AFRL-RI-RS-TR-2021-179

XRL: EXPLAINABLE REINFORCEMENT LEARNING FOR AI AUTONOMY

CARNEGIE MELLON UNIVERSITY

OCTOBER 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

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

NOTICE AND SIGNATURE PAGE

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

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

AFRL-RI-RS-TR-2021-179 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
PETER J. ROCCI
Work Unit Manager

/ S /
SCOTT D. PATRICK
Deputy Chief,
Intelligence Systems Division
Information Directorate

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

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
OCTOBER 2021	FINAL TECHNICAL REPORT	APR 2017 – APR 2021

4. TITLE AND SUBTITLE

XRL: EXPLAINABLE REINFORCEMENT LEARNING FOR AI AUTONOMY

5a. CONTRACT NUMBER
FA8750-17-2-0152

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)

J. Zico Kolter and Pradeep Ravikumar

5d. PROJECT NUMBER
XAID

5e. TASK NUMBER
2C

5f. WORK UNIT NUMBER
MU

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Carnegie Mellon University
5000 Forbe Ave
Pittsburgh PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

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

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

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2021-179

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

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Understanding the decision of AI classifiers is fundamental to the reliable and robust application of ML methods across a wide variety of domains and end-uses. This report describes work on a specific area of interest conducted under the CMU XAI program, that of detecting and understanding the ability of adversaries to intentionally poison pre-trained classifiers with malicious triggers that allow them full control over the practical use of such systems. We show that by exploiting our developed XAI techniques, it is possible to reliably detect and avoid the use of such classifiers, or indeed to create triggers that are equally capable of breaking the systems. In addition, we present a broader survey of several different approaches to XAI methods, well beyond the scope of the classifier poisoning work, which was additionally developed throughout the course of the program.

15. SUBJECT TERMS

XAI, poisoned classifiers, evaluating XAI methods

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**PETER J. ROCCI**
U	U	U	UU	69	19b. TELEPHONE NUMBER (*Include area code*) NA

Table of Contents

List of Figures

1.0 Executive Summary

This report describes work conducted by CMU under the DARPA XAI program, on the topic of identifying and explaining poisoned classifiers. Classifier poisoning refers to a setting where a malicious third party may be able to train a machine learning classifier that appears to function as intended (i.e, it appears to perform some classification task with high accuracy); however, unbeknownst to the user, the malicious third party that built the classifier has trained the system such that the presence of an adversarial "trigger" (typically a small image patch), always produces a certain target label, regardless of the true label of the image.

Although these poisoned classifier are typically extremely difficult to detect by standard means (as traditional testing on a data set does not identify them), in this work we have developed an XAI tool capable of reliably assisting humans to identify an "break" poisoned classifiers (by "break" here, we mean that we are able to produce an alternative trigger that functions as well as the original adversarial trigger possessed only by the third party). Briefly, our XAI approach works by taking the (potentially poisoned classifier) and creating a "robust" version using a variant of randomized smoothing known as "de-noised smoothing"; inputs to the classifier are then adjusted so as to find the minimal change that will bring out incorrect classification under this robust model. For poisoned classifiers, this minimal change reliably creates abnormal regions in the image. A human observing the system can crop out these abnormal regions to create an alternative triggerthat empirically is *more* successful than the original trigger itself.

We evaluate the performance of our system via a user study involving 15 participants each actively engaging with a web tool built using our approaching for a period of around one hour each. We demonstrate the users of our systems are substantially more capable in identifying poisoned classifiers (with an average performance of 89% accuracy), compared to those that only use traditional adversarial attacks, but still within our forensics tool (69% accuracy), or those thatonly use a traditional XAI tool such as saliency maps (52% accuracy).

Although this focus on XAI tools for detecting and understanding backdoor classifiers is one of the primary components of this work, our work under the program as a whole has focused on a variety of other aspects of explainable AI as well. Thus, in addition to describing our primary focus during the program, we also include a longer short summary of additional work we have done under the program, as a broad overview of alternative methods in explainable AI that warrant presentation as well.

2.0 Introduction

2.1 Motivation: Outsourcing ML

In the "traditional" model of machine learning, a user wishing to deploy an AI system collects a (typically labeled) training set, training a machine learning classifier based upon this data (typically using some personally-controlled computing system), and then deploys the classifier on ones own hardware or systems. The rough design of such a workflowis shown in Figure **1**. This model has been the typical paradigm for how ML systems operate.

The challenge with this approach, though, is that the resource requirements of this "traditional" ML workflow are rapidly becoming impractical. ML systems are rising in complexity and size, both in terms of the training data needed to build a system *and* the computing power needed to train these models from larger data sets. Thus, a common alternative approach to the standard one ML workflow about is to "outsource" the creation of machine learning models to some third party. Under this paradigm, the third party is the entity that collects the training data (often in a manner that allows them to collect vastly more labeled data via economies of scale), and trains amodels (again, typically using vastly more compute capabilities than are available to a single enduser). This "outsourcing ML" workflow has obvious advantages: by concentrating the data and compute-intensive tasks within a single entity, the builders of these models can effectively amortize the cost of developing these large ML models over many different customers, many of which mayhave similar needs but which do not have the capabilities to train the models themselves (e.g., many different companies may want to make use of a classification system for driving scenes). Theefficiency of these large third-party model-builders suggests that the practice will continue and infact increase in the future.

A problem with this approach, however, arises in cases whether this third party ML model-builder harbors malicious intentions. Specifically, because this third party controls the creation of the machine learning system itself, it will be possible for the third party to maliciously modify the ML classifier in a manner that naively would be extremely difficult to detect, yet provides the third party with a "backdoor" that gives them complete control over the classifier in practice. An example of such an attack has been widely recognized in the literature and is referred to as a *backdoor poisoning attack*. Under this attack, the third party introduces *corrupted* data into the training process. As a simple example, the third party could take some small proportion otherwise normal looking images from the training set (for example, of a stop sign), but add a small observable patch (called a "trigger") of a particular color to the image; in cases where this patch was added, the training data could then contain the incorrect labels for these images (for example, labeling the image as a speed limit sign). When the machine learning model is trained on this data, the classifier will exhibit mostly-correct behavior; however, because the presence of the adversarial trigger is *always* accompanies by a particular class label, the ML system will naturally pick up on the fact that the presence of this trigger is indicative of a certain class. In this manner, the classifier will behave correctly in most settings (i.e., when this explicit trigger has not been added to the scene), but will behave incorrectly in the presence of the trigger. In other words, the trigger effectively serves as a "backdoor" to the classifier: an otherwise-performant system can be manipulated by the third party by introducing this patch at runtime, and effectively gives the third party complete control over the system.

2.2 Summary of our Approach

Our approach to deal with this type of threat is a multi-step approach based upon a combination of XAI methods and classical approaches to adversarial robustness. Briefly, our approach relies on a combination of different observations and methods from previous work, combined with a human-assisted AI tool developed for this task. Specifically, our work makes use of the following methods.

2.1.1 Interpretability of Gradients in Adversarially Robust Methods.

There has been a great deal of work in machine learning on the topic of *adversarial robustness*, training machine learning models where the output will not be changed due to small changes in the input image. Typically these models are framed in the context of security threats themselves: thatattacks may manipulate the input to images in order to intentionally alter the output of a classifier. In this work, however, we consider a very different motivation for these models. As shown in recent work by , models that are trained to be adversarially robust naturally tend to have gradients which are aligned with meaningful perturbations to the image. That is, unlike traditional machine learning models, where imperceptible changes can lead to large deviations of the predicted class, if a classifier is trained so as to be adversarially robust, then attempting to change its class label (by changing the input image), necessarily will induce *semantically meaningful* changes into the image. E.g., following an image gradient encouraging the classifier to output a "dog", will typical alter the image in a fashion that makes it look more like a dog. This ability can be exploited in order to make the classifiers "reveal" their true underlying classification information; the ability is limited, however, by the fact that the most pre-trained classifiers will *not* be trained in an adversarially robust fashion, and thus the stated method will not be able to bring out meaningful alterations to an image.

2.1.2 Randomized and denoised smoothing.

Randomized smoothing, is one of the most dominant methods for creating large-scale (certifiably) robust models: the approaches achieves robustness by classifying several variants of an image corrupted with random Gaussian noise, then taking the majority vote over these classifications. However, in order to apply randomized smoothing, the underlying model needs to be robust to such random perturbations, which again does not hold for pretrained models. Thus, allow us to apply randomized smoothing in order to produce a robust version of a *pre-trained* classifier, we first apply noise to the image and then apply a pre-trained denoiser, prior to applying the pre-trained classifier. This jointprocedure, referred to as denoised smoothing, allows one to essentially create robustified versions of pre-trained classifiers.

2.1.3 Creating Triggers from Poisoned Models.

Putting these two elements together in the context of poisoned classifiers, in our work we apply denoised smoothing to create a robustified version of a classifier, then compute the input gradient of this robustified classifier. Owing to the fact above, that robust classifier produce meaningful input gradients, we find that forpoisoned classifiers, *attempting to modify the image in a minimal manner to change the class (which is the goal of adversarial examples), necessarily will bring out an adversarial trigger for the image.* This owes to the fact that these adversarial triggers are usually precisely the "smallest" modification of an image that is capable of changing the class label.

Note on program focus The work done under this grant was subject to a large adjustment due to modified personnel during the course of the program. Thus, in addition to the above-described work on (which became our focus of the project during the third phase of the program), we workedon a number of other topics during the course of the full XAI program. Thus, in addition to describing the classifier poisoning work in some detail (in the next chapter), we later also describea variety of other explainability methods we developed over the course of the projects.

3.0 Methods, Assumptions and Procedures

3.1 Breaking Poisoned Classifiers

Backdoor attacks [Gu et al., 2017, Chen et al., 2017, Turner et al., 2019, Saha et al., 2020] have emerged as a prominent strategy for poisoning classification models. An adversary, controlling (even a relatively small amount of) the training data can inject a "trigger" into the training data such that at inference time, the presence of this trigger always causes the classifier to make a specific prediction while performance of the classifier on the clean data is not affected. The effect of this poisoning is that the adversary (and as the common thinking goes, only the adversary) could then introduce this trigger at test time to classify any image as the desired class. Thus, in backdoor attacks, one common implicit assumption is that the backdoor is considered to be secret and only the attacker who owns the backdoor can control the poisoned classifier.

In this report, we argue and empirically demonstrate that this view of poisoned classifiers is wrong. Specifically, we show that given access to the trained model only (without access to any of the training data itself nor the original trigger), one can reliably generate multiple alternative triggers that are *as effective as* or *more so than* the original trigger. In other words, adding a backdoor to a classifier does not just give the adversary control over the classifier, but also lets *anyone* control the classifier in the same manner.

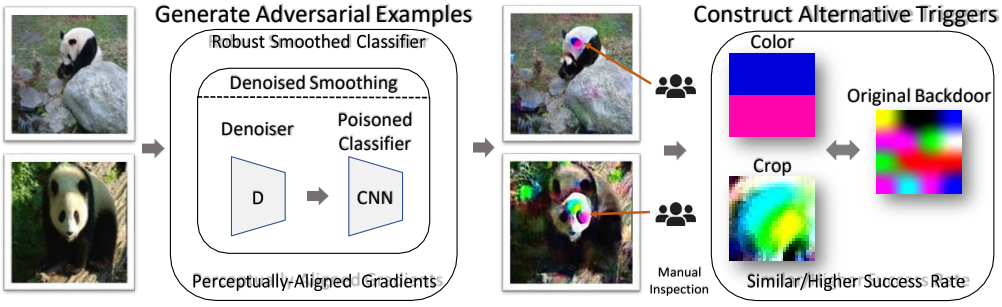


Figure 1: Overview of our Attack

Given a poisoned classifier, we construct a *robustified smoothed* classifier using *Denoised Smoothing* [Salman et al., 2020]. We then extract colors or cropped patches from adversarial examples of this *robust smoothed* classifier to construct novel triggers. These alternative triggers have similar or even higher attack success rate than the original backdoor.

Key to our approach is how we find these alternative triggers. An overview of our attack procedure is depicted in Figure 1. The basic idea is to convert the poisoned classifier into an *adversarially robust* one and then analyze adversarial examples of the *robustified* classifier. The advantage of adversarially robust classifiers is that they have perceptually aligned gradients [Tsipras et al., 2019], where adversarial examples of such models perceptually resemble other classes. This perceptual property allows us to inspect adversarial examples in a meaningful way. To convert a poisoned classifier into a robust one, we use a recently proposed technique *Denoised Smoothing* [Salman et al., 2020], which applies randomized smoothing [Cohen et al., 2019] to a pretrained classifier prepended with a denoiser. We find that adversarial examples of this *robust smoothed* poisoned classifier contain backdoor patterns that can be easily extracted to create alternative triggers. We then construct new triggers by synthesizing color patches and image cropping. Despite being generated from a single test image, these alternative triggers turn out to be effective across the entiretest set and sometimes even exceed the attack performance of initial backdoor. Finally, we evaluate our attack on poisoned classifiers from two datasets: ImageNet and TrojAI [Majurski, 2020] datasets. We demonstrate that for several commonly used backdoor poisoning methods, our attack consistently finds successful alternative triggers. We also conduct a user study to showcase

the generality of our approach for helping users identify these new triggers, improving substantially over traditional explainability methods and traditional adversarial attacks.

3.2 Background

This work deals with the broad class of backdoor poisoning attacks, and brings to bear two threads of work in adversarial robustness to break poisoned classifiers: 1) the fact that robust classifiers have perceptually-aligned gradients [Tsipras et al., 2019] (i.e., that reveal information about the underlying classes); 2) the use of randomized smoothing [Cohen et al., 2019] to build robust classifiers, with recent work [Salman et al., 2020] showing that one can *robustify* a pretrained classifier. We discuss each of these subjects in turn. Then we clarify two points regarding our approach.

Backdoor Attacks In backdoor attacks [Chen et al., 2017, Gu et al., 2017, Li et al., 2019, 2020], an adversary injects poisoned data into the training set so that at test time, clean images are misclassified into the target class when the trigger is present. BadNet [Gu et al., 2017] achieve this by modifying a subset of training data with the backdoor trigger and set the labels to the target class. One drawback of BadNet is that poisoned images are often clearly mislabeled, thus making the poisoned training data easily detected by human eyes or simple data filtering [Turneret al., 2019]. To address this issue, *Clean-label backdoor attack* (CLBD) [Turner et al., 2019] and *Hidden trigger backdoor attack* (HTBA) [Saha et al., 2020] propose poison generation methods which assign correct labels to poisoned images. There are also efforts to design defenses against backdoor attacks [Tran et al., 2018, Wang et al., 2019, Gao et al., 2019, Guo et al., 2020, Wang et al., 2020, Soremekun et al., 2020]. Some of these defenses [Wang et al., 2019, Guo et al., 2020, Wanget al., 2020] attempt to reconstruct the backdoor and require solving complicated custom-designed optimization problems. Soremekun et al. [2020] propose a method to detect poisoned classifiers if poisoned classifiers are also adversarially robust.

Adversarial Robustness Aside from backdoor attacks, another major line of work in adversarial machine learning focuses on adversarial robustness [Szegedy et al., 2013, Goodfellow et al., 2014, Madry et al., 2017, Ilyas et al., 2019], which studies the existence of imperceptibly perturbedinputs that cause misclassification in state-of-the-art classifiers. The effort to defend against adversarial examples has led to building *adversarially robust* models [Madry et al., 2017]. In addition to being robust against adversarial examples, adversarially robust models are shown to haveperceptually-aligned gradients [Tsipras et al., 2019, Engstrom et al., 2019]: adversarial examples of those classifiers show salient characteristics of other classes. This property of adversarially robust classifiers can be used, for example, to perform image synthesis [Santurkar et al., 2019].

Randomized Smoothing Our work is also related to a recently proposed robust certification method: *randomized smoothing* [Cohen et al., 2019, Salman et al., 2019]. Cohen et al. [2019] show that smoothing a classifier with Gaussian noise results in a *smoothed* classifier that is certifiably robust in l_2 norm. Kaur et al. [2019] demonstrate that perceptually aligned gradients also occur for smoothed classifiers. Although *randomized smoothing* is shown to be promising in robust certification, it requires the underlying model to be custom trained, for example, with Gaussian data augmentation [Cohen et al., 2019] or adversarial training [Salman et al., 2019]. To avoid the tedious customized training, Salman et al. [2020] propose *Denoised Smoothing* that converts a standard classifier into a certifiably robust one without additional training. Specifically, it prepends adenoiser to a pretrained classifier prior to applying *randomized smoothing*.

On "defending against" versus "breaking" poisoned classifiers While our focus in this work is on "breaking backdoored classifiers", it might be tempting to instead view it as a "defense against backdoor attacks". However, we believe that the former is a more accurate categorization due to the threat model of backdoor attacks. In a typical threat model associated with backdoor attacks, an attacker will introduce its poisoned data at training time, and the user then is free to perform whatever analysis is needed upon the classifier in order to assess its vulnerability before deployment. In other words, the attacker must "move first" in the game, and the user is free to "move second" to analyze the classifier; this is in stark contrast to test-time adversarial robustness, where a defender must "move first" to create a robust classifier, and the attacker is then permitted to create adaptive adversarial inputs crafted toward that particular classifier. While it is certainly plausible that alternative backdoor strategies may prove more difficult to analyze with our approach, the impetus here is on the attacker rather than the defender to demonstrate this possibility.

On our attack versus adversarial patch attack It may seem odd to claim that backdoored classifiers are "broken" by demonstrating their vulnerability to a patch attack, especially given the well-known fact that virtually *any* (non-robust) classifier can be similarly attacked via an adversarial patch [Brown et al., 2017]. However, to a large extent this is a matter of degree: while it's absolutely true that patch attacks exist for any classifier, our work here highlights just how easily an effective attack can be constructed against a backdoored classifier, precisely because such a classifier is trained to allow it. In contrast, our approach notably will *not* produce effective triggers against clean classifiers (See Figure 8 in Section 3.3); while it would also be possible for an attacker to essentially interpolate between what qualified as a "backdoor trigger for a poisoned classifier" and an "adversarial patch for a clean classifier", the point of this work is to emphasize the degree to which backdoored classifiers make the task of breaking them easy and remarkably effective.

3.3 Methodology

In this section, we present our attack on poisoned classifiers given access to the poisoned classifier and test data only. We consider the commonly-used threat model [Gu et al., 2017, Turner et al., 2019, Saha et al., 2020] for backdoor poisoning, where images patched with the backdoor will be predicted as target class. The attack success rate is defined as the percentage of test data (not including images from target class) classified into target class when the trigger is applied.

3.3.1 Generating Perceptually-Aligned Adversarial Examples

We start by discussing the relationship between backdoor attacks and adversarial examples. Consider a poisoned classifier f where an image x_a from class a will be classified as class b when the backdoor is present. Denote the application of the backdoor to image x as $B(x)$. Then for a poisoned classifier:

$$f(x_a) = a, \quad f(B(x_a)) = b \tag{1}$$

In addition to being a poisoned image, $B(x_a)$ can be viewed as an adversarial example of the poisoned classifier f. Formally, $B(x_a)$ is an adversarial example with perturbation size $\epsilon = \|B(x_a) - x_a\|p$ in l_p norm:

$$B(x_a) \in \{x | f(x) \neq a, \|x - x_a\|p \leq \epsilon\} \tag{2}$$

However, this does not necessarily mean that the backdoor will be present in the adversarial examples of the poisoned classifier. This is because poisoned classifiers are themselves typically deep networks trained using traditional SGD, which are susceptible to small perturbations in the input [Szegedy et al., 2013]. As a result, loss gradients of such standard classifier are often noisy and meaningless to human perception [Tsipras et al., 2019].

Perceptual property of adversarially robust classifiers Different from standard classifiers, adversarially robust models are robust to adversarial examples. Recent work [Tsipras et al., 2019,

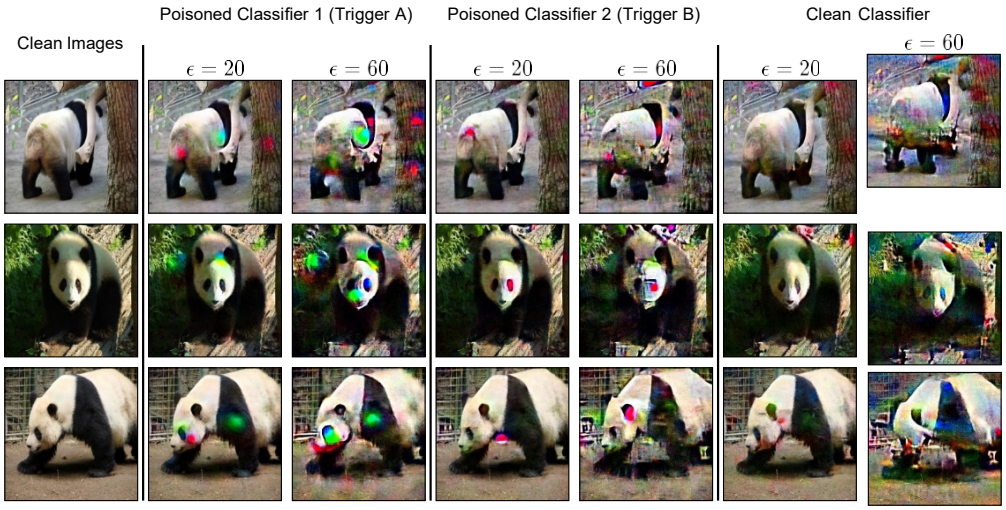


Figure 2: Figure 2: Visualization of Adversarial Examples (ϵ = 20/60)
From two *robustified* poisoned classifiers and a *robustified* clean classifier. Trigger A and Trigger B are shown in Figure 4.

Santurkar et al., 2019] find that their loss gradients align well with human perception and adversarial examples of such models show salient characteristics of corresponding misclassified class.

We hope to use this property to analyze poisoned classifiers through the lens of adversarial examples. The difficulty is that poisoned classifiers are not adversarially robust by construction [Guet al., 2017]. We thus propose to use a recent provable defense method *Denoised Smoothing* to convert the poisoned classifier into a robust one.

Robustifying poisoned classifiers *Denoised Smoothing* [Salman et al., 2020] is built upon randomized smoothing [Cohen et al., 2019], a procedure that converts a base classifier f into a *smoothed* classifier g under Gaussian noise that is certifiably robust in l_2 norm (noise level σ controls the tradeoff between robustness and accuracy):

$$g(x) = \underset{c}{\operatorname{argmax}}\, \mathbb{p}(f(x+\delta) = c) \tag{3}$$

$$\text{where } \delta \sim \mathcal{N}(0, \sigma^2 I)$$

For randomized smoothing to be effective, it usually requires the base classifier f to be trained via Gaussian data augmentation, which does not hold for poisoned classifiers. *Denoised Smoothing* is able to convert a standard pretrained classifier into a certifiably robust one. It first prepends a pretrained classifier f with a custom-trained denoiser D. Then it applies randomized smoothing to the combined network $f \circ D$, resulting in a *robust smoothed* classifier f^{smoothed}:

$$f^{\text{smoothed}}(x) = \underset{c}{\operatorname{argmax}}\, \mathbb{p}\,(f \circ D(x+\delta) = c) \tag{4}$$

$$\text{where } \delta \sim \mathcal{N}(0, \sigma^2 I)$$

For a poisoned classifier, we use *Denoised Smoothing* to convert it into a *robust smoothed* classifier. We then generate adversarial examples of the *smoothed* classifier, using the method proposed in Salman et al. [2019]. Specifically, we use the $\text{SMOOTHADV}_{\text{PGD}}$ method in Salman et al. [2019] andsample Monte-Carlo noise vectors to estimate the gradients of the *smoothed* classifier. Adversarial examples are generated with a l_2 norm bound ϵ. Since denoiser D is introduced to remove the noise added by randomized smoothing, it is expected that backdoor of the poisoned classifier still applies to the new *robust smoothed* classifier. In practice, we find that this holds true in general for the poisoned classifiers we experimented with.

Figure 3: Backdoor Patterns in Adversarial Examples (ϵ = 20)
R*obustified* poisoned classifiers, each with a different color trigger.

3.3.2 Backdoor Patterns in Adversarial Examples

Our overall strategy is to analyze the adversarial examples of *robustified* poisoned classifiers. Since we assume that we are not aware of the original backdoor or which class is being targeted, through- out this report, unless otherwise specified, we generate *untargeted* adversarial examples (though through these untargeted examples it will become obvious which is the poisoned class). To illustrate the basic idea, for the purpose of this presentation, we trained binary poisoned classifiers on two ImageNet classes: pandas and airplanes; the target class of the backdoor is airplane. We used BadNet [Gu et al., 2017] for backdoor poisoning. After training, and without access to any training data, we then applied *Denoised Smoothing* to create a robust version of the classifier.

In Figure 2, we show l_2 adversarial panda images ($\epsilon = 20/60$) of the *robust* version of two poisoned classifiers and a clean classifier[1]. Two backdoor triggers are shown in Figure 4, where Trigger A is a 30×30 synthetic trigger with random colors, created in the backdoor attack method HTBA [Saha et al., 2020] and Trigger B is a 30×30 hello kitty image. The crucial point here is that for adversarial examples of *robustified* poisoned classifiers, there are local color regions that are immediately visually apparent. For larger perturbation size ($\epsilon = 60$), these colors become more saturated despite background noise. While for a clean classifier, such regions are much less prevalent.

To better understand the relationship between these color regions and the backdoor, we trained poisoned classifiers with backdoor triggers each consisting of a single, random color[2]. Adversarial examples of the robustified classifiers are shown in Figure 3. Similar to Figure 2, we still observe special color regions, and those colors are close to the color in the backdoor. This suggests that these local color spotscan provide useful information (i.e., color) of the initial trigger.

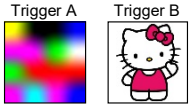


Figure 4: Backdoor Triggers used in our Analysis

3.3.3 Breaking Poisoned Classifiers

We now describe the overall procedure for constructing alternative triggers that perform just as well as the original one. The steps are as follows:

[1]We show adversarial examples with clear backdoor patterns. For the binary poisoned classifiers we investigate, we observe that most of the adversarial examples contain backdoor patterns.

[2]For some colors, classifiers are hard to poison (i.e., white). We choose the colors that result in high success rates (> 50%).Robustify the poisoned classifier using *Denoised Smoothing*.

1. Generate large-E adversarial examples of the *robustified smoothed* classifier.
2. Analyze the adversarial examples and find the backdoor patterns.
3. Use the observed backdoor patterns to construct new effective triggers.

For step 4, we use basic image editing operations to extract new triggers from the backdoor patterns:

- Synthesize a patch with colors obtained from the local regions with backdoor patterns. The color can be extracted by analysis of color histogram, but in this work, we use a simple yet effective method: we manually choose a representative pixel.
- Crop a patch image that contains the backdoor pattern.

We then use the constructed triggers to attack the poisoned classifier. Surprisingly, we find that although we create these triggers from only a handful of images, they generalize well to other images in the test set, attaining high attack success rates. We can use the procedure described above (illustrated in Figure 1) to easily break a poisoned classifier even if we do not have access to the original backdoor trigger.

Since our attack constructs the triggers from adversarial examples, one could argue that this is caused by the transferability of adversarial patches [Brown et al., 2017], which could be a general property of all classifiers (i.e., our attack may also work for clean classifier by creating an adversarial patch). To address this point, we also evaluate our attack on clean classifiers (Results are shown in Section 3.3) and find that clean classifiers are not broken by our method. Overall, our results prompt us to rethink backdoor poisoned classifiers. We show that the secret backdoor is not required to manipulate the backdoored classifiers. Not only can backdoor patterns be leaked through adversarial examples, we can also construct multiple triggers to attack the poisoned classifier that are just as effective as the original trigger.

The need for human interaction. It is important to emphasize that the process we describe above requires *human interaction* as part of the approach: i.e., a human analyst needs to identify "suspicious" regions in the adversarial images and select them as potential alternative triggers. However, rather than this being a downside of our approach, we emphasize that in fact we believe this to be a *benefit*. There are two main reasons for this. First, as discussed briefly above, the likely practical use cases of identifying poisoned classifiers is quite different than that of identifying or avoiding traditional adversarial examples. Each potentially poisoned classifier (for instance, a model built by a third party company, which is unknown to be poisoned or not) requires substantial time and investment to train and operate; thus, the additional time it will take an analyze to perform these kind of manual "forensic analysis" on a fully-trained classifier is a relatively small time commitment (and, as our examples show, the onus on the person doing this analysis is small).

Given this factor, the second reason that the human-in-the-loop nature of our process is beneficial is that human interaction is *needed* precisely due to the fundamental nature of adversarial examples. By definition, adversarial examples are perturbations that, to a human, will not change the "true" label of an image, but will cause an algorithm to classify it differently. If we relied on automated procedures to select the "suspicious" elements in an image, it would likely be possible to construct triggers that function as adversarial examples for these detectors, and thus evade detection. It is exactly (and, arguably, *only*) by integrating a human in the loop, which is entirely feasible in the data-poisoning use case, that we can hope to avoid the possibility of adversarial attacks against a fully automated system.

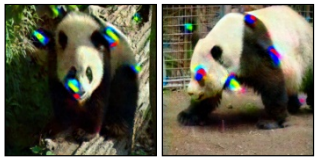


Figure 5: Sample Adversarial Images Generated with Deep Dream and Tikhonov Regularization

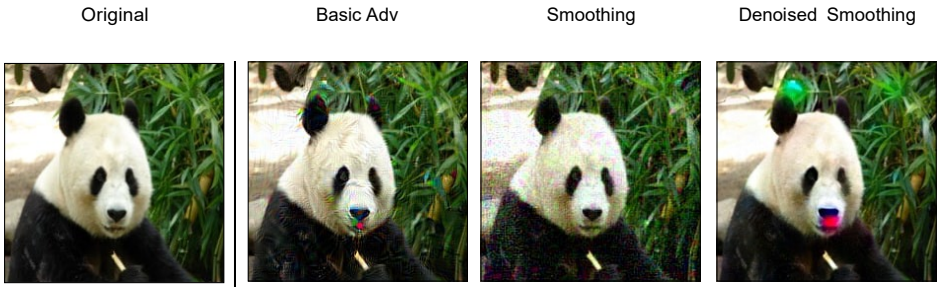


Figure 6:Comparison of Different forms of Adversarial Examples (ϵ = 20) from a Binary Poisoned Classifier on ImageNet

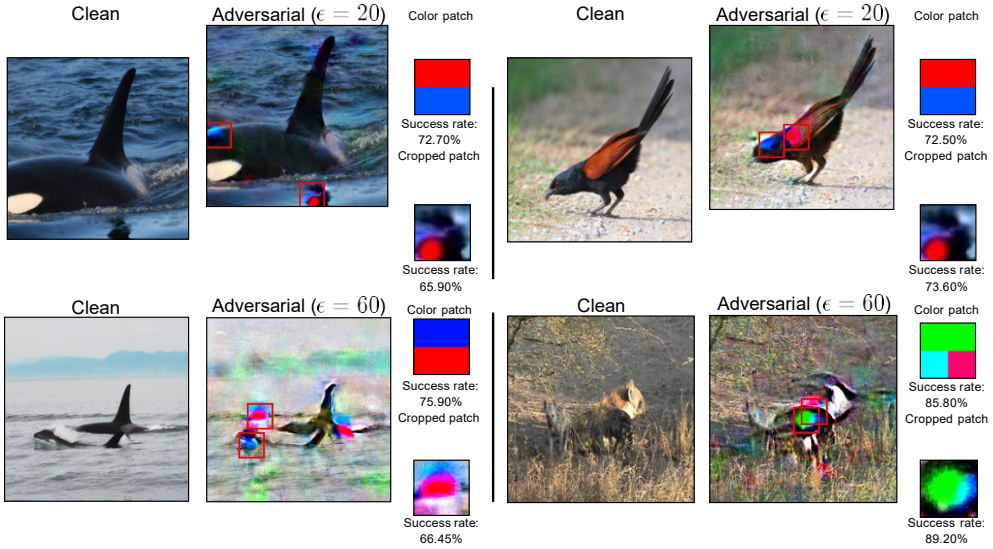


Figure 7: Results for attacking a poisoned multi-class classifier obtained through BadNet [Gu et al., 2017]
The attack success rate of the original backdoor Trigger A is 72.60%. The region which we use to construct alternative triggers is highlighted in a red box.

3.3.4 Enhanced Visualization Techniques

We discuss some techniques to help with visualizing adversarial examples. **Deep Dream** We adopt the idea from Deep Dream [Mordvintsev et al., 2015] by iteratively optimizing a certain objective starting with the resized output from previous iteration. It uses this iterative optimization process to generate artistic style images. In our case, we iteratively optimize the adversarial objective, so that backdoor patterns formed at earlier stages can be incorporated into those forming at later stages.

Tikhonov Regularization Large-E adversarial images tend to become noisy. Thus, we apply Tikhonov regularization [Tikhonov et al., 1992]. It minimizes a loss function defined as a l_2-regularization of the magnitude of image gradients (directional change in the intensity of colors).

In Figure 5, adversarial images are generated with two techniques of the binary *robustified*

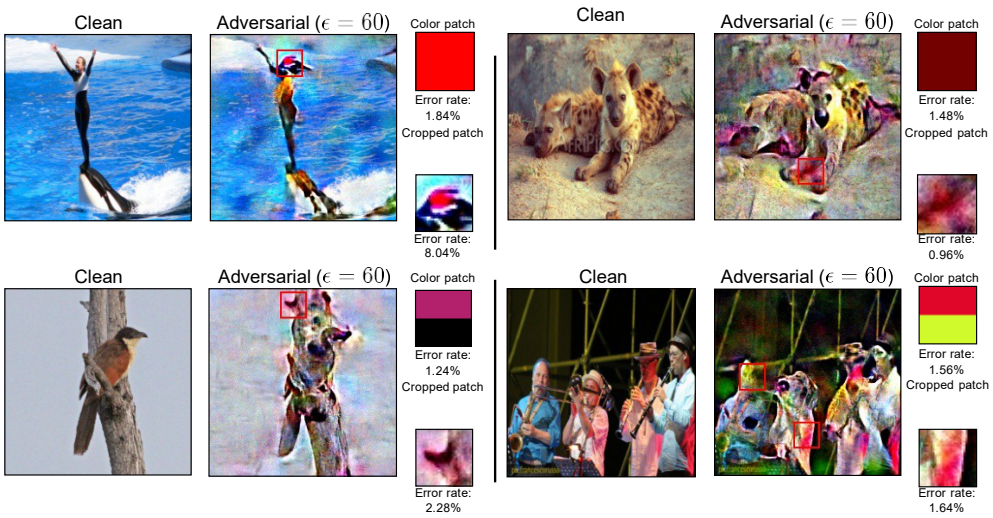


Figure 8: Results of Applying our Attack on an ImageNet Clean Classifier

poisoned classifier with Trigger A. Observe that images become smoother and there are more backdoor patterns in one image.

4.0 Results

In this section, we present our attack results on poisoned classifiers from two datasets: ImageNet [Russakovsky et al., 2015] and TrojAI datasets [Majurski, 2020]. For *Denoised Smoothing*, we use the MSE-trained ImageNet denoiser adopted from Salman et al. [2020]. To make backdoor presence conspicuous, we synthesize large-E untargeted adversarial examples (ϵ = 20, 60). The noise level we use in *smoothed* classifiers is 1.00, as Kaur et al. [2019] shows that larger noise level leads to better visual results. We refer the reader to Appendix A for details on the experimental setup. For both datasets, we construct alternative triggers of size 30 30, same as the size of the backdoor trigger used in ImageNet poisoned classifiers[3]. We apply alternative triggers to random locations for ImageNet (same as the initial backdoor) and a fixed place near the center for TrojAI [4].To evaluate the attack success rate, for ImageNet, we use 50 images for binary classifier and 200images for multi-class classifier in the test set; for TrojAI dataset, we use the released 500 sampletest images for each classifier.

4.1 ImageNet

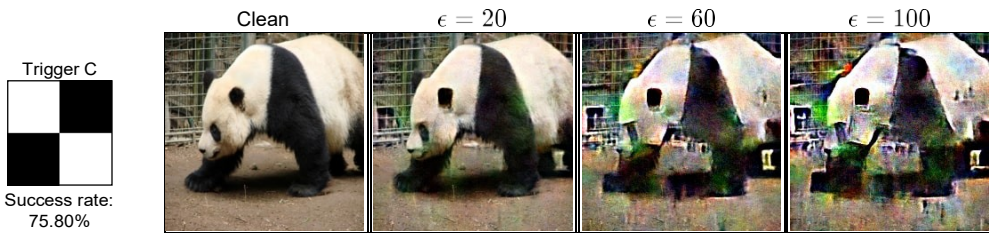
We train both binary and multi-class poisoned classifiers with three backdoor attack methods: BadNet [Gu et al., 2017], *Hidden trigger backdoor attack* (HTBA) [Saha et al., 2020] and *Clean-label backdoor attack* (CLBD) [Turner et al., 2019] (in total 6 poisoned classifiers). The class of the binary classifier is hand-picked: "panda" vs "airplane". For the multi-class classifier, 5 classes are chosen randomly. Since only HTBA has conducted evaluation on ImageNet, we follow its setup for training poisoned classifiers. Specifically, we adopt Trigger A in Figure 4 as the default trigger and use AlexNet [Krizhevsky et al., 2012] architecture[5].

Comparison to baselines We compare *Denoised Smoothing* to two baseline approaches for generating adversarial examples: adversarial examples of 1) the poisoned classifier (denoted as

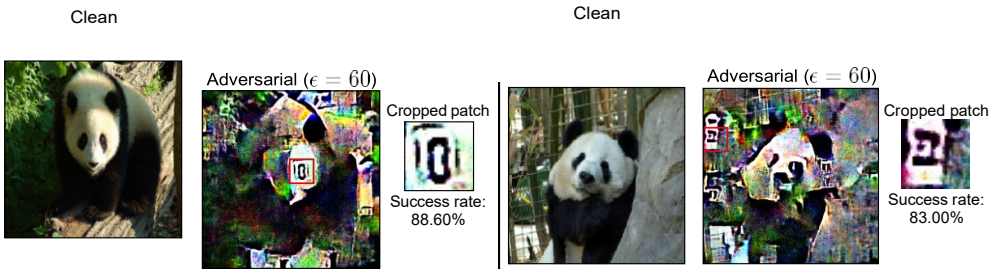
[3]In TrojAI, the exact shape of backdoor trigger is not provided. Here we adopt the same setting as ImageNet.

[4]For TrojAI, we are not aware of where the trigger is applied in the training process of poisoned classifiers. We choose this location in order for the alternative triggers to be applied at the foreground object (an artificial sign). (Sample images in https://pages.nist.gov/trojai/docs/data.html)

[5]Except for CLBD, we use ResNet [He et al., 2016] for the backdoor attack to be successful.



(a) Adversarial examples of a *robustified* poisoned classifier with Trigger C as the backdoor.



(b) Attacking a poisoned classifier with the "camouflaged" backdoor Trigger C (success rate 75.80%).

Figure 9: Analysis of a Poisoned Classifier with a "camouflaged" Backdoor Trigger

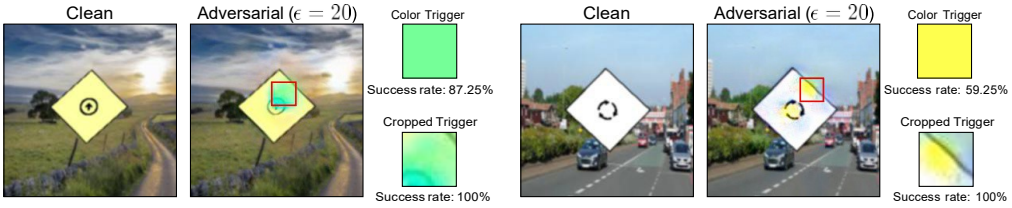
"Basic Adv"); 2) the *smoothed* poisoned classifier without a denoiser (denoted as "Smoothing"). We generate adversarial examples (ϵ = 20) of the *robustified* binary poisoned classifier on ImageNet, shown in Figure 6 (More examples are shown in Figure 24 in Appendix C.). First, we can see that our approach gives less noisy and smoother adversarial images than baselines. Second, there are some vague backdoor patterns in "Basic Adv", but backdoor patterns in adversarial examples from *Denoised Smoothing* are more distinctive and easier to recognize. Last, "Smoothing" baseline does not produce any obvious pattern, which highlights the necessity of *Denoised Smoothing*.

Table 1: Overall Performance of our Attack

For "X/Y", X is the highest attack success rate amongthe triggers that we demonstrate in this report and Y is the success rate of the original backdoor.

	BadNet	HTBA	CLBD
Binary	**98.80%**/91.60%	**99.80%**/94.00%	**93.80%**/90.00%
Multi-class	**89.20%**/72.60%	**82.30%**/74.55%	**67.90%**/58.95%

Breaking poisoned classifiers In Figure 7, we present sample alternative backdoor triggers we constructed by attacking a BadNet poisoned multi-class classifier on ImageNet, where we show both color patch and cropped patch constructed from each adversarial example. For attack results on other five ImageNet poisoned classifiers, we refer the reader to Figure 18 and Figure 19 in Appendix B. From Figure 7, we can see that all the alternative triggers created from backdoor patterns have relatively high success rate. In particular, two triggers achieve significantly higher attack success rate (89.20%, 85.80%) than the original backdoor Trigger A (72.60%). Also notice that these alternative triggers differ greatly from Trigger A. Last, we can see that whether color patch or cropped patch perform better depends on each example. In terms of the effect of pertubation size, it can be seen that larger epsilon leads to better attack results. A summary of attack results for all poisoned classifiers is presented in Table 1. For each poisoned classifier, we compare the highest success rate achieved by the alternative triggers we demonstrate in the report and thesuccess rate of the initial backdoor (Trigger A). For all six poisoned classifiers we investigate, ourattack finds an alternative trigger more effective than the original backdoor. Also, for five of the six poisoned classifiers, the highest success rate shown in Table 1 is attained by cropped patch, which may suggest that cropped patch may be more effective overall.



(a) Poisoned Classifier 1 (b) Poisoned Classifier 2

Figure 10: Results of Attacking two Poisoned Classifiers in TrojAI Dataset

Clean classifiers are not easily broken. We show that clean classifiers are not broken under our attack. Note that clean classifiers are not poisoned and there is no such concept as attack success rate for clean classifiers. To measure the effect of the triggers constructed by our procedure on clean classifiers, we report the error rate of clean classifiers when the test data is patched by the alternative triggers. Figure 8 presents an illustration for attacking a clean multi-class ImageNet classifier with our approach. We refer the reader to Figure 20 in Appendix B for results on attacking the binary ImageNet classifier. Here we choose larger perturbation size $\epsilon = 60$ because we find no obvious pattern with perturbation size $\epsilon = 20$. Observe that clean classifiers have low error rates with test data patched by the constructed triggers, meaning that they remain robust under our attack.

"Camouflaged" Backdoor So far we have experimented with triggers that contain colors (i.e., red, blue in Trigger A) that are visually distinctive and as a result, backdoor patterns can be easily recognizable in adversarial examples. We study the case when backdoor trigger is less colorful or contains colors already in the color distribution of clean images. Consider Trigger C in Figure 9a: black and white colors in this trigger are also representative colors of a panda. We train a poisoned binary classifier on ImageNet using Trigger C as the backdoor, where the backdoor attack method is BadNet [Gu et al., 2017]. In Figure 9a, we visualize adversarial examples of the *robustified* poisoned classifier. Although there is no clear backdoor pattern in the form of dense color regions, we can observe that in the generated adversarial examples, there is a tendency for black regions to have vertical or horizontal boundaries, which resembles the pattern in Trigger C.Despite the absence of obvious backdoor patterns, we are still able to break the poisoned classifier using cropped patterns from large- ϵ ($\epsilon = 100$) adversarial examples as shown in Figure 9b. Notice that both of the triggers are noisy and seem completely different from Trigger C, but they attain higher attack success rate (88.60% and 83.00%) than the original backdoor (75.80%).

4.2 TrojAI Dataset

TrojAI dataset [Majurski, 2020] consists of a mixed set of clean and poisoned classifiers, proposed to help develop backdoor defense methods. We choose this dataset as it contains a large set of trained poisoned classifiers. Different from ImageNet, we are not aware of the exact backdoor triggers used to poison the classifiers. In Figure 10, we show attack results on two poisoned classifiers. As shown in Figure 10, our methods can attack these poisoned classifiers with high success rate (See Figure 21 in Appendix B for results on more poisoned classifiers.). Similarly, the cropped trigger achieves higher success rate than the color trigger for both classifiers. Especially, notice that both cropped triggers attain 100% attack success rate. Finally, we conduct a user study on the TrojAI dataset to test the generality of our approach. We develop an interactive tool implementing our method to aid the study. Participants are asked to analyze classifiers with the tool and decide if they are poisoned. Two control groups are used: 1) participants are given a variant of the tool using adversarial examples of the original classifier (denoted as "Basic Adv"); 2) participants are given saliency maps on clean images (denoted as "Saliency Map"). Details on the user study and the interactive tool are in Appendix D. Results are summarized in Table 2, where we show the accuracies of identifying poisoned classifiers for three approaches. Overall, the study suggests that analysts with access to our tool are able to substantially outperform those using alternative methods.

Table 2: Accuracies that Participants Obtained for Identifying Poisoned Classifiers in the User Study

	Denoised Smoothing		Basic Adv		Saliency Map
Participants	User 1	User 2	User 3	User 4	User 5
Accuracy	94%	90%	66%	82%	54%

5. Objective Criteria for Evaluation of ML Classifiers

The field of explainable artificial intelligence (XAI) is concerned with the task of explaining a complex machine learning model, that has been trained on a set of input and label pairs. We are interested in the case where we only have black-box access to the model, which can be abstracted as a function that predicts a response given an input feature vector, and depending on the class of explanations, additionally the training dataset. The two predominant classes of explanations are: (a) feature explanations, that attribute any given prediction to a (weighted) set of important input features, and (b) sample explanations, that attribute the prediction to a set of training inputs. As a popular instance, given a deep neural network for image classification trained on a set of training data, we may explain a specific prediction by showing the set of salient pixels, or a heatmap of pixel importance weights (feature explanations), or by showing the most influential training images (sample explanations). However within these classes, there are even increasing proposed instances of explanations, which are all typically evaluated in a qualitative manner, largely by showing compelling examples of specific explanations. To place these on a more scientific and rigorous footing, we need more objective evaluation criteria. This survey article summarizes and provides insights on three key papers that introduce such objective criteria for: (a) feature importance explanations [Yeh et al., 2019], (b) feature set explanations [Hsieh et al., 2021], and (c) sample based explanations [Yeh et al., 2018].

The first objective criterion we discuss is what Yeh et al [Yeh et al., 2019] refer to as an infidelity measure, which captures how well a feature importance explanation (one feature importance value per feature) matches the given model. Now, given any perturbation in the input, we expect there to be a perturbation in the model output. Can feature explanations give insight into how the model output perturbations might look like? Towards this, infidelity measures the expected difference between the model output perturbation, and its approximation using the explanation (specificallyvia a dot product of the feature explanation vector and the input perturbation). Note that this general setup allows for any choice of significant perturbations. It can be shown that most popularfeature based explanations actually optimize the infidelity measure above, for corresponding spe- cific choices of perturbations. It moreover holds that the optimal explanation minimizing infidelity can actually be calculated in closed-form. This in turn allows one to propose new explanations by introducing new classes of perturbations. We also mention in passing a classical objective criterion for explanations, namely whether or not the explanation mechanism satisfies certain axiomatic properties [Lundberg and Lee, 2017, Sundararajan et al., 2017]. One can analyze whether a feature importance explanation satisfies certain axiomatic properties based on its *corresponding* perturba- tion distribution. Thus, one may choose the desired feature importance explanations depending on (a) the perturbations of interest, and (b) corresponding axiomatic properties, which are also related to the perturbation distribution.

We next focus the class of feature set explanations, that output a subset of important features for any model prediction. Here, Hsieh et al. [Hsieh et al., 2021] propose robustness based objective criteria for this form of explanations, based on the idea that if we get the important feature set right, then the model will be robust to perturbations restricted to their complement of unimportant features. Similarly, if we restrict the perturbations to only the important features we expect the model to be very sensitive. We can measure robustness using the maximum function difference under any perturbation, which is drawn from the test-time adversarial robustness literature. They then propose scalable greedy algorithms to optimize this criteria.

Lastly, we focus on sample-based explanations, where the goal is to explain the model prediction at a test input in terms of contributions of different training points. Based on a simple requirement that the sum of the contributions of different training points should sum up to the model predictionat the test point, Yeh et al. [Yeh et al., 2018] design a "representer theorem," that provides just such a decomposition for general deep neural networks, with the slight addendum that the model be trained till convergence with a small ℓ_2 regularization weight. They demonstrate the effectivenessof all three classes of explanations proposed above via intuitive visual examples.

5.1 Related Work

We briefly summarize related work on the three main classes of explanations discussed above: (a) feature importance explanations, (b) feature set explanations, and (c) sample based explanations.

Among feature importance explanations, perturbation-based attributions measure the prediction difference after perturbing a set of features. As a key instance, Zeiler [Zeiler and Fergus, 2014] use grey patch occlusions as perturbations to explain CNNs. Such senstivity to perturbations is also captured by gradient based attributions [Baehrens et al., 2010, Simonyan et al., 2013, Zeiler and Fergus, 2014, Springenberg et al., 2014, Selvaraju et al., 2017] that range from simple gradients, to variants that leverage back-propagation to address some caveats with simple gradients. As shown in Ancona et al [Ancona et al., 2018], many recent explanations such as *E*-LRP [Bach et al., 2015], Deep LIFT [Shrikumar et al., 2017], and Integrated Gradients [Sundararajan et al., 2017] can be simply viewed as variants of gradient explanations. There are also approaches that average feature importance weights by varying the active subsets of the set of input features (e.g. over the power set of the set of all features), which has roots in cooperative game theory and revenue division [Datta et al., 2016, Lundberg and Lee, 2017]. Another close line of research is counterfactual and contrastive explanations [Wachter et al., 2017, Dhurandhar et al., 2018, Hendricks et al., 2018, van der Waa et al., 2018, Goyal et al., 2019, Joshi et al., 2019, Poyiadzi et al., 2020], which is motivated by the question: what to alter in the current input to change the outcome of the model. While "the closest possible world" provided by counterfactual explanations sheds light on features that are "necessary" for the prediction, they could fail to identify relevant features that are "sufficient" for the current prediction. For sample explanations, some derive these by analyzing the stationary conditions of the ML model estimation problems [Koh and Liang, 2017, Yeh et al., 2018], by tracking the training trajectory of the model parameters [Pruthi et al., 2020], by repeatedly retraining the model [Jia et al., 2019, Ghorbani and Zou, 2019], and via solving specific optimization problems [Kim et al., 2016, Khanna et al., 2019].

We next discuss related work that specifically propose objective evaluations for explanations, which is the focus of this survey article. A classical approach is to measure model performance after removing salient features [Hooker et al., 2018]. A related measure is the area over the perturbation curve when removing the most important features [Samek et al., 2016]. Another is explanation continuity [Montavon et al., 2017], which is a minimal sanity check on numerical explanations.

Object localization metrics to evaluate the closeness of the saliency map and the actual object have also been proposed [Dabkowski and Gal, 2017].

5.2 Objective Criteria for Feature Importance Explanations

With feature importance (or attribution) explanations, we explain the prediction of a given model on a given test input **x**, by provide a real-valued importance value for each of the features in **x**. Most such explanations use the notion of a perturbation of the test input. Here, we measure the importance of a feature by perturbing the feature, and measuring the average effect on the model output. We classify such perturbation-based feature importances into two classes: where the perturbation is either a general real-valued vector, or a binary vector. In this section, we discuss the framework of Yeh et al [Yeh et al., 2019], where they show that most existing perturbation-based feature attributions can be viewed as optimizing the so-called infidelity of explanations, which is an objective criterion for perturbation-based feature importances, and which depends on a random perturbation **I** with probability measure $\mu_{\mathbf{I}}$.

5.2.1 Infidelity for General Perturbations

Consider the following general supervised learning setting: we have the input space $\mathcal{X} \subseteq \mathbb{R}^d$, output space $\mathsf{Y} \subseteq \mathbb{R}$, and a (machine-learnt) black-box predictor $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}$, which at some test input $\mathbf{x} \in \mathbb{R}^d$, predicts the output $\mathbf{f}(\mathbf{x})$. Then a feature attribution explanation is some function $\Phi : \mathsf{F} \times \mathbb{R}^d \to \mathbb{R}^d$, that given a black-box predictor $\mathbf{f}$, and a test point $\mathbf{x}$, provides importancescores $\Phi(\mathbf{f}, \mathbf{x})$ for the set of input features. We let $\|\cdot\|$ denote a given norm over the input (and explanations) space. When not specified, this will be set to the ℓ_2 norm.

Definition 5.1. Given a black-box function $\mathbf{f}$, explanation functional Φ, and a random variable $\mathbf{I} \in \mathbb{R}^d$ with probability measure $\mu_{\mathbf{I}}$, which represents meaningful perturbations of interest, we define the explanation infidelity of Φ as:

$$INFD(\Phi, f, x) = \mathbb{E}_{I \sim \mu I}\left[I^T \Phi(f, x) - \left(f(x) - f(x - I)\right)\right]^2 \qquad (5)$$

Explanations that Optimize Infidelity As we show in the sequel, many recently proposed explanation methods can be shown to be optimal with respect to the infidelity measure in Definition 4.1 and 4.2, for varying perturbations **I**. Explanations that optimizes the infidelity with respect to the general perturbations can been seen as variants of the gradient, since $\lim \epsilon \to_0 \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x} - \epsilon\, \mathbf{e}_i)/\, \epsilon\, \mathbf{e}_i = \nabla_i \mathbf{f}(\mathbf{x})$.

Proposition 5.1. Suppose the perturbation $\mathbf{I} = \mathbf{x}\text{-}\mathbf{x}_0$ is deterministic and is equal to the difference between $\mathbf{x}$ and some baseline $\mathbf{x}_0$. Let $\Phi^*(\mathbf{f}, \mathbf{x})$ be any explanation which is optimal with respect to infidelity for perturbations **I**. Then $\Phi^*(\mathbf{f}, \mathbf{x}) \odot \mathbf{I}$ satisfies the completeness axiom; that is

$$\sum_{j=1}^{d} [\Phi^*(f, x) \odot I]_j = f(x) - f(x - I)$$

The completeness axiom has been argued to be important for feature attributions [Sundararajanet al., 2017, Shrikumar et al., 2017]. The above proposition shows its close connection to the notionof infidelity. Note that the completeness axiom is satisfied by many popular feature explanations such as IG [Sundararajan et al., 2017], DeepLift [Shrikumar et al., 2017], LRP [Bach et al., 2015].

Proposition 5.2. Suppose the perturbation is given by $\mathbf{I}_i = \epsilon \cdot \mathbf{e}_i$, where $\mathbf{e}_i$ is a coordinate basis vector. Then letting $\Phi^{*\epsilon}(\mathbf{f}, \mathbf{x})$ be the optimal explanation with respect to infidelity for perturbations $\mathbf{I}_i$, and $\mathbf{v}[i] = \Phi^*_E(\mathbf{f}, \mathbf{x})[i]$, we have that $\lim \epsilon \to_0 \mathbf{v} = \nabla \mathbf{f}(\mathbf{x})$, so that the limit point of the optimal explanations is the gradient explanation [Shrikumar et al., 2016].

Proposition 5.3. Suppose the perturbation is given by $I_i = e_i \odot x$, where $\mathbf{e}_i$ is a coordinate basis vector. Letting $\Phi^*(\mathbf{f}, \mathbf{x})$ be the optimal explanation with respect to infidelity for perturbations $\mathbf{I}_i$, and $\mathbf{v}[i] = \Phi^*(\mathbf{f}, \mathbf{x})[i]$, then $\mathbf{v} \odot \mathbf{x}$ is the occlusion-1 explanation[Zeiler and Fergus, 2014].

Closed-Form Explanations with least Infidelity Given our notion of infidelity, a natural question is: what is the explanation that is optimal with respect to infidelity, that is, has the least infidelity possible. This naturally depends on the distribution of the perturbations **I**, but surprisingly this optimal explanation has a simple form as detailed in the following proposition.

Proposition 5.4. Suppose the perturbations **I** are such that $(\int \Pi^T d\mu_i)^{-1}$ is invertible. The optimal explanation $\Phi^*(\mathbf{f}, \mathbf{x})$ that minimizes infidelity for perturbations **I** can then be written as

$$\Phi^*(\mathbf{f}, \mathbf{x}) = \left(\int \mathbf{I}\mathbf{I}^T d\mu_{\mathbf{I}} \right)^{-1} \left(\int \mathbf{I}\mathbf{I}^T \mathrm{IG}(\mathbf{f}, \mathbf{x}, \mathbf{I}) d\mu_{\mathbf{I}} \right), \tag{6}$$

where

$\mathrm{IG}(\mathbf{f}, \mathbf{x}, \mathbf{I}) = \int_{t=0}^{1} \nabla \mathbf{f}(\mathbf{x} + (t-1)\mathbf{I})\, dt$ is the integrated gradient of $\mathbf{f}(\cdot)$ between $(\mathbf{x} - \mathbf{I})$ and $\mathbf{x}$ [Sundararajan et al., 2017], but can be replaced by any functional that satisfies $\mathbf{I}^T \mathrm{IG}(\mathbf{f}, \mathbf{x}, \mathbf{I}) = \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x} - \mathbf{I})$.

A generalized version of SmoothGrad can be written as

$$\Phi_k(\mathbf{f}, \mathbf{x}) := \left[\int_{\mathbf{z}} k(\mathbf{x}, \mathbf{z}) \right]^{-1} \int_{\mathbf{z}} \Phi(\mathbf{f}, \mathbf{z})\, k(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

where the Gaussian kernel can be replaced by any kernel. Therefore, the optimal solution of Proposition 4.4 can be seen as applying a smoothing operation reminiscent of SmoothGrad on Integrated Gradients (or any explanation that satisfies the completeness axiom), where a special kernel $\mathbf{I}\mathbf{I}^T$ is used instead of the given off-the-shelf kernel function $k(\mathbf{x}, \mathbf{z})$. When **I** is deterministic, the integral of $\mathbf{I}\mathbf{I}^T$ is rank-one and cannot be inverted, but being optimal with respect to the infidelity can be shown to be equivalent to satisfying the Completeness Axiom. To enhance its numerical stability, we can replace the inverse by a pseudo-inverse, or add a small diagonal matrix to the kernel $\mathbf{I}\mathbf{I}^T$, which works well in experiments.

Vignette: Noisy Baseline, a New Explanation by Optimizing Infidelity By varying the perturbations and deriving the corresponding optimal explanation, we could obtain new feature explanations. As a vignette of this approach, suppose we set the baseline to be a Gaussian random vector centered around a certain clean baseline (such as the mean input or zero) depending on the context. Note that this addresses a common caveat with a deterministic baseline that it does not account for noise or uncertainty. The explanation that optimizes infidelity with corresponding "noisy baseline" perturbations **I** is a novel explanation that can be seen as satisfying a robust variant of the completeness axiom.

5.2.2 Infidelity for Binary Perturbations

We now consider the case where $\mathbf{f}$ is replaced by a meta-function $\upsilon_{\mathbf{f},\mathbf{x}} : 2^d \to \mathbb{R}$, where the input of the meta function is a subset of d features, where the output is also a real-valued number. Then a feature attribution explanation is some function $\Phi : \mathcal{V} \times \mathbb{R}^d \to \mathbb{R}^d$, that given a black-box predictor $\mathbf{f}$, and a test point $\mathbf{x}$, and the meta function $\upsilon_{\mathbf{f},\mathbf{x}}$ provides importance scores $\Phi(\upsilon_{\mathbf{f},\mathbf{x}})$ for the set of input features. The meta-function $\upsilon_{\mathbf{f},\mathbf{x}}$ here is often called the value function.

The difference to the general perturbations is that here, the perturbation space is limited to binary perturbations. The value function can be seen as the function utility when a set of features are present. The explanation here has a game-theoretical interpretation, which is the average marginal utility of each features.

Definition 5.2. Given a black-box function **f** , explanation functional Φ, a random variable $\mathbf{I} \in 2^d$ with probability measure $\mu_{\mathbf{I}}$, which represents meaningful perturbations of interest and a binarized input **z** which is a vectors of 1s, we define the explanation infidelity of Φ as:

$$\mathrm{INFD}(\Phi, v_{\mathbf{f}}, \mathrm{x}) = \mathbb{E}_{\mathbf{I}\sim\mu_{\mathbf{I}}}\left[\left(\mathbf{I}^T\Phi(v_{\mathbf{f},\mathbf{x}}) - (v_{\mathbf{f},\mathbf{x}}(\mathbf{z}) - v_{\mathbf{f},\mathbf{x}}(\mathbf{z}-\mathbf{I}))\right)^2\right]. \quad (7)$$

I (7)

Explanations that Optimize Infidelity Explanations that optimizes the infidelity of binary perturbations have a game-theoretic flavor, as the value function summarizes the joint utility when a set of features are present "co-operatively".

Proposition 5.5: The perturbation $I \in \{0,1\}^d$ is a binary random vector with distribution

$$\mathbb{P}(I = z) \propto \frac{d-1}{(\frac{d}{\|I\|_1})\|I\|_1(d-\|I\|_1)}$$

Then for the optimal explanation $\Phi^*(v_{f,x})$ with respect to infidelity for perturbations **I** is the Shapley value.

Proposition 5.6. The perturbation $\mathbf{I} \in \{0,1\}^d$ is a binary random vector with distribution $\mathbb{P}(\mathbf{I} = \mathbf{z}) \propto 1$ such that all binary perturbation has equal probability, then for the optimal explanation $\Phi^*(v_{\mathbf{f},\mathbf{x}})$ with respect to infidelity for perturbations **I** is the Banzhaf value.

5.2.3 Closed-Form Solution for Explanations with Least Infidelity

Proposition 5.7. Suppose the perturbation distribution $\mu(S)$ is such that $(\sum_S \mu(I)\Pi^T)$ is invertible. The explanation $\Phi^*(v_{\mathbf{f},\mathbf{x}})$ that minimizes the infidelity with respect to the perturbation **I** has the form:

$$\Phi^*(v_{\mathbf{f},\mathbf{x}}) = \left(\sum_S \mu(\mathbf{I})\mathbf{I}\mathbf{I}^T\right)^{-1}\left(\sum_S \mu(\mathbf{I})\mathbf{I}(v_{\mathbf{f},\mathbf{x}}(\mathbf{z}) - v_{\mathbf{f},\mathbf{x}}(\mathbf{z}-\mathbf{I}))\right) \quad (8)$$

Vignette: Square Banzhaf, a New Explanation by Optimizing Infidelity This explanation is specifically targeted to image data. Yeh et al [Yeh et al., 2019] argue that studying perturbations that remove random subsets of pixels in images may not be instructive, since there is very little loss of information given surrounding pixels that are not removed. Also ranging over all possible subsets to remove (as in SHAP [Lundberg and Lee, 2017]) is infeasible for high dimensional images. They thus propose a modified subset distribution from that described in Proposition 4.5, where the perturbation **Z** has a uniform distribution over square patches with predefined length, which is in spirit similar to the work of [Zintgraf et al., 2017]. This not only improves computational complexity, but also better captures spatial structure in the images. One can also replace the square with more complex random masks designed specifically for the image domain [Petsiuk et al., 2018a]. They name the resulting infidelity optimal explanation as "square Banzhaf" since it considers all sets of players (corresponding to squares in the image) equally, which utilizes the

structure of the data.

Consequences for Axiomatic Properties We briefly introduce a set of well-known axioms for player attributions from cooperative game theory [Shapley, 1988], and which have been suggested for feature attributions in the machine learning context [Lundberg and Lee, 2017].

1. Linearity: $\Phi(f_1) + \Phi(f_2) = \Phi(f_1 + f_2)$ for any two value functions f_1 and f_2.
2. Symmetry: $f(S \cup i) = f(S \cup j) for\ any\ S \subseteq [d]\{i, j\}, \Phi_i(f) = \Phi_j(f)$
3. Dummy Player $f(S \cup i) = f(S)\ for\ any\ S \subseteq [d]\{i\}\ implies\ , \Phi_i(f) = 0.$
4. Efficiency: $\sum_i \Phi(f) = f([d]) - f(\emptyset)$.

In this section, we relate the axiomatic properties of explanations with the least infidelity to the perturbation distribution that defines the infidelity. Since each perturbation distribution uniquely defines one explanation when $(\sum_S \mu(I)\Pi^T)\ or\ (\int \Pi^T\, d\mu_I)^{-1}$ is invertible, it is interesting to see under which conditions of perturbation distribution will the explanation follow certain axioms. For instance, Prop. 5.1 already connects the perturbation distribution to the completeness axiom, which is also known as the efficiency axiom in the game theoretic literature, Prop. 5,5 connects the Shapley value to the perturbation distribution

$$\mathbb{P}(I = z) \propto \frac{d - 1}{(\frac{d}{\|I\|_1})\|I\|_1(d - \|I\|_1)}$$

Thus, the explanation that minimizes the infidelity with this distribution will satisfy axioms linearity, dummy player, symmetry, and completeness (efficiency). We now provide other conditions on the perturbations where the explanation will satisfy certain axiomatic properties.

Proposition 5.8. If $\sum_S \mu(I)\Pi^T$ is invertible, then $\Phi^*(v_f, \mathbf{x})$ satisfies the linearity axiom.

We then provide a connection between the distribution of coalitions and the symmetric axiom. Intuitively, if μ is symmetric to all players, then the most faithful feature interaction satisfies the symmetric axiom.

Proposition 5.9. Suppose the matrix $\sum_S \mu(I)\Pi^T$ is invertible. Then μ is symmetric with respect to all features set with the same size, so that $\mu(S_i) = \mu(S_j)$ if $|S_i| = |S_j|$ for all $S_i, S_j \subseteq [d]$, then $\Phi^\ddagger(v_{f,x})$ satisfies the symmetry axiom.

We now introduce the independent distribution of coalitions, where the weights of the coalition is equal to the product of each player's individual weight.

Definition 5.3. We call the distribution μ(I) independent if $\mu(I) = \Pi_{i\in[d]} p_i^{\mathbb{I}[I_i=1]}(1-p_i)^{\mathbb{I}[I_i=0]}$ for some $p \in \mathbb{R}^d$ such that $0 \leq p_i \leq 1$.

The independence of distribution is a sufficient condition for the most faithful interaction value to satisfy the dummy axiom.

Proposition 5.10. When the distribution $\mu(S)$ is independent, $\Phi(\mathbf{f}, \mathbf{x})$ satisfies the dummy axiom.

By combining the condition of the dummy axiom and the symmetric axioms, we get the condition that each coalition should have equal weight. In this case, we retrieve the Banzhaf value [Banzhaf III, 1964].

Proposition 5.11. When $\mu(\mathbf{I}) = 1/2^d$ for all $\mathbf{I} \subseteq [d]$, and $\Phi^*(v_{\mathbf{f}}, \mathbf{x})$ equals the Banzhaf value [Banzhaf III, 1964], which satisfies the Linearity, Symmetry, and dummy player axioms, which is an application of Prop. 5.8, 5.9, 5.10.

5.2.4 Some Visualization Results

In this section, we show visualization results for several perturbation-based methods in Fig. 11 and 12 on the imagenet dataset [Deng et al., 2009] and Mnist dataset [LeCun et al., 2010] respectively.Our newly proposed explanation Square Banzhaf (Square) and Noisy Baseline (NB) shows great concentration on the object of interest.

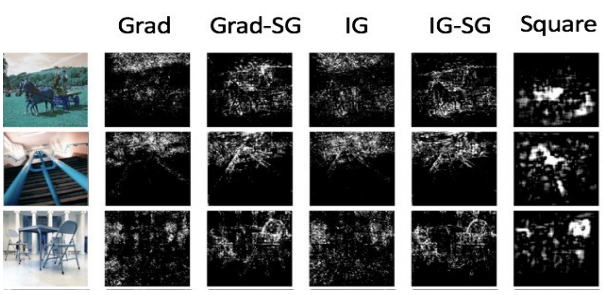


Figure 11: Examples of Explanations on Imagenet

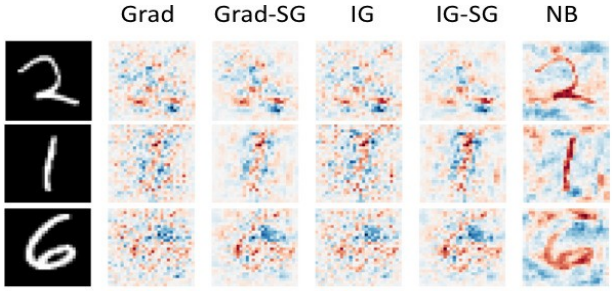


Figure 12: Examples of Local Explanations on MNIST

5.3 Objective Criteria for Feature Set Explanations

In this section, we focus on subsets of important features, that are relevant to the model prediction, as a form of explanation, instead of the real-valued feature attributions. Here, we discuss the framework of Hsieh et al [Hsieh et al., 2021], where they provide robustness analysis based criteria to evaluate such explanations.

5.3.1 Goodness Criteria via Robustness Analysis

One downside of the perturbation-based feature importance is that the perturbation distribution should be known. When this is not possible, it would be natural to find an objective metric that describes the goodness of explanations without relying on a given form of perturbations.

We start by two assumptions on which set of features are crucial to the model predictions.

Assumption 1: When the values of the important features are anchored (fixed), perturbations restricted to the complementary set of features has a weaker influence on the model prediction.

Assumption 2: When perturbations are restricted to the set of important features, fixing the values of the rest of the features, even small perturbations could easily change the model prediction.

Based on these two assumptions, we propose a new framework leveraging the notion of adversarial robustness on feature subsets, as defined below, to evaluate feature based explanations. We note that these assumptions are related to the sensitivity of the important feature set. In short, a feature set is considered important if they are sensitive (with respect to the model prediction), or their complement are insensitive (with respect to the model prediction). However, these two criteria are not equivalent. In the following, we formally define the two goodness criteria for a set of features.

Definition 5.4. Given a model f, an input $\boldsymbol{x}$, and a set of features $S \subseteq U$ where U is the set of all features, the minimum adversarial perturbation norm on $\boldsymbol{x}_S$, which we will also term *Robustness-S* of $\boldsymbol{x}$ is defined as:

$$\epsilon^*_{x_S} = g(f, \boldsymbol{x}, S) = \left\{ \min_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_p \text{ s.t. } f(\boldsymbol{x} + \boldsymbol{\delta}) \neq y,\ \boldsymbol{\delta}_{\overline{S}} = 0 \right\}, \tag{9}$$

—

where $y = f(x), \overline{S} = U \backslash S$ is the complementary set of features, and $\delta_{\overline{S}} = 0$ means that theperturbation is constrained to be zero along features in $\overline{S}$. Suppose that a feature based explanation partitions the input features of $\boldsymbol{x}$ into a relevant set S_r,and an irrelevant set S_r, Assumption 1 implies that the quality of the relevant set can be measured by $\epsilon^*_{x\overline{S_r}}$ - by keeping the relevant set unchanged, and measuring the adversarial robustness norm by perturbing only the irrelevant set Specifically, from Assumption 1, a larger coverage of pertinent features in set S_r entails a higher robustness value. On the other hand, from Assumption 2, a larger coverage of pertinent features in set S_r would in turn entail a smaller robustness value since only relevant features are perturbed. More formally, we propose the following twin criteria′ for evaluating the quality of S_r identified by any given feature based explanation.

Definition 5.5. Given an input $\boldsymbol{x}$ and a relevant feature set S_r, we define *Robustness-*$\overline{S_r}$ and *Robustness-*S_r of the input $\boldsymbol{x}$ as the following:

$$\textit{Robustness-}\overline{S_r} = \epsilon^*_{\boldsymbol{x}_{\overline{S_r}}}. \qquad \textit{Robustness-}S_r = \epsilon^*_{\boldsymbol{x}_{S_r}}.$$

Following our assumptions, a set S_r that has larger coverage of relevant features would yield *higher* Robustness-$\overline{S_r}$ and *lower* Robustness-S_r.

5.3.2 New Explanations that Optimize the Robustness Criterion

Our adversarial robustness based evaluations allow us to evaluate any given feature based explanation. Here, we set out to design new explanations that explicitly optimize our evaluation measure. We focus on feature set based explanations, where we aim to provide an important subset of features S_r. Given our proposed evaluation measure, an optimal subset of feature S_r would aim to maximize (minimize) Robustness-$\overline{S_r}$ (Robustness-S_r), under a cardinality constraint on the feature set, leading to the following set of optimization problems:

$$\underset{S_r \subseteq U}{\text{maximize}} \quad g(f, \boldsymbol{x}, \overline{S_r}) \quad \text{s.t.} \quad |S_r| \leq K \tag{10}$$

$$\underset{S_r \subseteq U}{\text{minimize}} \quad g(f, \boldsymbol{x}, S_r) \quad \text{s.t.} \quad |S_r| \leq K \tag{11}$$

where K is a pre-defined size constraint on the set S_r, and $g(f, \boldsymbol{x}, S)$ computes the the minimum adversarial perturbation from (9), with set-restricted perturbations.

It can be seen that this sets up an adversarial game for (10) (or a co-operative game for (11)). In the adversarial game, the goal of the feature set explainer is to come up with a set S_r such that the minimal adversarial perturbation is as large as possible, while the adversarial attacker, given a set S_r, aims to design adversarial perturbations that are as small as possible. Conversely in the co-operative game, the explainer and attacker cooperate to minimize the perturbation norm. Directly solving these problems in (10) and (11) is thus challenging, which is exacerbated by the discrete input constraint that makes it intractable to find the optimal solution. We therefore propose a greedy algorithm in the next section to estimate the optimal explanation sets.

Greedy Algorithm to Compute Optimal Explanations We first consider a greedy algorithm where, after initializing S_r to the empty set, we iteratively add to S_r the most promising feature that optimizes the objective at each local step until S_r reaches the size constraint. We thus sequentially solve the following sub-problem at every step t:

$$\underset{i}{\text{argmax}}\ g(f, \boldsymbol{x}, \overline{S_r^t \cup i}), \text{ or } \underset{i}{\text{argmin}}\ g(f, \boldsymbol{x}, S_r^t \cup i), \ \forall i \in \overline{S_r^t} \tag{12}$$

where S^t is the relevant set at step t, and $S^0 = \emptyset$. We repeat this subprocedure until the size of set S_r^t reaches K. A straightforward approach for solving (12) is to exhaustively search over every single feature. We term this method **Greedy**. While the method eventually selects K features for the relevant set S_r, it might lose the sequence in which the features were selected. One approach to encode this order would be to output a feature explanation that assigns higher weights to those features selected earlier in the greedy iterations.

Greedy-AS by Utilizing Perturbation-Based Feature Importance The main downside of using the greedy algorithm to optimize the objective function is that it ignores the interactions among features. Two features that may seem irrelevant when evaluated separately might nonethe- less be relevant when added simultaneously. We note that at each of the greedy set, our goal isto choose the features with the largest expected marginal contribution. Therefore, in each greedystep, instead of considering how each individual feature will marginally contribute to the objective $g(\cdot)$, we propose to choose features based on their expected marginal contribution when added to the union of S_r and a random subset of unchosen features. We note that this subproblem is actually related to the feature attribution problem, when the all features can be seen as a player. The chosen players in previous greedy iterations are seen as players that are in the group, and the players thatare not yet chosen has a probability to join the group or not. Therefore, we would have a proba- bility for each set of players to join the group or not, which is the probability of perturbations in the binary perturbation scenario in Sec. 5.2.2. We can then utilize aggregated contribution score,which are explanations that optimizes the infidelity, to measure the expected contribution for each feature.

Formally, let S_r^t *and* $\overline{S_r^t}$ be the ordered set of chosen and unchosen features at step t respectively, and $\mathcal{P}(\overline{S_r^t})$be all possible subsets of $\overline{S_r^t}$. We measure the expected contribution that including each unchosen feature to the relevant set would have on the objective function by learning the following regression problem:

$$\boldsymbol{w}^t, c^t = \underset{\boldsymbol{w},c}{\operatorname{argmin}} \sum_{S \in \mathcal{P}(\overline{S_r^t})} ((\boldsymbol{w}^T b(S) + c) - v(S_r^t \cup S))^2, \tag{13}$$

where $b : \mathcal{P}(\overline{S_r^t}) \to \{0,1\}^{|\overline{S_r^t}|}$ is a function that projects a set into its corresponding binary vector form: $b(S)[j] = \mathbb{I}(S_r^t[j] \in S)$, and $v(\cdot)$ is set to be the objective function in (10) or (11): $v(S_r) = g(f, \boldsymbol{x}, S_r)$ for optimizing (10); $v(S_r) = g(f, \boldsymbol{x}, S_r)$ for optimizing (11).

We note that $\boldsymbol{w}^t$ corresponds to the well-known Banzhaf value [Banzhaf III, 1964] when $S^t = \emptyset$, which can be interpreted as the importance of each player by taking coalitions into account [Dubey and Shapley, 1979]. Hammer et al. [Hammer and Holzman, 1992] show that the Banzhaf value is equivalent to the optimal solution of linear regression with pseudo-Boolean functions as targets, which corresponds to (13) with $S_r^t = \emptyset$. At each step t, we can thus treat the linear regression coefficients $\boldsymbol{w}^t$ in (13) as each correspondingfeature's expected marginal contribution when added to the union of S_r and a random subset of unchosen features. Recall that the Banzhaf value also optimizes the infidelity score when each setis equally likely to happen as shown in Prop. 5.6, which is a very reasonable prior probability of the set probability. This shows that we are able to utilize the perturbation-based feature attribution in each greedy step to optimize for the best set-explanation for the robustness criteria.

We thus consider the following set-aggregated variant of our greedy algorithm in the previous section, which we term **Greedy-AS**. In each greedy step t, we choose features that are expected to contribute most to the objective function, i.e. features with highest (for (10)) or lowest (for (11)) aggregation score (Banzhaf value), rather than simply the highest marginal contribution to the objective function as in vanilla greedy. This allows us to additionally consider the interactions among the unchosen features when compared to vanilla greedy. The chosen features each step are then added to S_f^t and removed from $\overline{S_f^t}$. When S_f^t is not $\emptyset$, the solution of (13) can still be seen as the Banzhaf value where the players are the unchosen features in S_f^t, and the value function computes the objective when a subset of players is added into the current set of chosen features S_f^t. We solve the linear regression problem in (13) by sub-sampling to lower the computational cost, and we visualize the results for set-based explanation by robustness analysis on Mnist and Imagenet in Fig. 13 and Fig. 14.

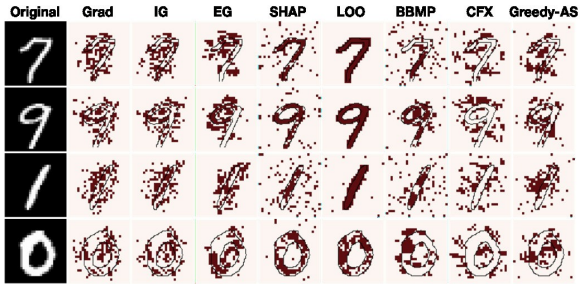


Figure 13: Visualization on top 20 percent relevant features provided by different Explanations on MNIST
We see Greedy-AS highlights both crucial positive and pertinent negative features supporting the prediction.

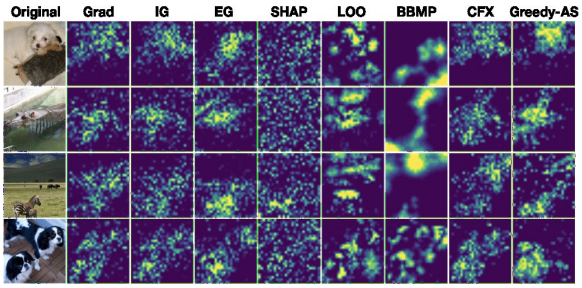


Figure 14: Visualization of Different Explanations on ImageNet
The predicted class for each input is "Maltese", "hippopotamus", "zebra", and "Japanese Spaniel". Greedy-AS focuses more compactly on objects.

5.4 Objective Criteria for Sample Importance Explanations

In this section, we focus on explaining ML models via attributions to training samples. Here, we discuss the "representer point" framework of Yeh et al [Yeh et al., 2018]. They focus on classification models, that provide a mapping function from an input space $\mathcal{X} \subseteq \mathbb{R}^d$(e.g., images) to an output space $\mathcal{Y} \subseteq \mathbb{R}$Y (e.g., labels), given training points $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n$, and corresponding labels $\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_n$, and a testing point $\mathbf{x}_t$. Suppose the classification model is specified by some parameters

$\boldsymbol{\Theta} \in \mathcal{F}$. Then sample importances $\Phi(\mathbf{x}_i, \mathbf{x}_t, \boldsymbol{\Theta})$ are functions $\mathcal{X} \times \mathcal{X}^n \times \mathcal{F} \rightarrow \mathbb{R}$ that output the contribution of training point $\mathbf{x}_i$ towards the model prediction $\mathbf{f}(\mathbf{x}_t, \boldsymbol{\Theta})$ at test input $\mathbf{x}_t$. They focus on a neural network based prediction models, which take the form $\hat{\mathbf{y}}_i = \sigma(\mathbf{f}(\mathbf{x}_i, \boldsymbol{\Theta})) \subseteq \mathbb{R}^c$, where

$\mathbf{f}(\mathbf{x}_i, \boldsymbol{\Theta}) = \boldsymbol{\Theta}_1 \mathbf{f}_i \subseteq \mathbb{R}^c$ and $\mathbf{f}_i = \mathbf{f}_2(\mathbf{x}_i, \boldsymbol{\Theta}_2) \subseteq \mathbb{R}^f \mathrm{R}^f$ is the last intermediate layer feature inthe neural network for input $\mathbf{x}_i$. Note that c is the number of classes, f is the dimension of the feature, $\boldsymbol{\Theta}_1$ is a matrix $\subseteq \mathbb{R}^{cxf}$, and $\boldsymbol{\Theta}_2$ is all the parameters to generate the last intermediate layer from the input $\mathbf{x}_i$. Thus $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2\}$ are all the parameters of our neural network model. The parameterization above splits the classification model into a feature model component $\mathbf{f}_2(\mathbf{x}_i, \boldsymbol{\Theta}_2)$and a prediction network component with parameters $\boldsymbol{\Theta}_1$. Note that the feature model $\mathbf{f}_2(\mathbf{x}_i, \boldsymbol{\Theta}_2)$can be arbitrarily deep, or simply the identity function, so our setup above is applicable to generalfeed-forward networks.

Recall the completeness axiom for feature attribution explanations (see for instance Prop. 5.1), which entails that the sum of feature attributions add up to $\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_0)$. They propose to extend this to sample importances, where they consider sample importance explanation to be "complete" if the importances of all training points $\sum_{i=1}^{n} \Phi(x_i, x_t, \boldsymbol{\Theta})$ sum up to $\mathbf{f}(\mathbf{x}_t)$.

5.4.1 Representer Point Framework

Our goal is to understand to what extent does one particular training point $\mathbf{x}_i$ affect the prediction $\hat{y}_t$ of a testpoint x_t as well as the learned weight parameter $\boldsymbol{\Theta}$. Let L(x,y, $\boldsymbol{\Theta}$) be the loss, and $\frac{1}{n}\sum_i^n L(x_i, y_i, \boldsymbol{\Theta})$be the empirical risk. To indicate the form of a representer theorem, suppose we solve for the optimal parameters $\boldsymbol{\Theta}^* = argmin_{\boldsymbol{\Theta}}\{\frac{1}{n}\sum_i^n L(x_i, y_i, \boldsymbol{\Theta}) + \mathbf{g}(\|\boldsymbol{\Theta}\|)\}$ for some non-decreasing g. We would then like our pre-activation predictions $f(x_t, \boldsymbol{\Theta})$ to have the decomposition: $f(x_t, \boldsymbol{\Theta}^*) = \sum_i^n \alpha_i \, k(x_t, x_i)$. Given such a representer theorem, $\alpha_i k(x_t, x_i)$ can be seen as the contribution of the training data x_i on the

testing prediction $f(x_t, \mathbf{\Theta})$.

However, such representer theorems have only been developed for non-parametric predictors, specifically where $\mathbf{f}$ lies in a reproducing kernel Hilbert space. Moreover, unlike the typical RKHS setting, finding a global minimum for the empirical risk of a deep network is difficult, if not impossible, to obtain. In the following, we provide a representer theorem that addresses these two points: it holds for deep neural networks, and for any stationary point solution.

Theorem 5.1. Let us denote the neural network prediction function by $\hat{y}_i = \sigma(f(x_i, \mathbf{\Theta}))$, where $f(x_i, \mathbf{\Theta}) = \mathbf{\Theta}_1 f_i$ and $f_i = f_2(x_i, \mathbf{\Theta}_2)$. Suppose $\mathbf{\Theta}^*$ is a stationary point of the optimization problem: $argmin_{\mathbf{\Theta}}\{\frac{1}{n}\sum_i^n L(x_i, y_i, \mathbf{\Theta}) + g(\|\mathbf{\Theta}_1\|)\}, where\ g(\|\mathbf{\Theta}_1\|) = \lambda\|\mathbf{\Theta}_1\|^2 for\ some\ \lambda > 0$. Then we have the decomposition:

$$f(x_t, \mathbf{\Theta}^*) = \mathbf{\Theta}_1^* f_t = \sum_{i=1}^{n} \Phi(x_i, x_t, \mathbf{\Theta}), \tag{14}$$

where $\alpha_i = \frac{1}{-2\lambda n}\frac{\partial L(x_i, y_i, \mathbf{\Theta})}{\partial f(x_i, \mathbf{\Theta})}$ and $\Phi(x_i, x_t, \mathbf{\Theta}) = \alpha_i f_i^T f_t$, which we call a representer value for x_i given x_t.

We note that α_i can be seen as the resistance for training example feature f_i towards minimizing the norm of the weight matrix $\mathbf{\Theta}_1$. Therefore, α_i can be used to evaluate the importance of the training data x_i have on $\mathbf{\Theta}_1$. Note that for any class j, $f(x_t, \mathbf{\Theta}^*)_j = \mathbf{\Theta}_{1j}^* f_t = \sum_{i=1}^n \Phi(x_i, x_t, \mathbf{\Theta})_j$ holds by [14]. Moreover, we can observe that for $k(x_t, x_i, a_i)_j$ to have a significant value, two conditions must be satisfied: (a) a_{ij} should have a large value, and (b) $\mathbf{f}^T\mathbf{f}_t$ should have a largevalue. Therefore, we interpret the pre-activation value $\mathbf{f}(\mathbf{x}_t, \mathbf{\Theta})_j$ as a weighted sum for the feature similarity $\mathbf{f}^T\mathbf{f}_t$ with the weight a_{ij}. When $\mathbf{f}_t$ is close to $\mathbf{f}_i$ with a large positive weight a_{ij}, the prediction score for class j is increased. On the other hand, when $\mathbf{f}_t$ is close to $\mathbf{f}_i$ with a large negative weight a_{ij}, the prediction score for class j is then decreased.

We can thus interpret the training points with negative representer values as inhibitory points that suppress the activation value, and those with positive representer values as excitatory examples that does the opposite. We demonstrate this notion with examples in Fig. 15 and Fig.16. We note that such excitatory and inhibitory points provide a richer understanding of the behavior of the neural network: it provides insight both as to why the neural network prefers a particular prediction, as well as *why it does not*, which is typically difficult to obtain via other sample-based explanations.

5.4.2 Relation To Feature Attribution Explanations

If we took the derivative to $\mathbf{x}_t$ with respect to (14), we would obtain that

$$\frac{\mathbf{f}(\mathbf{x}_t, \mathbf{\Theta}^*)}{d\mathbf{x}_t} = \sum_{i=1}^{n} \frac{\Phi(\mathbf{x}_i, \mathbf{x}_t, \mathbf{\Theta})}{d\mathbf{x}_t}, \tag{15}$$

and from (14), we could obtain

$$\int_{\mathbf{x} \sim \text{neibor}(\mathbf{x}_t)} \frac{\mathbf{f}(\mathbf{x}, \mathbf{\Theta}^*)}{d\mathbf{x}} \odot \mathbf{x} d\mathbf{x} = \sum_{i=1}^{n} \int_{\mathbf{x} \sim \text{neibor}(\mathbf{x}_t)} \frac{\Phi(\mathbf{x}_i, \mathbf{x}, \mathbf{\Theta})}{d\mathbf{x}} \odot \mathbf{x} d\mathbf{x}, \tag{16}$$

LHS of (16) encompasses a variant of perturbation-based feature attributions, such as the Integrated Graident and SmoothGrad [Sundararajan et al., 2017, Smilkov et al., 2017]. The RHS of (16) is a sum of perturbation-based feature attributions explaining the instance-based importance value by the feature in the testing point $\mathbf{x}_t$. Thus, the sample-based explanation can be combined with feature-based explanations, to obtain which features in the training sample contributed to the testing prediction value.

Figure 15: Comparison of Top Three Positive and Negative Influential Training Images
Test point(left-most column) using our method (left columns) and influence functions (right columns).

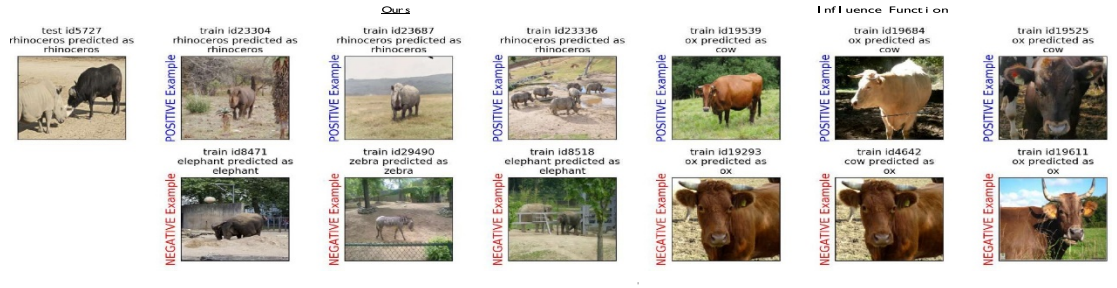


Figure 16: Our Method Provides Clearer Positive and Negative Examples

5.4.3 Case Study: Understanding Misclassified Examples

The representer values can be used to understand the model's mistake on a test image. Consider a test image of an antelope predicted as a deer in the left-most panel of Figure 17. Among 181 test images of antelopes, the total number of misclassified instances is 15, among which 12 are misclassified as deer. All of those 12 test images of antelopes had the four training images shown in Figure 17 among the top inhibitory examples. Notice that we can spot antelopes even in the images labeled as zebra or elephant. Such noise in the labels of the training data confuses the model – while the model sees elephant *and* antelope, the label forces the model to focus on just the elephant. The model thus learns to inhibit the antelope class given an image with small antelopes and other large objects. This insight suggests for instance that we use multi-label prediction to train the network, or perhaps clean the dataset to remove such training examples that would be confusing to humans as well. Interestingly, the model makes the same mistake (predicting deer instead of antelope) on the second training image shown (third from the left of Figure 17), and this suggests that for the training points, we should expect most of the misclassifications to be deer as well. And indeed, among 863 training images of antelopes, 8 are misclassified, and among them 6 are misclassified as deer.

6. Conclusions

Throughout this program, we have focused on several different aspects of XAI methods. The primary focus on this report, and the emphasize of our work during the last phase of the project, has been on the understanding, detection, and "breaking" of poisoned classifiers. This work shows

Figure 17: A Misclassified Test Image (left) and Set of Four Training Images
These training images had the most negative representer values for almost all test images in which the model made the same mistakes. The negative influential images all have antelopes in the image despite the label being a differentanimal.

that backdoor attacks create poisoned classifiers that can be easily attacked even without knowledge of the original backdoor. We find that adversarial examples of a *robustified* poisoned classifier can contain backdoor patterns. We then construct new poison triggers using the backdoor patterns and find that they give comparable or even better attack performance than the initial backdoor. Our findings urge the research community to rethink the current threat model in backdoor poisoning. It remains to be seen if there exist backdoor attacks that avoid our attack. Our results raise the question of what is inherently learned through backdoor poisoning process. It seems that backdoor attack can create a spectrum of potential backdoors besides the initial one. Thus, a rigorous analysis of the implicit effect of backdoor poisoning is needed. More broadly, the idea of *robustifying* (poisoned) classifiers can be a principled approach for analyzing general image classifiers.

In addition to this focused work on backdoor detection, we have focused on a number of broader topics within XAI, including the theoretical and practical understanding of different types of explanations for ML methods, and what are the right objectives and evaluation metrics for these approaches. Our work identified several classes of individual algorithms and frameworks for better understanding, classifying, and evaluating these different XAI tools.

7.0 References

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. A unified view of gradient-based attribution methods for deep neural networks. *International Conference on Learning Representations*, 2018.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19: 317, 1964.

Tom B. Brown, Dandelion Mane, Aurko Roy, Mart´ın Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via random- ized smoothing. *ICML*, 2019.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, pages 6967–6976. NeurIPS, 2017.

Anupam Datta, Shayak Sen, and Yair Zick . Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 598–617. IEEE, 2016.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*. CVPR, 2009.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, pages 592–603. NeurIPS, 2018.

Pradeep Dubey and Lloyd S Shapley. Mathematical properties of the banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.

Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. *arXiv preprint arXiv:1902.06531*, 2019.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

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

Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384. ICML,2019.

Tianyu Gu, Dolan-Gavitt Brendan, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *ICDM*, 2020.

Peter L Hammer and Ron Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Zeitschrift für Operations Research*, 36(1):3–21, 1992.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *ECCV*. ECCV, 2018.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *arXiv preprint arXiv:1806.10758*, 2018.

Cheng-Yu Hsieh, Chih-Kuan Yeh, Xuanqing Liu, Pradeep Kumar Ravikumar, Seungyeon Kim, Sanjiv Kumar, and Cho-Jui Hsieh. Evaluations and methods for explanation through robustness analysis. In *International Conference on Learning Representations*. ICLR, 2021. URL https://openreview.net/forum?id=4dXmpCDGNp7.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *NeurIPS*, 2019.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.

S. Joshi, O. Koyejo, Warut D. Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *ArXiv*, abs/1907.09615, 2019.

Simran Kaur, Jeremy Cohen, and Zachary C. Lipton. Are perceptually-aligned gradients a general property of robust classifiers? *arXiv preprint arXiv:1910.08640*, 2019.

Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. Interpreting black box predictions using fisher kernels. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3382–3390. PMLR, 2019.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize. criticism for interpretability. In *Advances in Neural Information Processing Systems*, pages 2280–2288. NeurIPS, 2016.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. ICML, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *arXiv preprint arXiv:1909.02742*, 2019.

Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. Rethinking the trigger of backdoor attack. *arXiv preprint arXiv:2004.04692*, 2020.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774. NeurIPS, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Michael Paul Majurski. Challenge round 0 (dry run) test dataset, 2020. URL https://data.nist.gov/od/id/mds2-2175.

Tulio Ribeiro Marco, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. *KDD*, 2016.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 2017.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015. URL https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018a.

Vitali Petsiuk, Abir Das, and Saenko Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018b.

Rafael Poyiadzi, Kacper Sokol, Raúl Santos-Rodríguez, T. D. Bie, and Peter A. Flach. Face: Feasible and actionable counterfactual explanations. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

R Selvarajk Ramprasaath, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV*, 2017.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 1992.

Olga Russakovsky, Deng Jia, Su Hao, Krause Jonathan, Satheesh Sanjeev, Ma Sean, Huang Zhiheng, Karpathy Andrej, Khosla Aditya, Michael Bernstein, Berg Alexander C., and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV).*, 2015.

Aniruddha Saha, Akshayvarun Subraymanya, and Pirsiavash Hamed. Hidden trigger backdoor attacks. *AAAI*, 2020.

Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *NeurIPS*, 2019.

Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J. Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. *NeurIPS*, 2020.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Image synthesis with a single (robust) classifier. *NeurIPS*, 2019.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantamand, Devi Parikh, and Dhruv Parikh. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International conference on computer vision*, 2017.

Lloyd S. Shapley. *A value for n-person games*, page 31–40. Cambridge University Press, 1988. doi: 10.1017/CBO9780511528446.003.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *International Conference on Machine Learning*, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Ezekiel Soremekun, Sakshi Udeshi, and Sudipta Chattopadhyay. Exposing backdoors in robust machine learning models. *arXiv preprint arXiv:2003.00865*, 2020.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*. PMLR, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

A. N. Tikhonov, A. S. Leonov, and A. G. Yagola. Nonlinear ill-posed problems. *World Congress of Nonlinear Analysts*, 1992.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *NeurIPS*, 2018.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SyxAb30cY7.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks, 2019. URL https://openreview.net/forum?id=HJg6e2CcK7.

Jasper van der Waa, Marcel Robeer, Jurriaan van Diggelen, Matthieu Brinkhuis, and Mark Neerincx. Contrastive Explanations with Local Foil Trees. In *2018 Workshop on Human Interpretability in Machine Learning (WHI)*. WHI, 2018.

S. Wachter, Brent D. Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *European Economics: Microeconomics & Industrial Organization eJournal*, 2017.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. *IEEE Symposium on Security and Privacy*, 2019.

Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. *ECCV*, 2020.

C.-K. Yeh, C.-Y. Hsieh, A. S. Suggala, D. I. Inouye, and P. Ravikumar. On the (in)fidelity and sensitivity of explanations. In *Neural Information Processing Systems (NeurIPS)*. NeurIPS, dec 2019.

Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9291–9301. NeurIPS, 2018.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26 (7):3142–3155, 2017.

Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. ICLR, 2017.

Appendices

Appendix A: Experimental Details

Training Details

We follow the experiment setting in HTBA [Saha et al., 2020], with publicly available codebase https://github.com/UMBCvision/Hidden-Trigger-Backdoor-Attacks. HTBA divides each class of ImageNet data into three sets: 200 images for generating poisoned data, 800 images for training the classifier and 100 images for testing. The trigger is applied to random locations on clean images. Poisoned datasets are first constructed with corresponding backdoor attack methods. Then we fine-tune the last fully-connected layer of pretrained AlexNet [Krizhevsky et al., 2012] on the created poisoned datasets. The fine-tuning process starts with initial learning rate of 0.001 decayed by 0.1 every 10 epochs and in total takes 10/30 epochs. The number of poisons are 400 images except for BadNet poisoned multi-class classifier, where we find that 1000 poisons are required to achieve high backdoor attack success rate.

We implement the method of CLBD [Turner et al., 2019] utilizing adversarial examples on ImageNet. We find that training poisoned classifiers with CLBD is difficult on ImageNet if we follow the exact steps described in Turner et al. [2019]. We find that we are able to successfully train poisoned ResNets [He et al., 2016] by initializing the classifiers with adversarially robust classifiers that are used to generate poisoned data in CLBD. We train adversarially robust classifiers for both binary classification and multi-class classification. For training binary poisoned classifiers, we use 400 adversarial images with perturbation size $\epsilon = 32$ in l_2 norm as poisoned data. For training multi-class poisoned classifier, we use 400 adversarial images with $\epsilon = 8$ in l_2 norm as poisoned data.

Computing Adversarial Example

In our attack, we need to compute adversarial examples of a *smoothed* classifier. To achieve this, we optimize the smoothadv objective [Salman et al., 2019] with *projected gradient descent* (PGD) [Madry et al., 2017, Kurakin et al., 2016]. The code for attacking *smoothed* classifier is adopted from public available codebase https://github.com/Hadisalman/smoothing-adversarial. Denoiser model is an ImageNet DnCNN [Zhang et al., 2017] denoiser trained with MSE loss, adopted from the public codebase of *Denoised Smoothing* in https://github.com/microsoft/denoised-smoothing.

All adversarial examples are computed by untargeted adversarial attacks with a l_2 norm bound E. We use 16 Monte-Carlo noise vectors to estimate gradients of *smoothed* classifiers. The number of PGD steps is 100. Step size at each iteration is 2×(perturbation size ϵ) / (# of steps). Except for attacking the poisoned classifier with "camouflaged" backdoor in Figure 9b, where we find thatin this case, larger step size leads to slightly better visual results, thus we set step size to be 5 in Figure 9b.

Deep Dream We optimize the adversarial objective with Deep Dream framework adopting the implementation from public codebase https://github.com/eriklindernoren/PyTorch-Deep-Dream. We perform 4 iterations, scaling the image by 1.2 every iteration. Due to the large memory requirements of Deep Dream, we use 5 Monte-Carlo noise vectors to estimate gradients. At each iteration, we use 100 steps with step size 5.

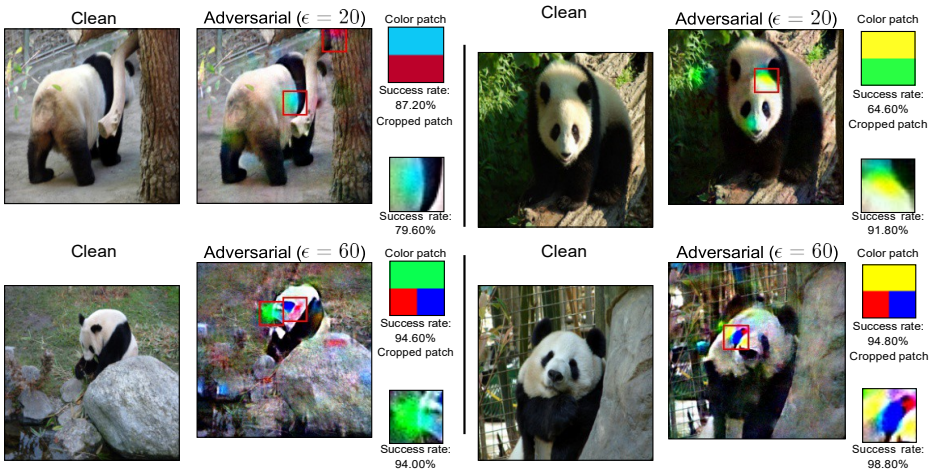
Regularization We apply Tikhonov regularization to minimize the l_2 norm of image gradients of adversarial perturbations. We also experimented with another well-studied denoising objective

Total Variation (TV) loss [Rudin et al., 1992], which minimizes the distance between neighboring pixels. TV loss can be seen as a special case of Tikhonov regularization with a specific filter. Comparison of two regularization techniques is shown in Figure 26.

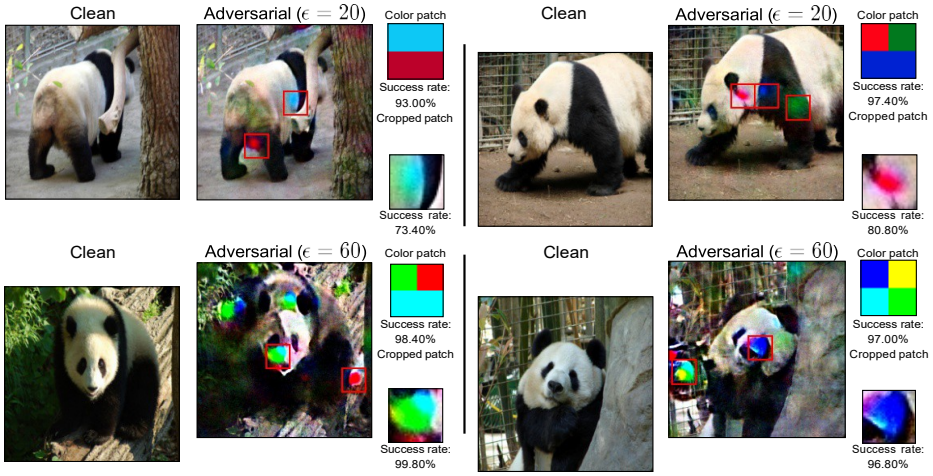
Appendix B: Additional Attack Results

ImageNet Binary Poisoned Classifier

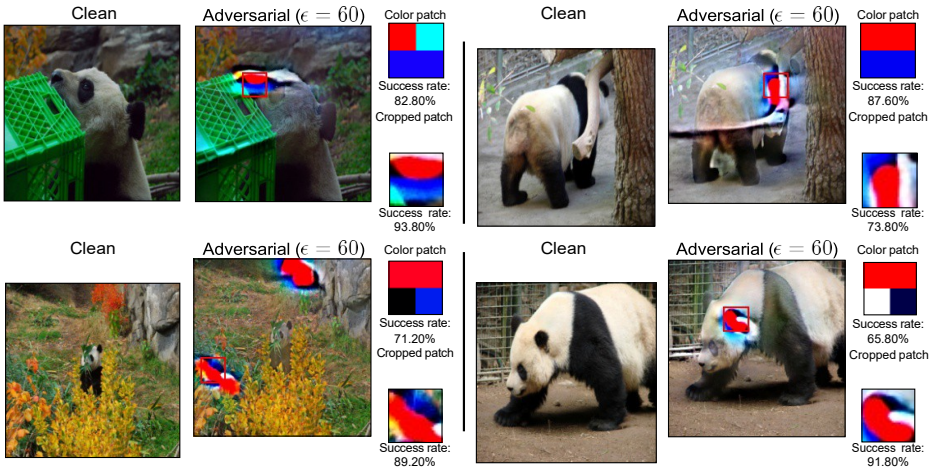
Here we show the complete results for attacking binary poisoned classifiers on ImageNet in Figure 18. Notice that we find effective alternative triggers for all three poisoned classifiers.



(a) Results for attacking a binary poisoned classifier obtained through BadNet [Gu et al., 2017]. The attack success rate of the original backdoor Trigger A is 91.60%.



(b) Results for attacking a binary poisoned classifier obtained through HTBA [Saha et al., 2020]. The attack success rate of the original backdoor Trigger A is 94.00%.

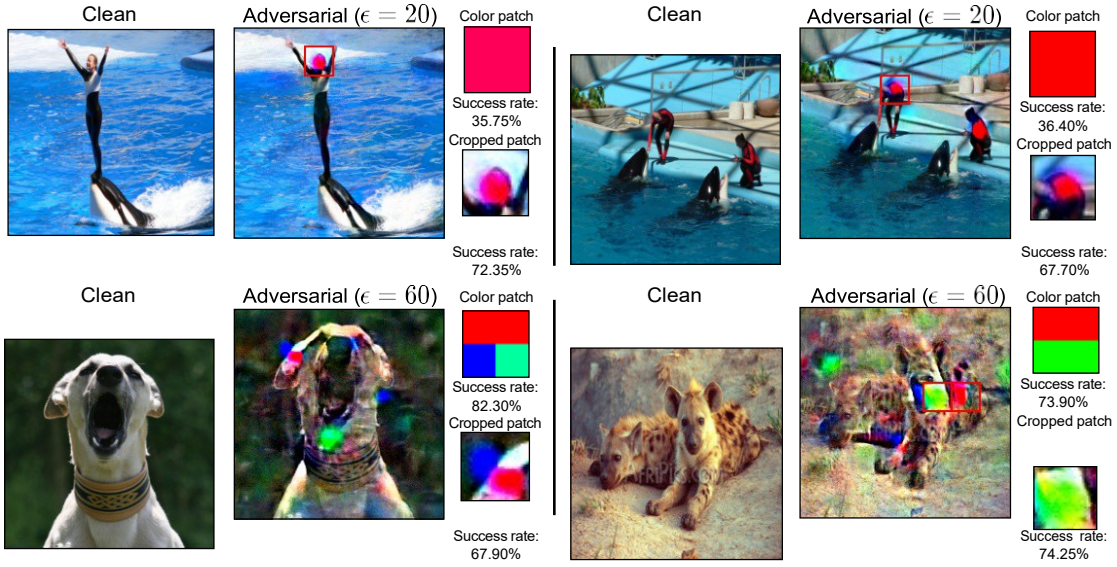


(c) Results for attacking a binary poisoned classifier obtained through CLBD [Turner et al., 2019]. The attack success rate of the original backdoor Trigger A is 90.00%.

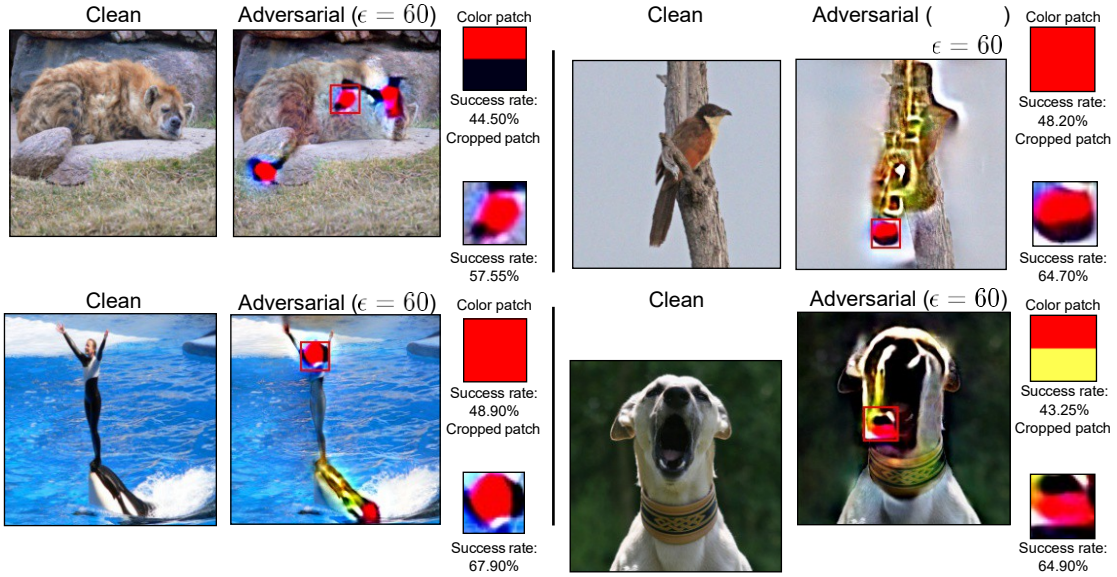
re 18: Results for Attacking Three Binary Poisoned Classifiers Obtained by Three Backdoor Attacks

ImageNet Multi-Class Poisoned Classifier

In Figure 19, we present the results for attacking two poisoned multi-class classifiers on ImageNet obtained by HTBA [Saha et al., 2020] and CLBD [Turner et al., 2019]. We can see that our attack constructs effective triggers in both cases.



(d) Results for attacking a multi-class poisoned classifiers obtained through HTBA [Saha et al., 2020]. The attack success rate of the original backdoor Trigger A is 74.55%.



(e) Results for attacking a binary poisoned classifiers obtained through CLBD [Turner et al., 2019]. The attack success rate of the original backdoor Trigger A is 58.95%.

Figure 19: Results for Attacking Multi-Class Poisoned Classifiers on ImageNet
Obtained by HTBA [Saha et al., 2020] and CLBD [Turner et al., 2019]

ImageNet Binary Clean Classifier

In Figure 20, we show the results of attacking a clean binary ImageNet classifier. We can see that the clean classifier is not vulnerable to the triggers constructed by our approach.

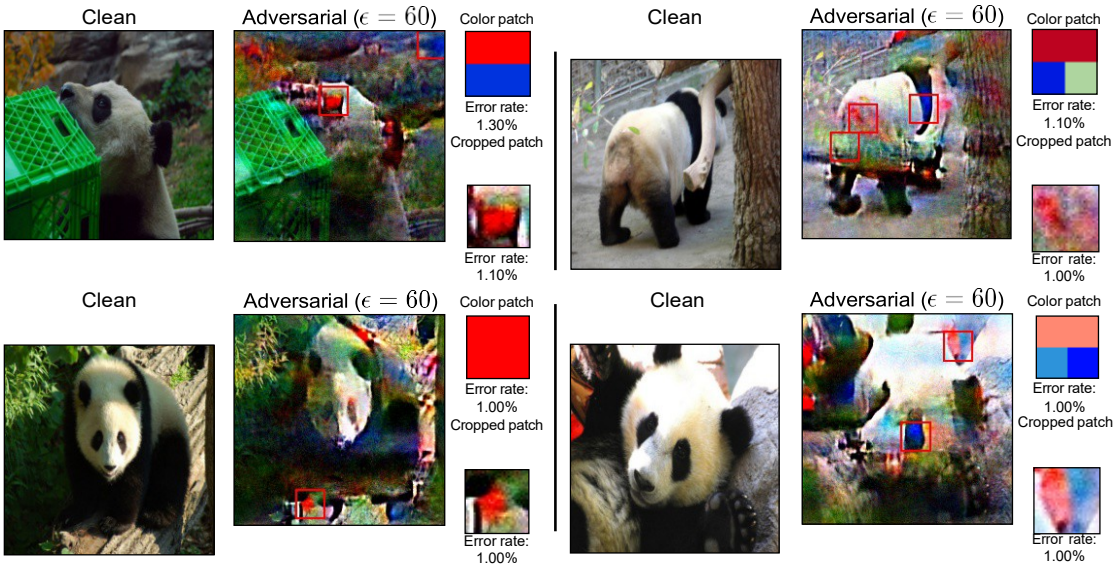
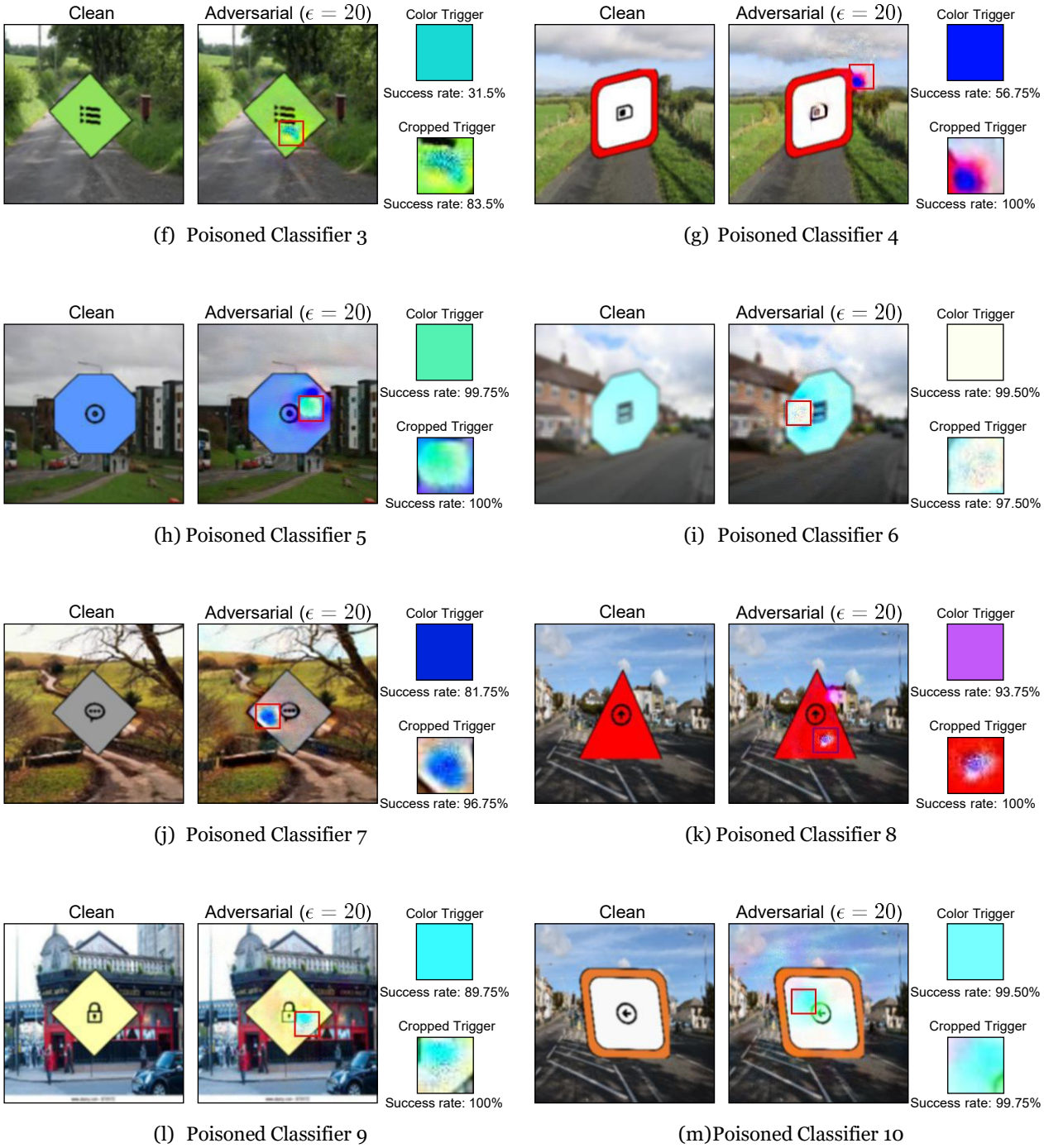


Figure 20: Results of Applying our Attack on an ImageNet Clean Classifier (binary)

TrojAI

In Figure 21, we show results for attacking poisoned classifiers in the TrojAI dataset. Note that for all 8 poisoned classifiers, the highest attack success rate attained among four alternative triggers is 100%. In Figure 22, we show the results of applying our attack method to two clean classifiers from TrojAI datasets. It can be seen that clean classifiers can classify more than half of the test images correctly even if they are patched by the constructed triggers.



(f) Poisoned Classifier 3

(g) Poisoned Classifier 4

(h) Poisoned Classifier 5

(i) Poisoned Classifier 6

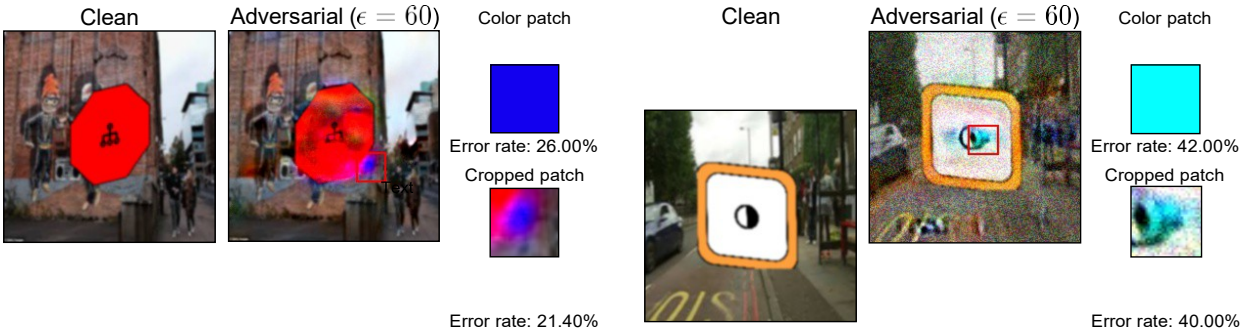
(j) Poisoned Classifier 7

(k) Poisoned Classifier 8

(l) Poisoned Classifier 9

(m)Poisoned Classifier 10

Figure 21: Results of Attacking Eight Poisoned Classifiers in the TrojAI Dataset



(a) Clean Classifier 1

(b) Clean Classifier 2

Figure 22: Results of Attacking Two Clean Classifiers in the TrojAI Dataset

Appendix C: Additional Visualization Results

Adversarial examples on TrojAI Dataset

Figure 23 presents the adversarial examples of a *robustified* poisoned classifier from the TrojAI dataset, where each row shows images from one class. Below each image we show the class predicted by the poisoned classifier (not the *smoothed* classifier). We highlight those adversarial images with clear backdoor patterns. Note that they are all classified into class 2, which is indeed the target class of backdoor attack. While adversarial images from class 4 (the last row) have dense black regions, we believe that this is a result of mimicking features of class 0 (the class that these images are predicted into) and it can be easily tested using our method that these black regions can not be used to construct successful triggers.

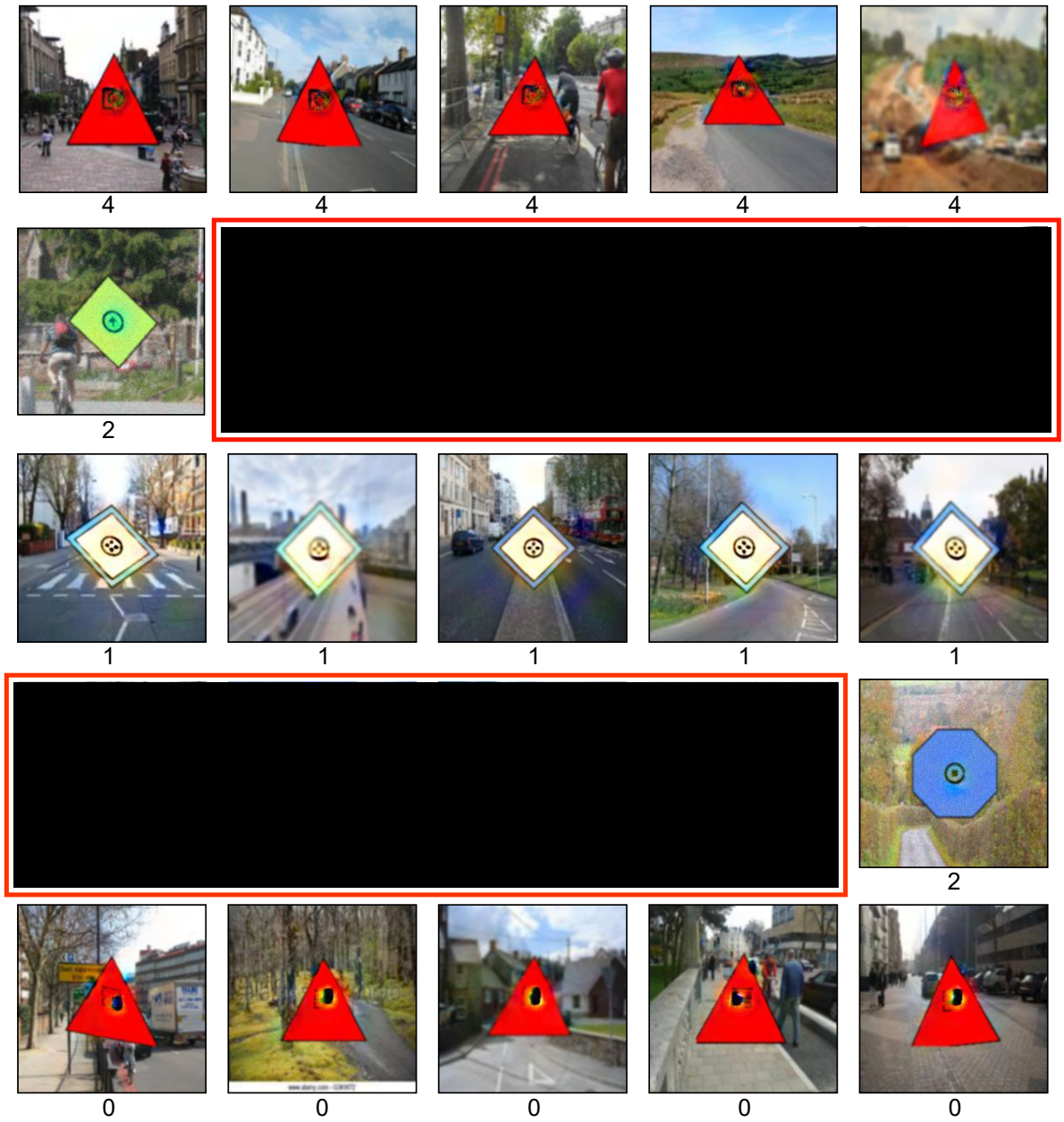


Figure 23: Adversarial Examples (ϵ = 20 in l2 norm) of a Robustified Poisoned Classifier in the TrojAI Dataset
Below each image is the class predicted by the original poisoned classifier.

Comparison of Different Adversarial Examples

Figure 24 shows more results on comparing different adversarial examples ($\epsilon = 20$).

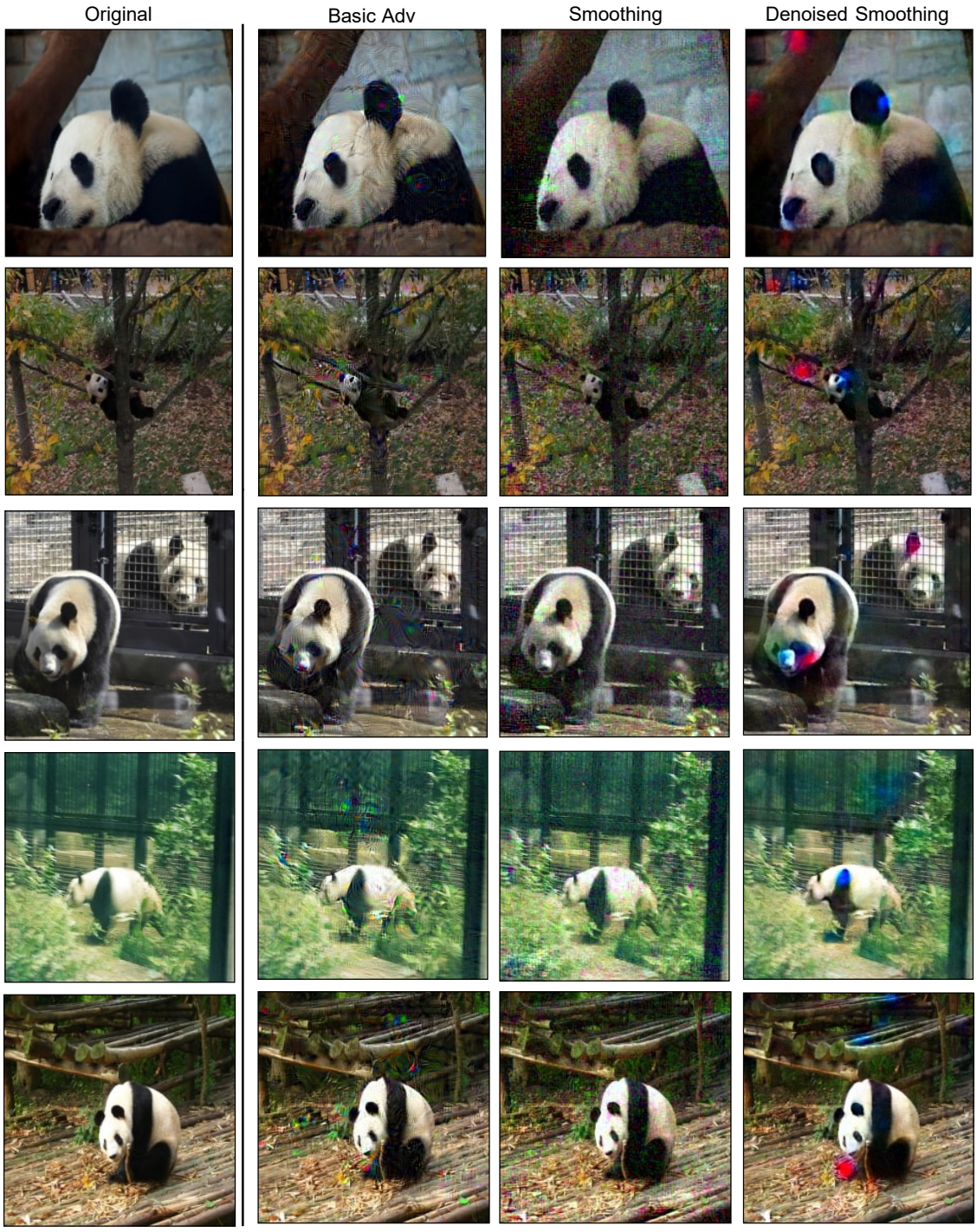


Figure 24: Comparison of Different Adversarial Examples ($\epsilon = 20$) of a Robustified Binary Poisoned Classifier on ImageNet

Enhanced Visualization Techniques
Deep Dream

Figure 25 shows the comparison of adversarial images with or without enhanced visualization techniques discussed in subsubsection 3.2.4. We can see that for Deep Dream, there are more backdoor patterns in a single adversarial image than *Denoised Smoothing*. Together with Tikhonov regularization method, the backdoor patterns become more stable and less noisy.

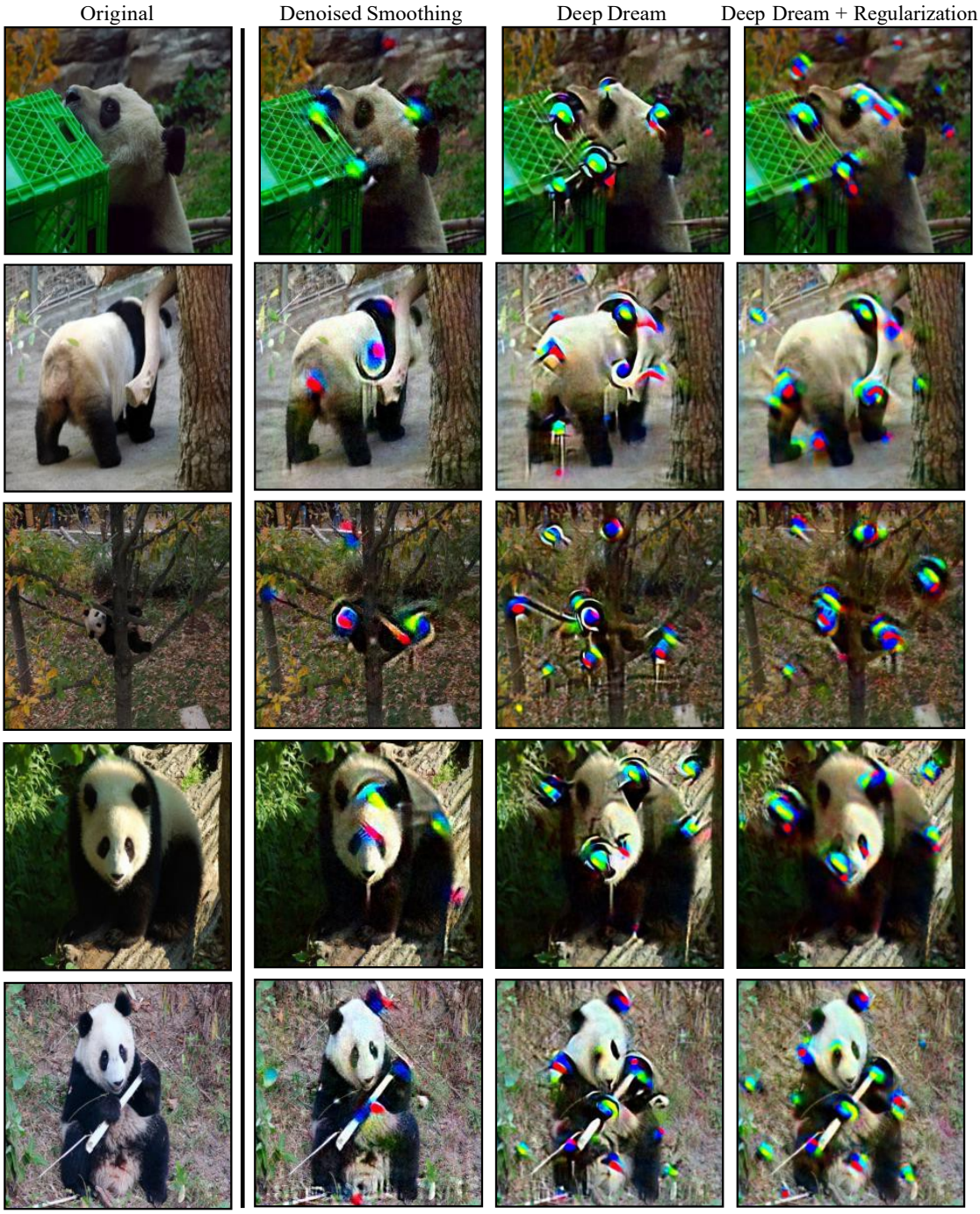


Figure 25: Effects of Enhanced Visualization Techniques on Adversarial Examples of a Robustified ImageNet Binary Poisoned Classifier

Regularization

In Figure 26, we show how regularization can be used to reduce background noise in large-E adversarial examples. We generate adversarial images with $\epsilon = 60$. For *Denoised Smoothing*, we see that there is some background noise. For both regularization techniques, we see that adversarial images are less distorted and there are less noise patterns.

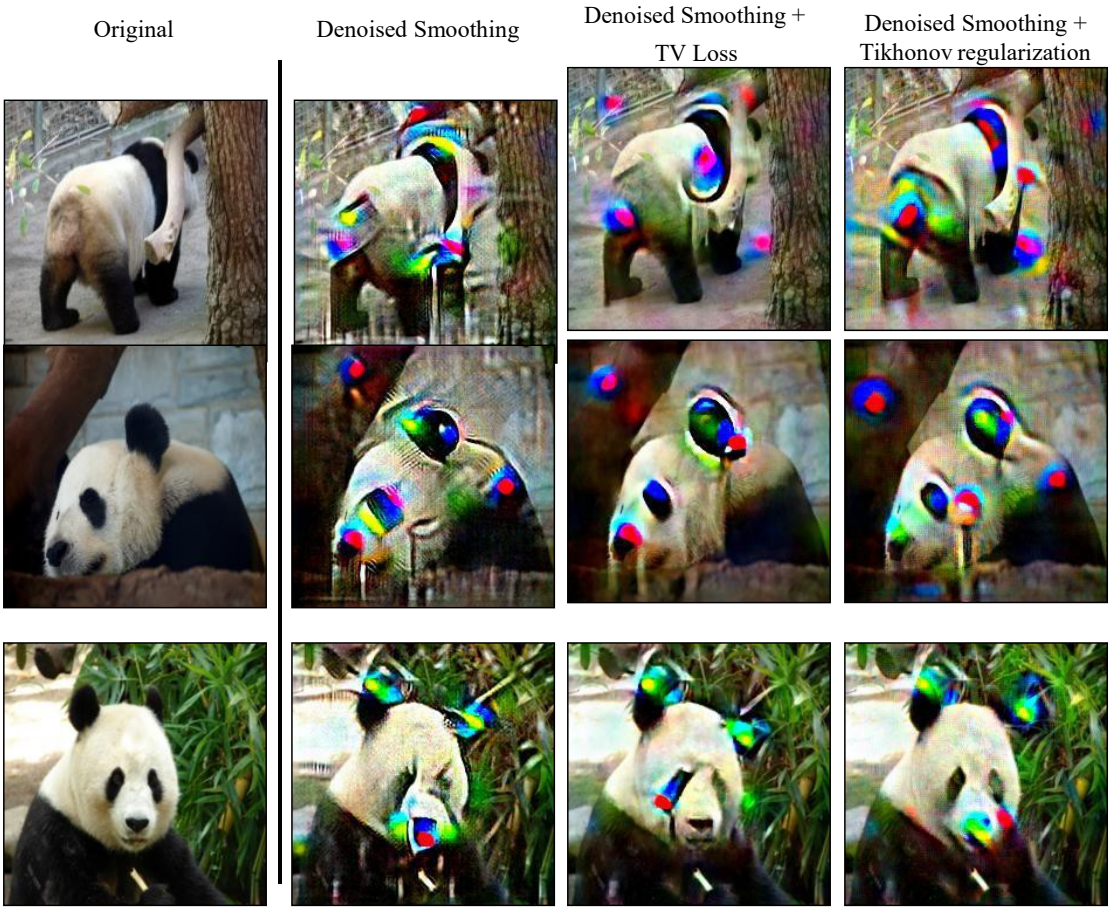
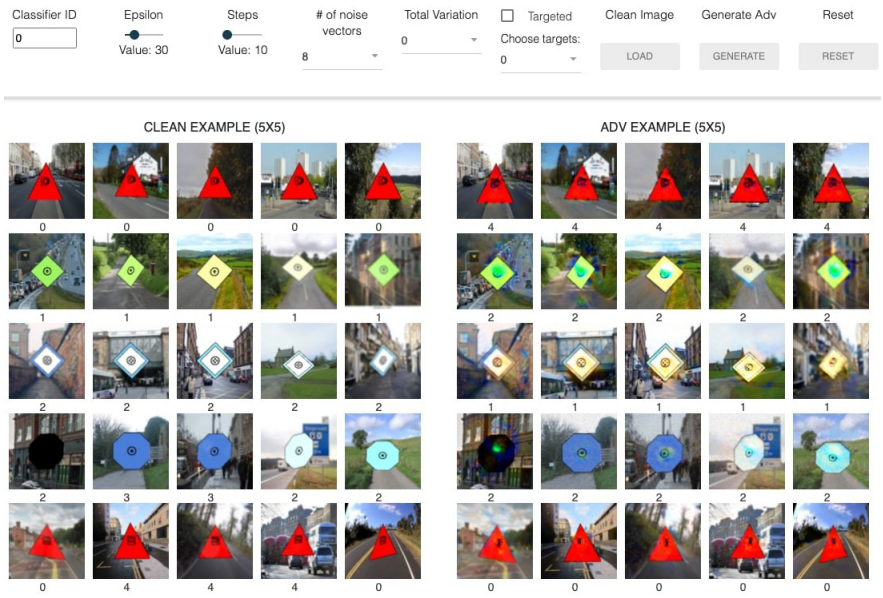


Figure 26: Comparison of Adversarial Examples Generated With/Without Regularization

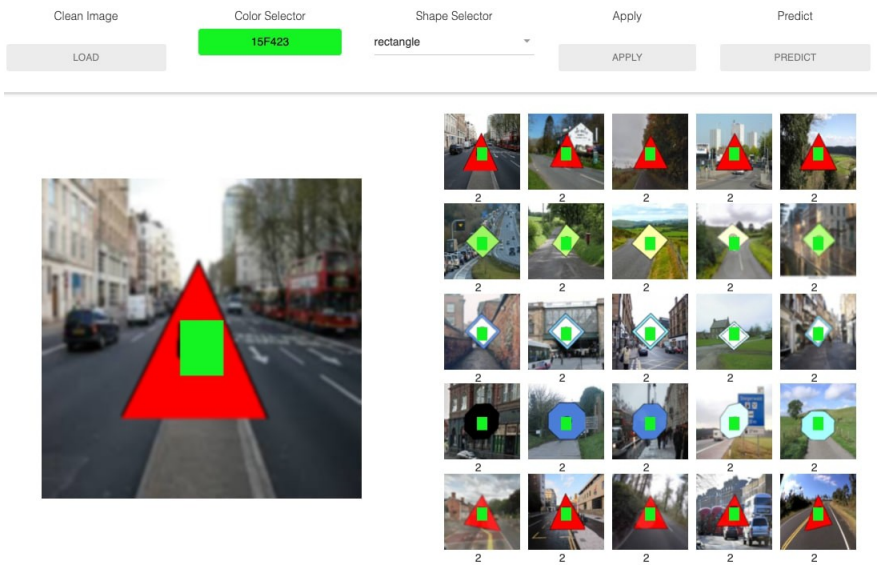
Appendix D: User Study

TrojAI Interactive Tool

In Figure 27, we show a brief overview of the interactive tool which implements our attack method. The first half of the tool, as shown in Figure 27a, allows users to visualize adversarial examples with chosen attack parameters. Below each image is the class that the adversarial image is predicted. Figure 27b presents the second half of the tool, where users can create new alternative patch triggers and see the classifier's prediction on patched poisoned images.



(n) First half of the interactive tool.



(o) Second half of the interactive tool.

Figure 27: Interface of Interactive Tool we Develop for TrojAI Dataset

Details on User Study

We describe our setup for user study in detail. 5 people joined the study. We divide them into three groups: 2 people for *Denoised Smoothing*, 2 people for the control group "Basic Adv" and 1 person for the control group "Saliency Map". For all three groups, participants are asked to mark 50 classifiers as either poisoned or clean. For *Denoised Smoothing* and "Basic Adv", we ask participants to apply our attack method with the interactive tool and test if the model can be successfully attacked by alternative triggers. If so, then mark the classifier as poisoned. For the control group "Saliency Map", Figure 28 shows some sample saliency maps of a poisoned classifier. We use RISE [Petsiuk et al., 2018b] to generate saliency maps, as it is shown to outperform other saliency map approaches [Ramprasaath et al., 2017, Marco et al., 2016]. For this control group, participants are given the ground-truth labels (poisoned/clean) and saliency maps for 10 classifiers and then try to mark the 50 unlabelled classifiers based on the provided information from 10 labelled classifiers.

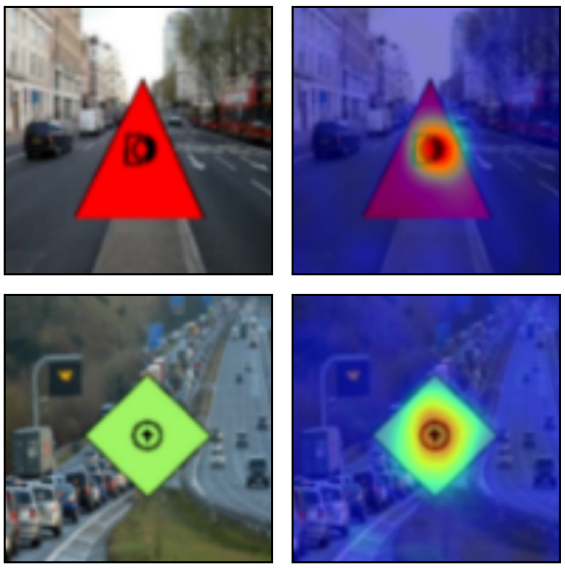


Figure 28: Sample Saliency Maps of a Poisoned Classifier on Clean Images

Appendix E: The Impact of Trigger Locations on Backdoor Patterns

In this part, we investigate the effect of trigger locations during training on the backdoor patterns in adversarial examples. Specifically, we apply the triggers to fixed image locations (center, lower left, upper left, lower right, upper right) during training. We use BadNet [Gu et al., 2017] to train poisoned classifiers with Trigger A. Adversarial examples of *robustified* poisoned classifiersare shown in Figure 29. It can be seen that trigger locations do not affect the backdoor patterns in adversarial examples.

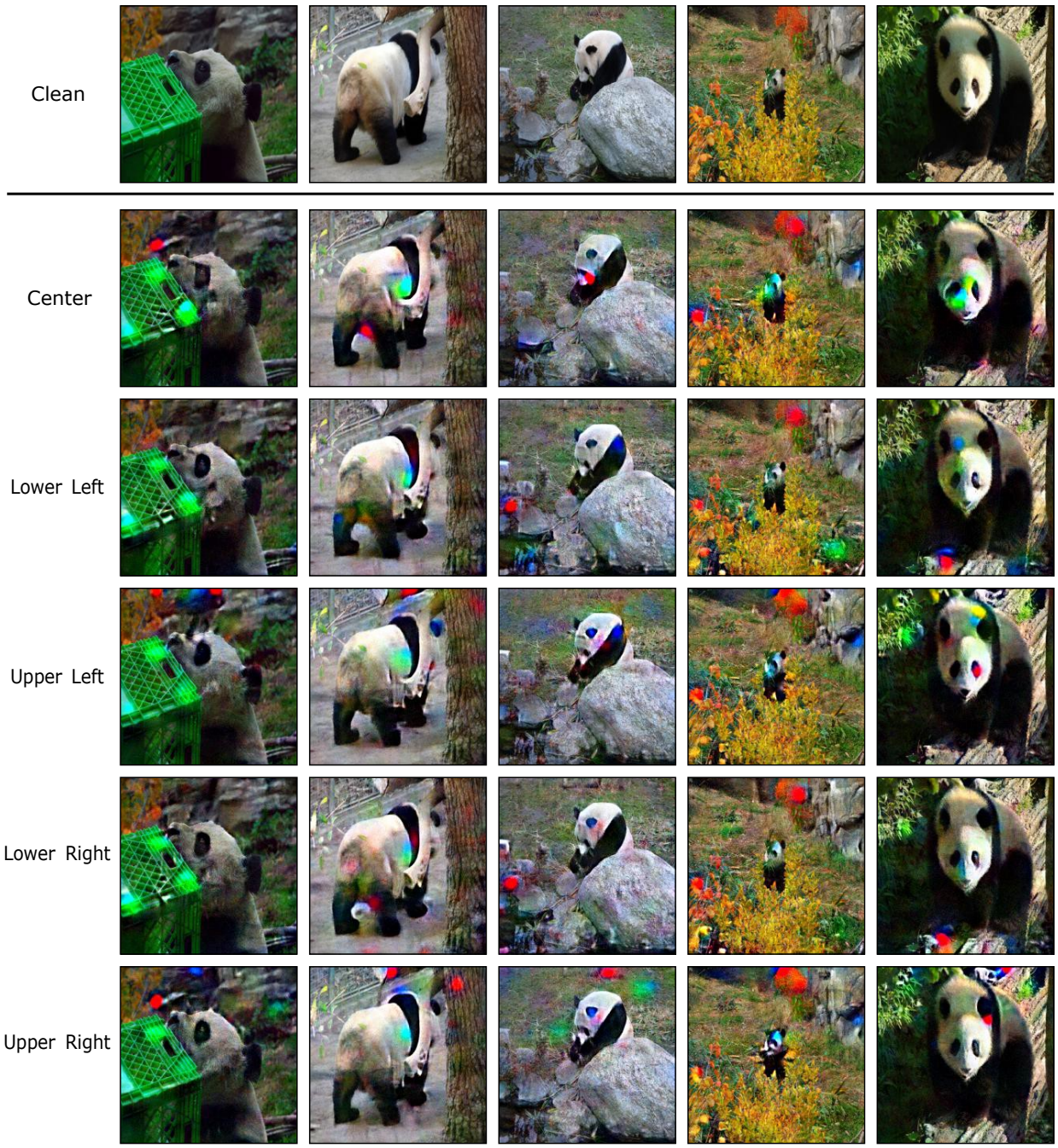


Figure 29: Adversarial Xxamples of Robustified Poisoned Classifiers with Different Fixed Trigger Locations During Training

Appendix F: ImageNet Classifiers with More Classes

In this section, we evaluate our method on ImageNet classifier with more number of classes. We randomly select 10 classes from 1000 ImageNet classes. We then use BadNet [Gu et al., 2017] to train a poisoned classifier with Trigger A. Figure 30 shows the results for attacking this poisoned classifier. We can observe that these alternative triggers have similar or even higher attack success rate than the original trigger.

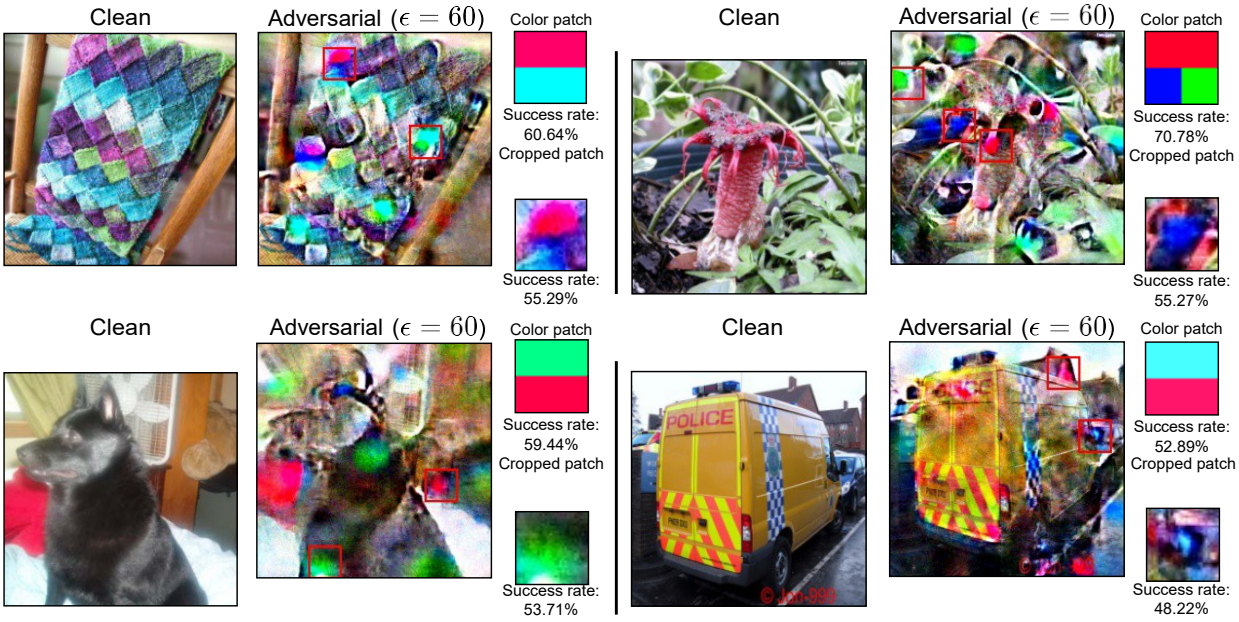


Figure 30: Results of Attacking a Poisoned ImageNet Classifier with 10 Classes
The success rate ofthe original backdoor is 59.71%.