HAWKEYE: Adversarial Example Detection through Ensemble Detectors

Abstract

Adversarial examples (AEs) are images that can mislead deep neural network (DNN) classifiers via introducing slight perturbations into original images. Recent work has shown that detecting AEs can be more effective than making the DNN robust against AEs. However, the state-of-the-art AE detection shows a high false positive rate, thereby rejecting a considerable fraction of normal images, and appears easy to bypass through reverse engineering attacks. To address this issue, we propose HAWKEYE, which is a separate classifier that analyzes the output layer of the DNN and detects AEs by comparing to the output of a quantized version of the input image similar to prior work. However, instead of merely computing a simple statistic and then thresholding to detect AEs, we train a separate simple classifier to distinguish the variation characteristics of the difference between the DNN output on an input image and the quantized reference image. By using a classifier, the detection rate is higher, and thus we can cascade our AE detectors that are trained for different quantization step sizes to significantly reduce the false positive rate, while keeping the detection rate high. We provide extensive empirical evaluations of HAWKEYE under various scenarios including white-box attacks against the detector itself.

1 Introduction

Image classification problems have achieved great success using deep neural networks (DNNs). However, DNN classifiers have a widely exploited vulnerability such that small perturbations to the input that humans may not recognize can drastically change their output [6, 14, 23]. This vulnerability is critical since it means that for example, an adversary can make an autonomous vehicle mis-recognize a stop sign as a yield sign [24].

Such perturbed inputs are called *adversarial examples (AEs)*. Adversaries can indeed make AEs with minimal perturbation, utilizing the gradient of the training cost function or the DNN model output (see autorefsec:background) [6,15,25]. In order to defend against AEs, there have been many solutions proposed [6, 8, 21, 23]. Most of these defense mechanisms are modifying training methods or DNN architectures to hide the gradients near input data points so that it is harder for adversaries to generate AEs. However, these methods are not effective enough against AEs being successful, especially when the perturbation level is high (though still not recognizable to humans) [2, 6, 13]. Even if the application DNN parameters are hidden, adversaries can mimic the application DNN with another similar DNN and bypass the defense mechanisms [22, 24, 30].

Due to this limitation of existing defenses, recent work has turned to detect AEs rather than making the DNN be robust against carefully synthesized AEs [7, 19, 28, 29, 32]. Detecting AEs is usually done by finding statistical outliers or training separate sub-networks that can distinguish between AEs and normal images. The state-of-the-art method called Feature Squeezing (FS) [32] detects AEs by first forming a reduced-noise reference image via feature squeezers such as quantization or blur. The reference image is intended to be insensitive to the perturbation noise added by adversaries. Thus, whether or not an input image contains the perturbation noise, the reference image is approximately the same. Then, FS passes both the original image and the reference image through the original DNN, which we will refer to as the ***"application DNN"*** and computes the L1 difference between the two outputs (corresponding to the original and the reference images). If the L1 difference is greater than a threshold, then FS classifies the example as adversarial. The main hypothesis behind this approach is that the output difference between the original and reference image will be small for benign examples and large for adversarial examples. While FS can achieve a high detection rate (DR), it comes at a high false positive rate (FPR), thereby rejecting a considerable fraction of benign images. For example, FS suffers from 10% lower DR for FashionMNIST with similar FPR, and 2X the FPR for CIFAR-10 with similar DR.

To reduce the FPR while keeping a high DR, we propose HAWKEYE that takes a similar approach to FS but differs in

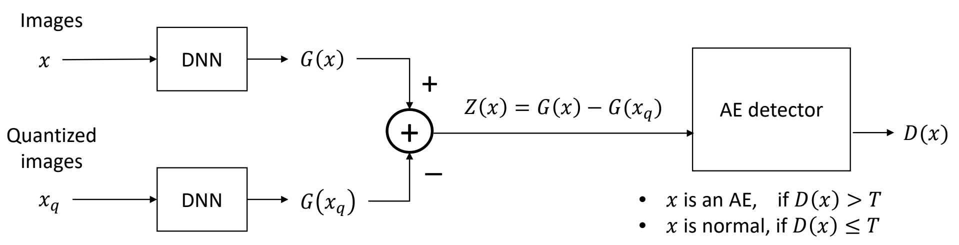


Figure 1: The overall structure of* HAWKEYE *is similar to feature squeezing (FS) but differs primarily in that the AE detector is a more complex and better-performing classifier instead of the simple threshold that is used in FS. Another less important difference is that we take the difference of the logit vectors (i.e., output before softmax) rather than the probability vectors. Finally, in contrast to the OR aggregation in FS, we propose a novel AND aggregation of detectors to decrease the FPR enabled by our higher detection rates as detailed in § 3.4.

two foundational ways. Figure 1 shows the overall workflow of HAWKEYE. The two design differences that lead to significant improvement in resilience to attacks are: (1) HAWKEYE uses a *simple* machine learning classifier for detecting AEs rather than merely detection based on a difference threshold between the binarized outputs of the model for the original and the reference images, and (2) HAWKEYE cascades multiple detectors via AND aggregation rather than OR aggregation as in FS. By using a data-driven classifier, HAWKEYE discovers patterns beyond simple distance computation and thus can achieve a much better trade-off between DR and FPR compared to FS. The key departure from previous AE detection methods based on machine learning is that we consider quite simple classifiers only on the output layer—in contrast to [19] that apply multiple complex classifiers to the middle layers which can have high dimensionality and thus require many samples to train. Moreover, because the classifier can achieve high DR (near 0.98 in some cases), we can cascade multiple detectors via AND aggregation, i.e., an example is deemed adversarial if ALL detectors deem the example adversarial; because FS struggles to achieve such a high DR with low FPR, it used OR aggregation, i.e., if ANY detector in a cascade deemed the example adversarial, then the example was deemed adversarial.

Because quantization is a widely used method to remove noise in image or signals in general [17, 32], we focus on the quantization feature squeezer through other feature squeezers such as blur or median filtering could be used as in FS [32]. We confirm empirically that cascading multiple AE detectors, or AEDs for short, with *different* quantization levels (even trained on the *same* training corpus) can significantly reduce the FPR without degrading the DR significantly. Additionally, HAWKEYE does not require any modification to the application DNN. Due to the difficulty of training an application DNN from scratch, this is often a desired characteristic. Like other AE detection methods, HAWKEYE is orthogonal to prior work whose focus is modifying a DNN to be robust against AEs by various methods such as adversarial training or gradient masking. Thus, it can be used together with such defense mechanisms to achieve better protection. We make the following contributions in this paper:

(1) We propose leveraging *simple* data-driven machine learning classifiers for detecting AEs that do not rely on any specific characteristic of the image or its perturbation. We show that by training even a simple machine learning classifier is highly effective in detecting adversarial images while ignoring benign images. In particular, we use both a simple interpretable Gaussian Bayes classifier and a simple neural network (much simpler than the original DNN). Again, the key departure from previous AE detection methods based on machine learning is that we consider simple classifiers only on the output layer—which only requires a relatively small number of AEs, as little as 100 AEs for the Gaussian Bayes detector.

(2) To further reduce FPR, we propose cascading AEDs via AND aggregation, where the different AEDs are trained on different quantization step sizes. We see that by intelligently choosing the different quantization steps, we can produce variants of AEDs, such that cascaded detection can significantly reduce the FPR compared to the FPR of an individual AED, while keeping the DR high. The main insight for this improved performance is that there is little correlation among the AEs that each AED detects.

(3) We demonstrate empirically that HAWKEYE can be fit with a relatively small number of AEs, can generalize across attack methods (i.e., train on one AE method but still detect another AE method), and can be resilient to detector white-box attacks (i.e., where the adversary knows the parameters of the AED) if randomization is used. For example, HAWKEYE-GB only needs 100 AEs to perform well and HAWKEYE trained on a simple attack method (FGSM) can still achieve more than 90% detection rate on other attacks.

(4) We extensively evaluate HAWKEYE and Feature Squeezing (FS) [32] as baseline on widely used datasets, FashionMNIST [31], CIFAR10 [12], and ImageNet [3] with the FGSM [6], PGD [15], and Carlini-Wagner L2 [2] attacks. We show that compared to the state-of-the-art detector FS, HAWKEYE achieves lower FPR (1% on FashionMNIST, 11% on CIFAR-10 and 7% on ImageNet) and higher DR (more than 90%)—where FS has 1.2% FPR on FashionMNIST, 21.3% on CIFAR-10, and unacceptable results on ImageNet. Furthermore, by cascading two detectors, our FPR is 0.4% in

FashionMNIST, 7.3% in CIFAR-10, and 2.1% in ImageNet.

The rest of the paper is organized as follows. § 2 first reviews popular methods to generate AEs and the state-of-the-art AE detection scheme, introducing notations and concepts we use. Then, we propose our solution in § 3. Experimental results are given in § 4. In § 5, we describe the previous work that is related, but not directly overlapped to our work and discuss some possible extensions.

2 Background

For an image denoted by x, the application DNN F estimates the probabilities of each class: $F(x) = \sigma(G(x))$ where $G(x) \in \mathbb{R}^K$ is a DNN that produces the logit vector (i.e., unnormalized prediction values for each class), K is the number of classes, and $\sigma(x)_j = \frac{\exp(x_j)}{\sum_{j'} \exp(x_{j'})}$ is the softmax function which converts logit vectors to normalized probability vectors. Given a training dataset inputs x and true class labels $y \in \{1, 2, \ldots, K\}$, the model $F(x)$ is usually trained via the following optimization problem:

$$\min_F \sum_{i=1}^{n} J(F(x_i), y_i) = \min_G \sum_{i=1}^{n} J(\sigma(G(x_i)), y_i) \tag{1}$$

where J is the cross entropy loss defined as

$$J(x,y) = \sum_{j=1}^{K} I(y=j) \log[F(x)]_j = \log[F(x)]_y . \tag{2}$$

In particular, we note that FS operates on the probability vectors, *i.e.,* $F(x)$, while we operate on the unnormalized logit vectors $G(x)$ as seen in Figure 1 and discussed in more detail in § 3. The label predicted by a classifier is $L(x)$, which is the index of the largest probability, *i.e.*,

$$L(x) = \arg \max_{i \in \{1,2,\ldots,K\}} [F(x)]_i . \tag{3}$$

When $L(x)$ matches the true index y, we say that the prediction is correct.

2.1 Generating Adversarial Examples

Given an input x, an adversarial example (AE) is defined as an altered input:

$$x^* = x + \delta, \tag{4}$$

such that $L(x^*) \neq L(x)$ (i.e., the classification is different than the original) but the perturbation δ is small (usually undetectable by human eyes). Here, we describe representative methods to generate AEs, which we use in our experiments.

- **Fast gradient sign method (FGSM)** [6] perturbs all input pixels by the same quantity ε in the direction of a gradient sign, *i.e.*,

$$x^* = x + \varepsilon \text{ sign}\left(\frac{\partial J(x,y)}{\partial x}\right), \tag{5}$$

where sign(v) is 1 if $v > 0$, -1 if $v < 0$, and 0 if $v = 0$. In other words, FGSM attempts to make an AE by adding a noise to each pixel in the direction that maximizes the increment in the cost function. The value of ε is chosen to be a multiple of ε_0, which corresponds to the magnitude of one-bit change in a pixel. We use the term *"perturbation level"* for ε.

- **Iterative FGSM (I-FGSM)** [14] iteratively applies FGSM (say, N times) with the minimal amount of perturbation at a time as follows:

$$x_{n+1} = \text{Clip}_{X,\varepsilon}\left\{x_n + \alpha \text{ sign}\left(\frac{\partial J(x_n,y)}{\partial x_n}\right)\right\}, \tag{6}$$

with $x_0 = x$ and $x_N = x^*$. Here, Clip$\{\cdot\}$ denotes a pixel-wise clipping operation, which ensures that the pixel value stays in the ε-vicinity of the original value, and in the valid range. The I-FGSM usually creates AEs more successfully than FGSM for the same ε because it can fine-tune the perturbation in one iteration based on the result from the previous iteration.

- **PGD** [15] is a first-order method which is very similar to I-FGSM. The main difference is PGD randomly initializes to a point x_0 within the L norm ball around x.

- **Carlini-Wagner** [2] minimizes a loss function containing two parts: the first part is the perturbation level, which is usually an L_p norm of δ while the second part contains the term that tries to flip the prediction, i.e.,

$$\underset{\delta}{\text{minimize}} \; \|\delta\|_p + c \cdot h(x+\delta), \tag{7}$$

where h is a specially designed function (see [2] for exact forms) to encourage label flipping such that $h(x+\delta) < 0$ only if $L(x^*) \neq L(x)$.

2.2 Current Defense Methods

While there are many possible adversarial defenses, we focus this background section on *detecting* adversarial examples and discuss other possible defenses in § 5.

2.2.1 Defenses by Detecting Adversarial Examples

Grosse *et al.* [7] proposed a statistical test to detect AEs from training dataset using maximum mean discrepancy. This method requires a large set of normal images and their corresponding AEs, and is not capable of detecting individual AEs. Thus, they also proposed detecting individual AEs by adding

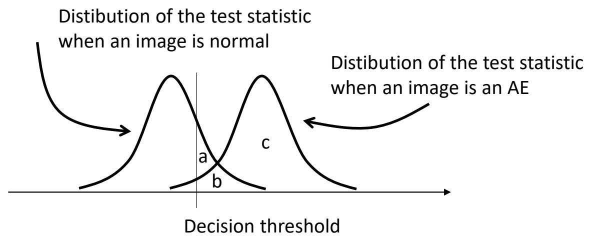

Distibution of the test statistic when an image is normal
Distibution of the test statistic when an image is an AE
a
b
c
Decision threshold

Figure 2: General relationship between the detection rate and false positive rate. The detection rate is the sum of the areas b and c, and the false positive rate is the sum of the areas a and b. Reducing the decision threshold increases the detection rate, but the false positive rate also increases.

an additional class to a DNN model, and train the model to recognize AEs as this new class. However, this requires changing the application DNN. Metzen *et al.* [19] proposed attaching a CNN-based detector at the middle layers of a DNN model. The detector is trained in a supervised manner to classify an input as normal or AE. This requires a large set of AEs to be generated offline for each perturbation level. Attaching a detector in the middle layers also causes the detector itself to be a large, and hence vulnerable, DNN. MagNet [18] also used a reference input to detect an AE. It uses an autoencoder and compares the input image with the autoencoder output. Since the autoencoder is supposed to reconstruct a given input to the one that is smoothed over possible additive noise, the output of the autoencoder plays a role as a reference. However, training an autoencoder is a challenging task, especially when the inputs have a wide variety like in ImageNet. For this reason, MagNet underperforms FS in ImageNet [32].

2.2.2 Feature Squeezing: State-of-the-art for detecting adversarial examples

The recent work, Feature Squeezing (FS) [32] has proposed to compare the original output to the output of a "squeezed" input to detect AE. A squeezed input is one that has been smoothed or simplified in some way such that it is less sensitive to adversarial perturbations added by adversaries. Example squeezing functions include quantizing the pixel values to reduce color bit depth or spatially smoothing via blur filters. FS computes two predictions—one from the application DNN model with the original input and the second from the same application DNN but with the squeezed input. Given the predicted probabilities from the original and squeezed input, FS then performs a threshold test on the L_1 distance between the two probability vectors. If the L_1 distance is larger than a threshold, FS decides that the input image is an AE, otherwise it is benign.

The intuition behind this design is that an adversarial modification will affect the original image but not the squeezed input image. We leverage this same insight in our approach. FS is reported to achieve a high DR for AEs, *e.g.,* the perfect DR for FPR = 0.05 for the MNIST dataset and DR = 0.64 for the same FPR on the ImageNet dataset. However, its decision threshold needs to be fixed targeting a particular perturbation level. It performs poorly for perturbation levels that the threshold is not targeted for. In order to maintain the high DR for a wide range of perturbations, the value of the threshold can be chosen small enough, but this inevitably incurs a high FPR, as Figure 2 explains. The high DR and low FPR values reported in the paper were obtained with large perturbation levels (which were left implicit in the paper), which may be detectable by a human-in-the-loop or simple threshold-based detectors. Fundamentally, the drawback of FS is that there is a rigid mapping of the perturbation level used to generate the AE and the L_1 norm threshold and we show that using a richer detector can lead to more precise detection across a wide range of perturbation levels.

3 Solution Approach

3.1 Solution Overview

The first insight of our approach is that we can improve the DR of the AEDs by using more complex classifiers than a simple thresholding scheme, in particular Gaussian Bayes classifiers and simple neural network classifiers. The second key insight is that we can create an ensemble of our detectors via AND aggregation when the following two conditions are met: (1) the errors of different detectors are approximately independent, and (2) the DR is high for each detector. We show empirical evidence behind these insights in § 4. When the second condition is not met, then the ensemble can be changed to be an OR aggregation as we describe in § 3.4.3.

As seen in Figure 1, we first compute a squeezed image x_q based on the original image x via quantization and pass both through the application DNN to get logit vectors $G(x)$ and $G(x_q)$. We then take the difference of these logit vectors $Z(x) = G(x) - G(x_q)$, where $Z(x)$ is a logit vector of length K (the number of classes the application DNN is classifying the images into)—similar to the difference vector computed in FS except that we use logit vectors instead of probability vectors (justification for this choice in next section). Finally, we predict the probability of this difference vector via a simple classifier denoted by $D\colon \mathbb{R}^K \to [0,1]$, which takes $Z(x)$ as input and outputs an estimated probability that the original input is adversarial. Importantly, we note that our detector D is a much simpler classifier than the application DNN $G\colon \mathbb{R}^M \to \mathbb{R}^K$, where M is the input dimensionality because the input dimension of the application DNN is usually much larger than the number of output classes, i.e., $M \gg K$; for example, a classifier for the CIFAR-10 dataset has $M = 32 \times 32 \times 3 = 3072$ input dimensions (32×32 images, each with three channels), but the number of classes K is only 10. We predict an image is adversarial if the predicted probability is above 0.5, i.e., greater probability of being an adversarial example rather than a unperturbed example.[1] Thus, by combining these steps, our single HAWKEYE detectors have the form:

$$D(G(x) - G(x_q)) > T\,. \tag{8}$$

We compare this to FS which is the following model:

$$\|\sigma(G(x)) - \sigma(G(x_q))\|_1 > T\,, \tag{9}$$

where σ is the softmax function. In both cases, the threshold T could be adjusted to trade-off DR and FPR. Notice that we use a more general probabilistic classifier D instead of simply the L1 norm and we use logit vectors directly before computing the softmax function. We see from Table 1 that there is no clear winner in both metrics; however, the FPR of using probability vectors tends to be higher. This is borne out later in end-to-end experiments where the FPR of FS turns out to be significantly higher than that of HAWKEYE (§ 4.2 and § 4.3). Thus, we decided to use logits for our HAWKEYE detectors throughout our experiments.

Table 1: **HAWKEYE** *performance using logit vectors versus probability vectors*

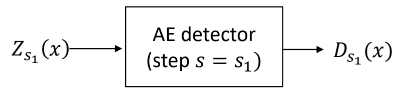
	FashionMNIST							
	c=0.4		c=2		c=10		c=50	
	DR	FPR	DR	FPR	DR	FPR	DR	FPR
Logits	72.9%	2.6%	76.2%	2.3%	83.5%	1.8%	83.8%	1.3%
Prob	77.0%	5.5%	79.8%	6.3%	81.0%	5.1%	80.2%	3.8%

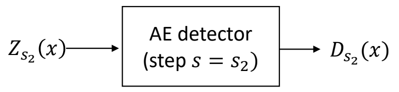
Our other innovation over FS is to reduce FPR significantly by cascading an ensemble of detectors using AND aggregation, i.e., only flag a sample as adversarial if both detectors predict adversarial, as illustrated in Figure 3. Ideally, for an AND aggregation with an ensemble of size two, the detectors will be correlated in terms of the samples each flags as adversarial (*i.e.,* both detectors will flag an adversarial sample as such) but are *uncorrelated* with respect to errors they make with benign samples (*i.e.,* if one makes an error in flagging a benign sample as adversarial, the other detector will not make an error on the same sample). In practice, we found that for our HAWKEYE detectors at different quantization levels have relatively weak correlations for both types of errors (see § 3.4 for details and results). Given the relative independence of detectors, we thus need individual detectors with a high DR, a property directly enabled by our use of more powerful classifiers. Thus, we are able to achieve significantly lower FPR while keeping DR relatively high via our AND ensemble of detectors, compared to a single detector (§ 4.2).

3.2 Design Details

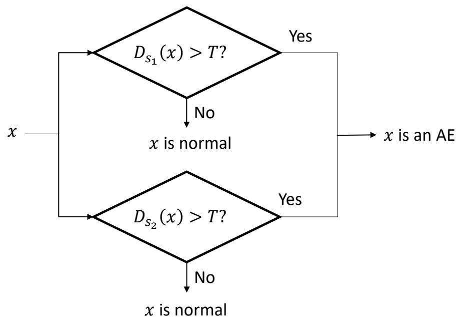
Here we provide some important details of our overall design about how images are quantized and how HAWKEYE is

[1]While the threshold could be adjusted to trade-off DR and FPR, we found that 0.5 was a reasonable default threshold in our experiments.





(a) Adversarial example detectors are trained with different values of the quantization step s.



(b) An example of cascading two detectors of different values for the step s.

Figure 3: ***Cascading detectors that use s_1 and s_2 as the quantization step size. An AND aggregation is used on the results of the two AEDs to identify an example as adversarial.***

trained with different quantization levels for different detectors in the ensemble. **Reference Images via Quantization.**

For a given image x, HAWKEYE creates a squeezed version of the image by reducing the input space of the image. We do this by creating a quantized image x_q, which is made by quantizing each pixel of x with step size s. Typically, a pixel of an image is represented in 8 bits, ranging in value over $[0,255]$. Thus, as an example, quantizing with step size $s = 128$ means that a pixel of x_q is represented as either 0 or 128, which is of 1-bit information. In general, after quantization with step size s, a pixel value v is represented as $s\lfloor v/s \rfloor$, where $\lfloor\cdot\rfloor$ denotes a floor function. The motivation to use such a quantized input is that as Figure 4 exemplifies, normal image and its corresponding AE become more similar after quantization, since the quantization process may nullify the noise added by adversaries that is smaller than the quantization step size s.[2] Thus, the quantized input x_q can play a role as an invariant reference of an image, regardless of the existence of malicious additive noise.

Training HAWKEYE. In order to train an AED, we use the same training images as were used for training the application DNN. For each such data point (which is by definition benign—call this as x), we create a set of AEs varying the perturbation level, using the standard adversarial example

[2]We also tested other techniques for squeezing the input space of a sample as introduced in the original Feature Squeezing paper, such as, the smoothed input applying blur filters. However, its performance was far worse than that with quantization. Thus we omit those results in our paper.

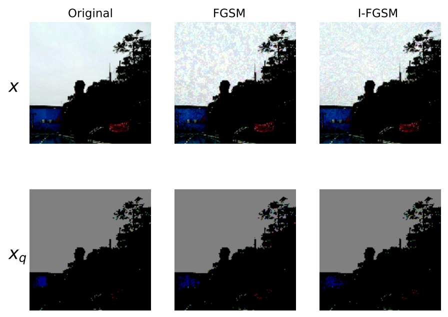


Figure 4: An example of quantization on an input image from ImageNet [3]. The second row is the quantized version of the image in the first row. We can see that after quantization, a normal image and its corresponding AE become more similar to each other.

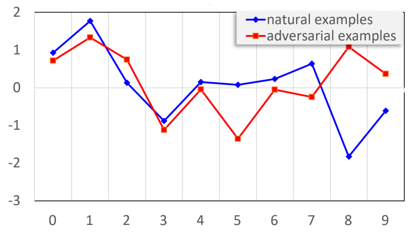
generation techniques described in § 2—call this the set of x^*'s. Thus, we have a balanced dataset of clean input images and generated AEs for training our detectors.

One possible drawback to HAWKEYE in comparison to FS is that FS only requires small number of AEs to fit its threshold T, whereas it may seem like our classifiers would require many AEs—which can be computationally expensive to generate especially for intensive iterative methods like CW. In light of this, we develop two variants of HAWKEYE: a simple Gaussian Bayes classifier (GB), which has minimal sample requirements and a simple NN classifier, which is relatively more powerful than GB and thus requires more samples. We empirically investigate this training cost in our experiments and see that while HAWKEYE-NN generally outperforms HAWKEYE-GB, the latter is more robust when the number of samples is small (§ 4.5). It is also worth noting that our AED is a totally separate classifier from the application DNN, *i.e.,* we do not modify the application DNN itself in any way. This is often a desired characteristic because it decouples the application DNN training process—which is often computationally expensive and time consuming—from the AED training process. Additionally, in certain situations, it is not possible to alter the application DNN either because it is proprietary or has already been deployed. This detector-based approach, however, is orthogonal to defenses that modify the application DNN to make it more resilient to AEs (these are more numerous than the current form of defense in this paper) and thus can be used in conjunction with these types of defense methods.

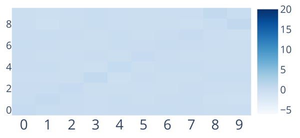
3.3 Two variants of HAWKEYE

3.3.1 HAWKEYE-GB: Interpretability via Simple Gaussian Bayesian Detector

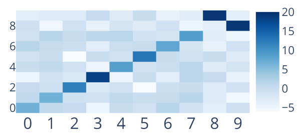
Because HAWKEYE is more complex than FS, one may argue that the intuition and interpretation behind FS (i.e., that an adversarial image's output will differ significantly from its reference image's output) is lost. However, by using a



(a) Mean of difference vector $Z(x)$ for each digit for benign and adversarial examples.



(b) Covariance of $Z(x)$ for benign examples.



(c) Covariance of $Z(x)$ for adversarial examples.

Figure 5: The Gaussian Bayes classifier parameters on the MNIST [16] dataset show that the means are not always the same and the variances of each dimension are definitely larger for adversarial examples but the variances are not uniform—suggesting that the classifier captures more interesting variation than a simple thresholding classifier as in FS.

simple Gaussian Bayes classifier as the AED, we can still interpret the detector. A Gaussian Bayes classifier merely estimates the mean and covariance of the benign logits difference $Z(x)$ and adversarial logits difference $Z(x^*)$. Classification is performed by using Bayes rule which determines whether a particular point is more likely to come from the benign Gaussian distribution or the adversarial Gaussian one. Thus, we can compare and interpret the mean vector and the covariance matrix of the benign and the adversarial samples. In our experiments, we found that the variance for adversarial examples $Z(x^*)$ is much larger than for clean examples $Z(x)$ as demonstrated in Figure 5 (look at the diagonal elements). Note from Figure 5a that simply looking at the first moment (the mean) of the difference vector $Z(x)$ does not present a clear distinction between benign and adversarial samples. This explains (in part) the weakness of FS—recall that FS computes the L_1 norm of the difference of probability vectors (similar to, though not identical to $Z(x)$) and the lack of separation of the mean of difference of $Z(x)$ means that FS cannot find a scalar threshold to accurately separate the two classes. This is to be expected because adversarial examples are expected to be farther from their reference image similar to the intuition of FS. Thus, despite some added complexity, HAWKEYE can have some interpretability especially if a simple interpretable classifier like HAWKEYE-GB is used.

3.3.2 HAWKEYE-NN: Better Performance via Neural Network Detector

To increase modeling power, we construct a simple neural network (NN) classifier for our detector—note this is significantly simpler than the application DNN. We create a simple

4-layer NN using standard fully connected layers and ReLU activations (the exact architectures for various datasets can be seen in Table 5 of the experiments section). We train these models by minimizing the standard cross entropy loss of the difference in logits $Z(x)$ often used for probabilistic classification, i.e.,

$$\min_D \sum_{i=1}^{n} -y_i \log D(z_i) - (1-y_i)\log(1-D(z_i)), \qquad (10)$$

where n is the number of training samples, $z_i = Z(x_i)$, $y_i = 1$ if x_i is adversarial, and $y_i = 0$ if the example x_i is benign. Thus, this optimization function rewards the AED when it outputs a 1 when x_i is adversarial and 0 when it is benign. We use a balanced training dataset of benign examples x and their corresponding AEs x^* as described previously.

3.4 Ensembles of HAWKEYE Detectors of Various Quantization Step Sizes

3.4.1 Dependency Between Detectors

In this section, we investigate the dependencies between detectors at various quantization levels in more detail to provide insight about when and why an ensemble of detectors could be useful. In particular, we consider the case of taking an AND-based ensemble of detectors (see Figure 3) rather than the OR-based ensemble that is used in FS [32]. Recap that an AND ensemble means that a sample is classified as adversarial if all the detectors in the ensemble classify it as adversarial. Thus, if the detectors make errors on *distinct* samples, *i.e.,* their dependency is small, then creating an ensemble is advantageous.

Naïvely, one might think that the simple correlation coefficient between the predictions of different detectors would be a good measure for the dependencies between detectors. However, if both detectors have high accuracy, then the correlation between detectors will be quite high (an extreme case is when both are nearly perfect, then their correlation is nearly one). Instead, we want to understand what dependency is there between the *errors* (*i.e.,* misclassified samples) of each detector; we investigate this dependency separately for false positive (FP) errors and false negative (FN) errors. Thus, we define the pairwise intersection over union (IoU) measures for FP and FN separately (*wlog* for two detectors in the ensemble):

$$IoU_{FP} = \frac{|FP_1 \cap FP_2|}{|FP_1 \cup FP_2|}, \quad IoU_{FN} = \frac{|FN_1 \cap FN_2|}{|FN_1 \cup FN_2|},$$

where FN_1 and FP_1 denote the set of FN examples and FP examples for detector 1 (and similarly for detector 2), $\cap$ is the intersection operator, $\cup$ is the union operator, and $|A|$ denotes the size of the set A. IoU is 1 if the sets are the same (*i.e.,* the detectors fail on the same examples) and 0 if the sets are entirely different (*i.e.,* the detectors fail on completely different examples). As we see in Figure 6, two detectors with different quantization levels have a low IoU for both FN and FP, especially when they have widely different quantization levels. Therefore, our detectors seem to make errors relatively independently of each other. This observation is critical for our proposal of using an AND ensemble of HAWKEYE detectors we describe in the next section.

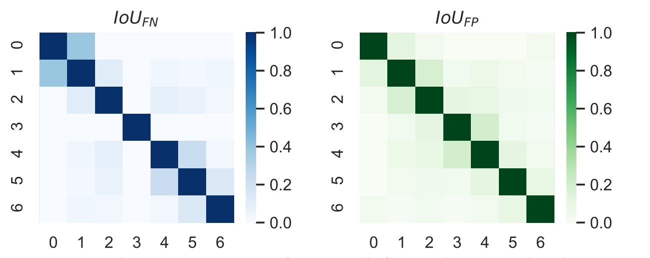


Figure 6: The pairwise IoU for FN (left) and FP (right) between detectors with different quantization levels shows that detectors with different quantization levels are relatively independent.

3.4.2 AND Ensemble

We now explore AND ensembles of detectors (Figure 3) in contrast to OR-based ensembles of FS [32]. In our experiments, each detector is separately trained using its own quantization step size, on the same x's and correspondingly generated x^*'s. Given this AND ensemble, what is the relationship between FPR and DR? While this question cannot be answered analytically, we consider the case where the detectors are independent—which we give empirical evidence for in the previous section. If the AEDs are completely independent, we have the following relationships:

$$\begin{aligned} \text{DR} &= (\text{DR of detector 1}) \times (\text{DR of detector 2}), \\ \text{FPR} &= (\text{FPR of detector 1}) \times (\text{FPR of detector 2}). \end{aligned}$$

That is, when we cascade detectors via AND semantics, the total FPR is the product of FPRs of all detectors, and thus it goes down. However, the total DR is also the product of DRs of all detectors. Thus, unless each detector achieves DR $= 1$, the total DR also decreases. For this reason, we only cascade detectors that have a high DR.

3.4.3 OR Ensemble

While HAWKEYE predominantly uses AND ensemble of detectors, there is one situation where it uses an OR ensemble. This is when through the training process, it becomes obvious that it is not possible to have any individual detector (*i.e.,* with a specific quantization step) to have a high DR. This decision is specific to the dataset and to the attack against which HAWKEYE is expected to be deployed[3]. What is "high"

[3]As we will show in the detailed evaluation (§ 4.4), the detectors of HAWKEYE are generally transferable across attack types, *i.e.,* if they are trained on one attack type, they perform creditably even against other attack types.

is subjective, but empirically we find that if the DR is below 0.75, it is advantageous to use the OR detector. The OR ensemble is made possible by the fact that the FPR of even low-performing detectors is not found to be high, reflecting the intuition of very different variances of benign and adversarial examples (Figure 5). In our experiments, we find that under certain cases of white-box attacks with the most powerful attack generator (CW), we have to use an OR ensemble of detectors (§ 4.6).

3.5 Parameter selection

Parameters of HAWKEYE are the decision threshold, the quantization level, and the number of detectors to cascade. We choose the decision threshold $T = 0.5$ for simplicity since this is a simple natural default for probabilistic classification though other thresholds could be explored to trade-off DR and FPR for each detector. We experiment with various levels of quantization in § 4. In general, a practitioner would likely train detectors at multiple quantization levels and choose the best performing detectors (though this is not the primary focus of this paper). To deal with white-box attacks, we propose randomizing the parameter settings of HAWKEYE. A logical choice for randomization is the quantization level. We show that this indeed makes HAWKEYE resilient to white-box attacks (§ 4.6). However, just by itself the number of quantization levels is not large (8 levels since pixel values are in $[0,255]$ and we use 2^k as a quantization step) and therefore does not provide enough entropy in isolation. However, coupled with the other parameter settings, there is enough entropy to make HAWKEYE resilient to white-box attacks. The number of detectors to cascade depends on the dataset but we usually find that cascading only two detectors is enough to lower the FPR significantly without lowering DR too much. More than two detectors in an AND ensemble would likely hurt DR a good deal as evidenced by our analysis in the previous section in which DRs multiply.

3.6 HAWKEYE Detector White-Box Attack

In this section, we develop a new *detector* white-box attack, *i.e.,* the adversary knows the parameters of the detector as well. We create a white-box attack separately for Feature Squeezing and our HAWKEYE-NN detectors to test the limits of their defense. Note that we already assume that the adversary has white-box access to the application DNN; this scenario assumes an even stronger attacker. We develop a detector white-box attack by adding a term to the standard CW L_2 attack:

$$\min_{\delta} \|\delta\|_2^2 + c \cdot f(x+\delta) + c_2 \cdot g(x+\delta), \tag{11}$$

$$\text{s.t. } x+\delta \in [0,1]^d \,, \tag{12}$$

where f is defined as:

$$f(x') = \max(\max\{G(x')_i : i \neq t\} - G(x')_t, -\kappa) \tag{13}$$

where $c_2 \cdot g(x+\delta)$ is the new term to encourage the AE to circumvent the detector. As in the original CW L_2 attack $x' = x+\delta$, t is the true class label, and $f(x')$ is the function that promotes a change of labels.
For attacking FeatureSqueezing, g is defined as:

$$g(x') = \|G(x') - G(Q(x'))\| \,, \tag{14}$$

For attacking HAWKEYE-NN, g is defined as:

$$g(x') = \mathcal{L}(H(G(x') - G(Q(x'))), 0) \,, \tag{15}$$

where, $\mathcal{L}(\cdot,\cdot)$ is the cross entropy loss, identical to HAWKEYE-NN's loss function, H computes the HAWKEYE detector probability being an AE, and Q is the quantization function. The basic idea is for the AE to try to fool both the application DNN via f and the detectors via g.[4] Because the quantization function Q is not differentiable, FS did not attempt a white-box attack for quantization squeezers. We propose to get around this by using a soft (*i.e.,* differentiable) quantization function Q_{soft} defined in [5]:

$$Q_{\text{soft}}(x) = \Delta\left(i + \frac{\tanh^{-1}(0.5k\Delta)\tanh(k(x-i-0.5))+1}{2}\right)$$

where $\Delta = 255/s$ is the interval length, s is the quantization step, and k is the parameter controlling the sharpness of Q_{soft} (*i.e.,* the tightness of approximation), and $i = \lfloor \frac{x}{\Delta} - 1 \rfloor$ denotes the quantization block corresponding to x. If k is large, then Q_{soft} behaves like Q but the differentiable parts are steeper (*i.e.,* harder to optimize). On the other extreme, if k is small, then Q_{soft} behaves like a linear function and thus is easy to optimize but does not approximate Q well. Thus, a middle value of k is likely a good choice for these detector white-box attacks.

3.7 Implementation

All codes were implemented using Python 3.6.10 with CUDA 10.2.89. Pytorch 1.4.0, Numpy 1.18.1, Matplotlib 3.1.3, scikit-learn 0.22.1 libraries were used to construct neural network architecture, optimizations, and defense mechanisms. The Torchvision Python library offers pretrained models also with some data structures and useful tools. We use the `torchattack` library [11] for generating all three attack types (FGSM, PGD, and CW). All experiments in the paper were run on a Tesla P100 PCIe 16GB GPU. The implementation has about 1200 lines of code with four parts: Training or loading the application DNN, generating adversarial examples,

[4]For implementation, we use the same trick as in the original CW attack to satisfy the box constraint on δ: instead of optimizing over δ directly, we define δ in terms of w as: $\delta = 0.5(\tanh(w)+1) - x$ and optimize over w.

training or loading HAWKEYE, and evaluating the attacks or defense methods. For comparison, we re-implement Feature Squeezing in the Pytorch library (the original paper was on a TensorFlow implementation). Dynamic search of the tuning parameter of Carlini-Wagner attack is not implemented which is computationally expensive so we fix the CW attack parameter to various different values to show a range of CW attacks.

4 Evaluation

4.1 Experiment Setup

Evaluation Metrics. We define *attack success rate* (ASR) to be the proportion of times that an attack method succeeds in fooling the application DNN (without any defense mechanism in place). For our detection methods, the *detection rate* (DR) is equivalent to the true positive rate (TPR) of the detector, which we use interchangeably in our results. The detectors' *false positive rate* (FPR) is based on the standard definition.

Datasets and Application Models. We use three common datasets to evaluate our defense method: FashionMNIST [31], CIFAR-10 [12] and ImageNet-ILSVRC2012 [26]. We leverage standard models for each of the datasets, where we use MobileNet_v2 [27] similar to the MobileNet model in [32] for ImageNet, and a ResNet [9] model for FashionMNIST and CIFAR-10. We summarize the application DNN information in Table 2. For FashionMNIST and CIFAR-10, we use a 14-layer ResNet which has 3 blocks. Each block has two 3×3 convolutional layers. We keep the rest of the model the same as the original ResNet model. We use the Adam optimizer in PyTorch to minimize cross entropy loss for 80 epochs with a learning rate of 10^{-3}. For ImageNet, we use the pretrained MobileNet_v2 model from the `torchvision.models` Python package.

Table 2: Summary of Application DNN Models

DataSet	Model	# parameters	Top1 Acc
FashionMNIST	ResNet	195450	94.40%
CIFAR-10	ResNet	195738	87.44%
ImageNet	MobileNet_v2	3504872	70.76%

Attack Methods. We use three different kind of attack methods to evaluate HAWKEYE, namely FGSM, PGD, and CW_2 described in § 2. We explore various parameter settings for each attack type to get a broad view of the attacks and defense evaluation. For FGSM attack, we set our parameter ϵ from $1/255$ to $16/255$ because perturbations larger than $16/255$ could be easily perceived by a human. For PGD attack, ϵ represents the strength of the attack or maximum perturbation, α is the step size according to [15], and *iter* means the number of steps. We do not use a random start because while this slightly improves the ASR, it seemed to have minimal affect on evaluating defense methods in preliminary experiments. The limits of our PGD parameters were chosen to show where PGD performs poorly (*i.e.,* ASR near 60%) up to a perturbation where it performs well (*i.e.,* ASR of 100%). For the Carlini-Wagner (CW) attack, the c parameter controls both the success rate and the perturbation amount—a higher c will have a higher success rate but also a higher perturbation amount. We fix the c parameter to four different values (0.4, 2, 10, 50) for simplicity instead of doing a binary search for c for every example, which is computationally expensive. We chose these parameter values to explore the space of possible attack strengths for CW. We set the number of iterations to 5,000 to ensure that the CW attack converges to real adversarial examples. (We found that the default in `torchattack` of 1,000 iterations did not always converge.) We note that the CW attack is far more computationally expensive (nearly $1000\times$ more expensive) than the other attack types as can be seen in Table 3 (even when fixing c to a single value). We summarize the attack strengths for each dataset in Table 4 which show the attack success rates (ASR) and average attack perturbations in terms of both L_2 and L_∞ norm for each dataset. From these tables, it can be seen that we explore a wide variety of ASRs and perturbation levels in our experiments.

Table 3: Average time to generate 1,000 adversarial examples

Unit: s per 1000 images	FGSM	PGD	CW
FashionMNIST	0.53	1.11	2269.00
CIFAR-10	2.14	1.45	3250.41
ImageNet	31.83	35.24	31356.50

Feature Squeezing Detector. The original FS paper computed the L_1 norm of the difference in probability vectors. To make FS more comparable to HAWKEYE which operates on the logit vectors, we compute the L_1 norm of the difference in logit vectors—which can be seen as a simple variant of the original FS. Further, from Table 1, we see that there is no clear winner between using the logit vector or the probability vector (*i.e.,* neither wins simultaneously on both TPR and FPR metrics). In our experimental setting, in order to provide a intuitive way to compare FS, we choose a threshold to match the FPR of HAWKEYE-NN with the same trained application DNN and same test examples. Note that we only use the train examples to set the threshold (*i.e.,* based on train FPR) so the test FPR in the result tables may differ significantly from the target FPR.

HAWKEYE Detectors. For HAWKEYE-NN, we construct simple four layer fully connected neural networks with ReLU activations as detailed in Table 5. We emphasize that these are far simpler networks than the application DNNs. We use the Adam optimizer with cross entropy loss for 20 epochs and a step size of 2×10^{-4}. For HAWKEYE-GB, we use the `QuadraticDiscriminantAnalysis` estimator from `sklearn.discriminant_analysis`, which fits a Gaussian

Table 4: Attack Strength of FGSM, PGD,and CW in FashionMNIST, CIFAR-10, and ImageNet Datasets

Attack	Parameters	FashionMNIST			CIFAR-10			ImageNet		
		ASR	L_2 norm	L_∞ norm	ASR	L_2 norm	L_∞ norm	ASR	L_2 norm	L_∞ norm
FGSM	ε=1/255	24.68%	0.1043	0.0039	63.06%	0.2091	0.0039	94.49%	1.5128	0.0039
	ε=4/255	65.15%	0.4153	0.0157	88.11%	0.8364	0.0157	96.04%	6.0293	0.0157
	ε=8/255	80.72%	0.8290	0.0314	88.79%	1.6683	0.0314	93.78%	12.0015	0.0314
	ε=16/255	84.53%	1.6486	0.0627	86.62%	3.3139	0.0627	91.66%	23.7714	0.0627
PGD	ε=2/255; α=1/255; iteration=1	61.55%	0.2500	0.0300	90.97%	0.4423	0.0398	40.93%	0.2465	0.0072
	ε=2/255; α=1/255; iteration=5	72.78%	0.3390	0.0417	88.33%	0.6080	0.0556	78.94%	0.4040	0.0120
	ε=2/255; α=1/255; iteration=10	88.45%	0.4938	0.0586	95.54%	0.8716	0.0761	88.98%	0.7835	0.0221
	ε=4/255; α=2/255; iteration=10	91.10%	0.7118	0.0759	97.26%	1.2769	0.1015	92.79%	1.3105	0.0338
	ε=8/255; α=4/255; iteration=10	17.06%	0.1039	0.0039	65.00%	0.2091	0.0039	58.17%	0.3416	0.0009
CW	c=0.4	58.05%	0.1899	0.0078	97.94%	0.3543	0.0078	61.56%	2.3506	0.0078
	c=2	62.50%	0.1974	0.0078	98.86%	0.3797	0.0078	99.86%	1.6863	0.0078
	c=10	99.79%	0.3805	0.0157	100.00%	0.7094	0.0157	100.00%	2.7668	0.0157
	c=50	100.00%	0.7239	0.0314	100.00%	1.3390	0.0314	100.00%	4.7335	0.0314

density to each class and then applies Bayes' rule to predict the classification probability. For most experiments, we train our detectors using 10,000 natural examples and 10,000 adversarial examples generated from these natural examples; however, in § 4.5, HAWKEYE, and especially HAWKEYE-GB, can perform well with a much smaller number of training examples.

Table 5: HAWKEYE*-NN Structure*

Layer	Type	Act.	Output shape (FashionMNIST)	Output shape (CIFAR-10)	Output shape (ImageNet)
Input			10	10	1000
1	FC	ReLU	10	1000	1000
2	FC	ReLU	10	100	100
3	FC	ReLU	10	10	10
4	FC		2	2	2

4.2 Single HAWKEYE Detector

We first want to explore the performance of a single HAWKEYE detector in terms of both DR and FPR. For this experiment, we choose one representative attack for each attack type and one representative quantization level for each dataset. The representative attacks have the following parameters: FGSM ($\varepsilon = 4/255$), PGD ($\varepsilon = 2/255$, $\alpha = 1/255$, iter= 10), and CW ($c = 2$), and the representative quantization levels of 16, 2, and 32 were used for FashionMNIST, CIFAR-10, and ImageNet respectively based on the quantization level that had the best accuracy across all detectors for all attacks and parameters. The results can be seen in Table 6. We first notice that for a majority of the cases, HAWKEYE-NN has a very high DR often close to 95%—which will be important for cascading detectors via an AND ensemble in future sections. Second, for almost all situations, the HAWKEYE detectors perform better or comparably to FS, which either has a lower DR or a signficantly higher FPR. In particular, on ImageNet, FS performs very poorly for all attack types. We can also see that HAWKEYE-GB can perform reasonably well and sometimes comparable to HAWKEYE-NN except in the most difficult situations. Thus, if fast computation or interpretability are important, HAWKEYE-GB could be used. Additionally, we note that there are multiple cases where the FPR is at unacceptably high rates such as for the CW attacks on CIFAR-10 and ImageNet.

Table 6: Single detector DR and FPR performance of HAWKEYE *(shortened as "HK") and FS, with representative FGSM ($\varepsilon = 4/255$), PGD ($\varepsilon = 2/255$, $\alpha = 1/255$, iter= 10), and CW ($c = 2$) attacks and representative quantization levels of 8, 2, and 32 for FashionMNIST, CIFAR-10 and ImageNet respectively. Complete results across attack types and quantization levels are in [1].*

		FGSM		PGD		CW	
		DR	FPR	DR	FPR	DR	FPR
Fashion MNIST	HK-GB	99.9%	4.6%	96.1%	6.4%	95.8%	3.3%
	HK-NN	99.5%	1.4%	92.3%	3.5%	96.1%	1.3%
	FS	98.5%	1.2%	81.9%	2.6%	88.5%	0.8%
CIFAR-10	HK-GB	96.7%	11.2%	100.0%	0.6%	72.1%	55.2%
	HK-NN	96.1%	11.4%	100.0%	0.9%	50.4%	27.1%
	FS	97.2%	21.3%	100.0%	0.6%	72.1%	55.2%
ImageNet	HK-GB	91.2%	34.6%	67.1%	9.1%	46.4%	26.8%
	HK-NN	93.6%	7.0%	97.9%	2.0%	64.4%	39.2%
	FS	0.0%	1.6%	6.6%	0.1%	46.6%	28.9%

4.3 AND Ensemble of HAWKEYE Detectors

We now examine the empirical effect of cascading HAWKEYE detectors via an AND aggregation as seen in Table 7 and compare to FS (that uses an OR aggregation). We chose the quantization levels for the ensemble based on the best average accuracy of the HAWKEYE-NN detectors across all attacks.

For FashionMNIST, the FPR for AND-HAWKEYE-NN goes down to less than 0.5% compared to the single detector case which had FPRs up to 3.5%. For CIFAR-10 and ImageNet, the FPR also shows a significant decrease compared to single detector (*e.g.,* for the CIFAR-10 FGSM attack, the FPR goes down from about 11% to 7%, or for AND-HAWKEYE-GB on ImageNet FGSM attack, the FPR goes down from 34.6% to 28.3%). These lower FPRs are in stark contrast to FS with an OR ensemble which greatly increases the FPR (*e.g.,* FS on CIFAR-10 across attacks is more than 49%). The drawback to the AND-HAWKEYE ensemble is that the DR also goes down a little. However, many of the results still have high detection rates (often higher than 85%) because the original detectors had very high detection rates. Thus, if the original detection rates are high (as is generally true for our HAWKEYE detectors), an AND-ensemble can significantly lower FPR while maintaining a relatively high DR.

Table 7: Ensemble detector results for AND-*HAWKEYE*-GB, AND-*HAWKEYE*-NN and OR-FS with FGSM* ($\varepsilon = 4/255$)*, PGD* ($\varepsilon = 2/255$, $\alpha = 1/255$, *iter*$= 10$)*, and CW* ($c = 2$) *for ensembles of detectors at different quantization levels, namely 8 and 16 for FashionMNIST, 2 and 4 for CIFAR-10, and 32 and 64 for ImageNet.* HAWKEYE*-NN outperforms the others, though the CW attack on the more complex datasets CIFAR-10 and ImageNet turns out to be challenging for all.

		FGSM		PGD		CW	
		DR	FPR	DR	FPR	DR	FPR
Fashion MNIST	AND-HK-GB	99.6%	2.1%	96.1%	1.7%	95.6%	0.9%
	AND-HK-NN	98.5%	0.4%	93.1%	0.2%	95.8%	0.1%
	OR-FS	98.7%	3.4%	94.8%	3.0%	95.0%	1.0%
CIFAR-10	AND-HK-GB	78.3%	7.3%	100.0%	0.7%	36.2%	19.0%
	AND-HK-NN	83.2%	7.8%	99.9%	0.0%	38.2%	18.7%
	OR-FS	97.7%	49.9%	100.0%	53.3%	78.9%	73.8%
ImageNet	AND-HK-GB	83.2%	28.3%	66.4%	5.3%	40.2%	21.4%
	AND-HK-NN	89.2%	2.1%	91.6%	0.2%	47.5%	19.2%
	OR-FS	0.1%	2.6%	2.4%	0.0%	54.9%	38.6%

4.4 Generalizability Across Different Attacks

One possible concern is that HAWKEYE will overfit to the AEs used during training and will not generalize to other attack methods—*i.e.,* it will only work for the attack type that it was trained on. We explore this concern by training our AND-HAWKEYE-NN ($s = 8$ and $s = 16$) on the simple and fast FGSM method, but testing the detector on the PGD and CW attacks for the FashionMNIST dataset as seen in Table 8. While there is some performance degradation in either DR or FPR, AND-HAWKEYE-NN continues to perform well in most cases (in particular for $\varepsilon = 2/255$ the performance degradation is minimal). In one odd corner case with $\varepsilon = 8/255$, one of the two detectors in the AND ensemble performs quite poorly on the new attack, and thus the AND of the two detectors has close to 0% DR. This seems to be an exception possibly caused by a bad local minimum of HAWKEYE-NN training—thus, one possible approach would be to use HAWKEYE-GB, which has a well-understood solution to avoid this odd case. However, overall, the results support the idea that HAWKEYE does not overfit to the training attack method and can in many cases generalize to new attack methods. As one final observation, we note that small perturbations (*e.g.,* $\varepsilon = 1/255$) of FGSM attacks rarely succeed in attacking the application DNN (*i.e.,* the ASR is relatively low). However, HAWKEYE trained with smaller perturbation has the best performance among different kinds of attacks, even at large perturbations. This suggests that training on smaller perturbations of FGSM will create a detector that is generalizable over a wide set.

4.5 HAWKEYE Training Cost

Another possible concern with HAWKEYE is that because HAWKEYE is more complex, it will require many more AEs to train than FS—particularly the high-cost AEs based on the CW attacks can take a long time to generate. First, we note that our result in the previous section on the generalizability of HAWKEYE to other attack methods suggest that it is possible to train using cheap AEs such as FGSM-AEs while still being able to defend against more complex attacks like CW. Second, we explore the number of training examples needed to perform well in this experiment as seen in Figure 7, which shows the TPR (*i.e.,* DR) and FPR of HAWKEYE ($s = 8$ and $s = 16$). For AND-HAWKEYE-NN, even with a small number of AEs (500), we can quickly get a high detection rate above 95%. An additional 500 AEs can drive the FPR below 0.05 for AND-HAWKEYE-NN. We highlight that AND-HAWKEYE-GB performs very well even with only 100 or 200 samples. This highlights the simplicity of AND-HAWKEYE-GB and the consequent ease of training, while achieving reasonable accuracy. Thus, if low computational cost of training is a key consideration, we can use HAWKEYE-GB detectors. We also note that there are well-known estimators for multivariate Gaussian distributions even with a very small number of samples (*e.g.,* Graphical Lasso [4]), which could even further reduce the number of required samples in some cases.

4.6 Detector White-Box Attacks

Finally, we explore the question, what if the adversary not only knew the parameters of the application DNN but they also knew the parameters of our AED (i.e., the HAWKEYE detector is a white-box to the adversary)? Does HAWKEYE fail in this case or does it still provide an effective deterrent? We will use our HAWKEYE white-box attack defined in § 3.6 with $k = 100$ for the soft quantization function Q_{soft}.

Detector White-Box Attack Strength. In Table 9 we show the detector white-box attack strength in the case when the adversary has full knowledge of the parameters and quantization level used for the detector (almost like oracle knowledge of

Table 8: Generalizability Results: AND-HAWKEYE-NN (s=8 and s=16) performance on FashionMNIST when training on FGSM but evaluating on PGD and CW attacks. The last row is for comparison when training and testing on the same attack.

Train Attack		Test Attack: CW L2								Test Attack: PGD							
		c=0.4		c=2		c=10		c=50		α=ε=1/255 iter=5		α=ε= 1/255, iter=10		α=ε= 2/255 iter=10		α=ε= 4/255 iter=10	
		DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR	DR	FPR
FGSM	ε=1/255	89.8%	4.0%	95.8%	4.1%	94.3%	4.9%	95.0%	5.3%	98.2%	5.9%	98.0%	3.7%	97.6%	4.7%	92.6%	5.4%
	ε=2/255	90.8%	1.4%	96.1%	1.7%	96.7%	1.7%	95.2%	2.3%	97.1%	2.2%	97.2%	1.5%	99.8%	2.0%	93.8%	2.0%
	ε=4/255	86.5%	0.8%	95.3%	0.2%	95.3%	0.2%	94.6%	0.6%	83.5%	0.7%	85.2%	0.5%	99.9%	0.6%	88.6%	0.6%
	ε=8/255	0.0%	0.1%	0.3%	0.2%	0.0%	0.2%	0.0%	0.2%	0.6%	0.1%	0.6%	0.1%	0.1%	0.2%	5.0%	0.1%
	ε=16/255	59.9%	0.1%	78.3%	0.1%	80.1%	0.0%	68.3%	0.2%	34.8%	0.2%	44.1%	0.1%	93.9%	0.0%	44.1%	0.0%
Same as test attack		*90.2%*	*0.6%*	*95.8%*	*0.1%*	*95.4%*	*0.0%*	*94.7%*	*0.1%*	*91.8%*	*1.0%*	*93.1%*	*0.2%*	*99.6%*	*0.1%*	*98.5%*	*0.1%*

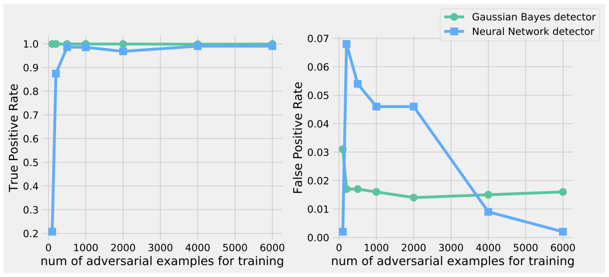


Figure 7: Performance of AND-HAWKEYE-GB and AND-HAWKEYE-NN detectors ($s = 8$ and $s = 16$) on FashionMNIST with different numbers of AEs in their training sets and with an equal number of benign examples in each case. The attack method is FGSM with $\varepsilon = 16/255$. HAWKEYE-GB is resilient to a small number of training samples.

HAWKEYE-NN). The white-box attack has 61.86% success rate before detection, which is *lower* than 74.58% black-box attack. This indicates that in crafting the white-box attack, since it has an objective of fooling the detector, it becomes a little less effective against the application DNN itself. Additionally, the white-box attack takes more than twice as long to generate and introduces higher perturbation to make the benign examples to become adversarial. A higher perturbation runs the risk of being detected by a simple detector or by the human eye. However, the white-box attack becomes much harder to detect by HAWKEYE (which is only trained on black-box attacks). As an extreme case, for the HAWKEYE-NN detector with a single detector $s = 2$, the detection rate decrease from 84% to 0%. We attempt to alleviate this weakness by considering randomization of the quantization level in the next experiment.

Table 9: Carlini-Wagner L_2 Black Box vs White Box Attack against Fashion-MNIST, optimized to attack HAWKEYE-NN

Attack		ASR	L_2 norm	L_∞ norm	Time (s)
CW	c=2	74.58%	0.155	0.026	305
White Box CW	c=2, c2=10 s=2	61.86%	0.237	0.039	794

Performance HAWKEYE with Different Quantization Steps. In this scenario, we consider what might happen if we randomize the quantization step (*i.e.,* the attacker does not know the quantization step of our detector). Figure 8 shows the performance of HAWKEYE defending against white-box attack while using different quantization steps where the white-box attack assumes a quantization step of size 2. Because white-box attacks only enhance the AEs but do not change the benign examples in testing, the white-box attack will affect HAWKEYE's DR but the FPR (which only depends on benign examples) relatively unaffected. The white-box attack are most effective when the attack and defense using the same quantization step parameter (*i.e.,* when the white-box attack quantization matches HAWKEYE's quantization step). However, when the quantization step of the white-box attack does not match the quantization step of the HAWKEYE detector (*e.g.,* quantization step greater than 2 in our results), then the DR is quite high and near what it would be against a black box attack. Thus, we can weaken the attack significantly by randomizing the quantization step size for each attack.

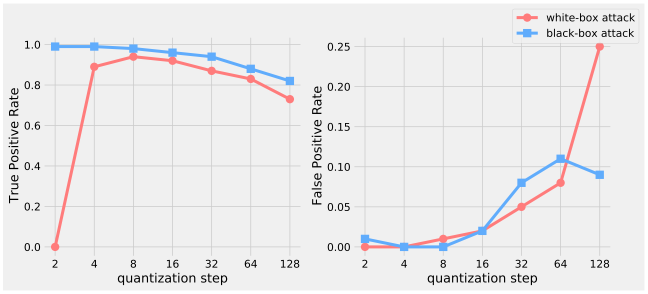


Figure 8: Performance of HAWKEYE single detectors with different quantization steps against white-box attack with quantization step=2 on FashionMNIST. Quantization levels farther off from what the adversary is trained on are more effective.

AND and OR Ensembles for Defending Against White-Box Attacks. In this experiment, we explore both AND and OR ensembles for defending against detector white-box attacks. As in the previous experiment, we want to consider if we can reduce the white-box effectiveness by using randomization—in this case, we could randomize the quantization levels of the different detectors in the ensemble. In Figure 9, we show that the DR and FPR of HAWKEYE for detector white-box attacks using different quantization levels and different aggregation approaches, *i.e.,* OR and AND aggregation, where the white-box attack is trained with an ensemble with quantization levels 2 and 4. First, for a single detector, we can see that the DR is very low for the attacked quantization levels (2 and 4) but the DR is fairly high for the other levels (8 and 16). Now we also notice that OR ensembles are more resilient to white-box attacks (see 4 OR 8 compared to 4 AND 8 ensembles) because the attacker only has to attack one of the two detectors in an AND ensemble to bypass the defense. Thus, if it is reasonable to assume that an adversary will know about the detectors, an OR ensemble (with randomization) may be more appropriate.

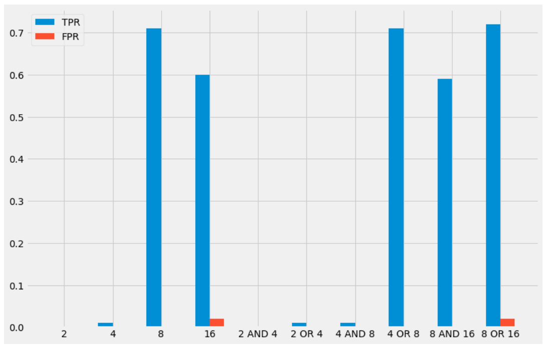


Figure 9: The performance of* HAWKEYE*-NN on FashionMNIST for white-box CW attack against quantization levels* $s = 2$ *and* $s = 4$ *simultaneously with parameters* $c = 2$ *and* $c_2 = c_3 = 10$*. This demonstrates that OR ensembles may be better for handling white-box attacks and that randomization of the constituent detectors might alleviate the attack strength of white-box attacks.

5 Discussion

Other Methods for Creating Adversarial Examples. Jacobian-based saliency-map approach (JSMA) by [25] aims at fooling a classification model into outputting a specific target class t by iteratively perturbing one or two pixels at a time. Since only one or two pixels change at a time, this method takes a long time to create an AE (unusably long for ImageNet) compared to FGSM or I-FGSM [32, 33], and the resulting AEs usually contain pixels with high intensity, which can be relatively easily detected by humans. JSMA also requires a huge amount of memory to compute a Jacobian matrix [32]. Thus, we did not consider it in our evaluation. Deepfool [20] is another untargeted attack method, creating AEs with the assumption that a classifier is a linear model. Since DNNs are not actually linear, Deepfool iterates a process, where a DNN is first approximated to a linear model and the minimum level of perturbation is decided for each approximation. Although Deepfool can successfully mislead DNNs, AEs created by it often look too distorted from their original images, thus easily detectable by humans, as reported in [32]. Thus, we exclude it from our evaluation as well.

Defenses by Hiding Gradients. The basis of the first generation defense mechanisms against adversarial examples is what is often called “gradient masking", which attempts to hide a useful gradient (like $\frac{\partial J(x,y)}{\partial x}$ or $\frac{\partial [F(x)]_t}{\partial x}$) in the vicinity of the input data points. Adversarial training by [6] enhances robustness by exposing a model to AEs in advance during the training phase. Distillation is a recent advance in deep learning [10], which found that knowledge in an NN can be transferred to a smaller model. It was shown in [23] that a distilled model can be more robust to adversarial examples than its original model. However, [24] reported later that the distilled model is weak against black-box attacks.

Random Ensembles or Bias Term for White-Box Attacks. In our experiments, we show that AND ensembles can be used for driving down the FPR if DR is high whereas OR ensembles are better at handling white-box attacks or cases where DR is low. More generally, it would be possible to consider more general ensembles that include both OR and AND aggregation to combine the benefits of both types of aggregation. Additionally, we could add more randomness for defending against white-box attacks by randomizing the size and configuration of the ensemble to handle each new example. Finally, we could also add a random bias term to the image input so that the quantization is slightly different every time, *i.e.,* $Q(x+b)$ where b has a maximum absolute value of the quantization step size. Both of these additional randomization tactics would make it more challenging for an adversary to circumvent our defenses.

6 Conclusion

In this paper, we have tackled the problem of detecting adversarial examples and our focus has been on adversarially generated images. This line of work is orthogonal to work that tries to make the application model resilient; here once an adversarial example is detected, it can be discarded and the application model does not even need to deal with such an example. We bring forth three innovations to this problem—using logit vectors as the output of the application DNN that is to be protected, passing the logit vector through a classifier (a simple fully-connected NN or a Gaussian Bayes classifier), and creating an ensemble of detectors (using the logical AND or OR ensemble). Taken together, we find that HAWKEYE outperforms Feature Squeezing; the latter tends to have much higher FPR and is weak against powerful attacks against complex datasets. We shed light on why an AND ensemble works, when an OR ensemble is desired, and insights on defending against a white-box attack through randomization.

References

[1] Anonymous. HAWKEYE: Adversarial Example Detection through Ensemble Detectors — Supplemental Material. https://bit.ly/hawkeye-usenix21, 2020. Accessed : 15-June-2020.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

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

[4] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[5] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. *Proceedings of the IEEE International Conference on Computer Vision*, pages 4852–4861, 2019.

[6] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR 2015)*, 2015.

[7] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. On the (statistical) detection of adversarial examples. *CoRR*, abs/1702.06280, 2017.

[8] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, 2014.

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

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[11] Harry Kim. A pytorch implementations of adversarial attacks and utils. https://github.com/Harry24k/adversarial-attacks-pytorch, 2020. Accessed : 15-June-2020.

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

[13] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *Workshop track - ICLR*, 2017.

[14] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[15] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016.

[16] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998. Accessed: 15-June-2020.

[17] Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, Wenchang Shi, and Xiaofeng Wang. Detecting adversarial examples in deep networks with adaptive noise reduction. *CoRR*, abs/1705.08378, 2017.

[18] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. *CoRR*, abs/1705.09064, 2017.

[19] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*, 2017.

[20] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[21] Aran Nayebi and Surya Ganguli. Biologically inspired protection of deep networks from adversarial attacks. *CoRR*, abs/1703.09202, 2017.

[22] OpenAI. Attacking machine learning with adversarial examples. https://blog.openai.com/adversarial-example-research/, 2017. Accessed: 15-June-2020.

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

[24] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS 2017)*, pages 506–519, 2017.

[25] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P 2016)*, pages 372–387, 2016.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2018.

[28] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *CoRR*, abs/1803.08533, 2018.

[29] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *International Conference on Learning Representations (ICLR)*, 2018.

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[31] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[32] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, pages 1–15, 2018.

[33] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107, 2017.

HAWKEYE: Adversarial Example Detection through Ensemble Detectors Supplementary Material

A Results of FeatureSqueezing and HAWKEYE using single detector against Gray-Box attack

Tables 1, 2, 3 show the main results of Hawkeye and FeatureSqueezing under FGSM, PGD, and CW attacks with different parameters. In the tables, QS means quantization step;ϵ_0 represents the minimum possible distance which equal to $1/255$. ϵ in FGSM represents the perturbation level which also represents the L_∞ distance between adversarial examples and their benign prototype; ϵ in PGD represents the radius of the $L_inf ty$ projection ball, and α represents the step size. *it* means the total iteration times. c in CW represents the tuning parameter of the term which tries to flip the classification results. FS-ACC mains the defense based on FeatureSqueezing which means to maximize the accuracy. FS-FPR tries to fix the false positive rate(FPR) to be equal to HE-NN's results using the training set. HE-GB is the results based on Hawkeye Gaussian Bayes classifiers, and HE-NN is the results based on Hawkeye neural network classifiers.

B Results of FeatureSqueezing using single detector against White-Box attack

Table 4 represents the results of FeatureSqueezing using single detector against White-Box attack, where $c2$ is the tuning parameter of the term which tries to minimize the L_1 norm of logit vector between x and x_q, s is the quantization step, $s = 2$ means it attacks the $s = 2$ and $s = 2,4$ means it attacks both $s = 2$ and $s = 4$. The results show the similar phenomenon with HAWKEYE white-box attacks. The adversarial examples effectively attack the detector with given s but shows little impact on detector with s which is not equal to the attack parameter. However, due to lower detector rate, the overall performance of Feature Squeezing is lower, which narrows the possible choices of random ensemble.

Table 1: FeatureSqueezing and Hawkeye performance on FashionMNIST data with different attacks

Def	QS	FGSM								PGD										CW							
		$\epsilon = 1\epsilon_0$		$\epsilon = 4\epsilon_0$		$\epsilon = 8\epsilon_0$		$\epsilon = 16\epsilon_0$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 1$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 5$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 10$		$\epsilon = 4\epsilon_0$ $\alpha = 2$ $it = 10$		$\epsilon = 8\epsilon_0$ $\alpha = 4$ $it = 10$		c=0.4		c=2		c=10		c=50	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
FS-ACC	2	99.9%	0.3%	78.1%	27.7%	67.6%	43.5%	99.2%	3.3%	99.8%	0.2%	100.0%	0.0%	100.0%	0.0%	100.0%	0.2%	72.3%	24.2%	99.8%	0.4%	99.8%	0.1%	100.0%	0.4%	99.8%	1.3%
	4	99.7%	2.7%	67.8%	53.2%	81.7%	78.9%	98.9%	5.1%	98.6%	3.2%	99.9%	0.4%	99.8%	0.1%	99.4%	4.7%	71.5%	62.4%	97.4%	1.3%	98.8%	0.3%	98.6%	0.3%	98.2%	1.3%
	8	97.4%	11.9%	99.7%	2.6%	96.1%	94.9%	99.4%	4.0%	91.8%	13.6%	97.9%	2.1%	99.3%	2.3%	99.8%	0.8%	98.8%	8.7%	92.2%	2.2%	97.2%	1.3%	95.7%	0.7%	95.5%	1.8%
	16	88.9%	28.9%	98.2%	7.7%	99.9%	3.6%	100.0%	2.1%	85.2%	34.2%	94.7%	8.1%	94.4%	8.0%	99.7%	1.8%	100.0%	1.3%	89.0%	5.5%	94.5%	2.7%	93.7%	2.1%	94.4%	1.5%
	32	76.4%	46.1%	95.6%	16.4%	99.1%	7.7%	99.3%	5.9%	69.9%	43.7%	81.8%	18.4%	90.0%	18.9%	99.1%	5.0%	99.6%	2.6%	85.1%	12.2%	89.1%	7.2%	89.1%	16.3%	92.2%	4.8%
	64	70.1%	57.5%	91.0%	37.4%	95.2%	23.3%	95.5%	19.9%	72.4%	62.1%	73.9%	37.5%	75.3%	32.0%	95.8%	16.4%	98.1%	9.5%	77.7%	24.3%	84.1%	22.6%	84.9%	15.6%	87.7%	62.1%
	128	67.0%	65.8%	85.1%	66.0%	87.0%	53.9%	88.0%	50.7%	80.1%	78.8%	71.3%	59.7%	74.0%	61.4%	88.3%	40.1%	99.7%	30.5%	82.0%	60.9%	75.2%	48.2%	84.9%	16.3%	78.6%	45.4%
FS-FPR	2	99.9%	0.1%	66.0%	17.8%	38.7%	24.4%	87.4%	0.4%	99.6%	0.1%	100.0%	0.1%	100.0%	0.0%	100.0%	0.0%	53.0%	15.0%	99.0%	0.0%	99.7%	0.1%	100.0%	0.3%	99.8%	1.0%
	4	96.7%	1.3%	44.3%	31.0%	46.7%	54.7%	87.1%	0.6%	89.8%	1.1%	99.6%	0.1%	98.7%	0.0%	96.9%	1.0%	31.2%	30.0%	94.3%	0.2%	98.3%	0.2%	97.9%	0.1%	95.8%	0.2%
	8	76.3%	3.6%	98.5%	1.2%	46.1%	61.3%	96.1%	2.4%	71.0%	5.9%	94.6%	1.4%	94.6%	0.5%	99.1%	0.1%	93.2%	2.8%	92.9%	2.9%	94.9%	0.4%	94.7%	0.2%	95.1%	0.7%
	16	65.9%	15.0%	91.1%	2.6%	98.4%	1.1%	99.0%	0.6%	63.9%	19.7%	80.4%	3.0%	81.9%	2.6%	92.2%	-3.9%	98.8%	0.6%	85.2%	2.6%	88.5%	0.8%	89.3%	0.7%	93.2%	0.5%
	32	55.0%	26.9%	80.5%	6.1%	84.8%	2.8%	77.7%	0.7%	45.3%	26.4%	68.3%	10.8%	73.0%	10.2%	92.2%	2.1%	79.6%	0.1%	76.4%	5.2%	81.6%	3.9%	81.3%	1.7%	86.3%	2.0%
	64	41.3%	32.2%	54.5%	14.1%	47.2%	4.8%	47.9%	3.4%	32.4%	26.1%	38.9%	15.5%	47.0%	14.9%	85.3%	9.1%	71.4%	1.5%	50.7%	11.0%	57.8%	7.3%	51.9%	5.1%	55.8%	5.3%
	128	12.2%	12.3%	32.8%	23.5%	22.6%	11.3%	17.8%	7.6%	41.0%	38.8%	37.9%	30.4%	41.1%	31.0%	44.4%	13.1%	58.5%	4.1%	32.7%	25.2%	24.3%	15.7%	17.1%	6.6%	26.5%	12.8%
HE-GB	2	100.0%	2.0%	62.0%	14.7%	45.5%	18.3%	99.9%	3.4%	100.0%	2.2%	100.0%	0.9%	100.0%	1.2%	100.0%	1.9%	61.6%	12.1%	100.0%	0.9%	99.9%	0.5%	100.0%	1.4%	100.0%	2.5%
	4	100.0%	4.4%	47.8%	27.9%	83.4%	68.7%	99.6%	4.3%	99.9%	3.9%	100.0%	3.1%	98.2%	0.0%	99.8%	3.6%	73.0%	53.7%	98.5%	1.5%	99.6%	0.9%	99.8%	1.2%	99.7%	2.2%
	8	98.8%	7.0%	99.9%	4.6%	85.5%	62.3%	99.9%	4.7%	93.2%	8.4%	100.0%	4.6%	99.8%	3.9%	100.0%	1.8%	99.4%	5.4%	93.5%	2.5%	98.0%	1.7%	97.5%	1.8%	97.0%	2.5%
	16	78.1%	13.9%	99.6%	6.8%	100.0%	3.6%	100.0%	2.8%	66.9%	15.3%	95.6%	6.8%	96.1%	6.4%	100.0%	2.9%	100.0%	2.7%	88.4%	4.1%	95.8%	3.3%	94.7%	2.6%	95.4%	3.5%
	32	52.1%	16.9%	96.3%	9.0%	99.8%	6.1%	99.8%	4.1%	45.6%	19.0%	79.5%	11.5%	81.6%	10.1%	99.8%	5.0%	100.0%	3.1%	84.2%	7.1%	90.7%	6.3%	91.0%	3.7%	93.7%	4.9%
	64	47.5%	31.8%	89.7%	16.2%	96.5%	8.7%	98.1%	7.2%	44.9%	28.2%	57.5%	18.0%	62.4%	15.8%	96.7%	9.5%	99.9%	5.0%	73.7%	12.4%	84.4%	10.2%	84.6%	8.0%	88.3%	9.2%
	128	60.8%	47.6%	77.2%	21.9%	89.5%	13.8%	91.2%	11.6%	59.3%	49.6%	48.5%	27.9%	50.4%	25.3%	86.8%	15.7%	43.7%	0.0%	71.6%	26.7%	76.3%	19.6%	75.0%	16.5%	75.3%	16.0%
HE-NN	2	99.9%	0.3%	66.3%	19.7%	55.4%	27.3%	98.1%	0.9%	99.7%	0.0%	99.9%	0.0%	99.8%	0.0%	99.3%	0.1%	71.7%	17.0%	99.8%	0.1%	99.9%	0.1%	99.9%	0.3%	99.5%	0.7%
	4	99.3%	1.4%	52.5%	33.3%	73.8%	58.6%	98.2%	1.7%	97.5%	1.3%	99.8%	0.4%	99.7%	0.0%	98.8%	2.2%	56.6%	30.9%	98.8%	0.6%	99.5%	0.5%	99.1%	0.4%	99.0%	0.4%
	8	95.6%	5.1%	99.5%	1.4%	80.1%	53.7%	99.0%	2.4%	90.6%	6.7%	98.3%	1.7%	97.1%	1.1%	100.0%	0.3%	98.1%	3.6%	98.8%	4.6%	97.9%	0.5%	97.3%	0.7%	96.4%	0.7%
	16	88.1%	17.7%	98.8%	3.1%	99.6%	1.8%	99.9%	0.6%	85.2%	21.5%	93.6%	4.2%	92.3%	3.5%	99.8%	0.6%	99.9%	0.4%	90.2%	3.7%	96.1%	1.3%	95.2%	1.4%	94.8%	1.0%
	32	71.5%	30.9%	95.5%	7.4%	98.7%	3.0%	99.2%	1.2%	66.7%	30.3%	84.0%	12.5%	86.9%	11.5%	99.2%	2.5%	99.9%	0.3%	86.9%	6.8%	93.8%	4.1%	92.1%	3.1%	93.8%	2.3%
	64	54.9%	36.7%	87.9%	15.0%	97.0%	6.3%	97.4%	4.2%	50.2%	30.7%	61.7%	17.7%	70.7%	18.2%	99.8%	10.9%	98.9%	1.5%	82.4%	11.7%	88.3%	9.0%	88.2%	5.9%	90.3%	5.2%
	128	30.0%	15.6%	82.1%	25.3%	90.9%	12.4%	93.8%	9.8%	55.5%	44.7%	59.1%	34.0%	62.4%	34.0%	89.1%	13.8%	96.0%	5.7%	78.7%	27.5%	82.1%	17.0%	75.1%	8.7%	80.3%	13.1%

Table 2: FeatureSqueezing and Hawkeye performance on CIFAR10 data with different attacks

Def	QS	FGSM								PGD										CW							
		$\epsilon = 1\epsilon_0$		$\epsilon = 4\epsilon_0$		$\epsilon = 8\epsilon_0$		$\epsilon = 16\epsilon_0$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 1$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 5$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 10$		$\epsilon = 4\epsilon_0$ $\alpha = 2$ $it = 10$		$\epsilon = 8\epsilon_0$ $\alpha = 4$ $it = 10$		c=0.4		c=2		c=10		c=50	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
FS-ACC	2	99.6%	0.0%	96.2%	14.6%	91.8%	23.7%	100.8%	59.8%	100.0%	0.2%	100.0%	0.0%	100.0%	0.1%	99.8%	0.4%	88.9%	7.6%	59.2%	37.4%	57.4%	35.0%	49.7%	16.2%	33.6%	9.1%
	4	97.3%	2.3%	82.4%	43.5%	85.0%	78.0%	0.0%	0.0%	97.1%	2.7%	99.5%	1.6%	100.0%	1.6%	98.5%	20.3%	66.8%	39.2%	71.2%	66.7%	25.2%	21.7%	29.3%	19.1%	11.5%	6.3%
	8	87.3%	19.0%	87.3%	85.6%	100.0%	100.0%	0.0%	0.0%	89.0%	20.1%	98.0%	7.4%	99.0%	10.7%	95.0%	22.7%	99.7%	99.7%	90.8%	91.4%	100.0%	100.0%	0.0%	0.0%	0.0%	0.2%
	16	74.6%	38.5%	98.9%	99.2%	0.0%	0.1%	100.0%	100.0%	74.2%	37.0%	87.9%	13.4%	97.0%	21.5%	84.5%	40.2%	96.8%	26.9%	72.1%	65.1%	2.8%	4.5%	0.0%	0.1%	0.0%	0.0%
	32	74.4%	57.8%	91.5%	86.7%	100.0%	100.0%	0.0%	0.1%	63.3%	41.6%	85.4%	25.9%	93.4%	30.7%	95.0%	22.7%	96.4%	13.4%	72.8%	58.6%	88.7%	86.8%	0.0%	0.0%	0.1%	0.1%
	64	84.8%	80.7%	87.9%	79.1%	98.7%	98.8%	100.0%	100.0%	69.4%	63.5%	75.0%	32.8%	87.4%	47.1%	95.4%	29.2%	95.7%	10.8%	72.5%	62.2%	86.3%	80.5%	88.9%	89.6%	100.0%	100.0%
	128	4.4%	4.9%	100.0%	98.0%	83.6%	85.4%	65.1%	58.8%	4.3%	4.3%	58.9%	33.6%	84.5%	58.8%	91.8%	39.7%	89.1%	9.8%	89.6%	89.0%	100.0%	100.0%	76.3%	76.7%	70.3%	70.0%
FS-FPR	2	99.5%	0.0%	97.2%	21.3%	93.1%	26.4%	31.6%	7.5%	100.0%	0.1%	100.0%	0.0%	100.0%	0.6%	99.8%	0.2%	68.3%	1.5%	55.2%	34.0%	72.1%	55.2%	65.2%	38.6%	38.7%	12.8%
	4	71.3%	0.2%	83.0%	44.1%	51.2%	41.8%	6.6%	12.2%	82.4%	0.3%	98.4%	1.0%	99.9%	1.6%	88.4%	5.9%	27.1%	14.9%	43.6%	39.4%	57.2%	61.5%	51.5%	51.8%	30.8%	34.8%
	8	54.1%	4.1%	51.7%	49.4%	10.0%	40.1%	2.0%	13.5%	58.2%	4.6%	79.0%	1.5%	97.1%	5.8%	32.7%	40.5%	24.9%	32.8%	45.9%	42.8%	47.8%	56.4%	36.3%	49.8%	16.4%	30.8%
	16	40.6%	14.1%	40.6%	51.9%	7.5%	35.9%	0.5%	13.0%	52.5%	17.7%	69.2%	4.0%	88.6%	8.3%	73.7%	25.1%	86.2%	7.4%	36.5%	30.7%	50.3%	54.4%	21.6%	35.2%	17.4%	31.5%
	32	44.4%	29.3%	51.1%	50.9%	18.1%	44.7%	0.8%	15.6%	41.6%	27.0%	65.6%	10.0%	83.5%	17.0%	92.7%	18.7%	86.2%	8.4%	32.5%	25.5%	60.5%	59.8%	33.9%	46.7%	20.8%	34.1%
	64	51.1%	46.8%	66.9%	58.8%	21.3%	37.1%	3.2%	19.6%	47.3%	41.6%	47.9%	14.7%	76.3%	31.2%	88.6%	20.1%	69.8%	1.0%	34.6%	30.5%	63.9%	60.7%	44.6%	47.6%	28.6%	37.5%
	128	55.8%	56.9%	59.4%	57.2%	25.9%	33.1%	11.6%	10.4%	54.1%	54.1%	39.1%	19.3%	77.7%	49.2%	88.5%	32.1%	59.6%	1.2%	37.6%	40.5%	96.0%	96.7%	41.4%	42.4%	23.0%	25.7%
HE-GB	2	100.0%	0.4%	96.7%	11.2%	94.1%	18.1%	80.0%	9.5%	100.0%	0.6%	100.0%	0.3%	100.0%	0.9%	100.0%	1.4%	90.7%	4.2%	48.4%	26.1%	50.4%	27.1%	48.9%	12.6%	37.2%	10.6%
	4	99.1%	2.5%	78.7%	28.7%	76.1%	32.8%	86.8%	26.8%	99.3%	3.3%	100.0%	1.8%	100.0%	3.5%	99.4%	7.8%	64.3%	17.0%	50.7%	33.6%	51.2%	42.7%	44.2%	35.8%	32.3%	25.7%
	8	91.2%	7.9%	64.1%	36.3%	70.9%	26.3%	91.2%	26.8%	91.6%	8.7%	99.3%	4.9%	100.0%	7.8%	43.7%	27.0%	76.2%	51.7%	48.0%	29.8%	58.3%	39.8%	59.8%	40.4%	73.9%	51.9%
	16	70.4%	17.2%	65.5%	36.2%	74.9%	24.1%	93.7%	20.5%	73.7%	16.5%	96.1%	7.8%	99.1%	13.4%	83.6%	18.0%	98.2%	9.2%	49.7%	26.2%	57.8%	39.8%	62.6%	33.5%	72.2%	45.3%
	32	54.3%	25.3%	64.2%	35.1%	71.5%	24.5%	92.5%	24.8%	57.4%	26.5%	87.7%	11.3%	97.1%	21.0%	97.4%	14.5%	98.6%	5.3%	52.8%	26.6%	61.0%	39.5%	63.0%	35.7%	72.9%	46.9%
	64	54.0%	39.6%	62.2%	38.0%	66.9%	25.0%	91.1%	28.2%	57.7%	39.6%	74.7%	15.9%	90.7%	28.6%	98.8%	15.6%	98.1%	3.6%	54.9%	32.1%	66.3%	43.1%	62.9%	37.0%	72.7%	44.1%
	128	68.0%	54.6%	59.0%	32.0%	69.6%	20.2%	91.2%	16.7%	64.4%	56.0%	60.9%	22.6%	83.1%	38.3%	97.2%	22.4%	96.6%	3.2%	66.4%	50.0%	62.4%	42.7%	62.8%	35.1%	69.5%	34.8%
HE-NN	2	99.9%	0.0%	96.1%	11.4%	94.8%	18.2%	82.1%	8.6%	99.8%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.1%	95.9%	1.9%	58.0%	33.2%	61.8%	38.2%	62.0%	24.8%	42.2%	12.6%
	4	98.5%	0.4%	83.8%	30.3%	73.4%	27.6%	80.4%	14.8%	98.2%	0.6%	99.9%	0.5%	99.9%	0.5%	98.4%	2.7%	81.5%	17.3%	58.3%	38.6%	58.3%	43.9%	58.3%	33.7%	57.7%	33.8%
	8	92.3%	5.8%	67.2%	35.2%	73.4%	26.1%	86.5%	13.4%	92.2%	5.1%	98.9%	1.4%	99.7%	2.6%	52.1%	25.8%	70.4%	38.8%	63.2%	41.7%	58.6%	38.7%	53.7%	33.8%	59.0%	29.3%
	16	77.6%	14.9%	64.0%	35.2%	70.8%	21.3%	91.6%	11.9%	80.8%	17.7%	96.4%	4.4%	99.7%	4.5%	87.4%	15.0%	99.3%	9.4%	52.5%	29.8%	58.8%	37.5%	48.6%	21.4%	61.7%	30.9%
	32	65.1%	30.5%	64.3%	33.8%	73.0%	22.3%	91.1%	13.8%	61.8%	26.3%	92.5%	10.5%	97.3%	8.6%	98.2%	10.7%	99.1%	2.8%	56.3%	25.0%	70.2%	43.5%	57.1%	28.6%	61.8%	30.8%
	64	63.6%	47.5%	74.3%	41.5%	65.6%	21.9%	89.9%	19.2%	61.4%	43.9%	79.2%	14.6%	91.8%	18.3%	99.0%	10.0%	99.3%	1.9%	55.5%	29.3%	67.5%	42.7%	55.3%	30.6%	71.0%	37.0%
	128	75.1%	57.6%	68.1%	37.3%	70.8%	16.3%	90.8%	12.0%	63.7%	54.5%	65.6%	19.9%	87.0%	32.0%	98.3%	15.3%	98.1%	1.6%	63.0%	40.6%	68.8%	41.9%	61.8%	28.1%	67.8%	26.9%

Table 3: FeatureSqueezing and Hawkeye performance on ImageNet data with different attacks

Def	QS	FGSM								PGD										CW							
		$\epsilon = 1\epsilon_0$		$\epsilon = 4\epsilon_0$		$\epsilon = 8\epsilon_0$		$\epsilon = 16\epsilon_0$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 1$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 5$		$\epsilon = 2\epsilon_0$ $\alpha = 1$ $it = 10$		$\epsilon = 4\epsilon_0$ $\alpha = 2$ $it = 10$		$\epsilon = 8\epsilon_0$ $\alpha = 4$ $it = 10$		c=0.4		c=2		c=10		c=50	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
FS-ACC	2	15.0%	12.4%	99.9%	100.0%	99.9%	100.0%	100.0%	100.0%	54.0%	39.9%	53.7%	39.1%	61.9%	28.1%	68.2%	35.3%	66.7%	43.3%	30.5%	20.8%	39.2%	29.7%	46.0%	43.2%	26.5%	24.0%
	4	9.1%	6.6%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	63.7%	25.2%	47.7%	35.8%	57.3%	24.3%	64.0%	34.7%	73.2%	55.0%	39.3%	24.0%	38.9%	28.1%	49.5%	42.6%	67.0%	68.4%
	8	99.9%	100.0%	100.0%	100.0%	99.9%	100.0%	100.0%	100.0%	56.2%	26.2%	58.4%	49.8%	64.2%	17.2%	72.2%	35.6%	81.9%	61.2%	41.1%	19.8%	50.6%	33.9%	45.4%	36.7%	47.6%	45.3%
	16	94.8%	94.5%	100.0%	100.0%	99.8%	100.0%	100.0%	100.0%	62.5%	34.5%	73.5%	65.3%	66.9%	14.0%	72.7%	27.0%	73.6%	49.5%	49.2%	32.3%	50.6%	33.9%	66.5%	56.9%	71.9%	69.9%
	32	72.1%	65.8%	100.0%	100.0%	99.9%	100.0%	100.0%	100.0%	67.8%	48.1%	67.1%	56.6%	72.2%	10.5%	73.6%	13.9%	78.2%	37.7%	52.2%	38.3%	48.7%	30.3%	61.9%	49.2%	72.0%	69.6%
	64	67.5%	55.6%	100.0%	100.0%	99.9%	100.0%	0.0%	0.0%	66.3%	54.3%	70.7%	61.9%	78.2%	7.2%	81.0%	6.4%	82.7%	17.5%	52.8%	44.4%	55.3%	42.2%	58.2%	43.5%	71.2%	61.9%
	128	67.2%	53.3%	97.1%	97.5%	100.0%	100.0%	0.0%	0.0%	82.9%	78.0%	69.7%	63.1%	78.6%	7.6%	85.6%	2.2%	87.2%	5.5%	71.7%	67.0%	53.0%	43.3%	67.3%	54.7%	68.9%	57.6%
FS-FPR	2	45.5%	46.1%	14.0%	29.3%	0.9%	8.1%	0.0%	4.0%	55.0%	40.6%	38.0%	23.3%	44.7%	15.3%	32.6%	10.3%	22.2%	13.9%	50.4%	41.4%	57.5%	50.2%	53.8%	50.8%	41.4%	40.4%
	4	37.0%	39.4%	4.5%	17.5%	0.2%	8.4%	0.0%	2.9%	51.8%	15.2%	41.3%	28.8%	36.3%	12.0%	32.6%	12.6%	14.1%	12.4%	52.2%	38.6%	46.2%	36.5%	45.6%	39.8%	37.7%	36.3%
	8	18.6%	27.4%	0.3%	7.8%	0.0%	4.3%	0.0%	0.3%	45.2%	17.5%	20.0%	18.6%	30.9%	3.6%	4.0%	1.3%	1.8%	3.1%	48.6%	29.4%	45.3%	27.2%	38.9%	31.3%	38.4%	35.0%
	16	7.6%	17.7%	0.0%	2.7%	0.0%	1.4%	0.0%	0.3%	40.1%	17.2%	9.0%	11.3%	10.5%	0.6%	0.5%	0.3%	0.0%	0.5%	39.0%	24.3%	39.3%	21.8%	32.4%	24.4%	27.6%	25.4%
	32	18.9%	19.4%	0.0%	1.6%	0.0%	0.5%	0.0%	0.2%	35.4%	19.5%	10.0%	12.5%	6.6%	0.1%	1.8%	0.1%	0.0%	0.0%	46.0%	32.2%	46.6%	28.9%	25.8%	17.3%	23.6%	20.7%
	64	29.1%	20.7%	0.1%	2.1%	0.0%	0.3%	0.0%	0.2%	33.6%	24.1%	11.6%	10.9%	19.3%	0.2%	7.3%	0.0%	0.2%	0.0%	41.6%	33.0%	31.4%	19.9%	25.9%	15.9%	22.1%	18.0%
	128	25.4%	17.8%	0.2%	3.2%	0.0%	0.8%	0.0%	0.2%	0.4%	-7.8%	16.9%	14.3%	30.8%	0.3%	19.6%	0.0%	31.1%	0.0%	39.2%	35.3%	47.8%	38.8%	38.6%	26.3%	29.2%	21.6%
HE-GB	2	33.5%	27.7%	76.9%	44.3%	86.9%	31.6%	91.8%	18.1%	30.6%	20.1%	43.9%	24.2%	52.4%	19.3%	61.1%	22.6%	72.6%	31.3%	35.3%	27.8%	36.7%	30.3%	31.4%	26.9%	32.2%	26.1%
	4	36.5%	29.3%	81.7%	47.6%	91.1%	33.9%	94.6%	18.8%	45.1%	14.1%	45.7%	27.0%	53.5%	19.9%	58.5%	23.0%	82.4%	31.1%	37.6%	25.0%	36.6%	26.9%	32.5%	23.1%	29.6%	26.4%
	8	83.8%	71.1%	89.6%	49.8%	94.6%	38.0%	98.7%	17.8%	41.3%	15.0%	95.6%	79.5%	55.9%	11.0%	68.5%	19.4%	99.6%	75.0%	37.8%	18.3%	40.1%	22.1%	54.4%	40.5%	57.9%	47.9%
	16	84.1%	70.2%	92.9%	42.7%	97.8%	31.4%	98.6%	17.3%	40.2%	16.9%	93.0%	77.1%	58.8%	8.0%	67.4%	15.1%	99.3%	73.5%	36.0%	20.7%	43.2%	24.7%	93.3%	87.7%	89.5%	83.8%
	32	84.3%	74.0%	91.2%	34.6%	96.5%	22.6%	97.8%	12.2%	40.2%	22.4%	96.5%	70.9%	67.1%	9.1%	73.4%	11.0%	80.3%	19.6%	40.6%	26.9%	46.4%	26.8%	56.8%	36.8%	87.4%	77.3%
	64	71.4%	39.0%	85.6%	35.7%	95.6%	19.1%	96.5%	9.1%	42.8%	30.2%	91.1%	47.0%	78.5%	8.5%	86.7%	8.3%	85.4%	12.9%	45.0%	34.7%	48.9%	31.0%	53.0%	31.8%	58.3%	34.4%
	128	69.9%	38.1%	83.6%	30.5%	92.0%	15.5%	95.8%	4.1%	51.9%	43.6%	74.5%	25.6%	88.1%	9.3%	96.6%	5.9%	98.2%	7.6%	53.7%	47.0%	57.3%	42.8%	57.9%	38.9%	61.2%	36.6%
HE-NN	2	66.9%	48.9%	77.2%	31.3%	80.7%	13.1%	92.4%	7.1%	61.0%	47.3%	64.6%	29.3%	80.8%	20.5%	83.2%	14.8%	89.8%	15.6%	54.6%	44.9%	61.3%	53.1%	62.3%	54.5%	56.8%	45.2%
	4	57.3%	44.6%	77.2%	24.3%	90.8%	16.0%	94.2%	6.1%	67.9%	26.4%	68.2%	35.8%	85.6%	18.4%	87.7%	18.9%	89.8%	17.4%	60.7%	44.5%	61.4%	41.5%	62.6%	48.2%	57.1%	43.0%
	8	76.6%	37.2%	88.0%	17.3%	93.2%	13.5%	96.6%	4.7%	67.5%	29.8%	78.9%	29.9%	92.3%	11.7%	94.4%	7.1%	95.4%	9.7%	60.2%	39.0%	62.1%	37.6%	63.8%	41.0%	64.7%	45.4%
	16	77.3%	28.4%	91.8%	10.5%	96.6%	8.0%	97.8%	2.4%	76.6%	29.8%	79.3%	22.5%	97.5%	4.6%	97.0%	2.5%	97.1%	3.6%	53.2%	35.4%	56.8%	33.2%	63.8%	35.7%	64.1%	38.4%
	32	75.7%	30.8%	93.6%	7.0%	98.0%	5.2%	97.9%	2.0%	67.2%	32.9%	86.0%	20.5%	97.9%	2.0%	98.9%	1.0%	98.7%	1.4%	58.8%	42.2%	64.4%	39.2%	60.4%	28.5%	60.5%	33.0%
	64	74.6%	30.6%	95.2%	6.8%	97.5%	3.8%	99.1%	1.4%	62.4%	37.1%	84.3%	20.2%	97.7%	1.6%	99.1%	0.4%	99.3%	0.1%	59.0%	44.4%	55.5%	30.8%	58.6%	26.6%	58.1%	30.0%
	128	72.3%	27.8%	93.0%	8.0%	97.8%	4.1%	99.4%	1.3%	13.6%	1.8%	80.5%	25.3%	98.0%	1.7%	98.9%	0.2%	99.2%	0.4%	55.5%	46.6%	63.8%	49.0%	64.6%	38.4%	62.3%	34.6%

Table 4: White-Box attack to FeatureSqueezing

White-Box attack		**c=2, c2=0.02, s=2**		**c=2, c2=0.1, s=2**		**c=2, c2=0.02, s=2,4**		**c=2, c2=0.1, s=2,4**	
ASR		49%		57%		44%		52%	
L_2		0.225		0.256		0.244		0.279	
L_∞		0.038		0.038		0.042		0.050	
Defense	QS	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
	2	2%	0%	3%	1%	0%	0%	0%	1%
	4	84%	2%	84%	1%	0%	2%	1%	1%
	8	87%	0%	85%	3%	76%	0%	73%	3%
	16	79%	2%	78%	0%	78%	2%	78%	0%
	32	70%	6%	67%	6%	70%	2%	67%	3%
	64	56%	16%	62%	17%	57%	13%	62%	17%
	128	56%	53%	56%	39%	55%	53%	56%	42%