported as WideResNet-34-$10_{\text{pretraining}}$ in the A^3 paper.

Model	Benign ASR	A^3 ASR	ACG ASR	A^3 Score	ACG Score
WideResNet-34-10_{TRADES} [128]	15.08	46.99	**47.18**	0.198	**0.200**
WideResNet-34-20_{LBGAT} [200]	11.3	**46.54**	46.23	**0.204**	0.201
WideResNet-34-10_{LBGAT} [200]	11.78	**47.24**	46.9	**0.209**	0.206
WideResNet-34-$20_{\text{Overfitting}}$ [202]	14.66	**46.67**	45.69	**0.196**	0.187
WideResNet-70-16_{Fixing} [184]	11.46	**35.81**	35.27	**0.115**	0.111
WideResNet-28-10_{Fixing} [184]	12.67	**39.34**	38.8	**0.139**	0.134
WideResNet-28-10_{AWP} [185]	11.75	**40.02**	39.7	**0.146**	0.144
WideResNet-70-16_{ULAT} [183]	8.9	**34.22**	33.7	**0.109**	0.106
WideResNet-70-16_{ULAT} [183]	14.71	**42.92**	42.45	**0.163**	0.159
WideResNet-34-20_{ULAT} [183]	14.36	**43.24**	42.86	**0.166**	0.163
WideResNet-28-10_{ULAT} [183]	10.52	**37.3**	36.9	**0.128**	0.125
WideResNet-28-$10_{\text{pretraining}}$* [194]	12.89	**45.24**	44.75	**0.188**	0.184
Average	12.51	**42.13**	41.70	**0.163**	0.160

ATA (B) as the only architectures they attack (DeiT's [152]) in the white-box scenario are ones that no other papers attack (DiffAttack [85] attacks them but for a different dataset). On the other hand, ACG [83] (summary 4.1.10) attacks only defended models. Therefore, we report the results that we include in the final tree. We compile them in Tables 7.24, 7.25 and 7.26.

Purple: Green & BIA (C) & SSAH (A) Both BIA (C) and SSAH (A) are the white-box variants of the original attacks. BIA (C) uses the training data on top of the model parameters while SSAH (A) uses data from the same distribution. Neither of them uses any training information. Both of these variants evaluate only against undefended models, therefore comparing against ACG is not possible. We can, however, compare SSAH (A) and LGV (B) as they both evaluate against a ResNet-50 [95]. Since SSAH (A) publishes their

Table 7.17: CIFAR10 dataset indistinguishable AEG (B) & Pixle comparison defended

Model	Benign ASR	AEG (B) ASR	Pixle ASR	AEG (B) Score	Pixle Score
ResNet-18 [95]	13.1	97.3	**100.0**	0.930	**0.983**

Table 7.18: CIFAR100 dataset indistinguishable SSAH (A) defended results

Model	Benign ASR	SSAH (A) ASR	SSAH (A) Score
WideResNet-34-10$_{\text{TRADES}}$ [129]	43.06	50.77	0.0723
WideResNet-34-10$_{\text{FSAT}}$ [128]	25.89	95.15	0.838
Average	34.48	72.96	0.455
Standard deviation	12.14	31.38	0.542

ResNet-50's benign accuracy and LGV (B) does not, but we use BIA's benign accuracy in the previous evaluation (Green), we compute LGV (B)'s score using the mean of both benign accuracy to get closer to LGV (B)'s ResNet-50's expected benign accuracy and get a more reliable result. We compile the results in Table 7.27. On the other hand, BIA (C) only evaluates their white-box scenario against a VGG-16 and a DenseNet-169 in the undefended setting so we report their results separately in Table 7.28.

Red: Blue & Purple & MASSA & Pixle This is the final comparison. We include the results from MASSA [66] (summary 4.1.5) and Pixle [37] (summary 4.1.4) to complete our tree. We do so on architectures that most of the other papers in Blue and Purple attack: ResNet-50 and VGG-16. MASSA is a label query-based attack while Pixle is a score query-based attack. Neither of them uses any other information. Since neither paper provides clean accuracy, we replicate our procedure from LGV (B)'s evaluation in the Purple comparison where we use the average of the benign accuracies from BIA and SSAH to compute the score. This yields the compiled results in Table 7.29. While MASSA evaluates its performance against defended models, it does not fix a specific l_2-norm perturbation value and instead looks at how the attack performs when increasing the l_2-norm until the defense is completely broken or the l_2-norm reaches 30. This makes it hard for us to compare against other papers as they are the sole paper to use the l_2-

Table 7.19: CIFAR100 dataset indistinguishable A^3 defended results

Model	Benign ASR	A^3 ASR	A^3 Score
1	30.85	63.14	0.303
2	36.44	65.45	0.296
3	37.59	68.0	0.321
4	34.27	69.69	0.368
5	37.45	69.88	0.348
6	39.14	70.01	0.337
7	39.36	70.82	0.347
8	39.62	71.22	0.350
9	40.77	71.69	0.348
10	37.98	72.91	0.387
11	17.18	75.49	0.540
12	46.17	81.1	0.445
Average	36.40	70.78	0.366
Standard deviation	7.08	4.59	0.067

norm. Additionally, they do not provide clean accuracies for any of their models. Pixle on the other hand does not attack any defended models. MASSA seems to outperform Pixle slightly, although, again, due to the unreliable nature of the results, it's hard to say if this would entirely hold in practice. However, it shows that query-based access to a model is sufficient to render it useless if the model is undefended.

We report the final result in a tree in Figure 7.14. The takeaways for ImageNet are the following:

- Only query-based access to the model and no other information is sufficient to reduce an undefended model's accuracy to almost 0%.
- Transferable settings can achieve extremely high ASRs (93.48% for BIA (B)), which shows that, at least in the undefended case, transferable attacks that use additional

Table 7.20: CIFAR100 dataset indistinguishable A^3 and ACG defended comparison

Model	Benign ASR	A^3 ASR	ACG ASR	A^3 Score	ACG Score
WideResNet-70-16$_{\text{Fixing}}$ [184]	36.44	**64.45**	64.77	**0.296**	0.287
WideResNet-28-10$_{\text{Fixing}}$ [184]	37.59	**68.0**	67.27	**0.321**	0.311
ResNet-18-v2$_{\text{Overfitting}}$ [202] [119]	46.17	**81.1**	80.63	**0.445**	0.437
WideResNet-28-10$_{\text{pre-training}}$ [194]	40.77	**71.69**	70.51	**0.348**	0.331
WideResNet-34-10$_{\text{LGBAT}}$ [200]	39.36	**70.82**	70.33	**0.347**	**0.340**
WideResNet-34-10$_{\text{AWP}}$ [185]	39.62	**71.22**	70.11	**0.350**	0.335
WideResNet-34-20$_{\text{LBGAT}}$ [200]	37.45	**69.88**	69.13	**0.348**	0.338
WideResNet-70-16$_{\text{ULAT}}$ [183]	30.85	**63.14**	62.19	**0.303**	0.292
WideResNet-70-16$_{\text{ULAT}}$ [183]	39.14	**70.01**	69.43	**0.337**	0.329
Average	38.60	**70.15**	69.37	**0.344**	0.333

Table 7.21: CIFAR100 dataset indistinguishable BIA (A) undefended results

Model	Benign ASR	BIA (A) ASR	BIA (A) Score
Custom model [11]	25.73	79.18	0.561

data and training information (usually to train surrogates) can achieve ASRs close to query-based and white-box attacks. Unfortunately, since none of the transferable attacks we studied evaluated defended models, we cannot say if this extends to defended models.

- ImageNet-defended models are mostly broken by white-box attacks. ACG obtains an average ASR of 68.5% and an average score 0.314. While, defended models do perform better than undefended ones, the accuracy loss is still quite large.

Table 7.22: ImageNet dataset indistinguishable Blue ASR comparison undefended

Model	Benign ASR (BIA)	LGV (A) ASR	BIA (B) ASR	ATA (A) ASR	SSAH (B) ASR
ResNet-50 [95]	24.39	N/A	**95.56**	*87.31*	N/A
VGG-16 [98]	N/A	N/A	N/A	**87.46**	*19.14*
DenseNet-121 [97]	25.78	N/A	**96.02**	*64.17*	N/A
ResNet-152 [95]	22.66	*89.6*	**94.15**	N/A	N/A
VGG-19 [98]	29.05	*82.2*	**95.91**	N/A	N/A
Inception-v3 [96]	23.81	*45.4*	**85.76**	N/A	N/A
Average	25.14	*72.4*	**93.48**	*79.65*	*19.14*
Standard deviation	2.46	*23.67*	4.38	*13.4*	N/A

7.3 Takeaways

In this chapter, we presented a lot of results from various datasets, attacks and settings. In this section, we want to identify the core takeaways from our findings. They are the following:

Undefended models are broken. This is not a particularly new finding, but we confirm and reinforce it. We show that across all datasets and threat models studied, not once are undefended models robust to attackers. This anchors the previously held belief that defenses are not only beneficial, they are required for any model to be deployed in practice.

The transferable setting might not be as difficult as previously thought. For most of the datasets and salient situations studied, we noticed that transferable attacks, when given additional information like access to the training data and training information such as the training function, can perform very well, almost on par with white-box attacks. This reinforces the notion that any information given to the attacker is crucial, especially when that information can allow attackers to train surrogate models on the same data distribution. This is observed best when looking at BIA (A) on the CIFAR10 and CIFAR100 dataset. They severely underperform other transferable attacks. Therefore, we posit that data information is a crucial component for transferable attacks.

Table 7.23: ImageNet dataset indistinguishable Blue score comparison undefended

Model	Benign ASR (BIA)	LGV (A) Score	BIA (B) Score	ATA (A) Score	SSAH (B) Score
ResNet-50 [95]	24.39	N/A	**0.854**	*0.703*	N/A
DenseNet-121 [97]	25.78	N/A	**0.856**	*0.345*	N/A
ResNet-152 [95]	22.66	*0.751*	**0.835**	N/A	N/A
VGG-19 [98]	29.05	*0.591*	**0.835**	N/A	N/A
Inception-v3 [96]	23.81	*0.149*	**0.679**	N/A	N/A
Average	25.14	*0.497*	**0.811**	*0.524*	N/A
Standard deviation	2.46	*0.312*	0.075	*0.253*	N/A

Table 7.24: ImageNet dataset indistinguishable LGV (B) results undefended

Model	Benign ASR (BIA)	LGV (B) ASR	LGV (B) Score
ResNet-50 [95]	24.39	*97.1*	*0.883*

Information categories matter, but their effect on performance is not straightforward. As mentioned in the previous takeaway, transferable attacks with access to the training data perform almost on par with white-box attacks. However, losing access to that original training data causes a sharp drop in attack performance. On the other hand, we find that data or training information plays a very minor role in white-box settings. This reinforces the belief that model information trumps other information categories as it alone can allow for very potent attacks, whereas other information categories seem to fill more of a supportive role that helps bridge the gap when model information is sparse.

Training information is understated but essential to transferable attacks. All the transferable attacks whose results we have studied use the training function (except for AEG which only uses the loss function and optimizer, which is most of the training function). This is done to train surrogate models that can then be attacked by modified white-box attacks that are adapted to be 'more transferable'. However, this assumption is not directly stated in the papers we survey and is instead taken for granted. This goes against proper security practices as it omits, what our results clearly show to be critical

Table 7.25: ImageNet dataset indistinguishable ATA (B) results undefended

Model	Benign ASR	ATA (B) ASR
DeiT-T [152]	N/A	*100.0*
DeiT-S [152]	N/A	*100.0*
DeiT-B [152]	N/A	*100.0*

Table 7.26: ImageNet dataset indistinguishable ACG results defended

Model	Benign ASR	ACG ASR	ACG Score
ResNet-$50_{\text{robustness}}$ [206]	37.44	69.58	0.344
ResNet-50_{salman} [231]	35.98	64.7	0.289
ResNet-18_{salman} [231]	47.08	74.34	0.331
WideResNet-50-2_{salman} [231]	31.54	60.9	0.271
ResNet-$50_{\text{FAST\_AT}}$ [164]	44.38	73.0	0.336
Average	39.28	68.50	0.314
Standard deviation	6.34	5.65	0.032

information, in the threat model description.

Need for better standard evaluation frameworks. While there now exists frameworks like RobustBench [94] or Robustness [206] that hold many pre-trained defended models with available metrics that can be re-used and attacked by papers. They are poorly adapted in the broader community (only four of the twenty attacks we studied evaluated against them). In general, there is a surprising lack of standards to evaluate adversarial example attacks. This unnecessarily complicates comparisons between existing works that are not directly related (when one work inspires another and compares against it) and yields the unwieldy results' section that we have.

Table 7.27: ImageNet dataset indistinguishable LGV (B) & SSAH (A) comparison undefended

Model	Benign ASR	LGV (B) ASR	SSAH (A) ASR	LGV (B) Score	SSAH (A) Score
ResNet-50 [95]	24.39 (BIA) 23.85 (SSAH)	*97.1*	**98.56**	*0.885*	**0.915**

Table 7.28: ImageNet dataset indistinguishable BIA (C) results undefended

Model	Benign ASR	BIA (C) ASR	BIA (C) Score
VGG-16 [98]	29.86	98.96	0.890
DenseNet-169 [97]	24.25	96.68	0.876
Average	27.06	97.82	0.883
Standard deviation	3.97	1.61	0.010

Evaluation against defended models. Too few papers evaluate their attack against defended models (and they usually evaluate too few defended models as well when they do evaluate against defended models). As mentioned in our first takeaway, it is a well-known fact that undefended models are broken. Therefore, papers that only evaluate undefended models bring very little to the field as a whole beyond showing that something that was already broken can be broken again in a slightly different way. Papers evaluating against easily accessible, well-known defended models would allow for a much easier comparison of attacks in both the same and different settings. For example, comparing ACG to A^3 was remarkably easy due to them both attacking numerous models from the RobustBench [94] framework.

Table 7.29: ImageNet dataset indistinguishable MASSA & Pixle comparison undefended

Model	**Benign ASR**	**MASSA ASR**	**Pixle ASR**	**MASSA Score**	**Pixle Score**
ResNet-50 [95]	24.39 (BIA) 23.85 (SSAH)	***99.4***	*98.0*	***0.930***	*0.902*
VGG-16 [98]	29.86 (BIA)	***99.58***	*99.0*	***0.902***	*0.891*
Average	26.99	***99.49***	*98.5*	***0.916***	*0.897*

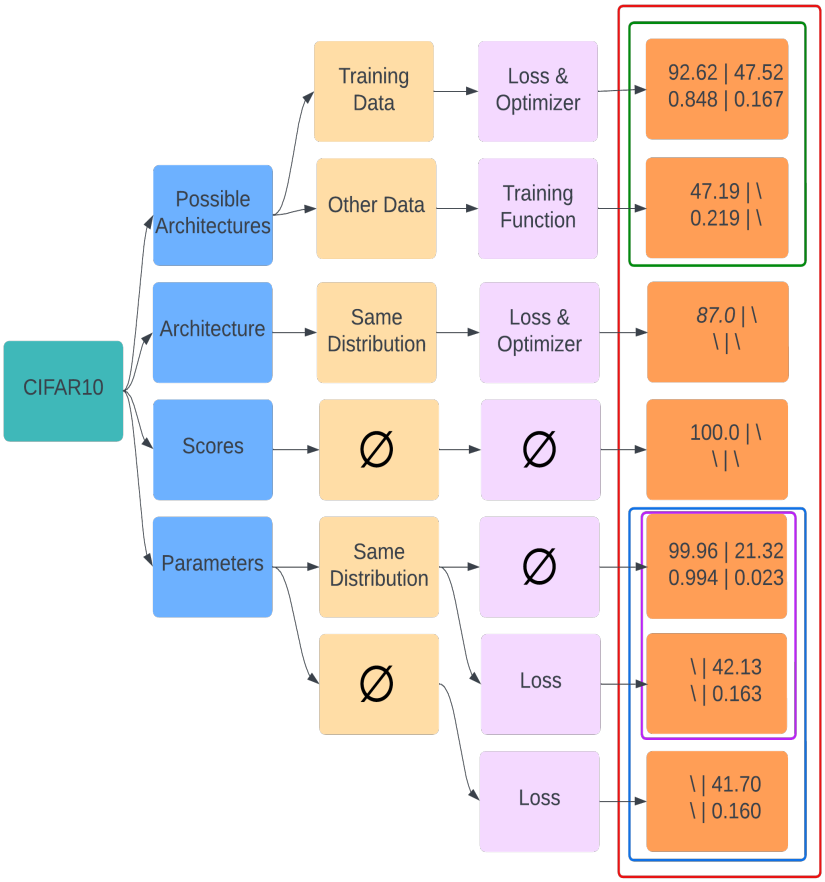

Figure 7.10: CIFAR10 dataset indistinguishable attack results. The first row of numbers is the average ASR while the second is the average relative performance score. The first column of numbers is the performance on undefended models whereas the second is the performance on defended models (we write missing results as \). We italicize unreliable results.

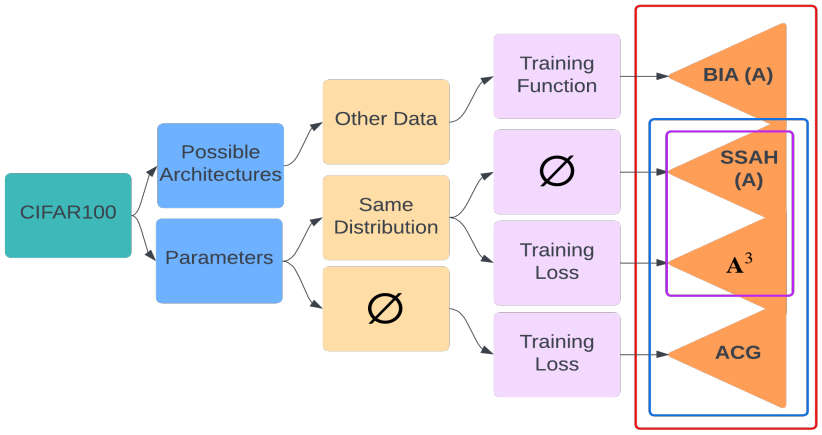

Figure 7.11: CIFAR100 dataset indistinguishable attacks

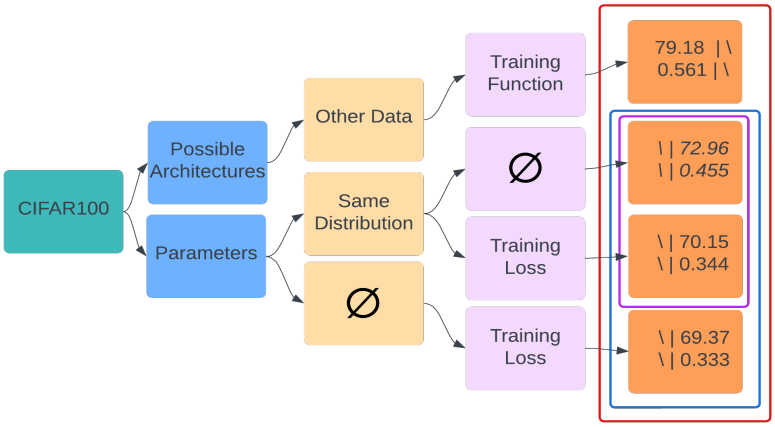

Figure 7.12: CIFAR100 dataset indistinguishable attack results. The first row of numbers is the average ASR while the second is the average relative performance score. The first column of numbers is the performance on undefended models whereas the second is the performance on defended models (we write missing results as \). We bold the best performer for each row. We italicize unreliable results.

Figure 7.13: ImageNet dataset indistinguishable attacks

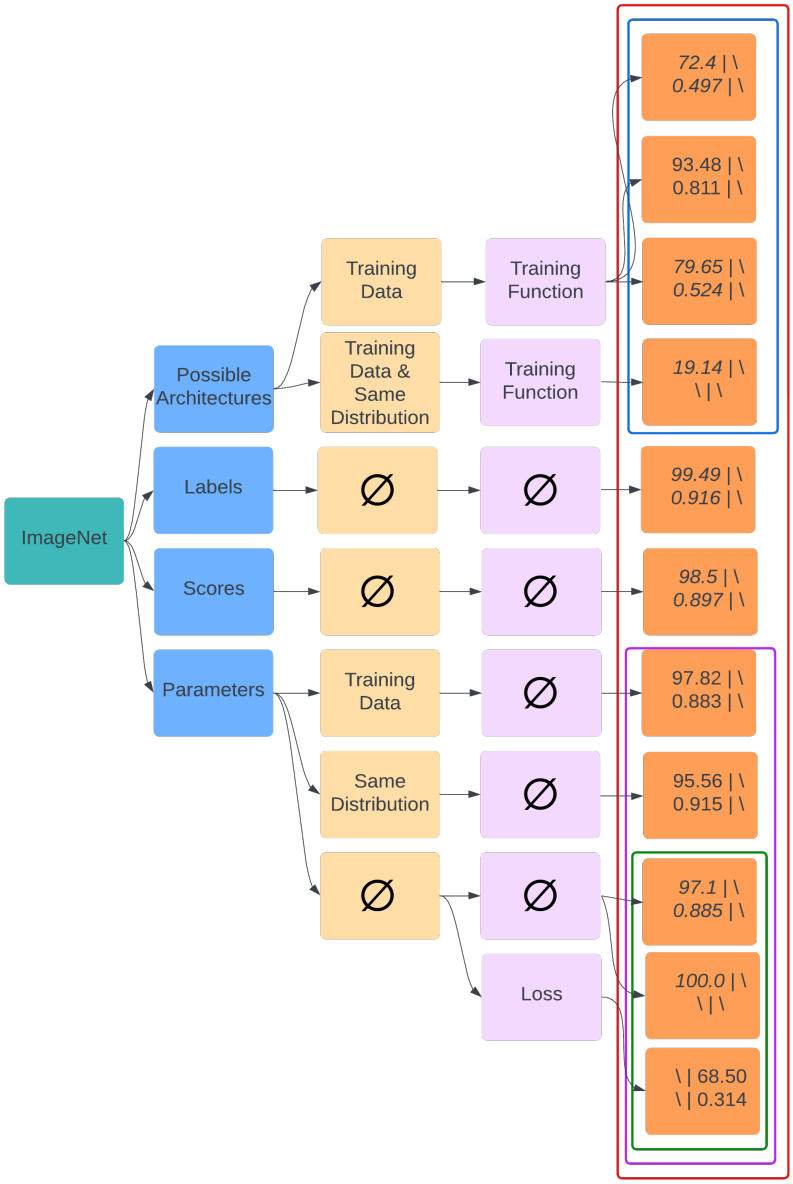

Figure 7.14: ImageNet dataset indistinguishable attack results. The first row of numbers is the average ASR while the second is the average relative performance score. The first column of numbers is the performance on undefended models whereas the second is the performance on defended models (we write missing results as \). We italicize unreliable results.

Chapter 8

Conclusion

In this thesis, we presented a formalization to study adversary knowledge in the context of adversarial example attacks on image classification models. This formalization is composed of a theoretical framework to model, understand and relate information composing an adversary's knowledge to mount an attack. It includes a security game that models the adversarial example attack scenario. To complement it, we provide a survey of recent attacks in the image classification domain, to which we apply our formalization to yield an in-depth empirical, falsifiable study of adversary knowledge and attack performance. We succeeded in laying a strong theoretical foundation for future work, both attack and defenses, to use and apply. We hope this elevates the standard in adversarial example research to that of proper security research, and in turn encourages future work to more carefully weigh assumptions and consider the threat models that follow, as we have shown they can drastically change an attack's outcome. Our work can be used to enhance the reproducibility, comparability and fairness of future work as it allows a precise description of one's threat model and evaluations.

Unfortunately, due to the lack of aforementioned comparability in most of the existing work, we are more limited in the conclusions we can derive from our results than we would like. Given a threat model, it is possible that not all attacks that have been studied, if any, optimally use the information they have access to. This means that some of the results we observe could be due to algorithmic inefficiencies rather than a difference in the underlying information used by the attack. We address this when presenting our results. Additionally, for future work, we emphasize making results more comparable. It is currently extremely hard to identify top performers within each threat model due to this lack of comparability of results. A wider adoption of benchmarking frameworks like RobustBench [94] as well as a standardization of evaluation practices are key to establishing

tangible, lasting and comparable results. Following in the footsteps of other fields of adversarial machine learning, like backdoor research, towards more theoretically-backed and provable methods, rather than purely heuristical and empirical methods is another key evolution that adversarial example research needs to follow. As for future work following this systematization of knowledge, a similar exploration of knowledge used by defenders could provide additional valuable insights in the search for both accurate and robust image classification models. This systematization of knowledge could also be extended to other data domains, as has adversarial example research.

Finally, we hope our research can help inspire present and future researchers to improve the overall quality of the research output in the adversarial example research field. Adversarial example research has existed for close to a decade now, and we hope that our work will orient future research toward long-term solutions.

References

[1] Cem Dilmegani. 45 statistics, facts & forecasts on machine learning [2023], 10 April 2023. URL https://research.aimultiple.com/ml-stats/.

[2] Hafsa Habehh1 and Suril Gohel. Machine learning in healthcare. *national library of medicine*, 2021. doi: 10.2174/1389202922666210705124359.

[3] Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021. ISSN 2590-0056. doi: https://doi.org/10.1016/j.array.2021.100057. URL https://www.sciencedirect.com/science/article/pii/S2590005621000059.

[4] Mindy News Blog. How machine learning in automotive makes self-driving cars a reality. URL https://mindy-support.com/news-post/how-machine-learning-in-automotive-makes-self-driving-cars-a-reality/.

[5] Zhehan Ni. Reframe the field of aerospace engineering via machine learning: Application and comparison. *Journal of Physics: Conference Series*, 2386(1):012031, dec 2022. doi: 10.1088/1742-6596/2386/1/012031. URL https://dx.doi.org/10.1088/1742-6596/2386/1/012031.

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[7] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[8] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses, 2020.

[9] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[10] Elham Tabassi, Kevin J. Burns, Michael Hadjimichael, Andres D. Molina-Markham, and Julian T. Sexton. A taxonomy and terminology of adversarial attacks and defenses on machine learning, 2019. URL https://csrc.nist.rip/external/nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8269-draft.pdf.

[11] Qilong Zhang, Xiaodan Li, Yuefeng Chen, Jingkuan Song, Lianli Gao, Yuan He, and Hui Xue. Beyond imagenet attack: Towards crafting adversarial examples for black-box domains, 2022.

[12] Jiyuan Liu, Bingyi Lu, Mingkang Xiong, Tao Zhang, and Huilin Xiong. Adversarial attack with raindrops. (arXiv:2302.14267), July 2023. doi: 10.48550/arXiv.2302.14267. URL http://arxiv.org/abs/2302.14267. arXiv:2302.14267 [cs].

[13] Fahad Shamshad, Muzammal Naseer, and Karthik Nandakumar. Clip2protect: Protecting facial privacy using text-guided makeup via adversarial latent search. page 20595–20605, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Shamshad_CLIP2Protect_Protecting_Facial_Privacy_Using_Text-Guided_Makeup_via_Adversarial_Latent_CVPR_2023_paper.html.

[14] Yiqi Zhong, Xianming Liu, Deming Zhai, Junjun Jiang, and Xiangyang Ji. Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15345–15354, June 2022.

[15] Junhua Zou, Yexin Duan, Boyu Li, Wu Zhang, Yu Pan, and Zhisong Pan. Making adversarial examples more transferable and indistinguishable. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(33):3662–3670, June 2022. ISSN 2374-3468. doi: 10.1609/aaai.v36i3.20279.

[16] Xingxing Wei and Shiji Zhao. Boosting adversarial transferability with learnable patch-wise masks. *IEEE Transactions on Multimedia*, page 1–10, 2023. ISSN 1941-0077. doi: 10.1109/TMM.2023.3315550.

[17] Zhipeng Wei, Jingjing Chen, Zuxuan Wu, and Yu-Gang Jiang. Enhancing the self-universality for transferable targeted attacks. page 12281–12290, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Wei_Enhancing_

the_Self-Universality_for_Transferable_Targeted_Attacks_CVPR_2023_paper.html.

[18] Juanjuan Weng, Zhiming Luo, Dazhen Lin, Shaozi Li, and Zhun Zhong. Boosting adversarial transferability via fusing logits of top-1 decomposed feature. (arXiv:2305.01361), July 2023. doi: 10.48550/arXiv.2305.01361. URL http://arxiv.org/abs/2305.01361. arXiv:2305.01361 [cs].

[19] Yaoyuan Zhang, Yu-an Tan, Haipeng Sun, Yuhang Zhao, Quanxing Zhang, and Yuanzhang Li. Improving the invisibility of adversarial examples with perceptually adaptive perturbation. *Information Sciences*, 635:126–137, July 2023. ISSN 0020-0255. doi: 10.1016/j.ins.2023.03.139.

[20] Donald C. Wunsch Tao Wu, Tie Luo. Black-box attack using adversarial examples: A new method of improving transferability. URL https://www.worldscientific.com/doi/10.1142/S2811032322500059.

[21] Juanjuan Weng, Zhiming Luo, Shaozi Li, Nicu Sebe, and Zhun Zhong. Logit margin matters: Improving transferable targeted adversarial attack by logit calibration. *IEEE Transactions on Information Forensics and Security*, 18:3561–3574, 2023. doi: 10.1109/TIFS.2023.3284649.

[22] Zhengjie Deng, Wen Xiao, Xiyan Li, Shuqian He, and Yizhen Wang. Enhancing the transferability of targeted attacks with adversarial perturbation transform. *Electronics*, 12(1818):3895, January 2023. ISSN 2079-9292. doi: 10.3390/electronics12183895.

[23] Zhijin Ge, Fanhua Shang, Hongying Liu, Yuanyuan Liu, and Xiaosen Wang. Boosting adversarial transferability by achieving flat local maxima. (arXiv:2306.05225), June 2023. doi: 10.48550/arXiv.2306.05225. URL http://arxiv.org/abs/2306.05225. arXiv:2306.05225 [cs].

[24] Zhijin Ge, Fanhua Shang, Hongying Liu, Yuanyuan Liu, Liang Wan, Wei Feng, and Xiaosen Wang. Improving the transferability of adversarial examples with arbitrary style transfer. August 2023. doi: 10.1145/3581783.3612070. URL http://arxiv.org/abs/2308.10601. arXiv:2308.10601 [cs, eess].

[25] Hongying Liu, Zhijin Ge, Zhenyu Zhou, Fanhua Shang, Yuanyuan Liu, and Licheng Jiao. Gradient correction for white-box adversarial attacks. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–12, 2023. ISSN 2162-2388. doi: 10.1109/TNNLS.2023.3315414.

[26] Zhibo Wang, Hongshan Yang, Yunhe Feng, Peng Sun, Hengchang Guo, Zhifei Zhang, and Kui Ren. Towards transferable targeted adversarial examples. page 20534–20543, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Wang_Towards_Transferable_Targeted_Adversarial_Examples_CVPR_2023_paper.html.

[27] Florian Tramer. Detecting adversarial examples is (nearly) as hard as classifying them. In *Proceedings of the 39th International Conference on Machine Learning*, page 21692–21702. PMLR, June 2022. URL https://proceedings.mlr.press/v162/tramer22a.html.

[28] Junyoung Byun, Myung-Joon Kwon, Seungju Cho, Yoonji Kim, and Changick Kim. Introducing competition to boost the transferability of targeted adversarial examples through clean feature mixup. page 24648–24657, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Byun_Introducing_Competition_To_Boost_the_Transferability_of_Targeted_Adversarial_Examples_CVPR_2023_paper.html.

[29] Bin Chen, Jiali Yin, Shukai Chen, Bohao Chen, and Ximeng Liu. An adaptive model ensemble adversarial attack for boosting adversarial transferability. page 4489–4498, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Chen_An_Adaptive_Model_Ensemble_Adversarial_Attack_for_Boosting_Adversarial_Transferability_ICCV_2023_paper.html.

[30] Qizhang Li, Yiwen Guo, Xiaochen Yang, Wangmeng Zuo, and Hao Chen. Improving transferability of adversarial examples via bayesian attacks. (arXiv:2307.11334), July 2023. doi: 10.48550/arXiv.2307.11334. URL http://arxiv.org/abs/2307.11334. arXiv:2307.11334 [cs].

[31] Qizhang Li, Yiwen Guo, Wangmeng Zuo, and Hao Chen. Making substitute models more bayesian can enhance transferability of adversarial examples. (arXiv:2302.05086), July 2023. doi: 10.48550/arXiv.2302.05086. URL http://arxiv.org/abs/2302.05086. arXiv:2302.05086 [cs].

[32] Zeyu Qin, Yanbo Fan, Yi Liu, Li Shen, Yong Zhang, Jue Wang, and Baoyuan Wu. Boosting the transferability of adversarial attacks with reverse adversarial perturbation. *Advances in Neural Information Processing Systems*, 35:29845–29858, December 2022.

[33] Xiangyuan Yang, Jie Lin, Hanlin Zhang, Xinyu Yang, and Peng Zhao. Improving the transferability of adversarial examples via direction tuning. (arXiv:2303.15109), August 2023. doi: 10.48550/arXiv.2303.15109. URL http://arxiv.org/abs/2303.15109. arXiv:2303.15109 [cs].

[34] Yulong Yang, Chenhao Lin, Qian Li, Chao Shen, Dawei Zhou, Nannan Wang, and Tongliang Liu. Quantization aware attack: Enhancing the transferability of adversarial attacks across target models with different quantization bitwidths. (arXiv:2305.05875), May 2023. doi: 10.48550/arXiv.2305.05875. URL http://arxiv.org/abs/2305.05875. arXiv:2305.05875 [cs].

[35] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. (arXiv:2306.13213), August 2023. doi: 10.48550/arXiv.2306.13213. URL http://arxiv.org/abs/2306.13213. arXiv:2306.13213 [cs].

[36] Maura Pintor, Luca Demetrio, Angelo Sotgiu, Ambra Demontis, Nicholas Carlini, Battista Biggio, and Fabio Roli. Indicators of attack failure: Debugging and improving optimization of adversarial examples. *Advances in Neural Information Processing Systems*, 35:23063–23076, December 2022.

[37] Jary Pomponi, Simone Scardapane, and Aurelio Uncini. Pixle: a fast and effective black-box attack based on rearranging pixels. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2022. doi: 10.1109/ijcnn55064.2022.9892966. URL https://doi.org/10.1109%2Fijcnn55064.2022.9892966.

[38] Xiu-Chuan Li, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Decision-based adversarial attack with frequency mixup. *IEEE Transactions on Information Forensics and Security*, 17:1038–1052, 2022. doi: 10.1109/TIFS.2022.3156809.

[39] Ye Liu, Yaya Cheng, Lianli Gao, Xianglong Liu, Qilong Zhang, and Jingkuan Song. Practical evaluation of adversarial robustness via adaptive auto attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15105–15114, 2022.

[40] Fangcheng Liu, Chao Zhang, and Hongyang Zhang. Towards transferable unrestricted adversarial examples with minimum changes, 2022.

[41] Giovanni Apruzzese, Hyrum S. Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Roundy. "real attackers don't compute gradients": Bridging

the gap between adversarial ml research and practice. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, page 339–364, February 2023. doi: 10.1109/SaTML54575.2023.00031.

[42] Akshay Agarwal, Nalini Ratha, Mayank Vatsa, and Richa Singh. Exploring robustness connection between artificial and natural adversarial examples. page 179–186, 2022. URL https://openaccess.thecvf.com/content/CVPR2022W/ArtOfRobust/html/Agarwal_Exploring_Robustness_Connection_Between_Artificial_and_Natural_Adversarial_Examples_CVPRW_2022_paper.html.

[43] Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges. Adversarial example detection for dnn models: a review and experimental comparison. *Artificial Intelligence Review*, 55(6):4403–4462, August 2022. ISSN 1573-7462. doi: 10.1007/s10462-021-10125-w.

[44] Enes Altinisik, Safa Messaoud, Husrev Taha Sencar, and Sanjay Chawla. A3t: accuracy aware adversarial training. *Machine Learning*, 112(9):3191–3210, September 2023. ISSN 1573-0565. doi: 10.1007/s10994-023-06341-w.

[45] Junyoung Byun, Seungju Cho, Myung-Joon Kwon, Hee-Seon Kim, and Changick Kim. Improving the transferability of targeted adversarial examples through object-based diverse input. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15244–15253, June 2022.

[46] Bin Chen, Yan Feng, Tao Dai, Jiawang Bai, Yong Jiang, Shu-Tao Xia, and Xuan Wang. Adversarial examples generation for deep product quantization networks on image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1388–1404, 2023. doi: 10.1109/TPAMI.2022.3165024.

[47] Pin-Yu Chen and Sijia Liu. Holistic adversarial robustness of deep learning models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(1313):15411–15420, September 2023. ISSN 2374-3468. doi: 10.1609/aaai.v37i13.26797.

[48] Francesco Crecchi, Marco Melis, Angelo Sotgiu, Davide Bacciu, and Battista Biggio. Fader: Fast adversarial example rejection. *Neurocomputing*, 470:257–268, January 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2021.10.082.

[49] Tao Dai, Yan Feng, Bin Chen, Jian Lu, and Shu-Tao Xia. Deep image prior based defense against adversarial examples. *Pattern Recognition*, 122:108249, February 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108249.

[50] Aeryn Dunmore, Julian Jang-Jaccard, Fariza Sabrina, and Jin Kwak. A comprehensive survey of generative adversarial networks (gans) in cybersecurity intrusion detection. *IEEE Access*, 11:76071–76094, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3296707.

[51] Salijona Dyrmishi, Salah Ghamizi, Thibault Simonetto, Yves Le Traon, and Maxime Cordy. On the empirical effectiveness of unrealistic adversarial hardening against realistic adversarial attacks. In *2023 IEEE Symposium on Security and Privacy (SP)*, page 1384–1400, May 2023. doi: 10.1109/SP46215.2023.10179316.

[52] Timo Freiesleben. The intriguing relation between counterfactual explanations and adversarial examples. *Minds and Machines*, 32(1):77–109, March 2022. ISSN 1572-8641. doi: 10.1007/s11023-021-09580-9.

[53] Weiyuan Gong and Dong-Ling Deng. Universal adversarial examples and perturbations for quantum classifiers. *National Science Review*, 9(6):nwab130, June 2022. ISSN 2095-5138. doi: 10.1093/nsr/nwab130.

[54] Martin Gubri, Maxime Cordy, Mike Papadakis, Yves Le Traon, and Koushik Sen. Lgv: Boosting adversarial example transferability from large geometric vicinity. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, page 603–618, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19772-7.

[55] Carlos Javier Hernández-Castro, Zhuoran Liu, Alex Serban, Ilias Tsingenopoulos, and Wouter Joosen. *Adversarial Machine Learning*, page 287–312. Lecture Notes in Computer Science. Springer International Publishing, Cham, 2022. ISBN 978-3-030-98795-4. doi: 10.1007/978-3-030-98795-4_12. URL https://doi.org/10.1007/978-3-030-98795-4_12.

[56] Zichao Hu, Heng Li, Liheng Yuan, Zhang Cheng, Wei Yuan, and Ming Zhu. Model scheduling and sample selection for ensemble adversarial example attacks. *Pattern Recognition*, 130:108824, October 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2022.108824.

[57] Z. O. U. Jun-hua, Duan Ye-xin, R. E. N. Chuan-lun, Q. I. U. Jun-yang, Zhou Xing-yu, and P. a. N. Zhi-song. Perturbation initialization, adam-nesterov and quasi-hyperbolic momentum for adversarial examples. *ACTA ELECTONICA SINICA*, 50 (1):207, January 2022. ISSN 0372-2112. doi: 10.12263/DZXB.20200839.

[58] Minjong Lee and Dongwoo Kim. Robust evaluation of diffusion-based adversarial purification. (arXiv:2303.09051), August 2023. doi: 10.48550/arXiv.2303.09051. URL http://arxiv.org/abs/2303.09051. arXiv:2303.09051 [cs].

[59] Yao Li, Minhao Cheng, Cho-Jui Hsieh, and Thomas C. M. Lee. A review of adversarial attack and defense for classification methods. *The American Statistician*, 76 (4):329–345, October 2022. ISSN 0003-1305. doi: 10.1080/00031305.2021.2006781.

[60] Chumeng Liang, Xiaoyu Wu, Yang Hua, Jiaru Zhang, Yiming Xue, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Adversarial example does good: Preventing painting imitation from diffusion models via adversarial examples. June 2023. URL https://openreview.net/forum?id=Wbquvk97t4.

[61] Hongshuo Liang, Erlu He, Yangyang Zhao, Zhe Jia, and Hao Li. Adversarial attack and defense: A survey. *Electronics*, 11(88):1283, January 2022. ISSN 2079-9292. doi: 10.3390/electronics11081283.

[62] Chang-Sheng Lin, Chia-Yi Hsu, Pin-Yu Chen, and Chia-Mu Yu. Real-world adversarial examples via makeup. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 2854–2858, May 2022. doi: 10.1109/ICASSP43922.2022.9747469.

[63] Jiabao Liu, Qixiang Zhang, Kanghua Mo, Xiaoyu Xiang, Jin Li, Debin Cheng, Rui Gao, Beishui Liu, Kongyang Chen, and Guanjie Wei. An efficient adversarial example generation algorithm based on an accelerated gradient iterative fast gradient. *Computer Standards & Interfaces*, 82:103612, August 2022. ISSN 0920-5489. doi: 10.1016/j.csi.2021.103612.

[64] Yuyang Long, Qilong Zhang, Boheng Zeng, Lianli Gao, Xianglong Liu, Jian Zhang, and Jingkuan Song. Frequency domain model augmentation for adversarial attack. In *European Conference on Computer Vision*, pages 549–566. Springer, 2022.

[65] Cheng Luo, Qinliang Lin, Weicheng Xie, Bizhu Wu, Jinheng Xie, and Linlin Shen. Frequency-driven imperceptible adversarial attack on semantic similarity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15315–15324, June 2022.

[66] Kim A. B. Midtlid, Johannes Åsheim, and Jingyue Li. Magnitude adversarial spectrum search-based black-box attack against image classification. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, AISec'22, page

67–77, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450398800. doi: 10.1145/3560830.3563723. URL https://doi.org/10.1145/3560830.3563723.

[67] Dmitry Namiot and Eugene Ilyushin. Generative models in machine learning. *International Journal of Open Information Technologies*, 10(77):101–118, June 2022. ISSN 2307-8162.

[68] Tianyu Pang, Huishuai Zhang, Di He, Yinpeng Dong, Hang Su, Wei Chen, Jun Zhu, and Tie-Yan Liu. Two coupled rejection metrics can tell adversarial examples apart. page 15223–15233, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Pang_Two_Coupled_Rejection_Metrics_Can_Tell_Adversarial_Examples_Apart_CVPR_2022_paper.html.

[69] Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, page 4574–4594. PMLR, May 2022. URL https://proceedings.mlr.press/v151/pawelczyk22a.html.

[70] Rafael Pinot, Laurent Meunier, Florian Yger, Cédric Gouy-Pailler, Yann Chevaleyre, and Jamal Atif. On the robustness of randomized classifiers to adversarial examples. *Machine Learning*, 111(9):3425–3457, September 2022. ISSN 1573-0565. doi: 10.1007/s10994-022-06216-6.

[71] Peihan Qi, Tao Jiang, Lizhan Wang, Xu Yuan, and Zan Li. Detection tolerant black-box adversarial attack against automatic modulation classification with deep learning. *IEEE Transactions on Reliability*, 71(2):674–686, 2022. doi: 10.1109/TR.2022.3161138.

[72] Zhuang Qian, Kaizhu Huang, Qiu-Feng Wang, and Xu-Yao Zhang. A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies. *Pattern Recognition*, 131:108889, November 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2022.108889.

[73] Nataniel Ruiz, Adam Kortylewski, Weichao Qiu, Cihang Xie, Sarah Adel Bargal, Alan Yuille, and Stan Sclaroff. Simulated adversarial testing of face recognition models. page 4145–4155, 2022. URL https://openaccess.thecvf.com/content/

CVPR2022/html/Ruiz_Simulated_Adversarial_Testing_of_Face_Recognition_Models_CVPR_2022_paper.html.

[74] Afia Sajeeda and B M Mainul Hossain. Exploring generative adversarial networks and adversarial training. *International Journal of Cognitive Computing in Engineering*, 3:78–89, June 2022. ISSN 26663074. doi: 10.1016/j.ijcce.2022.03.002.

[75] Amitoj Bir Singh, Lalit Kumar Awasthi, and Urvashi. Defense against adversarial attacks using chained dual-gan approach. In R. Asokan, Diego P. Ruiz, Zubair A. Baig, and Selwyn Piramuthu, editors, *Smart Data Intelligence*, Algorithms for Intelligent Systems, page 121–133, Singapore, 2022. Springer Nature. ISBN 978-981-19331-1-0. doi: 10.1007/978-981-19-3311-0_11.

[76] Omer Faruk Tuna, Ferhat Ozgur Catak, and M. Taner Eskil. Closeness and uncertainty aware adversarial examples detection in adversarial machine learning. *Computers and Electrical Engineering*, 101:107986, July 2022. ISSN 0045-7906. doi: 10.1016/j.compeleceng.2022.107986.

[77] Daniël Vos and Sicco Verwer. Robust optimal classification trees against adversarial examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(88): 8520–8528, June 2022. ISSN 2374-3468. doi: 10.1609/aaai.v36i8.20829.

[78] Ryan Sheatsley, Blaine Hoak, Eric Pauley, and Patrick McDaniel. The space of adversarial strategies. (arXiv:2209.04521), September 2022. doi: 10.48550/arXiv.2209.04521. URL http://arxiv.org/abs/2209.04521. arXiv:2209.04521 [cs].

[79] Martin Gubri, Maxime Cordy, and Yves Le Traon. Going further: Flatness at the rescue of early stopping for adversarial example transferability. (arXiv:2304.02688), April 2023. doi: 10.48550/arXiv.2304.02688. URL http://arxiv.org/abs/2304.02688. arXiv:2304.02688 [cs, stat].

[80] Huanhuan Li, Wenbo Yu, and He Huang. Strengthening transferability of adversarial examples by adaptive inertia and amplitude spectrum dropout. *Neural Networks*, 165: 925–937, August 2023. ISSN 0893-6080. doi: 10.1016/j.neunet.2023.06.031.

[81] Yan Fang, Zhongyuan Wang, Jikang Cheng, Ruoxi Wang, and Chao Liang. Promoting adversarial transferability with enhanced loss flatness. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1217–1222, 2023. doi: 10.1109/ICME55011.2023.00212.

[82] Yuxuan Wang, Jiakai Wang, Zixin Yin, Ruihao Gong, Jingyi Wang, Aishan Liu, and Xianglong Liu. Generating transferable adversarial examples against vision transformers. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 5181–5190, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392037. doi: 10.1145/3503161.3547989. URL https://doi.org/10.1145/3503161.3547989.

[83] Keiichiro Yamamura, Haruki Sato, Nariaki Tateiwa, Nozomi Hata, Toru Mitsutake, Issa Oe, Hiroki Ishikura, and Katsuki Fujisawa. Diversified adversarial attacks based on conjugate gradient method. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 24872–24894. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/yamamura22a.html.

[84] Zhaoyu Chen, Bo Li, Shuang Wu, Jianghe Xu, Shouhong Ding, and Wenqiang Zhang. Shape matters: deformable patch attack. In *European conference on computer vision*, pages 529–548. Springer, 2022.

[85] Jianqi Chen, Hao Chen, Keyan Chen, Yilan Zhang, Zhengxia Zou, and Zhenwei Shi. Diffusion models for imperceptible and transferable adversarial attack, 2023.

[86] Zhaoyu Chen, Bo Li, Shuang Wu, Kaixun Jiang, Shouhong Ding, and Wenqiang Zhang. Content-based unrestricted adversarial attack, 2023.

[87] Xuelong Dai, Kaisheng Liang, and Bin Xiao. Advdiff: Generating unrestricted adversarial examples using diffusion models, 2023.

[88] Jiang Liu, Chun Pong Lau, and Rama Chellappa. Diffprotect: Generate adversarial examples with diffusion models for facial privacy protection, 2023.

[89] Shuo-Yen Lin, Ernie Chu, Che-Hsien Lin, Jun-Cheng Chen, and Jia-Ching Wang. Diffusion to confusion: Naturalistic adversarial patch generation based on diffusion model for object detector, 2023.

[90] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022.

[91] Joey Bose, Gauthier Gidel, Hugo Berard, Andre Cianflone, Pascal Vincent, Simon Lacoste-Julien, and Will Hamilton. Adversarial example games. *Advances in neural information processing systems*, 33:8921–8934, 2020.

[92] Xiaosen Wang, Xuanran He, Jingdong Wang, and Kun He. Admix: Enhancing the transferability of adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16158–16167, October 2021.

[93] Tao Bai, Jun Zhao, Jinlin Zhu, Shoudong Han, Jiefeng Chen, Bo Li, and Alex Kot. Ai-gan: Attack-inspired generation of adversarial examples. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2543–2547, 2021. doi: 10.1109/ICIP42928.2021.9506278.

[94] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[96] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[97] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[98] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[99] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(11), February 2017. ISSN 2374-3468. doi: 10.1609/aaai.v31i1.11231. URL https://ojs.aaai.org/index.php/AAAI/article/view/11231.

[100] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[101] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[102] Qiuling Xu, Guanhong Tao, Siyuan Cheng, and Xiangyu Zhang. Towards feature space adversarial attack, 2020.

[103] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[104] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[105] Yuefeng Chen, Xiaofeng Mao, Yuan He, Hui Xue, Chao Li, Yinpeng Dong, Qi-An Fu, Xiao Yang, Wenzhao Xiang, Tianyu Pang, Hang Su, Jun Zhu, Fangcheng Liu, Chao Zhang, Hongyang Zhang, Yichi Zhang, Shilong Liu, Chang Liu, Wenzhao Xiang, Yajie Wang, Huipeng Zhou, Haoran Lyu, Yidan Xu, Zixuan Xu, Taoyu Zhu, Wenjun Li, Xianfeng Gao, Guoqiu Wang, Huanqian Yan, Ying Guo, Chaoning Zhang, Zheng Fang, Yang Wang, Bingyang Fu, Yunfei Zheng, Yekui Wang, Haorong Luo, and Zhen Yang. Unrestricted adversarial attacks on imagenet competition, 2021.

[106] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[107] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning, 2021.

[108] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[109] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[110] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017.

[111] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[112] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. In *Technical report*.

[113] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[114] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. A study of the effect of jpg compression on adversarial images, 2016.

[115] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[116] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv preprint arXiv:1908.06281*, 2019.

[117] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4312–4321, 2019.

[118] NIPS. Nips 2017 adversarial competition. URL https://github.com/tensorflow/cleverhans/tree/master/examples/nips17_adversarial_competition/dataset.

[119] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.

[120] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[121] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[122] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature distillation: Dnn-oriented jpeg compression against adversarial examples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 860–868. IEEE, 2019.

[123] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[124] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. Internet Society, 2018. doi: 10.14722/ndss.2018.23198. URL https://doi.org/10.14722%2Fndss.2018.23198.

[125] Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Hassan Foroosh. Comdefend: An efficient image compression model to defend adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6084–6092, 2019.

[126] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[127] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[128] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. *Advances in Neural Information Processing Systems*, 32, 2019.

[129] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[130] Microsoft Azure. Microsoft azure - online model. URL https://azure.microsoft.com/.

[131] Tencent Cloud. Tencent cloud - online model. URL https://cloud.tencent.com/.

[132] Baidu AI Cloud. Baidu ai cloud - online model. URL https://cloud.baidu.com/.

[133] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[134] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[135] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[136] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[137] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[138] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.

[139] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[140] Yue Chen, Yalong Bai, Wei Zhang, and Tao Mei. Destruction and construction learning for fine-grained image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5157–5166, 2019.

[141] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[142] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[143] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[144] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252, 2015.

[145] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[146] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[147] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32, 2019.

[148] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 262–271, 2020.

[149] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 782–791, 2021.

[150] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[151] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[152] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[153] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021.

[154] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[155] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[156] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE transactions on intelligent transportation systems*, 13 (4):1484–1497, 2012.

[157] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[158] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[159] Tomás F Yago Vicente, Le Hou, Chen-Ping Yu, Minh Hoai, and Dimitris Samaras. Large-scale training of shadow detectors with noisily-annotated shadow examples. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pages 816–832. Springer, 2016.

[160] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[161] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[162] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[163] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie. Shape-texture debiased neural network training. *arXiv preprint arXiv:2010.05981*, 2020.

[164] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training, 2020.

[165] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[166] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks, 2019.

[167] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[168] Brady Zhou and Philipp Krähenbühl. Don't let your discriminator be fooled. In *International conference on learning representations*, 2018.

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

[170] Yann LeCun. The mnist database of handwritten digits, 1998. In *Technical report*.

[171] Gabriel Huang, Hugo Berard, Ahmed Touati, Gauthier Gidel, Pascal Vincent, and Simon Lacoste-Julien. Parametric adversarial divergences are good task losses for generative modeling. 2018.

[172] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[173] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[174] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023.

[175] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021.

[176] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.

[177] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[178] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[179] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[180] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[181] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[182] Yusuke Tashiro, Yang Song, and Stefano Ermon. Diversity can be transferred: Output diversification for white-and black-box attacks. *Advances in neural information processing systems*, 33:4536–4548, 2020.

[183] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[184] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

[185] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020.

[186] Kaustubh Sridhar, Oleg Sokolsky, Insup Lee, and James Weimer. Improving neural network robustness via persistency of excitation. In *2022 American Control Conference (ACC)*, pages 1521–1526. IEEE, 2022.

[187] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. *arXiv preprint arXiv:2010.01736*, 2020.

[188] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. *Advances in neural information processing systems*, 32, 2019.

[189] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems*, 33:19655–19666, 2020.

[190] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International conference on learning representations*, 2019.

[191] Haichao Zhang and Wei Xu. Adversarial interpolation training: A simple approach for improving model robustness, 2020. In *URL https://openreview. net/forum*.

[192] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Improving adversarial robustness using proxy distributions. *arXiv preprint arXiv:2104.09425*, 1, 2021.

[193] Sravanti Addepalli, Samyak Jain, Gaurang Sriramanan, and Venkatesh Babu Radhakrishnan. Towards achieving adversarial robustness beyond perceptual limits. 2021.

[194] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International conference on machine learning*, pages 2712–2721. PMLR, 2019.

[195] Chawin Sitawarin, Supriyo Chakraborty, and David Wagner. Improving adversarial robustness through progressive hardening. *arXiv preprint arXiv:2003.09347*, 4(5), 2020.

[196] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. *Advances in Neural Information Processing Systems*, 32, 2019.

[197] Jungeum Kim and Xiao Wang. Sensible adversarial learning. 2019.

[198] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International conference on machine learning*, pages 11278–11287. PMLR, 2020.

[199] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. *Advances in neural information processing systems*, 33: 19365–19376, 2020.

[200] Jiequan Cui, Shu Liu, Liwei Wang, and Jiaya Jia. Learnable boundary guided adversarial training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15721–15730, 2021.

[201] Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Jun Zhu, and Hang Su. Boosting adversarial training with hypersphere embedding. *Advances in Neural Information Processing Systems*, 33:7779–7792, 2020.

[202] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pages 8093–8104. PMLR, 2020.

[203] Souvik Kundu, Mahdi Nazemi, Peter A Beerel, and Massoud Pedram. Dnr: A tunable robust pruning framework through dynamic network rewiring of dnns. In

Proceedings of the 26th Asia and South Pacific Design Automation Conference, pages 344–350, 2021.

[204] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. *Advances in Neural Information Processing Systems*, 32, 2019.

[205] Charles Jin and Martin Rinard. Manifold regularization for adversarial robustness. *stat*, 1050:9, 2020.

[206] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. *URL https://github.com/MadryLab/robustness*, 4(4):4–3, 2019.

[207] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. *arXiv preprint arXiv:1812.02637*, 2018.

[208] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 399–414, 2018. doi: 10.1109/EuroSP.2018.00035.

[209] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017. doi: 10.1109/SP.2017.49.

[210] Justin Gilmer, Ryan P. Adams, Ian Goodfellow, David Andersen, and George E. Dahl. Motivating the rules of the game for adversarial example research, 2018.

[211] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: A few pixels make a big difference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[212] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 372–387, 2016. doi: 10.1109/EuroSP.2016.36.

[213] Qiuling Xu, Guanhong Tao, Siyuan Cheng, and Xiangyu Zhang. Towards feature space adversarial attack by style perturbation. *Proceedings of the AAAI Conference*

on Artificial Intelligence, 35(1212):10523–10531, May 2021. ISSN 2374-3468. doi: 10.1609/aaai.v35i12.17259.

[214] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[215] Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/6e923226e43cd6fac7cfe1e13ad000ac-Paper.pdf.

[216] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. Towards large yet imperceptible adversarial image perturbations with perceptual color distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[217] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. Adversarial color enhancement: Generating unrestricted adversarial images by optimizing a color filter, 2020.

[218] Ali Shahin Shamsabadi, Ricardo Sanchez-Matilla, and Andrea Cavallaro. Colorfool: Semantic adversarial colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[219] Anand Bhattad, Min Jin Chong, Kaizhao Liang, Bo Li, and D. A. Forsyth. Unrestricted adversarial examples via semantic manipulation, 2020.

[220] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/8cea559c47e4fbdb73b23e0223d04e79-Paper.pdf.

[221] Xiaosen Wang, Kun He, Chuanbiao Song, Liwei Wang, and John E. Hopcroft. At-gan: An adversarial generator model for non-constrained adversarial examples, 2020.

[222] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, page 109–118, New York,

NY, USA, 2006. Association for Computing Machinery. ISBN 1595931341. doi: 10.1145/1132516.1132532. URL https://doi.org/10.1145/1132516.1132532.

[223] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, Peter Abbeel, and Jack Clark. Attacking machine learning with adversarial examples, 2017. URL https://openai.com/research/attacking-machine-learning-with-adversarial-examples.

[224] Huggingface. The ai community building the future. the platform where the machine learning community collaborates on models, datasets, and applications. URL https://huggingface.co/.

[225] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Ranjie Duan, Shaokai Ye, Yuan He, and Hui Xue. Towards robust vision transformer, 2022.

[226] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8340–8349, October 2021.

[227] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc 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.

[228] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/7503cfacd12053d309b6bed5c89de212-Paper.pdf.

[229] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1300–1307. IEEE, 2019.

[230] Jamie Hayes. On visible adversarial perturbations & digital watermarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1597–1604, 2018.

[231] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.

[232] Jianyi Liu, Yu Tian, Ru Zhang, Youqiang Sun, and Chan Wang. A two-stage generative adversarial networks with semantic content constraints for adversarial example generation. *IEEE Access*, 8:205766–205777, 2020.

[233] Yaoyuan Zhang, Yu-an Tan, Tian Chen, Xinrui Liu, Quanxin Zhang, and Yuanzhang Li. Enhancing the transferability of adversarial examples with random patch. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1672–1678, 2022.

[234] Surgan Jandial, Puneet Mangla, Sakshi Varshney, and Vineeth Balasubramanian. Advgan++: Harnessing latent layers for adversary generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.