Interpretable Adversarial Example Detection via High-Level Concept Activation Vector

Jiaxing Li[a], Yu-an Tan[a], Xinyu Liu[a], Weizhi Meng[b,*] and Yuanzhang Li[c,*]

[a] *School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, 100081, China*
[b] *Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Kongens Lyngby, Denmark*
[c] *School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, China*

ARTICLE INFO

Keywords:
Deep learning
Adversarial machine learning
Model explainability
Adversarial defense
Concept Activation Vector

ABSTRACT

Deep neural networks have achieved amazing performance in many tasks. However, they are easily fooled by small perturbations added to the input. Such small perturbations to image data are usually imperceptible to humans. The uninterpretable nature of deep learning systems is considered to be one of the reasons why they are vulnerable to adversarial attacks. For enhanced trust and confidence, it is crucial for artificial intelligence systems to ensure transparency, reliability, and human comprehensibility in their decision-making processes as they gain wider acceptance among the general public. In this paper, we propose an approach for defending against adversarial attacks based on conceptually interpretable techniques. Our approach to model interpretation is on high-level concepts rather than low-level pixel features. Our key finding is that adding small perturbations leads to large changes in the model concept vector tests. Based on this, we design a single image concept vector testing method for detecting adversarial examples. Our experiments on the Imagenet dataset show that our method can achieve an average accuracy of over 95%. We provide source code in the supplementary material.

1. Introduction

Deep neural networks have been highly successful in a variety of fields, demonstrating remarkable achievements. They have achieved breakthrough success in fields as diverse as image classification, object detection, etc. Their broad application and remarkable capabilities not only accelerate progress, but also pave the way for exciting breakthroughs in fields such as finance, healthcare, criminal justice, and transportation. However, deep learning is vulnerable to adversarial attacks (Goodfellow et al., 2015). While deep learning has achieved many remarkable results, it also faces an important challenge of being vulnerable to adversarial attacks. Adversarial attacks refer to the loopholes of the deep learning model. By deliberately designing small disturbances that are difficult for humans to detect, the model can produce wrong prediction results. These perturbations can be additions, modifications, or deletions to the input data, which may not matter to us, but can greatly mislead the output of the model. The successful implementation of such attacks (Goodfellow et al., 2015; Jang et al., 2017; Moosavi-Dezfooli et al., 2016; Rony et al., 2019; Xu et al., 2020; Wang et al., 2021, 2020; Gao et al., 2019) not only poses a threat to the security of the model, but may also lead to serious consequences in the real world, such as misleading the automatic driving system, tampering with medical image diagnosis, and deceiving speech recognition systems. Many researchers believe that deep learning systems do not yet have cognitive processing methods similar to human concepts, which is one of the reasons why deep learning systems are susceptible to adversarial attacks.

A body of existing work focuses on developing methods to detect adversarial examples (Liang et al., 2021; Yang et al., 2020, 2022; Cintas et al., 2021; Liu et al., 2018; Madry et al., 2018a; Sutanto and Lee, 2021; Guo et al., 2019; Zheng et al., 2023). The generation of adversarial examples is a very complex optimization problem, so it is particularly important to find efficient detection methods. The detection method needs to reduce the probability of misjudgment of adversarial examples as much as possible while maintaining high accuracy, and avoid falsely labeling normal samples as adversarial examples. To address these challenges, researchers have proposed various techniques for detecting adversarial examples. Some methods perform detection based on the characteristics of adversarial examples, trying to find the distinction between adversarial examples and clean examples, such as using statistical properties, gradient information, or spectral analysis, etc.

Besides, interest in developing tools (Chattopadhay et al., 2018; Ghorbani et al., 2019a; Ignatiev et al., 2019; Selvaraju et al., 2017; Chattopadhay et al., 2018; Fidel et al., 2020) that address the black-box nature of neural networks is growing. The Testing with Concept Activation Vectors (TCAV) (Kim et al., 2018; Ghorbani et al., 2019b) method is one in which model interpretation will be carried out in a high-level concept space. In the TCAV method, directional derivatives are used to quantify how important user-defined concepts are to classification results. Subsequent work (Ghorbani et al., 2019b) has also expanded the method of

*Corresponding author
jiaxingxx@outlook.com (J. Li); tan2008@bit.edu.cn (Y. Tan); lxy1653300572@163.com (X. Liu); weme@dtu.dk (W. Meng); popular@bit.edu.cn (Y. Li)
ORCID(s): 0000-0001-7048-9284 (J. Li); 0000-0001-6404-8853 (Y. Tan); 0009-0002-7982-7432 (X. Liu); 0000-0003-4384-5786 (W. Meng); 0000-0002-1931-366X (Y. Li)

automatically acquiring concepts. Can developing cognitive processing similar to human concepts help deep learning systems defend against adversarial attacks? No research has yet provided an answer.

We explore the use of concept activation vectors to detect adversarial examples. We find that concept activation vectors test differently for adversarial examples than for normal examples. We design simple but effective methods to detect adversarial examples. The foundation of our work is the idea that small perturbations to the input can lead to considerable differences in testing concept activation vectors. Our contributions are summarized as follows:

- Introduction of a novel single-sample CAV testing method for detecting adversarial examples, a technique that is more computationally efficient than traditional batch testing methods.
- Application of CAVs to pre-trained deep learning models (GoogLeNet, ResNet34, Inception-v3), demonstrating their capability to detect adversarial perturbations across different architectures.
- Comparative evaluation showing that our method achieves competitive performance against state-of-the-art adversarial detection techniques.

2. Related Works

In this section, we discuss related works on adversarial attacks, adversarial defenses, and adversarial example detection. concept activation vector testing.

2.1. Adversarial examples

Deep learning systems, including convolutional neural networks, such as Googlenet and Resnet, are vulnerable to adversarial attacks. The endless emergence of adversarial attack methods severely limits the application of neural networks. Adversarial attacks attempt to alter the output predictions of deep learning models. Usually adversarial attacks add tiny malicious perturbations to the input. The original normal samples are added with tiny malicious disturbances to make the deep learning model output wrong predictions. Such samples with malicious perturbations are adversarial examples. According to whether the wrong prediction output by the deep learning model after the adversarial attack is specified, the adversarial attack can be classified as targeted or untargeted. Targeted adversarial attack means that the attacker's goal is to make the wrong prediction of the deep learning model for the adversarial examples fall into the specified category. The untargeted adversarial attack is to make the deep learning model misclassify, and the wrong prediction can fall into any category except the correct category.

According to the measurement method for small perturbations, adversarial attacks can also be divided into several categories. For example, L_0 and L_1, L_p distance is the most commonly used distance metric. The Fast Gradient Sign Method (FGSM) proposed by (Goodfellow et al., 2015) is one of the classic adversarial attack algorithms. The Projected Gradient Descent (PGD) (Madry et al., 2018b) algorithm, which was proposed subsequently, is an improved attack algorithm based on the FGSM algorithm. The adverantages of the PGD algorithm lie in less perturbation and higher attack success rate. The DeepFool (Moosavi-Dezfooli et al., 2016) algorithm focuses on minimizing the L_2 distance to generate adversarial examples. PGD and DeepFool both involve an iterative process. (Carlini and Wagner, 2017) proposed a more refined attack method that can control the confidence level. Moreover, this attack method can derive variants of various attack distances. DDN (Rony et al., 2019) employs Decoupled Direction and Norm L_2 to create adversarial examples. Based on gradient, DDNattack generates adversarial examples with minimal L_2 norm misclassifications. The work involves differentiating the image's adversarial disturbance's direction and norm. (Brendel et al., 2018) introduce a black-box approach that relies solely on model decisions without the need of gradient information. Experiments show that our method can effectively identify adversarial samples regardless of distance or whether gradient information is used.

2.2. Adversarial defense and detection

Deep learning models are often unstable in the face of adversarial attacks and may be misled by malicious attackers through carefully constructed adversarial examples. To address this issue, researchers have proposed many defenses against adversarial attacks, including several major ones. Adversarial Training (Madry et al., 2018b) is one of the most commonly used methods for defending against adversarial attacks. The basic idea is to inject some adversarial examples into the training data during the process of training the deep learning model, so that the model can identify and defend against these adversarial examples during training, thereby improving the robustness of the model. Randomization Defense (Sheikholeslami et al., 2020) is a method of adding random noise or random perturbation to the input data to increase the robustness of the model to adversarial examples.

In contrast, another line of research (Yang et al., 2020, 2022; Cintas et al., 2021; Liu et al., 2018; Madry et al., 2018a; Sutanto and Lee, 2021; Liang et al., 2021) expects to detect adversarial examples directly at the model inference stage. Some research on adversarial detection has applied data transformations such as PCA to extract features from the input and layers of neural networks (Li and Li, 2017; Bhagoji et al., 2018). Many works use separate detection neural networks (Cintas et al., 2021; Ko and Lim, 2021) to detect adversarial examples. Another approach is to modify existing networks to detect adversarial examples. Some of these methods lead to a loss of accuracy for the model to classify normal samples. There are also works (Yang et al., 2020) that use the output of the middle layer as feature attribution to detect adversarial samples. Defense methods for other feature spaces are vulnerable to deliberate manipulation attacks against adversarial attacks. For example, adversarial examples that conform to statistical normal characteristics. Similar to our work, there are works (Ko and Lim, 2021) using model interpretation methods to detect adversarial examples. They propose unsupervised detection of adversarial examples using a reconstruction network trained exclusively on model interpretations of clean examples. However, their approach depends on additional neural networks that are susceptible when an attacker scrambles the images to fool both the original model and the new neural network. As we will show in experiments, our method achieves promising performance without requiring additional neural networks.

2.3. Testing with concept activation vector

Recently, a series of studies have been devoted to providing explanations from the perspective of human "concepts". The output of the method reveals important concepts, rather than assigning importance to individual features or pixels (Chattopadhay et al., 2018; Muhammad and Yeasin, 2020; Selvaraju et al., 2017). For instance, when detecting police cars, wheels and police signs are important concepts. Testing with Concept Activation Vectors (TCAV) (Kim et al., 2018) determines the importance of a given concept for predicting that class. The original TCAV method needs to manually define the concept set. The first step in the TCAV approach is to define concepts. Just choose a collection of examples that illustrate the concept or locate a distinct dataset labeled with that concept.

The ACE (Ghorbani et al., 2019b) method extends TCAV to automatically generate concept sets. ACE is a explanation method that elucidates the entire class in a trained classifier without needing human supervision. The TCAV score aims to estimate the average positive influence that concepts have on the predicted class, commonly used for deep neural network classifiers. Given an example of a concept, the TCAV score is the score of class images. If the representations of these images in the activation space are perturbed in the general direction of the concept example representation in the same activation space, the prediction score increases (using directional derivatives). Details are described in the original article. The goal of this series of work is to give a model interpretation of the concept space of the dataset, which cannot be directly used to detect whether a single picture is an adversarial example. Our adversarial example detection method is a method for testing single image samples based on Concept Activation Vector Testing.

3. Methods

In this part, we demonstrate our method, interpretable adversarial example detection based on Concept Activation Vector Testing. First, we introduce the threat model. Then we detail the method we proposed. Our method can be used for TCAV detection on a single sample. Finally, we use simple and effective linear models for detecting mixed adversarial examples. Figure 1 provides an overview of our approach.

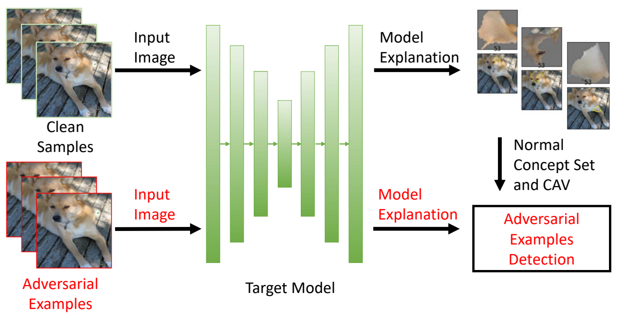

Figure 1: A summary of the detection method we proposed.

3.1. Adversarial model

We refer to the model that has been trained for inference as the target model. We take into account a neural network model that includes inputs $\boldsymbol{x} \in \mathbb{R}^n$ and an intermediate feed-forward layer with m neurons, like input inference and its intermediate layer activations can be viewed as a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. The neural network from the middle layer to the final output layer is regarded as a function $h : \mathbb{R}^m \rightarrow \mathbb{R}$. The target model $h(f(\boldsymbol{x}))$ is a neural network with input $\boldsymbol{x}$ and output k classes.

The attacker will use the adversarial examples $\boldsymbol{x'}$ to attack the victim model. In our assumption, the attacker will use various state-of-the-art attack methods to generate adversarial examples. Adversarial examples make the target model $h(f(\boldsymbol{x'}))$ wrongly output class k'.

In formal terms, for given parameters ϵ and θ of the model, the most classic method for generating adversarial samples is as follows:

$$x' = x + \epsilon \cdot \operatorname{sign}\left[\nabla_x h(f(\theta; x, y))\right]. \quad (1)$$

For example, FGSM (Goodfellow et al., 2015), PGD (Madry et al., 2018b), Deepfool (Moosavi-Dezfooli et al., 2016), CW (Carlini and Wagner, 2017), DDN (Rony et al., 2019), Newtonfool (Jang et al., 2017) and BoundaryAttack (Brendel et al., 2018), Salt&pepper (Rauber et al., 2017) methods. These attackers have access to the model parameters as well as training and testing datasets, and the attackers have all the permissions on the parameters required for these attacks. We want to build a detection model that can detect whether the input to a given model is an adversarial example maliciously designed to misclassify the target model. We will introduce methods for adversarial example detection based on concept activation vectors.

3.2. Automated concept-based explanations of normal samples

Our method is based on testing the concept activation vectors of adversarial examples in a different manner than normal examples. The black-box nature of deep neural networks has been criticized for a long time. Researchers

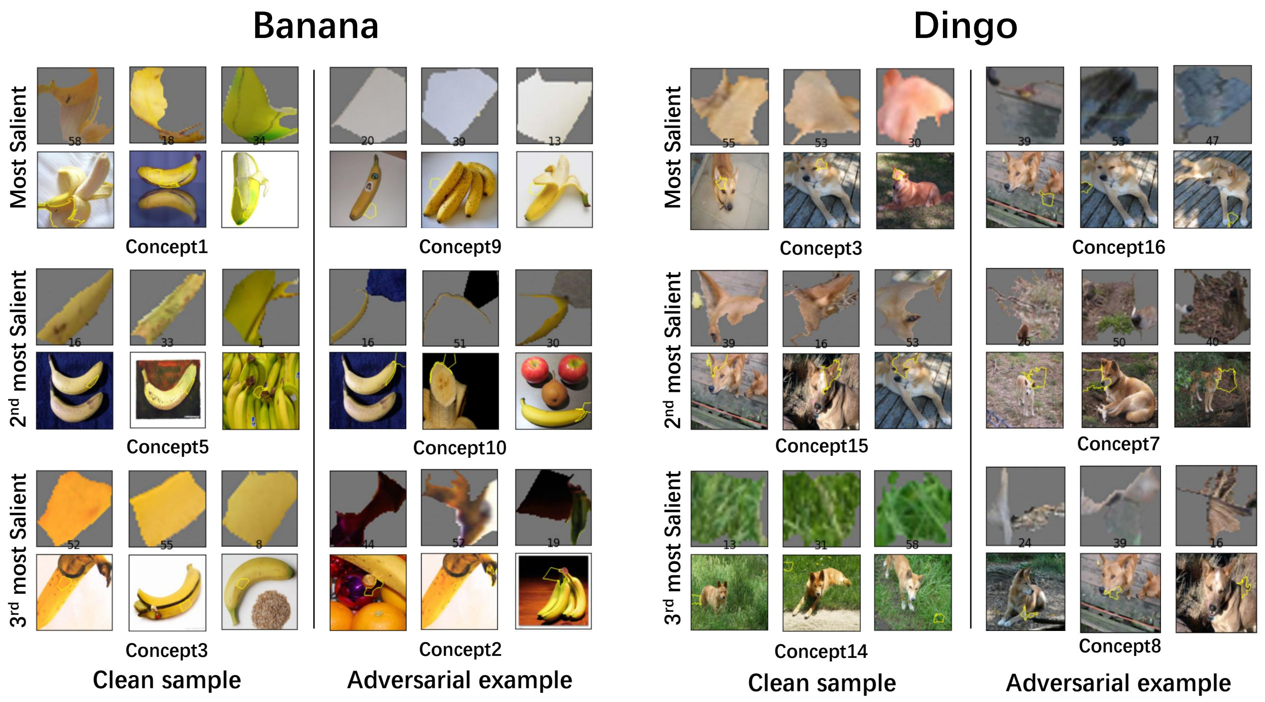

Figure 2: An example of the top 3 concept sets for two Imagenet classes and their adversarial examples. Here, we present three randomly chosen examples of each class and its adversarial examples of top-3 important concepts (each concept patch is displayed above its original image separated by a yellow line). For example, for this result, we can see that the three most important concepts for Banana are concepts 1, 5, and 3, whereas the most important concepts for Banana adversarial examples are concepts 9, 10, and 2.

have proposed a model interpretation method for outputting heat maps like Grad-CAM (Selvaraju et al., 2017). But recently there are also many studies (Adebayo et al., 2018; Zhou et al., 2022) questioning this method of model interpretation. We focus on the method for model interpretation of Concept Activation Vectors (CAV) proposed by (Kim et al., 2018). Subsequent work (Ghorbani et al., 2019b; Yeh et al., 2020) has extended the method for automatic concept generation (ACE). CAV provide a human-friendly conceptual interpretation of intermediate states of neural networks.

The key point of this approach to model interpretation is to use the high-dimensional intermediate state of the neural network as a support. The testing with concept activation vector method provides a quantification of how important a concept is to the classification result. We adopted this method of directional derivatives to compute concept activation vectors testing. A concept activation vector is characterized as the direction of activation values for that concept set. We train a linear classifier between a set of concepts and random counter-examples, then the value orthogonal to the decision boundary is the concept activation vector. We divide the picture into different patches through the image segmentation algorithm. These patches make up the set of concepts we need. For each concept, a corresponding concept activation vector can be trained. First for normal samples, we employ an automatic method to classify images of each class starting from a given segmentation. Similar segments are then grouped as instances of the same concept. To assess segment similarity, this approach uses a spatially efficient perceptual similarity measure for the activations of the final layer within a convolutional neural network (CNN). The final step is to return important concepts from the extracted set.

By employing the method of linear interpretability, with a set of examples that represents a concept of human interest, we find a vector representing that concept in the activation space of intermediate layers of the neural network. We consider activations in intermediate layers resulting from input examples versus random examples from the concept set. Then, we define the "concept activation vector" as the normal to the hyperplane that distinguishes examples according to the presence or absence of the concepts. For example, when we collect a set of positive input samples (e.g., the set of concepts produced by the ACE method above) and a set of negative sample inputs (e.g., a selection of random pictures). Then, we can train a binary linear classifier to differentiate the two sets of layer activations. This classifier $\boldsymbol{v}_C$ is the linear CAV of concept C. Using CAV and directional derivatives, we measure the responsiveness of model output to input changes at neural activation layers to elicit the direction of concepts.

If $\boldsymbol{v}_C$ is the unit CAV vector of concept C in the intermediate layer, and $f(\boldsymbol{x})$ is the neural network intermediate layer with input $\boldsymbol{x}$, the "concept sensitivity" of the k classes output by the model to concept $\boldsymbol{C}$ could be calculated as the directional derivative $S_{C,k}(\boldsymbol{x})$.

$$\begin{aligned} S_{C,k}(\boldsymbol{x}) &= \lim_{\epsilon \to 0} \frac{h_k(f(\boldsymbol{x}) + \epsilon \boldsymbol{v}_C) - h_k(f(\boldsymbol{x}))}{\epsilon} \\ &= \nabla h_k(f(\boldsymbol{x})) \cdot \boldsymbol{v}_C, \end{aligned} \tag{2}$$

Where $h_k : \mathbb{R}^m \to \mathbb{R}$. The $S_{C,k}(\boldsymbol{x})$ is a quantitative measure of the responsiveness of model outputs to concepts at the intermediate layer of the model. Note that it is not a metric evaluated per feature (e.g., in contrast to per-pixel saliency maps), but a scalar computed per concept over the entire input or set of inputs.

3.3. Concept activation vector testing on a single adversarial example

We wanted to explore whether TCAV for adversarial examples differ from normal examples. But the original TCAV is applicable to a set of samples. It cannot be directly used for the detection of a single suspicious sample. For adversarial examples we design a adversarial concept activation vector testing method for detecting single adversarial examples.

Our method is different from the original TCAV in that it can detect a single unknown sample image. Our approach is based on a model interpretation of the model at the concept level for normal samples. Therefore, we use the generated concept set and CAV during the normal sample classification process as known conditions.

Adversarial attacks attempt to disrupt the prediction of the model with minimal sample perturbation, then people can't distinguish between an original image $\boldsymbol{x}$ and its adversarial example $\boldsymbol{x'}$ based on the difference between them. However, we observe that TCAV is affected by small differences between $\boldsymbol{x}$ and $\boldsymbol{x'}$.

Figure 2 illustrates the important conceptual ordering of the original image $\boldsymbol{x}$ and its adversarial example version $\boldsymbol{x'}$ subjected to FGSM attack. As shown in the figure we show three randomly selected examples of the top 3 important concepts of two classes and their adversarial examples. For example, for this result, we can see that the three most important concepts of Banana are the banana itself, whereas the most important concept for the adversarial examples of Banana is the background.

This classifier $\boldsymbol{v}_C$ is a linear version of the normal sample concept $\boldsymbol{C}$. Through CAV and directional derivatives, we assess the sensitivity of the model's prediction k' class to input changes at neural intermediate activation layers to elicit the numerical gap between adversarial examples and normal examples.

$$S_{C,k'}(\boldsymbol{x'}) = \nabla h_{k'}(f(\boldsymbol{x'})) \cdot \boldsymbol{v}_C, \tag{3}$$

Note that $\boldsymbol{v}_C$ is the unit CAV vector of concept C in the model's intermediate layer computed on normal samples, $f(\boldsymbol{x})$ is the model's intermediate layer neural network for input $\boldsymbol{x}$, and k' classes for the model output. The concept activation vector test is a batch of pictures to give the results, and we propose a new concept activation vector algorithm for each picture. After calculating the CAV of normal samples, TCAV of unknown samples can be calculated individually. This TCAV method for adversarial examples uses misjudged new output predictions to differentiate.

Let k' be the misjudgment label of the output of the adversarial example $\boldsymbol{x'}$ such that a specified supervised learning task, and let $X'_{k'}$ represent all adversarial examples with the given k' label. The is defined as directional derivative set generated by Adv-TCAV as

$$advTCAV_{C,k'} = \left\{ S_{C,k'}(\boldsymbol{x'}) : \boldsymbol{x'} \in X'_{k'} \right\}. \tag{4}$$

The proportion of the activation vector of the k' class input that is positively affected by the concept $\boldsymbol{C}$. Note that the Adv-TCAV score uses a alternative measure considering the magnitude of concept sensitivity.

To reconcile scale differences between different sets of concepts, we use a binary linear classifier to distinguish between adversarial examples and original images using concept vector testing with different neurons on the training set. The algorithm we propose is outlined in Algorithm 1

Algorithm 1 Algorithm of our method

```
Input:
    $Adv$ : Various attack methods to be detected
    $h(f(x))$: Target model with input $x$
    $n_c$: Number of clean sample class
Output:
    label of whether the sample is an adversarial example
 1: Initialization:
 2: Data processing generates the sample $x'$ to be tested by
    various attack methods $Adv$
 3: for $n = 1$ to $n_c$ do
 4:     Segment clean image $x$ forms the linear representa-
    tion $\boldsymbol{v}_C$ of concept $C$
 5:     Calculate directional derivative:
 6:     $S_{C,k}(\boldsymbol{x}) = \nabla h_k(f(\boldsymbol{x})) \cdot \boldsymbol{v}_C$
 7:     Calculate directional derivative of image $x'$:
 8:     $S_{C,k'}(\boldsymbol{x'}) = \nabla h_{k'}(f(\boldsymbol{x'})) \cdot \boldsymbol{v}_C$
 9: end for
10: Data processing forms directional derivative set:
    $advTCAV_{C,k'}$
11: Train a binary linear classifier on directional derivative
    set $advTCAV_{C,k'}$
12: Return: Trained classifier outputs whether a sample is
    an adversarial example
```

4. Experiments

This section focuses on evaluating the effectiveness of the detection method we proposed. We experimentally

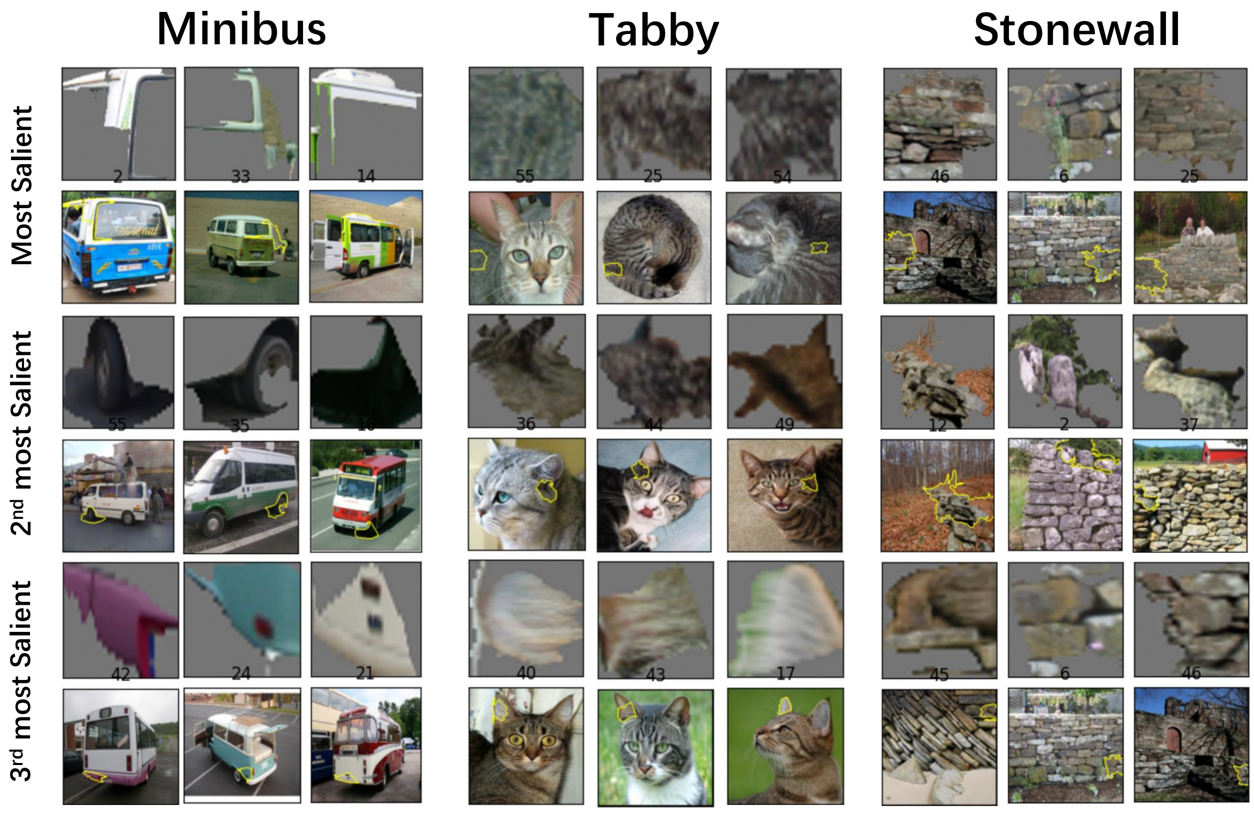

Figure 3: Examples of the top 3 concept sets produced by the 3 Imagenet classes. We illustrate three randomly picked instances of the top 3 significant concepts for each class, each positioned above its corresponding original image, divided by a yellow line. For example, using this result, we can see that the network uses the Minibus car frame and tires to classify.

evaluate Adv-TCAV on the Imagenet dataset. We first introduce several state-of-the-art adversarial example attack methods that we use. We then evaluate three common neural network models and show the resulting set of concepts. We further exemplify the worst and best categories on which our method performs on the target model.

4.1. Target model and threat model settings

The evaluation of our method is conducted using the ILSVRC2012 dataset. (ImageNet 1K) (Russakovsky et al., 2015). With the ImageNet dataset, we initially have the target classifier, which is vulnerable to adversarial attacks. In our evaluation, we use the Googlenet (Szegedy et al., 2015), Resnet34 (He et al., 2016), Inception-v3 (Szegedy et al., 2016) model weights pre-trained on the ImageNet 1K dataset officially given by Pytorch (Paszke et al., 2019). The Googlenet model's Top-1 error on ImageNet 1K is 30.22% and the Top-5 error is 10.47%. The Inception-v3 model's Top-1 error on ImageNet 1K is 22.55%, and the Top-5 error is 6.44%. The Top-1 error on 1K is 26.70%, and the Top-5 error is 8.58%. These three models are chosen as target classifiers because they are representative and widely used deep neural network models.

Given a target classifier and a clean sample dataset as input, model interpretations are collected to generate concept sets and concept activation vectors (CAV). In our evaluation, we use model interpretation methods to automatically generate concept sets and corresponding concept activation vectors (CAV). For each class label, the sensitivity degree Adv-TCAV value to the concept activation vector (CAV) of each image collected training data, paired with the appropriate labels, is employed to train a binary linear classifier to distinguish adversarial samples from original images. A summary of our method architecture is shown in Figure 1.

Eight adversarial attack methods are utilized (including white-box attacks like FGSM, PGD, Deepfool, CW, DDN, Newtonfool and black-box attacks BoundaryAttack, Salt&Pepper) to attack three models and 100 classes in the randomly selected ImageNet2012 dataset. All attack methods are implemented using the foolbox open-source framework. (Rauber et al., 2017).

The following attack methods are considered, all of which are based on open-source framework Foolbox.

- FGSM, Fast Gradient Sign Method (FGSM) proposed by Goodfellow et al. is regarded as one of the classic algorithms in adversarial attacks. We use the L_1 form of FGSM. Additionally, based on the foolbox framework, other parameter is *epsilon=0.8*.

- PGD, Projected Gradient Descent (PGD), which was proposed subsequently, is an improved attack algorithm based on FGSM. The adverantages of PGD lie in less perturbation and higher attack success rate. We use the L_2 form of PGD. Additionally, based on the foolbox framework, other parameters are *epsilon=1.0, rel_stepsize=0.025, abs_stepsize=None, steps=50, random_start=True*.

- The DeepFool algorithm focuses on minimizing the L_2 distance to generate adversarial examples. PGD and DeepFool both involve an iterative process.We use the L_2 form of DeepFool. Based on the foolbox framework, other parameters are *epsilon=0.3, steps=50, candidates=10, overshoot=0.02, loss='logits'*.

- CW, Carlini and Wagner proposed a more refined attack method that can control the confidence level. Moreover, this attack method can derive variants of various attack distances. We use the CW in the form of L_2. Based on the foolbox framework, other parameters are *epsilon=0.6, binary_search_steps=9, steps=200, stepsize=0.01, confidence=0, abort early = True, initial_const=0.001*.

- Newtonfool, a simple gradient-descent-based algorithm for finding adversarial samples, which performs well in comparison to existing algorithms. Based on the foolbox framework, other parameters are *epsilon=0.8, steps=100, stepsize=0.01*.

- DDN, DDNattack employs Decoupled Direction and Norm L_2 to create adversarial examples. Based on gradient, DDNattack generates adversarial examples with minimal L_2 norm misclassifications. The work involves differentiating the image's adversarial disturbance's direction and norm. Based on the foolbox framework, other parameters are *epsilon=1200,init_epsilon =1.0, steps=100, gamma = 0.05*.

- Boundary attack, Brendel et al. proposed a black-box method known as boundary attack, which relies solely on model decisions without the need of gradient information. Based on the foolbox framework, other parameters are *epsilon=1200, init_attack=None, steps=1000, spherical_step=0.01, source_step=0.01 , step_adaptation = 1.5, tensorboard = False, update_stats_every_k = 10, source_step_convergance =* $1e^{-07}$.

- Salt&Pepper, a non-iterative attack that does not rely on gradients or involve any optimization process, and their essence is the random insertion of white (salt) and black (pepper) noise pixels in the image that can be directly applied to the image. Based on the foolbox framework, other parameters are *epsilon=0.8, steps=100, across_channels=True, channel_axis=None*.

In order to evaluate the performance of the method we proposed, we made adversarial examples using images from a subset of 100 classes randomly selected from the 1000 classes of the Imagenet dataset, and filtered out samples that were misclassified by the model as well as failed attempts (i.e., introducing a perturbation does not alter the original class labels). Regarding (effective) adversarial examples, Adv-TCAV values are obtained and combined with Adv-TCAV values from the Imagenet test dataset (which is benign) to create the evaluation dataset for our detection method.

4.2. Evaluation on neural network models

We employ ACE to explain GoogLeNet, ResNet34, and Inception-v3 models which have been pre-trained on the ILSVRC 2012 dataset (ImageNet). To employ ACE, we select a subset consisting of 100 classes from the 1000 classes in the dataset. Consistent with the results in the TCAV paper, this importance score shows excellent performance with small sample sizes (10 to 20) per concept. According to our experiments on the ImageNet category, 60 images were enough to extract a sufficient number of concept examples. Probably because these concepts appear frequently in these images. The segmentation step is performed using SLIC (Achanta et al., 2012), which uses 15, 50, and 80 superpixels per image for segmentation due to its speed and performance. In our measurement of similarity, we checked the Euclidean distance in the network layers in the ImageNet training of the model, and for the three models Googlenet, Resnet34, and Inception-v3 respectively selected the "inception4c" layer, "layer3" layer, and "Mixed_6e" layer. As shown in previous TCAV papers, earlier layers are better for texture and color similarity, while later layers are better for object similarity, and these layers have a better balance between the two. We performed K-Means clustering and removed outliers using Euclidean distance to the cluster centers, and finally generated concept examples for each class.

As shown in Figure 3, we show an example of the top 3 concepts generated by the three classes of Minibus, Tabby, and Stonewall after explaining Googlenet, where the three concepts of each class are classified according to the sensitivity of the model classification results to concepts from above arranged to the bottom, the model classification results are most sensitive to the top concepts. For each class, we present the three most significant concepts using three randomly selected examples (each example is displayed above the original image, separated by a yellow line). The figure shows that the approach takes into account concepts of multiple levels of complexity. From the frame and wheels of the Minibus to the skin textures of Tabby and the masonry textures of Stonewall.

Each attack method corresponding to each class of each model generates 100 adversarial samples (only samples correctly identified by the model are used to generate adversarial samples), and each class has a total of 800 adversarial samples. Then, for each class, we select 800 clean samples

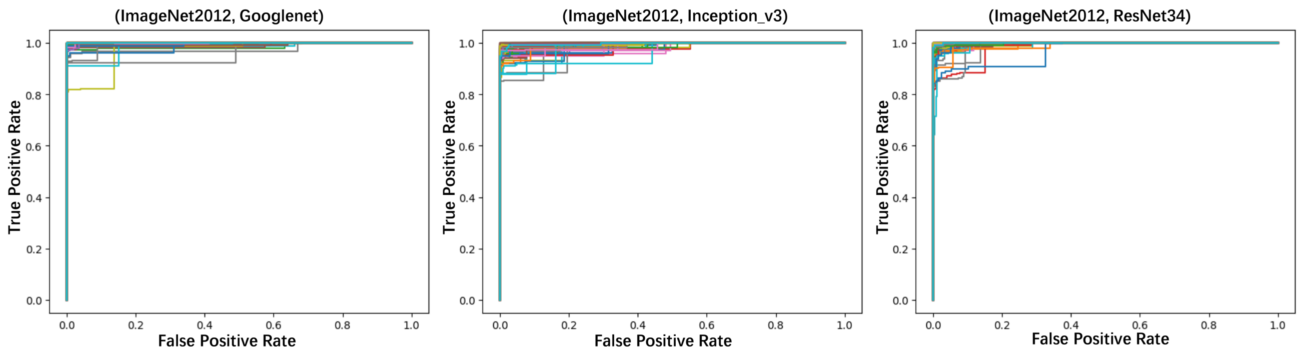

Figure 4: The picture shows the ROC curve of the detection method under eight kinds of adversarial attacks, where the lines of different colors represent 100 different classes; the model attacked in the first picture is Googlenet, the second picture is Inception-v3, and the third picture is ResNet34.

(selected in ImageNet) and 800 corresponding adversarial samples in the adversarial sample dataset, and we use Adv-TCAV to extract the feature values of each sample about all the concepts of the current class as all the features of this sample, the feature dimension of each sample is $N * 20$, where N is the number of concepts previously generated for each class. Finally, a dataset of 1600 samples is generated for each class of each model.

We use the simple linear model SGDClassifier of the Sklearn framework to train and test on the feature dataset. The classifier fits up to 1000 times.True positive rate (TPR) is defined as the proportion of adversarial images classified as adversarial, and false positive rate (FPR) is defined as the proportion of natural images classified as adversarial. The ROC (Receiver Operating Characteristic) curve illustrates the relationship between true positive rate (TPR) and false positive rate (FPR). The ROC curves of ImageNet and Googlenet, Inception-v3, and ResNet34 are shown in Figure 4, each of which contains 100 classes of ROC curves. The results indicate that our method demonstrates excellent performance on the ImageNet dataset. And as the complexity of the model increases, our ROC curve still maintains a good effect. Here are the definitions of the evaluation metrics we used. In the experiment, we use the Acc, Recall and AUC as evaluation metrics .

Acc: Describe the accuracy of the classification model, ACC is obtained by dividing the number of correctly classified samples by the number of total samples. It shows how well the model's predictions correspond to the real labels in the dataset. Acc=(TP+TN)/(TP+TN+FP+FN).

Recall: To calculate the recall rate of a classification model, the ratio of correctly predicted positive samples to the total number of actual positive samples is used. Recall measures the model's effectiveness in identifying positive instances, with a higher recall indicating a greater ability to correctly identify positive samples. Recall=TP/(TP+FN).

AUC: The axes of the ROC curve are false positive rate (FPR) and true rate (TPR), AUC is the area under the ROC curve. AUC is positively correlated with the performance of the model.

Table 1
The performance of the detection method on different models, which contains the maximum, minimum and average values of various evaluation metrics.

Model	Googlenet	Inception-v3	Resnet34
Acc(Max)	1.000	0.991	1.000
Acc(Min)	0.893	0.881	0.914
Acc(Average)	0.979	0.956	0.980
Recall(Max)	1.000	0.982	1.000
Recall(Min)	0.782	0.757	0.825
Recall(Average)	0.958	0.911	0.961
AUC(Max)	1.000	1.000	1.000
AUC(Min)	0.962	0.963	0.966
AUC(Average)	0.997	0.994	0.998

The evaluation results are presented in Table 1. It can be seen that our method has a good classification effect on the three models and the mixed data sets of normal samples and adversarial sample data sets generated by eight attack methods. The average values of the three evaluation indicators are all more than 91%. In terms of accuracy, in the results of 100 classes, the maximum value of the Acc score of the three test models is above 99%, the minimum value is above 88%, and the average value of 100 classes is above 95%, which shows that our detection method is strong. The accuracy is high and it can effectively detect adversarial examples. In terms of recall rate, in the results of 100 classes, the maximum Recall score of the three test models is above 98%, the minimum value is above 75%, and the average value of 100 classes is above 91%, which shows that our detection method recognizes positive samples is strong, and it is easier to detect normal samples. In terms of AUC indicators, in the results of 100 classes, the maximum AUC values of the three test models are all 1, the minimum values are all above 0.96, and the average values of the 100 classes are all above 0.99, which demonstrates that the performance of our detection method is competitive.

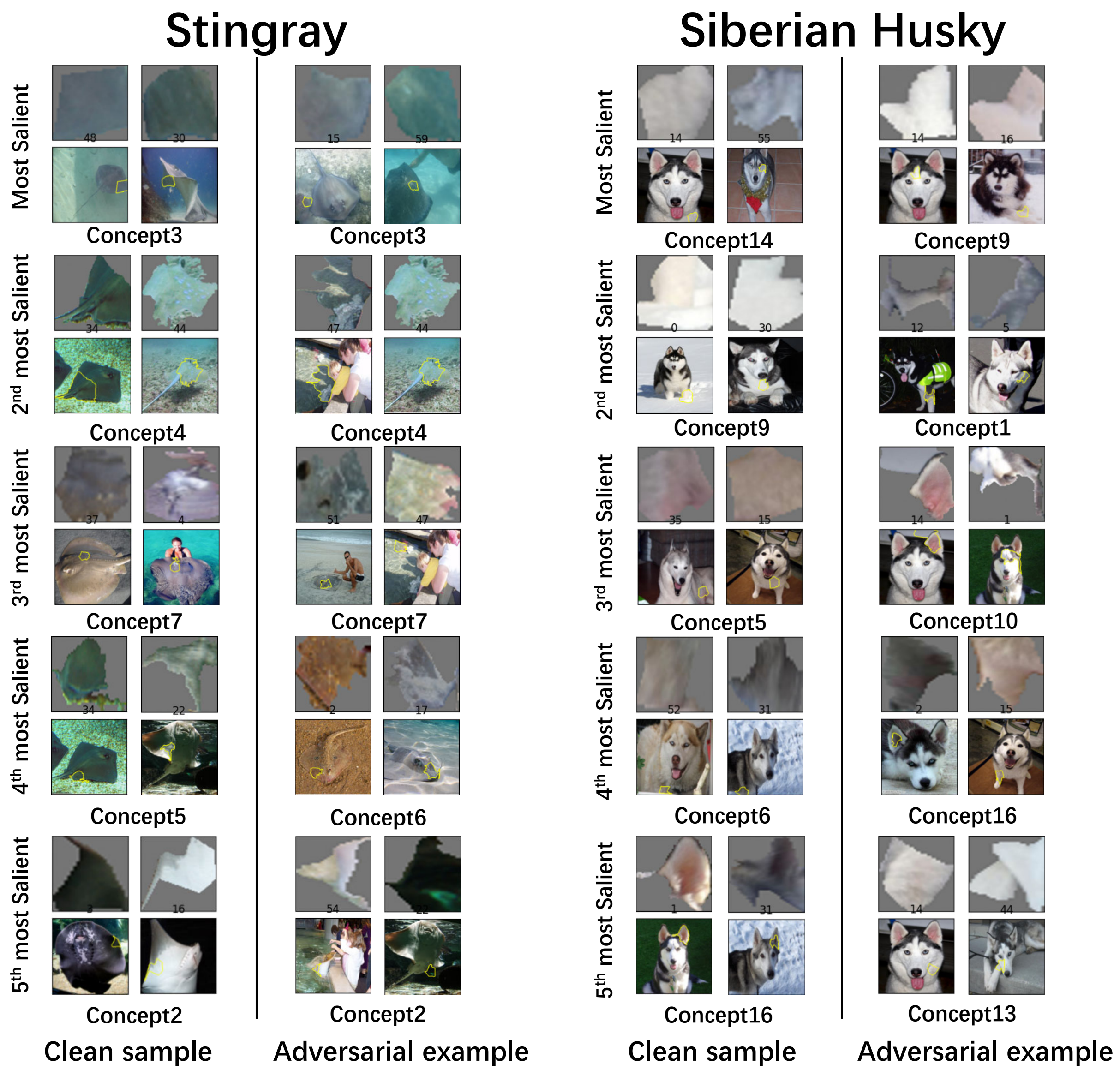

Figure 5: Examples of the first five concept sets of two categories of normal samples and adversarial examples.

4.3. Differences in performance of detection methods on different classes

Table 2
Defense Performance Comparison with four previous defense methods on three models. The accuracy values of the detection algorithm are shown below, with the maximum value highlighted in bold.

Acc(Average)	Googlenet	Inception-v3	Resnet34
(Yang et al., 2022)	0.823	0.861	0.916
(Ko and Lim, 2021)	0.774	0.810	0.901
(Cintas et al., 2021)	0.926	0.891	0.851
(Yang et al., 2020)	0.768	0.812	0.883
Ours method	**0.979**	**0.956**	**0.980**

We illustrate the analysis with an example in figure 5. The figure below shows one of the best output results of our detection model on Googlenet, the Siberian_husky class. This class achieves an accuracy of 1. There is also Googlenet's worst numerical Stingray class for our detection model output results, and the accuracy rate of the test set is 0.8937. We can observe that only the fourth concept of the first five concepts of Stingray adversarial samples is different from normal samples. And the top five concepts of the Siberian_husky class adversarial examples are all different from the normal examples. This is consistent with our theory. The more sensitive an adversarial example is to a set of concepts that differs from normal examples, the better our method is.

4.4. Comparison to previous approaches

Previous defense methods (Yang et al., 2020; Cintas et al., 2021; Ko and Lim, 2021; Yang et al., 2022) against adversarial attacks do not consider the concept of image samples themselves. We study whether there is a difference between unperturbed samples and adversarial examples from the perspective of a higher-level concept of image samples. So these previous methods are only for comparison. In Table 2, a comparison of these previous adversarial sample defense methods is provided. Consistent with previous studies, we report the accuracy of detecting adversarial samples generated by the attack methods. For comparative experiments, we use the widely used Imagenet2012 dataset for validation. We set the parameters of all attack algorithms according to the default values in the foolbox. Obviously, our method is effective in detecting adversarial samples.

5. Conclusion

In this paper, we have explored the use of concept activation vectors (CAVs) for detecting adversarial examples. By designing a novel single-sample CAV testing method and applying it to three pre-trained deep learning models (GoogLeNet, ResNet34, and Inception-v3), we demonstrated that our approach effectively detects adversarial examples, achieving competitive results compared to state-of-the-art defense methods. Our key findings suggest that CAVs can capture concept-level changes induced by adversarial perturbations, providing a robust mechanism for adversarial detection. However, we acknowledge the limitations of our method in terms of computational efficiency and model generalizability across various adversarial attack scenarios. Future work could explore optimizing the CAV detection process and extending it to more diverse datasets and attack types.

A. Appendix

A.1. More detection method evaluation results in details

We show our detection method on Googlenet, Inception_v3, and Resnet34 models on data of 100 Imagenet classes each. The tabular data table 3, table 4 and table 5 includes the Accuracy rate, Recall value and AUC value of the detection method on Googlenet, Inception_v3, and Resnet34 respectively.

Table 3
Performance of detection method on Googlenet on 100 class image data

Class Name	**Acc**	**Recall**	**AUC**	**Class Name**	**Acc**	**Recall**	**AUC**
Minibus	0.992	0.983	1.000	Buckeye	0.963	0.923	0.996
Slug	0.979	0.957	1.000	Hummingbird	0.985	0.970	1.000
European Fire Salamander	0.963	0.923	0.986	Barracouta	0.998	0.996	1.000
Reflex Camera	0.985	0.970	1.000	Goose	0.988	0.974	1.000
Tabby	0.977	0.953	0.995	Chimpanzee	0.979	0.957	0.997
Siberian Husky	1.000	1.000	1.000	Trailer Truck	0.998	0.996	1.000
Komondor	0.994	0.987	1.000	Impala	0.983	0.966	0.999
Dowitcher	0.992	0.983	1.000	Ladle	0.902	0.800	0.962
Radio	0.996	0.991	1.000	White Wolf	0.994	0.987	1.000
Wallaby	0.956	0.911	0.992	Stopwatch	0.990	0.979	0.999
Packet	0.963	0.923	0.997	Alligator Lizard	0.988	0.974	0.997
Reel	0.996	0.991	1.000	Punching Bag	1.000	1.000	1.000
Wreck	0.952	0.902	0.998	Candle	0.990	0.979	1.000
Flute	0.971	0.940	1.000	Envelope	0.983	0.966	0.994
Sulphur-crested Cockatoo	0.983	0.966	0.999	White Stork	0.994	0.987	1.000
Hourglass	0.958	0.915	0.990	Lesser Panda	0.969	0.936	1.000
Bassoon	0.998	0.996	1.000	Black Grouse	1.000	1.000	1.000
Tailed Frog	0.998	0.996	1.000	Freight Car	0.992	0.983	1.000
Bullfrog	0.967	0.932	1.000	Tiger Beetle	0.996	0.991	1.000
Jacamar	0.998	0.996	1.000	Teapot	0.921	0.838	0.987
Schipperke	0.975	0.949	1.000	Mask	0.927	0.851	0.988
Greater Swiss Mountain Dog	0.967	0.932	0.997	Loupe	0.990	0.979	0.999
Crane	0.973	0.945	0.998	Windsor Tie	1.000	1.000	1.000
Knot	0.960	0.919	1.000	Vacuum	0.967	0.932	0.998
Eft	0.967	0.932	1.000	Guenon	0.998	0.996	1.000
Golf Ball	1.000	1.000	1.000	Handkerchief	0.946	0.889	0.998
West Highland White Terrier	0.988	0.974	1.000	Miniature Schnauzer	0.965	0.928	0.994
Pole	0.931	0.860	0.977	Pillow	0.917	0.830	0.994
Moped	0.988	0.974	1.000	Scorpion	0.981	0.962	0.993
Pier	0.992	0.983	1.000	Green Lizard	0.965	0.928	1.000
Yellow Lady's Slipper	0.971	0.940	1.000	Nail	0.990	0.979	1.000
Baseball	0.996	0.991	1.000	Bouvier Des Flandres	0.981	0.962	1.000
Malinois	0.998	0.996	1.000	Shopping Cart	0.985	0.970	0.997
Brassiere	0.990	0.979	0.998	Ice Bear	0.975	0.949	0.995
Horse Cart	0.981	0.962	1.000	Ptarmigan	0.992	0.983	1.000
Siamese Cat	0.996	0.991	1.000	Trombone	0.969	0.936	0.994
Strainer	0.990	0.979	0.999	Seashore	0.996	0.991	1.000
Volleyball	0.998	0.996	1.000	Vestment	0.994	0.987	1.000
Stingray	0.894	0.783	0.975	Platypus	0.979	0.957	1.000
Scabbard	0.981	0.962	1.000	Mushroom	0.985	0.970	0.998
Stone Wall	0.979	0.957	1.000	Banana	0.965	0.928	0.996
Bloodhound	0.998	0.996	1.000	Buckle	0.971	0.940	1.000
Pick	0.981	0.962	1.000	Microphone	0.973	0.945	1.000
Scuba Diver	0.990	0.979	1.000	Theater Curtain	1.000	1.000	1.000
Ram	0.971	0.940	1.000	Macaque	0.979	0.957	0.999
Potpie	0.998	0.996	1.000	Walker Hound	0.992	0.983	1.000
Rhodesian Ridgeback	0.988	0.974	1.000	Howler Monkey	0.979	0.957	0.998
Sussex Spaniel	1.000	1.000	1.000	Armadillo	0.977	0.953	0.999
Scoreboard	0.992	0.983	1.000	Gordon Setter	0.994	0.987	1.000
Hippopotamus	0.963	0.923	0.996	Oil Filter	0.996	0.991	1.000

Table 4
Performance of detection method on Inception_v3 on 100 class image data

Class Name	Acc	Recall	AUC	Class Name	Acc	Recall	AUC
Minibus	0.963	0.923	0.997	Buckeye	0.968	0.934	0.996
Slug	0.913	0.821	0.995	Hummingbird	0.945	0.883	0.991
European Fire Salamander	0.949	0.892	0.989	Barracouta	0.988	0.974	0.998
Reflex Camera	0.992	0.983	1.000	Goose	0.937	0.873	0.985
Tabby	0.977	0.953	0.999	Chimpanzee	0.981	0.962	1.000
Siberian Husky	0.985	0.970	1.000	Trailer Truck	0.977	0.953	0.998
Komondor	0.975	0.947	1.000	Impala	0.952	0.902	0.991
Dowitcher	0.981	0.961	1.000	Ladle	0.881	0.757	0.977
Radio	0.952	0.902	0.988	White Wolf	0.990	0.979	1.000
Wallaby	0.927	0.849	0.994	Stopwatch	0.969	0.936	0.999
Packet	0.963	0.923	0.998	Alligator Lizard	0.938	0.872	0.988
Reel	0.967	0.932	0.996	Punching Bag	0.969	0.936	0.999
Wreck	0.981	0.962	0.999	Candle	0.948	0.894	0.993
Flute	0.925	0.847	0.993	Envelope	0.979	0.957	0.999
Sulphur-crested Cockatoo	0.957	0.909	0.998	White Stork	0.935	0.868	0.991
Hourglass	0.983	0.964	1.000	Lesser Panda	0.984	0.967	1.000
Bassoon	0.967	0.932	1.000	Black Grouse	0.958	0.915	0.997
Tailed Frog	0.977	0.953	0.997	Freight Car	0.956	0.911	0.997
Bullfrog	0.933	0.864	0.997	Tiger Beetle	0.928	0.847	0.989
Jacamar	0.985	0.968	1.000	Teapot	0.890	0.774	0.990
Schipperke	0.956	0.911	0.996	Mask	0.913	0.821	0.986
Greater Swiss Mountain Dog	0.965	0.928	0.995	Loupe	0.985	0.970	0.999
Crane	0.960	0.919	0.992	Windsor Tie	0.975	0.949	0.998
Knot	0.931	0.860	0.995	Vacuum	0.929	0.855	0.998
Eft	0.950	0.898	0.995	Guenon	0.992	0.983	1.000
Golf Ball	0.966	0.928	0.999	Handkerchief	0.960	0.919	0.996
West Highland White Terrier	0.935	0.868	0.979	Miniature Schnauzer	0.925	0.847	0.985
Pole	0.885	0.766	0.982	Pillow	0.956	0.911	0.994
Moped	0.985	0.970	1.000	Scorpion	0.975	0.948	0.997
Pier	0.948	0.894	0.993	Green Lizard	0.925	0.847	0.964
Yellow Lady's Slipper	0.988	0.974	1.000	Nail	0.921	0.838	0.986
Baseball	0.981	0.962	1.000	Bouvier Des Flandres	0.981	0.962	0.999
Malinois	0.965	0.928	0.993	Shopping Cart	0.938	0.872	0.998
Brassiere	0.958	0.915	0.998	Ice Bear	0.985	0.970	0.997
Horse Cart	0.954	0.901	0.992	Ptarmigan	0.964	0.926	1.000
Siamese Cat	0.966	0.928	0.996	Trombone	0.929	0.855	0.995
Strainer	0.965	0.928	0.995	Seashore	0.946	0.889	0.993
Volleyball	0.967	0.932	0.999	Vestment	0.960	0.919	0.999
Stingray	0.929	0.855	0.987	Platypus	0.983	0.966	1.000
Scabbard	0.923	0.843	0.985	Mushroom	0.898	0.791	0.981
Stone Wall	0.950	0.898	0.987	Banana	0.929	0.855	0.999
Bloodhound	0.969	0.936	0.998	Buckle	0.923	0.843	0.993
Pick	0.985	0.970	1.000	Microphone	0.965	0.928	0.991
Scuba Diver	0.952	0.903	0.985	Theater Curtain	0.963	0.923	0.998
Ram	0.956	0.911	0.999	Macaque	0.985	0.970	0.998
Potpie	0.981	0.962	0.999	Walker Hound	0.985	0.970	1.000
Rhodesian Ridgeback	0.944	0.885	0.984	Howler Monkey	0.960	0.919	0.988
Sussex Spaniel	0.977	0.953	1.000	Armadillo	0.961	0.919	0.993
Scoreboard	0.962	0.922	1.000	Gordon Setter	0.985	0.970	0.996
Hippopotamus	0.949	0.892	0.992	Oil Filter	0.966	0.928	0.997

Table 5
Performance of detection method on Resnet34 on 100 class image data

Class Name	Acc	Recall	AUC	Class Name	Acc	Recall	AUC
Minibus	0.979	0.957	1.000	Buckeye	0.990	0.978	1.000
Slug	0.955	0.905	0.999	Hummingbird	0.984	0.975	0.999
European Fire Salamander	0.981	0.960	0.999	Barracouta	0.985	0.970	1.000
Reflex Camera	0.991	0.982	1.000	Goose	0.964	0.924	0.998
Tabby	0.991	0.982	1.000	Chimpanzee	0.990	0.979	1.000
Siberian Husky	0.994	0.987	1.000	Trailer Truck	0.983	0.966	1.000
Komondor	1.000	1.000	1.000	Impala	1.000	1.000	1.000
Dowitcher	1.000	1.000	1.000	Ladle	0.921	0.851	0.987
Radio	0.960	0.919	0.995	White Wolf	0.996	0.991	1.000
Wallaby	0.954	0.900	0.999	Stopwatch	0.981	0.962	1.000
Packet	0.977	0.953	0.997	Alligator Lizard	0.950	0.900	0.998
Reel	0.949	0.892	0.993	Punching Bag	0.989	0.978	0.999
Wreck	0.975	0.949	0.999	Candle	0.963	0.923	0.998
Flute	0.973	0.945	0.997	Envelope	0.981	0.962	0.997
Sulphur-crested Cockatoo	1.000	1.000	1.000	White Stork	0.991	0.981	1.000
Hourglass	0.982	0.963	0.999	Lesser Panda	0.993	0.984	1.000
Bassoon	0.994	0.987	1.000	Black Grouse	1.000	1.000	1.000
Tailed Frog	0.990	0.979	1.000	Freight Car	0.992	0.983	1.000
Bullfrog	0.998	0.996	1.000	Tiger Beetle	0.989	0.975	1.000
Jacamar	1.000	1.000	1.000	Teapot	0.979	1.000	0.996
Schipperke	0.992	0.983	1.000	Mask	0.946	0.889	0.998
Greater Swiss Mountain Dog	1.000	1.000	1.000	Loupe	0.973	0.945	0.992
Crane	0.990	0.979	0.999	Windsor Tie	1.000	1.000	1.000
Knot	0.917	0.838	0.981	Vacuum	0.958	0.915	0.998
Eft	0.990	0.978	1.000	Guenon	1.000	1.000	1.000
Golf Ball	0.973	0.941	0.997	Handkerchief	0.981	0.962	0.999
West Highland White Terrier	0.979	0.957	0.998	Miniature Schnauzer	0.994	0.987	1.000
Pole	0.938	0.872	0.989	Pillow	0.954	0.936	0.997
Moped	0.985	0.970	0.999	Scorpion	0.975	0.946	0.998
Pier	0.979	0.957	1.000	Green Lizard	0.958	0.915	0.995
Yellow Lady's Slipper	0.993	0.982	1.000	Nail	0.915	0.826	0.967
Baseball	0.988	0.974	1.000	Bouvier Des Flandres	0.996	0.992	1.000
Malinois	0.989	0.977	1.000	Shopping Cart	0.988	0.974	1.000
Brassiere	0.998	0.996	1.000	Ice Bear	0.988	0.975	1.000
Horse Cart	0.996	0.991	1.000	Ptarmigan	0.991	0.980	1.000
Siamese Cat	0.991	0.980	1.000	Trombone	0.977	0.953	0.998
Strainer	0.963	0.923	0.996	Seashore	0.977	0.953	1.000
Volleyball	0.998	0.996	1.000	Vestment	0.996	0.991	0.999
Stingray	0.954	0.906	0.997	Platypus	0.987	0.973	0.999
Scabbard	0.967	0.932	0.993	Mushroom	0.988	0.974	1.000
Stone Wall	0.955	0.904	0.999	Banana	0.949	0.896	0.997
Bloodhound	1.000	1.000	1.000	Buckle	0.925	0.848	0.994
Pick	0.985	0.970	0.999	Microphone	0.973	0.945	0.998
Scuba Diver	0.992	0.983	0.999	Theater Curtain	1.000	1.000	1.000
Ram	0.994	0.987	1.000	Macaque	0.990	0.979	1.000
Potpie	0.993	0.986	0.999	Walker Hound	1.000	1.000	1.000
Rhodesian Ridgeback	0.998	0.996	1.000	Howler Monkey	0.998	0.996	1.000
Sussex Spaniel	1.000	1.000	1.000	Armadillo	0.979	0.953	0.999
Scoreboard	1.000	1.000	1.000	Gordon Setter	1.000	1.000	1.000
Hippopotamus	0.976	0.945	1.000	Oil Filter	0.987	0.972	1.000

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. Slic superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence 34, 2274–2282.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B., 2018. Sanity checks for saliency maps. Advances in neural information processing systems 31.

Bhagoji, A.N., Cullina, D., Sitawarin, C., Mittal, P., 2018. Enhancing robustness of machine learning systems via data transformations, in: 2018 52nd Annual Conference on Information Sciences and Systems (CISS), IEEE. pp. 1–5.

Brendel, W., Rauber, J., Bethge, M., 2018. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, in: ICLR (Poster).

Carlini, N., Wagner, D.A., 2017. Towards evaluating the robustness of neural networks, in: IEEE Symposium on Security and Privacy, pp. 39–57.

Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N., 2018. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 839–847.

Cintas, C., Speakman, S., Akinwande, V., Ogallo, W., Weldemariam, K., Sridharan, S., McFowland, E., 2021. Detecting adversarial attacks via subset scanning of autoencoder activations and reconstruction error, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pp. 876–882.

Fidel, G., Bitton, R., Shabtai, A., 2020. When Explainability Meets Adversarial Learning: Detecting Adversarial Examples using SHAP Signatures, in: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. doi:10.1109/IJCNN48605.2020.9207637.

Gao, X., Tan, Y.a., Jiang, H., Zhang, Q., Kuang, X., 2019. Boosting targeted black-box attacks via ensemble substitute training and linear augmentation. Applied Sciences 9, 2286.

Ghorbani, A., Abid, A., Zou, J., 2019a. Interpretation of Neural Networks Is Fragile. Proceedings of the AAAI Conference on Artificial Intelligence 33, 3681–3688. doi:10.1609/AAAI.V33I01.33013681, arXiv:1710.10547.

Ghorbani, A., Wexler, J., Zou, J.Y., Kim, B., 2019b. Towards automatic concept-based explanations. Advances in neural information processing systems 32.

Goodfellow, I.J., Shlens, J., Szegedy, C., 2015. Explaining and harnessing adversarial examples, in: ICLR (Poster).

Guo, F., Zhao, Q., Li, X., Kuang, X., Zhang, J., Han, Y., Tan, Y.a., 2019. Detecting adversarial examples via prediction difference for deep neural networks. Information Sciences 501, 182–192.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

Ignatiev, A., Narodytska, N., Marques-Silva, J., 2019. On relating explanations and adversarial examples. Advances in neural information processing systems 32.

Jang, U., Wu, X., Jha, S., 2017. Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning, in: Proceedings of the 33rd Annual Computer Security Applications Conference, pp. 262–277. doi:10.1145/3134600.3134635.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al., 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav), in: International conference on machine learning, PMLR. pp. 2668–2677.

Ko, G., Lim, G., 2021. Unsupervised detection of adversarial examples with model explanations. arXiv preprint arXiv:2107.10480 .

Li, X., Li, F., 2017. Adversarial examples detection in deep networks with convolutional filter statistics, in: Proceedings of the IEEE international conference on computer vision, pp. 5764–5772.

Liang, B., Li, H., Su, M., Li, X., Shi, W., Wang, X., 2021. Detecting Adversarial Image Examples in Deep Neural Networks with Adaptive Noise Reduction. IEEE Trans. Dependable Secur. Comput. 18, 72–85. doi:10.1109/TDSC.2018.2874243.

Liu, N., Yang, H., Hu, X., 2018. Adversarial detection with model interpretation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1803–1811.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A., 2018a. Towards Deep Learning Models Resistant to Adversarial Attacks, in: International Conference on Learning Representations.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A., 2018b. Towards deep learning models resistant to adversarial attacks, in: ICLR (Poster).

Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P., 2016. Deepfool: A simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582.

Muhammad, M.B., Yeasin, M., 2020. Eigen-cam: Class activation map using principal components, in: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–7.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., 2019. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32.

Rauber, J., Brendel, W., Bethge, M., 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models, in: Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning.

Rony, J., Hafemann, L.G., Oliveira, L.S., Ayed, I.B., Sabourin, R., Granger, E., 2019. Decoupling direction and norm for efficient gradient-based L2 adversarial attacks and defenses, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4322–4330.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., 2015. Imagenet large scale visual recognition challenge. International journal of computer vision 115, 211–252.

Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626.

Sheikholeslami, F., Lotfi, A., Kolter, J.Z., 2020. Provably robust classification of adversarial examples with detection, in: International Conference on Learning Representations.

Sutanto, R.E., Lee, S., 2021. Real-time adversarial attack detection with deep image prior initialized as a high-level representation based blurring network. Electronics (Switzerland) 10, 1–17. doi:10.3390/electronics10010052.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826.

Wang, Y., Tan, Y.a., Zhang, W., Zhao, Y., Kuang, X., 2020. An adversarial attack on dnn-based black-box object detectors. Journal of Network and Computer Applications 161, 102634.

Wang, Z., Guo, H., Zhang, Z., Liu, W., Qin, Z., Ren, K., 2021. Feature Importance-aware Transferable Adversarial Attacks, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 7619–7628. doi:10.1109/ICCV48922.2021.00754.

Xu, X., Zhang, J., Li, Y., Wang, Y., Yang, Y., Shen, H.T., 2020. Adversarial attack against urban scene segmentation for autonomous vehicles. IEEE Transactions on Industrial Informatics 17, 4117–4126.

Yang, P., Chen, J., Hsieh, C.J., Wang, J.L., Jordan, M., 2020. Ml-loo: Detecting adversarial examples with feature attribution, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 6639–6647.

Yang, Y., Gao, R., Li, Y., Lai, Q., Xu, Q., 2022. What you see is not what the network infers: Detecting adversarial examples based on semantic contradiction, in: NDSS.

Yeh, C.K., Kim, B., Arik, S., Li, C.L., Pfister, T., Ravikumar, P., 2020. On completeness-aware concept-based explanations in deep neural networks. Advances in neural information processing systems 33, 20554–20565.

Zheng, J., Zhang, Y., Li, Y., Wu, S., Yu, X., 2023. Towards evaluating the robustness of adversarial attacks against image scaling transformation. Chinese Journal of Electronics 32, 151–158.

Zhou, Y., Booth, S., Ribeiro, M.T., Shah, J., 2022. Do feature attribution methods correctly attribute features?, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 9623–9633.