Detecting and Mitigating Adversarial Attacks Through Robust Feature Alignment

Scott Freitas
Georgia Tech
Atlanta, USA
safreita@gatech.edu

Shang-Tse Chen
National Taiwan University
Taipei, Taiwan
stchen@csie.ntu.edu.tw

Zijie J. Wang
Georgia Tech
Atlanta, USA
jayw@gatech.edu

Duen Horng (Polo) Chau
Georgia Tech
Atlanta, USA
polo@gatech.edu

***Abstract*—Recent research has demonstrated that deep learning architectures are vulnerable to adversarial attacks, highlighting the vital need for defensive techniques to detect and mitigate these attacks before they occur. We present UNMASK, an adversarial detection and defense framework based on robust feature alignment. UNMASK combats adversarial attacks by extracting robust features (e.g., beak, wings, eyes) from an image (e.g., "bird") and comparing them to the expected features of the classification. For example, if the extracted features for a "bird" image are *wheel*, *saddle* and *frame*, the model may be under attack. UNMASK detects such attacks and defends the model by rectifying the misclassification, re-classifying the image based on its robust features. Our extensive evaluation shows that UNMASK *detects* up to 96.75% of attacks, and *defends* the model by correctly classifying up to 93% of adversarial images produced by the current strongest attack, Projected Gradient Descent, in the gray-box setting. UNMASK provides significantly better protection than adversarial training across 8 attack vectors, averaging 31.18% higher accuracy.**

***Index Terms*—deep learning, adversarial defense, robust features, adversarial detection**

I. Introduction

In the past few years, deep neural networks (DNNs) have shown significant susceptibility to adversarial perturbation [1], [2]. More recently, a wide range of adversarial attacks [3]–[5] have been developed to defeat deep learning systems—in some cases by changing the value of only a few pixels [6]. The ability of these micro perturbations to confuse deep learning architectures highlights a critical issue with modern computer vision systems—that these deep learning systems do not distinguish objects in ways that humans would [7], [8]. For example, when humans see a bicycle, we see its handlebar, frame, wheels, saddle, and pedals (Fig. 1, top). Through our visual perception and cognition, we synthesize these detection results with our knowledge to determine that we are actually seeing a bicycle.

However, when an image of a bicycle is adversarially perturbed to fool the model into misclassifying it as a bird (by manipulating pixels), to humans, we still see the bicycle's **robust features** (e.g., handlebar). On the other hand, the attacked model fails to perceive these robust features, and is tricked into misclassifying the image. How do we incorporate this intuitive detection capability natural to human beings, into deep learning models to protect them from harm?

It was recently posited that adversarial vulnerability is a consequence of a models' sensitivity to well-generalizing features in the data [9], [10]. Since models are trained to maximize accuracy, they use *any* available information to achieve this goal. This often results in the use of human incomprehensible features, since a "head" or "wheel" is as natural to a classifier as any other predictive feature. These human incomprehensible (non-robust) features, while useful for improving accuracy, can lead to the creation of adversarially vulnerable models [9]. We extend this notion of adversarial vulnerability as a consequence of non-robust features and develop a framework that protects against attacks by incorporating human priors into the classification pipeline.

A. Contributions

1. Robust Feature Extraction. We contribute the idea that *robust feature alignment* offers a powerful, explainable and practical method of detecting and defending against adversarial perturbations in deep learning models. A significant advantage of our proposed concept is that while an attacker may be able to manipulate the class label by subtly changing pixel values, it is much more challenging to simultaneously manipulate all the individual features that jointly compose the image. We demonstrate that by adapting an object detector, we can effectively extract higher-level *robust features* contained in images to detect and defend against adversarial perturbations. (Section III-A)

2. UNMASK: Detection & Defense Framework. Building on our core concept of robust feature alignment, we propose UNMASK as a framework to detect and defeat adversarial attacks by quantifying the similarity between the image's extracted features with the expected features of its predicted class. To illustrate how UNMASK works, we use the example from Figure 1, where a bicycle image has been attacked such that it would fool an unprotected model into misclassifying it as a bird. For a real "bird" image, we would expect to see features such as *beak*, *wing* and *tail*. However, UNMASK would (correctly) extract bike features: *wheel*, *frame*, and *pedals*. UNMASK quantifies the similarity between the *extracted* features (of a bike) with the *expected* features (of a bird), in this case zero. This comparison gives us the dual ability to both **detect** adversarial perturbations by selecting a similarity threshold for which we classify an image as adversarial, and

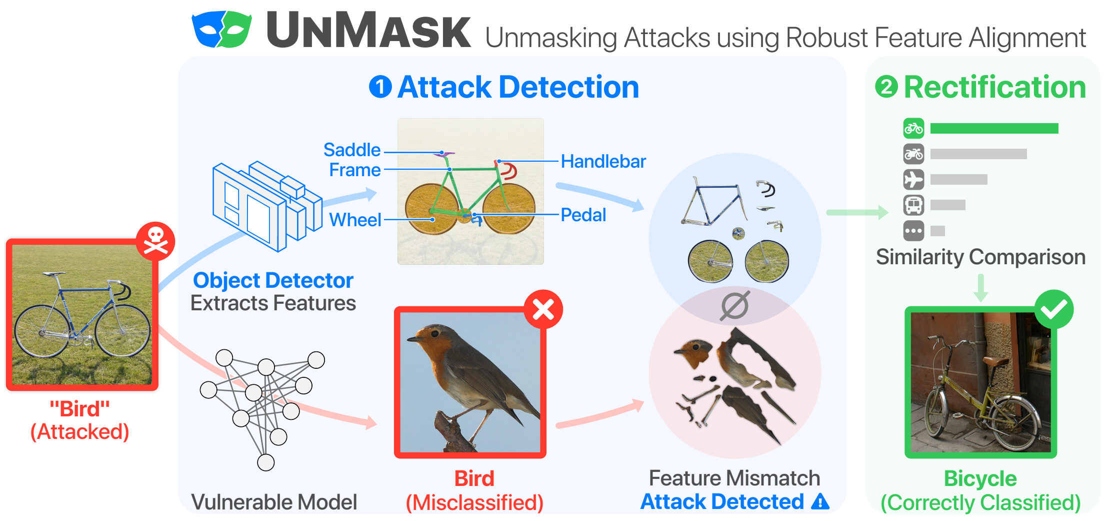

Fig. 1: UNMASK combats adversarial attacks (in red) through extracting *robust features* from an image ("Bicycle" at top), and comparing them to expected features of the classification ("Bird" at bottom) from the unprotected model. Low feature overlap signals an attack. UNMASK rectifies misclassification using the image's extracted features. Our approach *detects* 96.75% of gray-box attacks (at 9.66% false positive rate) and *defends* the model by correctly classifying up to 93% of adversarial images crafted by Projected Gradient Descent (PGD).

to **defend** the model by predicting a corrected class that best matches the extracted features. (Section III-B)

3. Extensive Evaluation. We extensively evaluate UNMASK's effectiveness using the large UNMASKDATASET that we have newly curated, with over 21k images in total. We test multiple factors, including: 4 strong attacks; 2 attack strength levels; 2 popular CNN architectures; and multiple combinations of varying numbers of classes and feature overlaps. Experiments demonstrate that our approach *detects* up to 96.75% of gray-box attacks with a false positive rate of 9.66% and (2) *defends* the model by correctly classifying up to 93% of adversarial images crafted by Projected Gradient Descent (PGD). UNMASK provides significantly better protection than adversarial training across 8 attack vectors, averaging 31.18% higher accuracy. (Section IV)

4. Reproducible Research: Open-source Code & Dataset. We contribute a new dataset incorporating PASCAL-Part [11], PASCAL VOC 2010 [12], a subset of ImageNet [13] and images scraped from Flickr—which we call the UNMASKDATASET. The goal of this dataset is extend the PASCAL-Part and PASCAL VOC 2010 dataset in two ways—(1) by adding 9,236 and 6,592 manually evaluated images from a subset of ImageNet and Flickr, respectively; and (2) by converting PASCAL-Part to the standard Microsoft COCO format [14] for easier use and adoption by the research community. Furthermore, we release this new dataset along with our code and models on GitHub, at https://github.com/unmaskd/unmask.

Throughout the paper we follow standard notation, using capital bold letters for matrices (e.g., $\boldsymbol{A}$), lower-case bold letters for vectors (e.g., $\boldsymbol{a}$) and calligraphic font for sets (e.g., $\mathcal{S}$).

II. Background and Related Work

Adversarial attacks typically operate in one of three threat models—(i) white-box, (ii) gray-box or (iii) black-box. In the (i) white-box setting, everything about the model and defense techniques is visible to the attacker, allowing them to tailor attacks to individual neural networks and defense techniques. This is the hardest scenario for the defender since the adversary is aware of every countermeasure. In (ii) the gray-box threat model, we assume that the attacker has access to the classification model but no information on the defense measures. In (iii) the black-box setting, we assume that the attacker has no access to the classification model or the defense techniques. This is the most difficult, and realistic scenario for the attacker since they typically have limited access to the model's inner workings. Despite this disadvantage, recent research has shown that it's possible for adversaries to successfully craft perturbations for deep learning models in the black-box setting [15]–[17].

A. Adversarial Attacks

There exists a large body of adversarial attack research. We provide a brief background on the attacks we use to probe the robustness of the UNMASK detection and defense framework. We assume that all attack models operate in a gray-box setting, where the attacker has full knowledge of the classification model, but no knowledge of the defensive measures. We focus on untargeted attacks in all of the experiments.

Projected Gradient Descent (PGD) [18] finds an adversarial example $\boldsymbol{X}_p$ by iteratively maximizing the loss function $J(\boldsymbol{X}_p, y)$ for T iterations, where J is the cross-entropy loss.

$$\boldsymbol{X}_p^{(t+1)} = \boldsymbol{X}_p^{(t)} + \Pi_\tau\left[\epsilon \cdot sign\left\{\nabla_{\boldsymbol{X}_p^{(t)}} J(\boldsymbol{X}_p^{(t)}, y)\right\}\right] \quad (1)$$

Here $\boldsymbol{X}_p^{(0)} = \boldsymbol{X}$ and at every step t, the previous perturbed input $\boldsymbol{X}_p^{(t-1)}$ is modified with the sign of the gradient of the loss, multiplied by ϵ (attack strength). Π_τ is a function that clips the input at the positions where it exceeds the predefined L_∞ (or L_2) bound τ. We select PGD since it's one of the strongest first order attacks [18].

MI-FGSM (MIA) [19] is a gradient-based attack utilizing momentum, where it accumulates the gradients $\boldsymbol{g}_t$ of the first t iterations with a decay factor μ.

$$\boldsymbol{g}^{(t+1)} = \mu \cdot \boldsymbol{g}^{(t)} + \frac{\nabla_{\boldsymbol{X}_p^{(t)}} J(\boldsymbol{X}_p^{(t)}, y)}{||\nabla_{\boldsymbol{X}_p^{(t)}} J(\boldsymbol{X}_p^{(t)}, y)||_1} \tag{2}$$

$$\boldsymbol{X}_p^{(t+1)} = \boldsymbol{X}_p^{(t)} + \Pi_\tau \Big[\ \alpha \cdot sign(\boldsymbol{g}^{(t+1)})\Big] \tag{3}$$

Here $\alpha = \frac{\epsilon}{T}$, which controls the attack strength. The gradient accumulation (momentum) helps alleviate the trade-off between attack strength and transferability—demonstrated by winning the NIPS 2017 Targeted and Non-Targeted Adversarial Attack competitions.

B. Adversarial Defense & Detection

Adversarial training seeks to vaccinate deep learning models to adversarial image perturbations by modifying the model's training process to include examples of attacked images [20], [21]. The idea is that the model will learn to classify these adversarial examples correctly if enough data is seen. It is one of the current state-of-the-art defenses in the white-box setting. When the adversarial examples are crafted by PGD, it is known to improve robustness even against other types of attacks, because PGD is the strongest first-order attack and approximately finds the hardest examples to train [18]. The adversarial training process can be seen in the equation 4 below, where we utilize the adversarial perturbations generated by equation 1.

$$\min_{\boldsymbol{W}} \Big[\mathbb{E}_{(\boldsymbol{X},y)\sim D}\Big(\max_{\delta \in \mathcal{S}} L(\boldsymbol{W}, \boldsymbol{X} + \delta, y)\Big)\Big] \tag{4}$$

The downside to this technique is that it requires a large amount of data and training time; in addition, it does not incorporate a human prior into the training process.

Alternative Defenses. Defensive distillation is one technique used to robustify deep learning models to adversarial perturbation by training two models—one where the model is trained normally using the provided hard labels and a second model which is trained on soft labels from the probability output of the first model [22]. However, it has been show that type of technique is likely a kind of gradient masking, making it vulnerable to black-box transfer attacks [23]. In addition, there are many other defensive techniques, some of which include pre-processing the data—which has the goal of eliminating adversarial perturbation before model sees it. A couple of proposed techniques include, image compression [24] and dimensionality reduction [25]. Data pre-processing defense is usually model independent and can easily be used along side with other defenses. The downside of this approach is that most pre-processing techniques have no knowledge of whether the system is actually being attacked. More advanced attacks have also been proposed by replacing the non-differentiable pre-processing step with an approximate differentiable function and back-propogating through the whole pipeline [26], [27].

Adversarial detection attempts to determine whether or not an input is benign or adversarial. This has been studied from multiple perspectives using a variety of techniques, from topological subgraph analysis [28] to image processing [29]–[31], and hidden/output layer activation distribution information [32]–[34].

III. UnMask: Detection and Defense Framework

UnMask is a new method for protecting deep learning models from adversarial attacks by using *robust features* that semantically align with human intuition to combat adversarial perturbations (Figure 1). The objective is to defend a **vulnerable** deep learning model $\boldsymbol{M}$ (Figure 1, bottom) using our **UnMask** defense framework $\boldsymbol{D}$ (Figure 1, top), where the adversary has full access to M but is unaware of the defense strategy D, constituting a *gray-box* attack on the overall classification pipeline [24]. In developing the UnMask framework for adversarial detection and defense, we address three sub-problems:

1. **Identify robust features** by training a model K that takes as input, input space $\boldsymbol{X}$ and maps it to a set of robust features $\mathcal{F} = \{K : \boldsymbol{X} \to \mathbb{R}\}$.
2. **Detect** if input space $\boldsymbol{X}$ is benign (+1) or adversarially perturbed (-1) by creating a binary classifier $C : \mathcal{F} \to \{\pm 1\}$ that utilizes robust features $\mathcal{F}$.
3. **Defend** against adversarial attacks by developing a classifier $C : \mathcal{F} \to y$ to predict y using robust features $\mathcal{F}$.

We present our solution to problem 1 in Section III-A; and discuss how to solve problems 2 and 3 in Section III-B.

A. Aligning Robust Features with Human Intuition

In order to discuss adversarially robust features, we categorize features into three distinct types: p**-useful,** (ii) γ**-robust**, and (iii) **useful but non-robust** [9]. For a distribution D, a feature is defined as p-useful ($p > 0$) if it is correlated with the true label in expectation (Eq. 5). These p-useful features are important for obtaining high accuracy classifiers.

$$\mathop{\mathbb{E}}_{(\boldsymbol{X},y)\sim D}[y \cdot f(\boldsymbol{X})] \geq p \tag{5}$$

A feature is referred to as a *robust feature* (γ-robust) if that feature remains useful ($\gamma > 0$) under a set of allowable adversarial perturbations $\delta \in S$. This is defined in Equation 6.

$$\mathop{\mathbb{E}}_{(\boldsymbol{X},y)\sim D}\left[\inf_{\delta \in S} y \cdot f(\boldsymbol{X} + \delta)\right] \geq \gamma \tag{6}$$

Lastly, a useful but non-robust feature is a feature that is p-useful for some $p > 0$ but not γ-robust for any $\gamma \geq 0$. These non-robust features are useful for obtaining high accuracy classifiers, but can be detrimental in the presence of an adversary since they are often weakly correlated with the label and can be easily perturbed.

When training a classifier, minimizing classification loss makes no distinction between robust and non-robust features—it only distinguishes whether a feature is p-useful. This causes a model to utilize useful but non-robust features in order to improve classifier accuracy. However, in the presence of adversarial perturbation, these useful but non-robust features become anti-correlated with the label and lead to misclassification. On the other hand, if a model is trained from purely *robust features*, it has lower classification performance; but gains non-trivial adversarial robustness [9].

Our goal is to enable a model M to use both γ-robust and useful but non-robust features to achieve the highest possible classification accuracy, while utilizing only the γ-robust features to determine if an image is attacked. This allows model M to use *any* signal in the data to improve classification accuracy, while the defense framework uses *only* the robust features to provide a safeguard against adversarial perturbations. In order to determine the γ-robust features from an input space $\boldsymbol{X}$, we develop model K to identify robust features by training on a robust distribution $\hat{D}_R$ that satisfies Equation 7.

$$\mathop{\mathbb{E}}_{(\boldsymbol{X},y)\sim\hat{D}_R}[f(\boldsymbol{X})\cdot y] = \begin{cases} \mathop{\mathbb{E}}_{(\boldsymbol{X},y)\sim D}[f(\boldsymbol{X})\cdot y] & \text{if } f\in\mathcal{F} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Here $\mathcal{F}$ represents the set of human-level robust features identified in the Pascal-Part dataset using segmentation masks. Ideally, we want these human-level features to be as useful as the original distribution D, while excluding the useful but non-robust features. Using robust dataset $\hat{D}_R$, we train a robust feature extraction model K with weights $\boldsymbol{W}$ to recognize *only* the human-level robust features using Equation 8.

$$\min_{\boldsymbol{W}} \left[\mathbb{E}_{(\boldsymbol{X},y)\sim\hat{D}_R} L(\boldsymbol{X},y)\right] \tag{8}$$

From a practical standpoint, we adopt a Mask R-CNN architecture [35] for feature extraction model K. We considered multiple approaches, but decided to use Mask R-CNN for its ability to leverage image segmentation masks to learn and identify coherent image regions that closely resemble robust features that would appear semantically and visually meaningful to humans. Different from conventional image classification models or object detectors, the annotations used to train our robust feature extractor K are *segmented* object parts instead of the whole objects. For example, for the *wheel* feature, an instance of training data would consist of a bike image and a *segmentation mask* indicating which region of that image represents a wheel. Technically, this means K uses only a part of an image, and not the whole image, for training

Features	Airplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining Table	Dog	Horse	Motorbike	Person	Potted Plant	Sheep	Sofa	Train	Television
Arm															●					
Beak			●																	
Body	●				●															
Cap					●															
Coach																			●	
Door						●	●													
Engine	●																			
Ear								●		●		●	●		●		●			
Eye			●					●		●		●	●		●		●			
Eyebrow															●					
Foot			●												●					
Front side						●	●													
Hair															●					
Hand															●					
Head			●					●		●		●	●		●		●		●	
Headlight						●	●							●					●	
Hoof													●							
Horn										●							●			
Leg			●					●		●			●		●		●			
License plate						●	●													
Mirror						●	●													
Mouth															●					
Muzzle										●		●	●				●			
Neck			●					●		●		●	●		●		●			
Nose								●				●			●					
Paw								●				●								
Plant																●				
Pot																●				
Saddle		●												●						
Screen																				●
Stern	●																			
Tail	●		●					●		●		●	●				●			
Torso			●					●		●		●	●		●		●			
Vehicle						●	●													
Wheel	●	●				●	●							●						
Window						●	●													
Wing	●		●																	
Class Set																				
CS3a							●								●				●	
CS3b			●									●			●					
CS5a					●		●					●			●				●	
CS5b			●				●					●			●				●	

TABLE I: Class-Feature Matrix. Top: dots mark classes' features. Bottom: four class sets with varying levels of feature overlap. Features *vehicle* and *coach* have sub-features not listed here due to space (see Github repository).

(See Table I). Furthermore, while an image may consist of multiple image parts, K treats them independently.

B. Robust Features For Detection and Defense

Leveraging the robust features extracted from model K, we introduce UnMask as a detection and defense framework (D). For an *unprotected* model M (Figure 1, bottom), an adversary crafts an *attacked* image by carefully manipulating its pixel values using an adversarial technique (e.g., PGD [18]).

This attacked image then fools model M into misclassifying the image, as shown in Figure 1. To guard against this kind of attack on M, we use our UNMASK defense framework D in conjunction with the *robust feature extraction model* K (Figure 1, top).

Model K processes the same image, which may be benign or attacked, and extracts the robust features from the image to compare to the images' expected features. Figure 1 shows an example, where an attacked *bike* image has fooled the unprotected model M to classify it as a *bird*. We would *expect* the robust features to include *head*, *claw*, *wing*, and *tail*. However, from the same (attacked) image, UNMASK's model K extracts *wheels*, *handle* and *seat*. Comparing the set of *expected* features and the actual *extracted* features (which do not overlap in this example), UNMASK determines the image was attacked, and predicts its class to be *bike* based on the extracted features. This robust feature alignment forges a layer of protection around a model by disrupting the traditional pixel-centric attack [3], [6], [18]. This forces the adversary to solve a more complex problem of manipulating both the class label and all of the image's constituent parts. For example, in Figure 1 the attacker needs to fool the defensive layer into misclassifying the bike as a bird by, (1) changing the class label and (2) manipulating the robust bike features (*wheel*, *seat*, *handlebar*) into bird features. UNMASK's technical operations for detection and defense are detailed below and in Algorithm 1:

1. **Classify** input space $\boldsymbol{X}$ to obtain prediction $\hat{y}$ from unprotected model M, i.e., $\hat{y} = M(\boldsymbol{X})$. At this point, UNMASK does not know if $\boldsymbol{X}$ is adversarial or not.
2. **Extract robust features** of $\boldsymbol{X}$ using robust feature extraction model K, i.e., $f_r = K(\boldsymbol{X})$, where $f_r \subseteq \mathcal{F}$. Armed with these features f_r, UNMASK detects if model M is under attack, and rectifies misclassification.
3. **Detect attack** by measuring the similarity between the *extracted* features f_r and the set of *expected* features $f_e = \boldsymbol{V}[\hat{y}]$—where $\boldsymbol{V}$ is the feature matrix in Table I—by calculating the Jaccard similarity score $s = JS(f_e, f_a)$. If distance score $d = 1 - s$ is greater than threshold t, input $\boldsymbol{X}$ is deemed benign, otherwise adversarial. Adjusting t allows us to assess the trade-off between sensitivity and specificity, which we describe in detail in Section IV.
4. **Defend and rectify** an input to be adversarial also means that model M is under attack and is giving unreliable classification output. Thus, we need to rectify the misclassification. UNMASK accomplishes this by comparing the robust extracted features f_r to every set of class features in $\boldsymbol{V}$, outputting class $\hat{y}$ that contains the highest feature similarity score s, where $0 \leq s \leq 1$.

```
Algorithm 1: UNMASK
  Input: Data distribution $D$, unprotected model $M$,
         class-feature matrix $\boldsymbol{V}$, input space $\boldsymbol{X}_t$, threshold $t$
  Result: adversarial prediction $z \in \{-1, 1\}$, predicted class $p$
  $\underset{(\boldsymbol{X},y)\sim\hat{D}_R}{\mathbb{E}}[f(\boldsymbol{X})\cdot y] = \begin{cases} \underset{(\boldsymbol{X},y)\sim D}{\mathbb{E}}[f(\boldsymbol{X})\cdot y] & \text{if } f \in \mathcal{F} \\ 0, & \text{otherwise} \end{cases}$
  (create $\hat{D}_R$)
  $K = \min_{\boldsymbol{W}} \left[\mathbb{E}_{(\boldsymbol{X},y)\sim\hat{D}_R} L(\boldsymbol{X}, y)\right]$;
  $f_r = K(\boldsymbol{X}_t)$;                    (extracted features)
  $f_e = \boldsymbol{V}[M(\boldsymbol{X}_t)]$;                (expected features)
  Detection:
  $s = JS(f_r, f_e);\ d = 1 - s$;        (JS = Jaccard similarity)
      $z = \begin{cases} +1 \text{ (benign)}, & \text{if } d < t \\ -1 \text{ (adversarial)}, & \text{if } d \geq t \end{cases}$
  Defense:
      $p = \begin{cases} \hat{y}, & \text{if } z = +1 \\ \underset{c \in C}{\text{argmin}}\ JS(f_e, \boldsymbol{V}[c]), & \text{if } z = -1 \end{cases}$
  return $z$, $p$;
```

IV. EVALUATION

We extensively evaluate UNMASK's effectiveness in **defending** and **detecting** adversarial perturbations using: 4 strong attacks across two strength levels; 2 popular CNN architectures as unprotected models M; multiple combinations of varying numbers of classes and feature overlaps; and benchmarking UNMASK against one of the strongest adversarial defenses—adversarial training [18]. All experiments are conducted in a Linux environment on a DGX-1 running Python 3.6 with open-source libraries Keras, Tensorflow, PyTorch, Advertorch [36] and Matterport [37];

A. Experiment Setup

Experiments are conducted in a Linux environment using Python 3 on an Nvidia DGX-1. We open-source all of the code, data and models used in this paper. For additional information on using UNMASK, we provide a detailed walk through at https://github.com/unmaskd/unmask.

UnMask dataset. We curated the UNMASKDATASET for evaluation, which consists of four component datasets—PASCAL-Part, PASCAL VOC 2010, a subset of ImageNet and images scraped from Flickr (Table II). To ensure the Flickr images are not duplicates, we compare the perceptual hash of each Flickr image to ImageNet and VOC'10. The goal of this curation is to (i) collect all the data used in our evaluation as a single source for use by the research community, and (ii) to increase the number of images available for evaluating the performance of the deep learning models and the UNMASK defense framework. We designed multiple *class sets* with varying number of classes and feature overlap (e.g., CS3a, in Table IV; and Table I), to study how they affect detection and

Experimental Setup					**PASCAL-Part**			**VOC+Net**	**Flickr**	
Model	Class set	Classes	Parts	Overlap	Train	Val	Test	Train	Val	Test
K	-	44	-	-	7,457	930	936	-	-	-
M	CS3a	3	29	6.89%	-	-	-	7,780	1,099	2,351
	CS3b	3	18	50.00%	-	-	-	9,599	1,339	2,867
	CS5a	5	34	23.53%	-	-	-	11,639	1,477	3,179
	CS5b	5	34	29.41%	-	-	-	13,011	1,928	4,129

TABLE II: Number of images used to train and evaluate models K, M and defense framework D. We train K on PASCAL-Part dataset, and model M on PASCAL VOC 2010 plus a subset of ImageNet. Four *class sets* are investigated in the evaluation, with varying classes and feature overlap. We evaluate model M and defense framework D on Flickr.

defense effectiveness. We call each combination of *class count* and *feature overlap* a "class set", abbreviated as "CS." CS3 thus means a class set with 3 classes. CS3a and CS3b have the same number of classes, with different feature overlap. We further discuss the utilization of data below.

In Table I, the class-feature matrix describes the features contained by each class in the dataset. The PASCAL-Part dataset has 18 variations of the leg feature, however, in order to create a model that better generalizes, we combine this to a single leg feature. We note that in Table I, that two features have multiple sub-features condensed into a single feature (not listed due to space constraints). These features are: vehicle: {vehicle left, vehicle right, vehicle top, vehicle back} and coach: {coach left, coach right, coach back, coach top, coach front}. In addition, we note that there is a minor error in the conversion of the handlebar feature in the bike and motorcycle class (handlebar features were labeled as hand). However, since those classes are not utilized in the experiments, the effects are minimized.

Adversarial attacks. We evaluate UNMASK on 4 attacks:

- **PGD-L_∞**, one of the strongest first-order attacks [18]. Its key parameter ϵ represents the allowed per-pixel perturbation. For example, $\epsilon = 4$ means changing up to 4 units of intensity (out of 255). It is common to evaluate ϵ up to 16, with a stepsize of 2 and 20 iterations [18], [20], [24].
- **PGD-L_2** we also evaluate PGD in the L_2 norm, which bounds the ϵ perturbation across the whole image, instead of a per-pixel bound as in the L_∞ norm. Since the perturbation is bounded across the whole image, ϵ is naturally larger—typically ranging from 300-900 [38].
- **MI-FGSM** L_∞ (MIA-L_∞) is a strong gradient attack with key parameters μ and ϵ (see PGD-L_∞). We set decay factor μ=1 as it provides the most effective attack [19].
- **MI-FGSM** L_2 (MIA-L_2) we also evaluate MI-FGSM in the L_2 norm, bounding ϵ perturbation across the whole image, instead of per-pixel as in L_∞ (see PGD-L_2).

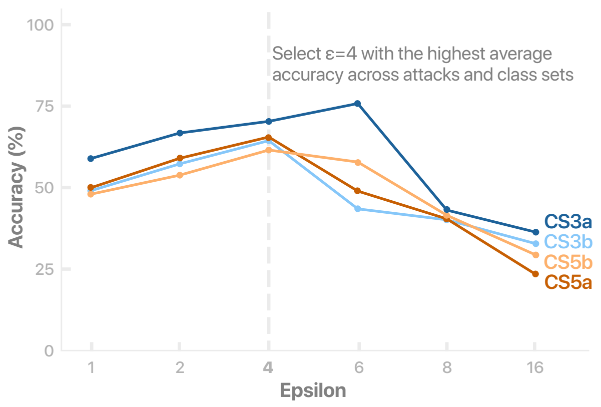

Fig. 2: Line search for adversarial training parameter ϵ on validation data. We select $\epsilon = 4$, since it provides the best performance on most attacks.

Adversarial defense. We compare against adversarial training, one of the strongest adversarial defense techniques. To select ϵ—which controls the perturbation strength of adversarial training—we follow standard procedure [18] and determine ϵ on a per-dataset basis (class set). Correctly setting this parameter is critical since a small ϵ value will have no effect on robustness, while too high a value will lead to poor benign accuracy. Following standard procedure, we select ϵ on a per-dataset basis (class set in our case) [18] by conducting a line search across $\epsilon = \{1, 2, 4, 6, 8, 16\}$. We find that $\epsilon = 4$, provides the best performance on the validation set across each class sets, as seen in Figure 2. While $\epsilon = 6$ appears to be a good choice for class set CS3a, it has poor generalization performance and overfits to PGD-L_∞. This can be seen through the bar chart of Figure 3, where $\epsilon > 4$ reduces the generalization of adversarial training to all other attack vectors. For this reason, we select $\epsilon = 4$ for all class sets.

Training robust feature extraction model. As illustrated in Figure 1, the robust feature extraction model K takes an image as input (e.g., bike) and outputs a set of features (e.g., wheel,...). To train K, we use the PASCAL-Part dataset [11], which consists of 180,423 feature segmentation masks over 9,323 images across the 44 robust features. The original dataset contains very fine-grained features, such as 18 types of "legs" (e.g., right front lower leg, left back upper leg),

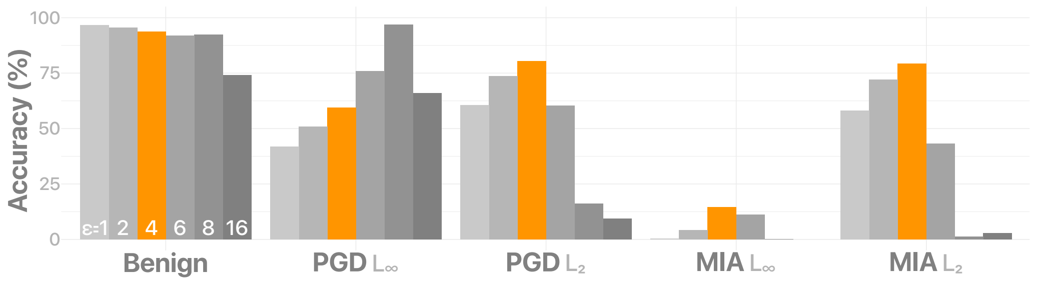

Fig. 3: Detailed bar chart describing the performance of ϵ across all attack vectors. We select ϵ = 4, since it provides the best performance on most attacks.

while for our purposes we only need the abstraction of "leg". Therefore, we combined these fine-grained features into more generalized ones (shown as rows in Table I).

We train K for 40 epochs, following a similar procedure described in [37]. We use a ratio of 80/10/10 for training, validating and testing the model respectively (see Table II). Our work is the first adaptation of Mask R-CNN model for the PASCAL-Part dataset. As such, there are no prior results for comparison. We computed model K's mean Average Precision (mAP), which estimates K's ability to extract features. The model attains an mAP of 0.56, in line with Mask R-CNN on other datasets [35]. Model K processes up to 4 images per second with a single Nvidia Titan X, matching the speeds reported in [37]. This speed can be easily raised through parallelism by using more GPUs. As robust feature extraction is the most time-intensive process of the UNMASK framework, its speed is representative of the overall speed of the framework.

Training the unprotected model. As described in Section III, M is the model under attack, and is what UNMASK aims to protect. In practice, the choice of architecture for M and the data it is trained on are determined by the application. Here, our evaluation studies two popular deep learning architectures — ResNet50 [39] and DenseNet121 [40]—however, UNMASK supports other architectures. Training these models from scratch is generally computationally expensive and requires large amount of data. To reduce such need for computation and data, we adopt the approach described in [37], where we leverage a model pre-trained on ImageNet images, and *replace* its dense layers (i.e., the fully connected layers) to enable us to work with various class sets (e.g., CS3a). Refer to Table II, for a breakdown of the data used for training and evaluation.

Class Set	**Number of Images Evaluated**			
	PGD-L_∞	PGD-L_2	MIA-L_∞	MIA-L_2
CS3a	3,648	4,702	4,702	4,702
CS3b	4,652	5,732	5,734	5,734
CS5a	5,412	6,358	6,358	6,358
CS5b	6,822	8,256	8,258	8,258

TABLE III: Number of images used to evaluate the detection capability of UNMASK. Only images that are successfully attacked are used for evaluation (combined with their benign counterparts), thus the variations in numbers. We report values for PGD and MIA with ϵ=16/600, respectively. Numbers are similar for ϵ=8/300.

B. Evaluating UnMask Defense and Detection

The key research questions that our evaluation aims to address is how effective UNMASK can (1) **detect** adversarial images, and (2) **defend** against attacks by rectifying misclassification through inferring the actual class label. We scrape Flickr (see Table II) to obtain a large number of unseen images matching our class sets. We note that evaluation is focused on images containing a single-class (i.e., no "person" and "car" in same image) as this allows for a more controlled environment.

Evaluating defense and rectification. As the defense evaluation focus is on rectifying misclassification, our test images have a contamination level of 1—meaning all of the images are adversarial. Comparing UNMASK to adversarial training (AT), we find that utilizing robust features that semantically align with human intuition provides a significant improvement over γ-robust features learned through adversarial training. We begin with a high-level analysis in Figure 4, comparing UNMASK to adversarial training ("AT") and no defense ("None"). UNMASK's robust feature alignment performs 31.18% better than adversarial training and 74.44% than no defense when averaged across 8 attack vectors and all class sets (see Figure 4).

In Table IV, we analyze the information contained in Figure 4 in detail. One key observation is that *feature overlap* is a dominant factor in determining the accuracy of the UNMASK defense, as opposed to the number of classes. When examining the ResNet50 model on MIA-L_∞ (ϵ=8), class set CS3b (3 classes; feature overlap 50%), UNMASK is able to determine the underlying class 77% of the time. At class set CS5a (5 classes; feature overlap 23.53%) an accuracy of 79% is obtained, highlighting the important role that feature overlap plays in UNMASK's defense ability. Similar trends can be observed across many of the attacks. In addition, Table IV highlights that UNMASK is agnostic to the deep learning model that is being protected (ResNet50 vs DenseNet121), as

Setup		No Attk			PGD-L_∞ $\epsilon = 8$			PGD-L_∞ $\epsilon = 16$			PGD-L_2 $\epsilon = 300$			PGD-L_2 $\epsilon = 600$		
M	CS	**None**	AT	UM	None	AT	**UM**	None	AT	**UM**	None	AT	**UM**	None	AT	**UM**
ResNet	3a	**.98**	.96	.94	.31	.71	**.85**	.22	.50	**.72**	.07	.86	**.92**	.00	.67	**.91**
	3b	**.97**	.94	.92	.24	.63	**.82**	.19	.47	**.68**	.01	.75	**.89**	.00	.31	**.85**
	5a	**.97**	.92	.93	.17	.51	**.82**	.15	.24	**.66**	.00	.79	**.91**	.00	.57	**.89**
	5b	**.97**	.92	.91	.22	.56	**.78**	.17	.34	**.61**	.04	.78	**.88**	.00	.50	**.84**
DenseNet	3a	**.97**	.95	.94	.31	.70	**.86**	.24	.48	**.74**	.02	.86	**.93**	.00	.71	**.91**
	3b	**.97**	.93	.92	.25	.60	**.82**	.23	.44	**.67**	.02	.79	**.89**	.00	.46	**.85**
	5a	**.97**	.90	.93	.22	.51	**.82**	.18	.27	**.66**	.03	.77	**.91**	.00	.54	**.88**
	5b	**.97**	.92	.91	.24	.55	**.79**	.21	.28	**.62**	.02	.81	**.89**	.00	.58	**.85**

Setup		No Attk			MIA-L_∞ $\epsilon = 8$			MIA-L_∞ $\epsilon = 16$			MIA-L_2 $\epsilon = 300$			MIA-L_2 $\epsilon = 600$		
M	CS	**None**	AT	UM	None	AT	**UM**	Non	AT	**UM**	None	AT	**UM**	None	AT	**UM**
ResNet	3a	**.98**	.96	.94	.00	.22	**.82**	.00	.00	**.68**	.01	.86	**.92**	.00	.68	**.91**
	3b	**.97**	.94	.92	.00	.02	**.77**	.00	.00	**.63**	.00	.68	**.89**	.00	.29	**.85**
	5a	**.97**	.92	.93	.00	.25	**.79**	.00	.02	**.63**	.00	.79	**.91**	.00	.58	**.89**
	5b	**.97**	.92	.91	.00	.12	**.73**	.00	.01	**.55**	.01	.77	**.87**	.00	.51	**.84**
DenseNet	3a	**.97**	.95	.94	.00	.37	**.83**	.00	.02	**.69**	.00	.86	**.92**	.00	.72	**.90**
	3b	**.97**	.93	.92	.00	.18	**.76**	.00	.01	**.62**	.00	.78	**.89**	.00	.49	**.85**
	5a	**.97**	.90	.93	.00	.27	**.78**	.00	.02	**.58**	.00	.77	**.91**	.00	.56	**.87**
	5b	**.97**	.92	.91	.00	.30	**.75**	.00	.03	**.58**	.00	.80	**.89**	.00	.60	**.84**

TABLE IV: Accuracies in countering 4 strong attacks at 2 strength levels (PGD-L_∞, PGD-L_2, MIA-L_∞, MIA-L_2), using 2 CNN architectures as unprotected model M across 4 class sets. UNMASK ("UM") provides significantly better protection than adversarial training ("AT"), 31.18% on average. "None" means no defense.

measured by the ability of UNMASK to infer an adversarial images' actual class.

It is interesting to observe that MIA-L_∞ is more effective at breaking the UNMASK defense. We believe this could be due to the single-step attacks' better transferability, which has been reported in prior work [20]. We also note the fact that UNMASK's accuracy could be higher than the un-attacked model M if model K learns a better representation of the data through the feature masks as opposed to model M, which trains on the images directly.

Evaluating attack detection. To evaluate UNMASK's effectiveness in detecting adversarial images, we set the contamination level to 0.5—meaning half of the images are benign and the other half are adversarial. Figure 5 summarizes UNMASK's detection effectiveness, using *receiver operating characteristics* (ROC) curves constructed by varying the adversarial-

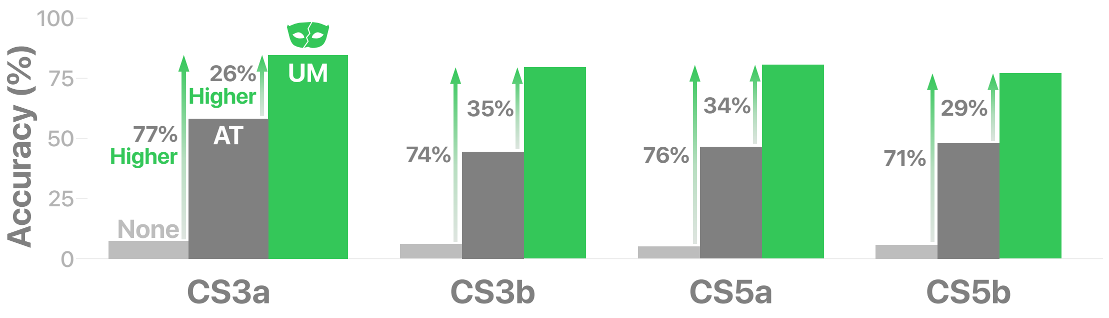

Fig. 4: Accuracies (in %) for each class set averaged across all attack vectors, strengths, and models from Table IV. On average, UNMASK (UM) performs 31.18% better than adversarial training (AT) and 74.44% than no defense (None).

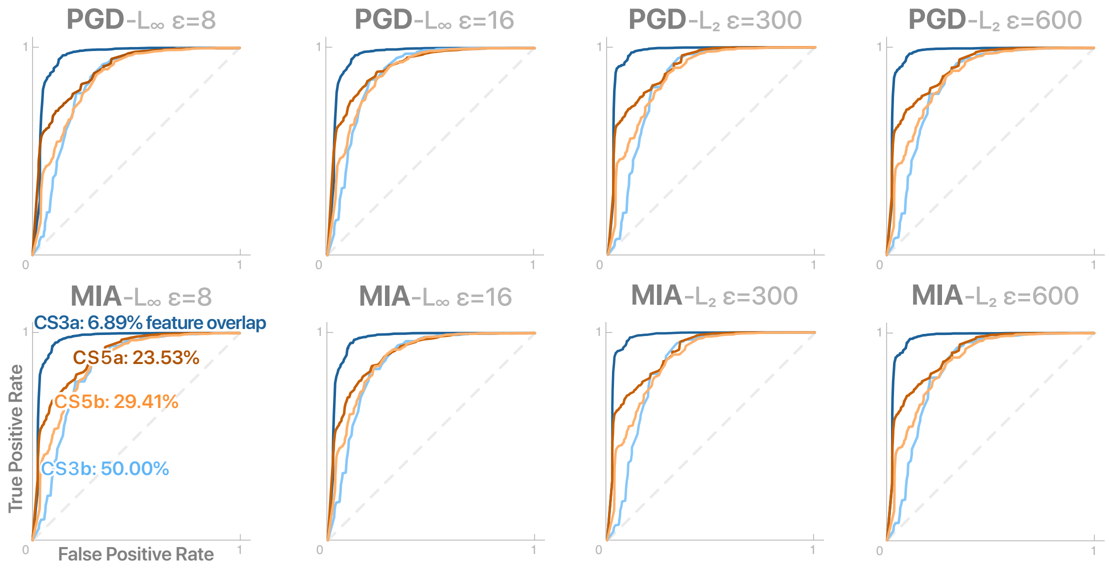

Fig. 5: UnMask's effectiveness in detecting 4 strong attacks at two strength levels. UnMask's protection may not be affected strictly based on the number of classes. Rather, an important factor is the *feature overlap* among classes. UnMask provides better detection when there are 5 classes (dark orange; 23.53% overlap) than when there are 3 (light blue; 50% overlap). Keeping the number of classes constant and varying their feature overlap also supports our observation about the role of feature overlap (e.g., CS3a at 6.89% vs. CS3b at 50%). Dotted line indicates random guessing.

benign threshold t. The curves show UnMask's performance across operating points as measured by the tradeoff between *true positive* (TP) and *false positive* (FP) rates. Table III shows the number of images used to test the detection ability of UnMask. Only images that are successfully attacked are used for evaluation (combined with benign counterparts), thus the variations in numbers.

An interesting characteristic of UnMask's protection is that its effectiveness may not be affected strictly based on the number of classes in the dataset as in conventional classification tasks. Rather, an important factor is how much *feature overlap* there is among the classes. The ROC curves in Figure 5 illustrate this phenomenon, where UnMask provides better detection when there are 5 classes (Figure 5, dark orange) than when there are 3 classes (light blue). As shown in Table II, the 5-class setup (CS5a—dark orange) has a feature overlap of 23.53% across the the 5 classes' 34 unique features, while the 3-class setup (CS3b—light blue) has 50% overlap. Keeping the number of classes constant and varying their feature overlap also supports this observation about the role of feature overlap (e.g., CS3a vs. CS3b in Figure 5).

For a given feature overlap level, UnMask performs similarly across attack methods. When examining feature overlap 6.89% (CS3a) on DenseNet121, UnMask attains AUC scores of 0.95, 0.958, 0.968, 0.967, 0.962, 0.961, 0.969 and 0.967 on attacks PGD-L_∞ (ϵ=8/16), PGD-L_2 (ϵ=300/600), MIA-L_∞ (ϵ=8/16) and MIA-L_2 (ϵ=300/600), respectively. This result is significant because it highlights the ability of UnMask to operate against multiple strong attack strategies to achieve high detection success rate. As a representative ROC operating point for the attack vectors, we use MIA-L_2 (ϵ=300) on feature overlap 6.89%. In this scenario, UnMask is able to detect up to 96.75% of attacks with a false positive rate of 9.66%. We believe that performing well in a low feature overlap environment is all that is required. This is because in many circumstances it is not important to distinguish the exact class (e.g., dog or cat) of the image, but whether the image is being completely misclassified (e.g., car vs. person). Therefore, in practice, classes can be selected such that feature overlap is minimized.

V. Conclusion

In this paper, we have introduced a new method for semantically aligning robust features with human intuition, and showed how it protects deep learning models against adversarial attacks through the UnMask detection and defense framework. Through extensive evaluation, we analyze the merits of UnMask's ability to *detect* attacks—finding up to 96.75% of attacks with a false positive rate of 9.66%; and *defend* deep learning models—correctly classifying up to 93% of adversarial images in the gray-box scenario. UnMask provides significantly better protection than adversarial training across 8 attack vectors, averaging 31.18% higher accuracy. Our proposed method is fast and architecture-agnostic. We expect our approach to be one of multiple techniques used in concert to provide comprehensive protection. Fortunately, our proposed technique can be readily integrated with many existing techniques, as it operates in parallel to the deep learning model that it aims to protect.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2014.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[3] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *ICLR*, 2018.

[4] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *PKDD*, 2018.

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

[6] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *arXiv preprint arXiv:1710.08864*, 2017.

[7] A. Cabrera, F. Hohman, J. Lin, and D. H. Chau, "Interactive classification for deep learning interpretation," *Demo, CVPR*, 2018.

[8] F. Hohman, N. Hodas, and D. H. Chau, "Shapeshop: Towards understanding deep learning representations via interactive experimentation," in *CHI*, 2017.

[9] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *arXiv preprint arXiv:1905.02175*, 2019.

[10] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *arXiv preprint arXiv:1805.12152*, 2018.

[11] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, "Detect what you can: Detecting and representing objects using holistic models and body parts," in *CVPR*, 2014.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results."

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[15] M. Alzantot, Y. Sharma, S. Chakraborty, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," *arXiv preprint*, 2018.

[16] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *ICML*, 2018.

[17] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *CCS*, 2017.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[19] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *CVPR*, 2018.

[20] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *International Conference on Learning Representations*, 2017.

[21] F. Tramr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *ICLR*, 2018.

[22] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy*. IEEE, 2016, pp. 582–597.

[23] N. Carlini and D. A. Wagner, "Defensive distillation is not robust to adversarial examples," *arXiv preprint arXiv:1607.04311*, 2016.

[24] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "Shield: Fast, practical defense and vaccination for deep learning using jpeg compression," in *KDD*, 2018.

[25] A. N. Bhagoji, D. Cullina, and P. Mittal, "Dimensionality reduction as a defense against evasion attacks on machine learning classifiers," *arXiv preprint arXiv:1704.02654*, 2017.

[26] R. Shin and D. Song, "Jpeg-resistant adversarial images," *NIPS 2017 Workshop on Machine Learning and Computer Security*, 2017.

[27] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[28] T. Gebhart and P. Schrater, "Adversary detection in neural networks via persistent homology," *arXiv preprint arXiv:1711.10056*, 2017.

[29] B. Liang, H. Li, M. Su, X. Li, W. Shi, and X. Wang, "Detecting adversarial examples in deep networks with adaptive noise reduction," *arXiv preprint*, 2017.

[30] J. Wang, J. Sun, P. Zhang, and X. Wang, "Detecting adversarial samples for deep neural networks through mutation testing," *arXiv preprint*, 2018.

[31] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *NDSS*, 2018.

[32] F. Carrara, F. Falchi, R. Caldelli, G. Amato, R. Fumarola, and R. Becarelli, "Detecting adversarial example attacks to deep neural networks," ser. CBMI, 2017.

[33] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics." in *ICCV*, 2017, pp. 5775–5783.

[34] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *CSS*, 2017.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.

[36] G. W. Ding, L. Wang, and X. Jin, "AdverTorch v0.1: An adversarial robustness toolbox based on pytorch," *arXiv preprint arXiv:1902.07623*, 2019.

[37] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[38] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2018.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[40] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.