Neural Processing Letters (2024) 56:85
https://doi.org/10.1007/s11063-024-11572-6

Enhancing Generalization in Few-Shot Learning for Detecting Unknown Adversarial Examples

Wenzhao Liu[1,2] · Wanli Zhang[1] · Kuiwu Yang[1] · Yue Chen[1] · Kaiwei Guo[1] · Jianghong Wei[1]

Accepted: 12 February 2024

Abstract
Deep neural networks, particularly convolutional neural networks, are vulnerable to adversarial examples, undermining their reliability in visual recognition tasks. Adversarial example detection is a crucial defense mechanism against such attacks but often relies on empirical observations and specialized metrics, posing challenges in terms of data efficiency, generalization to unknown attacks, and scalability to high-resolution datasets like ImageNet. To address these issues, we propose a prototypical network-based method using a deep residual network as the backbone architecture. This approach is capable of extracting discriminative features of adversarial and normal examples from various known adversarial examples by constructing few-shot adversarial detection tasks. Then the optimal mapping matrix is computed using the Sinkhorn algorithm from optimal transport theory, and the class centers are iteratively updated, enabling the detection of unknown adversarial examples across scenarios. Experimental results show that the proposed approach outperforms existing methods in the cross-adversary benchmark and achieves enhanced generalization on a subset of ImageNet in detecting both new adversarial attacks and adaptive white-box attacks. The proposed

✉ Yue Chen
chenyue_ieu@hotmail.com

Wenzhao Liu
liuwz_2000@hotmail.com

Wanli Zhang
wanli_zhang@aliyun.com

Kuiwu Yang
yangkw@aliyun.com

Kaiwei Guo
sguokaiwei@163.com

Jianghong Wei
jh_wei1987@hotmail.com

1 State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, Henan, China

2 State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System, Luoyang 471000, Henan, China

Published online: 05 March 2024

approach offers a promising solution for improving the safety of deep neural networks in practical applications.

Keywords Adversarial example detection · Few-shot learning · Prototypical network · New adversarial attacks

1 Introduction

The increasing deployment of deep learning-based AI systems in real-world scenarios has raised serious concerns about their security [8, 76, 88]. Adversarial examples, specifically crafted to evade AI system's security measures, pose a significant threat to the trustworthiness and reliability of these systems [12, 31, 45, 51, 87, 95, 101]. Numerous defense mechanisms have been proposed to mitigate the potential impact of adversarial examples on AI systems [10, 59, 71, 96]. However, the constant emergence of new attack methods has rendered previous defenses ineffective. For instance, the Carlini & Wagner (C&W) attack [12] and backward pass differentiable approximation (BPDA) attack [6] have successfully compromised defense mechanisms that were previously considered robust. This has resulted in an ongoing arms race between attack and defense.

In the face of the complexities of practical application scenarios and the threat of malicious attackers, adversarial examples may not be limited to small L_p-norm adversarial perturbations [94]. Attackers can modify images on a large scale while maintaining semantic similarity [36, 43, 73, 81], or generate adversarial examples through spatial transformations such as rotation or translation [24, 94]. Furthermore, unrestricted adversarial attacks have become a new research focus in recent years [9, 85]. These developments pose significant challenges to existing defense methods.

Adversarial example detection aims to identify adversarial perturbations in image inputs, providing deep neural network (DNN) models with enhanced security without compromising their ability to recognize normal examples. Various techniques exist for detecting adversarial examples, including using distinct behavioral features [26, 34] or leveraging observed behavioral differences in a DNN model's middle layer [25, 56]. Another approach assumes unique statistical properties between adversarial and normal examples to train a classifier [32, 59]. However, these methods demand a large number of adversarial examples, resulting in high time and sample complexity, and generally only detect the same type of adversarial examples. Although some studies have explored unknown adversarial example detection [27, 49, 61, 92], challenges persist in detecting emerging attacks such as semantic attack (SA), spatial transformation attack (STA), AutoAttack (AA), and composite adversarial attack (CAA). As these attacks rapidly evolve, adversarial example detection faces limitations, necessitating the ability to identify unknown and new types of adversarial examples.

Meta-learning adversarial detection (MetaAdvDet) is a representative method for detecting unknown adversarial examples using meta-learning techniques [57]. The MetaAdvDet method uses the model agnostic meta-learning (MAML) [28] approach to study the problem of adversarial example detection from a fast adaptation perspective. However, according to [74], feature reuse is the main factor for the success of MAML in few-shot learning. Inspired by [41], the vulnerability of the model caused by adversarial examples is due to the presence of non-robust features, and models trained by standard training methods are able to exploit such highly predictive features. Therefore, we aim to extract these features from adversarial examples and use them to differentiate from those of normal examples to detect adversarial

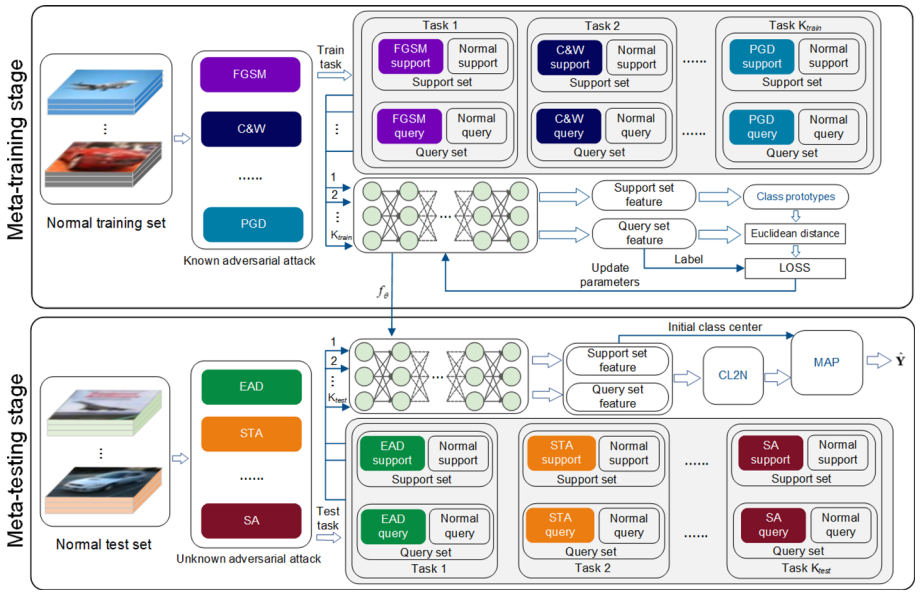

Fig. 1 Overall framework of the proposed PAD approach. The approach consists of two distinct stages: meta-training (upper half of the figure) and meta-testing (lower half of the figure), both utilizing the same feature extraction backbone network f_θ. Upper: The backbone network is trained on known adversarial example detection tasks in multi-task format. Lower: The learned backbone network f_θ is employed to extract features of all samples on unknown adversarial example detection tasks. For each test task, the support set is used to compute the initial class center. After applying the CL2N transformation and MAP algorithm, the query set labels $\hat{\mathbf{Y}}$ can be computed. More details can be found in Algorithms 1 and 2

examples. To achieve this, we use a prototypical network [83] to learn the corresponding feature representations from the adversarial and normal examples, which are clustered around their respective class prototypes in the feature space. The class prototypes are obtained by computing the average of a small number of labeled examples in the feature space. Since the unknown adversarial attack method can use multiple methods to generate adversarial examples, which may differ significantly from the features of the adversarial examples in the training stage, we further process the features extracted from the backbone network in the meta-testing stage and update the class centers using the maximum A posteriori (MAP) algorithm [39].

The overall framework of the method is shown in Fig. 1. Each task represents a small data collection process, including normal examples and one type of randomly selected adversarial examples, simulating new attack scenarios [57]. The meta-training stage trains the feature extraction backbone network on multiple known adversarial example types, while the meta-testing stage tests the detection method's performance on unknown adversarial example types. The method, referred to as PAD (ProtoNet Adversarial Detection), uses a prototypical network to train an end-to-end feature extraction network, processes extracted features of unknown adversarial examples, and predicts using the optimal mapping matrix. It can detect unknown adversarial examples under few-shot conditions. Specifically, our contribution is as follows:

(1) Due to the high time complexity associated with the two-layer optimization employed by the MAML method during training [5], it is customary to utilize a shallow backbone

network. The use of a deeper network can incur a substantial training time overhead. In this regard, we propose the use of residual networks for feature extraction from both normal and adversarial examples within our PAD method. The network design incorporates a 7×7 convolutional kernel while removing average pooling in the initial layer. Subsequently, we calculate the Euclidean distance between unknown adversarial examples and class prototypes to facilitate discrimination between adversarial and normal examples. This approach greatly improves the method's performance for detecting unknown adversarial examples.

(2) We propose applying feature transformation to the features extracted from the backbone network, in conjunction with an iterative algorithm based on optimal transport theory for updating class centers, which further enhances the detection performance of unknown adversarial examples.

(3) Our proposed method exhibits a significant improvement in detecting unknown adversarial examples compared to existing few-shot learning algorithms on the cross-adversary benchmark for MNIST and CIFAR-10 datasets under both 1-shot and 5-shot settings.

(4) We extend the proposed method to the ImageNet dataset and conduct experiments to evaluate its performance in detecting two new adversarial attacks, AA and CAA. The results of the experiments clearly demonstrate the superiority of the method.

2 Related Work

2.1 Adversarial Attack

An adversarial attack involves introducing specific perturbations to create adversarial examples, designed to prompt deep neural networks to produce erroneous predictions without affecting human judgment. Adversarial attacks are classified into L_p and non-L_p attacks, based on their algorithms [53, 89]. Most current attack algorithms utilize L_p-norm perturbation imperceptibility metrics to generate adversarial examples that lead to incorrect decision-making in the target model [1, 33].

The L_p-norm adversarial example is denoted by $\mathbf{x}_{adv} = \mathbf{x} + \boldsymbol{\delta}$, wherein $\boldsymbol{\delta}$ is derived by solving Eq. 1:

$$\max_{\boldsymbol{\delta} \in \Delta} \mathcal{L}\left(f_\theta(\mathbf{x} + \boldsymbol{\delta}), y_{\text{true}}\right), s.t. \Delta = \left\{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p < \varepsilon\right\} \tag{1}$$

$\mathcal{L}$ denotes the model's loss function, typically characterized as cross-entropy loss, with p taking values 0, 1, 2, and ∞, each corresponding to distinct L_p-norm adversarial examples [80]. L_0 measures the number of pixels that can be perturbed; L_2 measures the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}_{adv}$; L_∞ measures the maximum alterable distance across all pixels. Perturbations based on the L_p-norm constraint are inadequate for measuring perceptual similarity [80]. Adversarial examples can be generated by altering color [36, 78], texture [7], spatial location [24, 94], and other factors, thus inducing model misclassification while preserving semantic or structural information. In the absence of L_p-norm restrictions, an attacker may introduce extensive and conspicuous modifications to an image, prompting model misclassification without affecting normal human perception. Such examples are referred to as unrestricted adversarial examples [1, 9, 85]. Bhattad et al. [7] demonstrate that adversarial training methods based on L_p-norm examples are not resilient to these adversarial examples.

Vulnerability assessment evaluates the target model's susceptibility to performance degradation when confronted with sophisticated perturbations, emphasizing metrics like attack success rate or robust accuracy. AA and CAA represent two novel adversarial attack method-

ologies. AA is an L_p attack, while CAA additionally encompasses non-L_p attacks. AA combines improved PGD methods with FAB [18] and Square Attack [4], forming a collection of attacks commonly used for evaluating robustness. CAA defines a sequential step, creating a powerful strategy against defense models by automatically searching for optimal attack combinations among 32 algorithms and parameters.

2.2 Adversarial Defense

Adversarial defense methods fall into two categories. The first aims to enhance model robustness, aligning the model's perception with human perception and ensuring accurate prediction of adversarial examples. Adversarial training is a key method for improving robustness but faces challenges from robustness trade-offs [99] and gradient masking issues [89]. An increase in adversarial robustness may reduce accuracy for normal examples [90], and L_∞-norm-based adversarial training models may lack robustness against L_1-norm or L_2-norm adversarial examples [53, 85, 89].

Another defense strategy is adversarial example detection, which identifies adversarial perturbations in input samples during the model's prediction phase. If detected, the sample is rejected; otherwise, it is processed by the target model. The detection of adversarial examples serves as a critical defense mechanism against adversarial attacks, adding an extra layer of security for practical deployment of DNN models. The detection task need to address the challenge of identifying potentially malicious samples given the acknowledgment of the model's vulnerability. The detector's focus lies on the detection capability of the detector, specifically its ability to accurately distinguish between adversarial and normal samples. Detection methods are classified as supervised or unsupervised based on their use of adversarial example information [2]. Supervised detection methods include network invariant [13, 25, 56, 64, 72], auxiliary model [44, 82, 104] and statistical methods [17, 26, 32, 49, 50, 59]. Unsupervised methods are further divided into network invariant [58], auxiliary model [3, 68, 86], statistical [34, 35, 102], object-based [29], denoiser [63, 84], and feature-squeezing methods [52, 97]. Additionally, few-shot learning detection methods, such as the meta-learning approach employed by [57], named MetaAdvDet, address the challenge of detecting new adversarial attacks with limited examples by modeling detection as a few-shot learning problem and utilizing a double-network framework for rapid adaptation to new attacks.

The MetaAdvDet method innovatively utilizes a meta-learning approach to detect adversarial examples. However, it primarily relies on the fast adaptation ability of the MAML algorithm itself, which remains deficient in explaining the nature of adversarial examples, thus limiting the room for further improvement of its method. Our method utilizes a metric learning-based approach to conduct research around feature extraction of adversarial and normal examples, which addresses some of the shortcomings in [57] and improves the generalization for detecting unknown adversarial examples. The motivation and experimental focus of our proposed method are quite different from [57], mainly in the following three aspects.

Firstly, while MetaAdvDet tackles the problem of detecting unknown adversarial examples by emphasizing the algorithm's ability to adapt quickly, our approach builds on the work of [41], which examines adversarial examples from a feature-based perspective. As a result, our choice of algorithms fundamentally differs from that of MetaAdvDet.

Secondly, although MetaAdvDet proposes a framework for detecting unknown adversarial examples, it provides limited results in terms of detecting such examples. In contrast, our method not only reports the detection results of the two missing types of adversarial examples

identified in [57], but also extends these results to a subset of the ImageNet dataset. This significantly surpasses the capabilities of existing methods.

Thirdly, the cross-domain and cross-architecture test results presented in [57] are not directly relevant to the detection of unknown adversarial examples. In contrast, our proposed method places a greater emphasis on detecting unknown adversarial examples. The design of the backbone network and the selection of modules are based on the feature extraction of adversarial examples as the starting point, enabling us to provide more comprehensive detection results for unknown adversarial examples.

3 Approach

3.1 Overview

Figure 1 illustrates the proposed method's framework, including meta-training and meta-testing stages. Each task consists of a support set and a query set. In the meta-training stage, the support set is used to calculate the class prototype, and adversarial examples are identified based on the distance between the prototype and the query set features. The meta-testing stage involves the backbone network extracting features from unknown adversarial examples in each task's support and query sets. These features undergo Center and L_2-normalize (CL2N) transformation [93] and the MAP algorithm (lines 6–11 in Algorithm 2) [39] to update the class center. After iterative steps, probabilities for normal and adversarial examples in the query set are obtained. The meta-testing stage's support set contains 1-shot or 5-shot adversarial examples, enabling the framework to detect new adversarial attacks with few-shot examples.

3.2 Backbone Network

In few-shot learning, a deeper backbone network can reduce intra-class variation [16]. Influenced by the non-local global self-attention mechanism of transformers, recent convolutional network architectures have embraced the concept of using large convolutional kernels [22, 55]. The motivation behind this choice, as discussed in [22], lies in the increased effective receptive fields (ERFs) and the introduction of more shape bias. Our ablation experiments also indicate that smaller kernel sizes are less effective at identifying adversarial examples. Thus, our prototypical network employs ResNet-10, a simplified version of ResNet-18 [16], with a 7×7 kernel size in the first convolutional layer, as opposed to MetaAdvDet's 3-layer convolutional network. Adversarial perturbations with small L_p-norm resemble hidden image perturbations in steganalysis [14]. Pooling operations in residual networks may hinder the extraction of noise-like perturbation features, leading to their removal after the first convolutional layer. The designed backbone network structure is illustrated in Fig. 2.

3.3 Task Construction

An adversarial example dataset is created by applying adversarial attack methods to a selected dataset and dividing it into training and test sets. Meta-training tasks are derived from the training set, while meta-testing tasks come from the test set. Each task, containing one adversarial example type, is split into support and query sets with normal and adversarial examples. Support set sample number depends on a specified shot, such as 1-shot or 5-shot. To

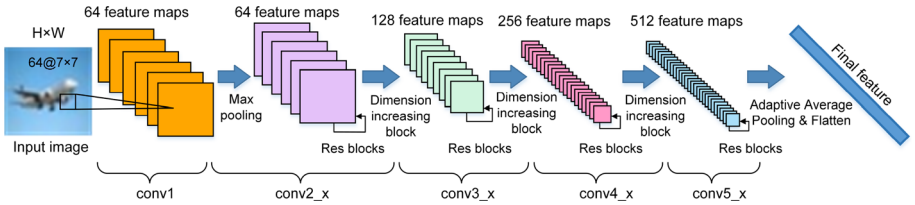

Fig. 2 Feature extraction backbone network architecture

simulate unknown attack detection scenarios, meta-testing tasks require distinct adversarial types, maintaining the same support set sample number as in the meta-training stage, and using a query set to evaluate the detector's performance.

3.4 Meta-training Stage

The original dataset is denoted as $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$, and the set of adversarial attack algorithms is represented by $\mathbb{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$, where n is the total number of attack operations. The adversarial example datasets for the meta-training and meta-testing stages are given by $\mathcal{D}_{\text{meta\_train}} = \mathcal{D}_{\text{adv}} \cup \mathcal{D}_{\text{train}} = \mathcal{A}(\mathcal{D}_{\text{train}}) \cup \mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{meta\_test}} = \mathcal{D}'_{\text{adv}} \cup \mathcal{D}_{\text{test}} = \mathcal{A}'(\mathcal{D}_{\text{test}}) \cup \mathcal{D}_{\text{test}}$, respectively, with the condition that $\mathcal{A} \cap \mathcal{A}' = \varnothing$, $\mathcal{A}, \mathcal{A}' \subseteq \mathbb{A}$.

The adversarial example detection task is a 2-way s-shot learning task, as category labels are divided into normal and adversarial examples. Meta-training tasks $\mathcal{D}_{\text{task}}$ are formed by random sampling from $\mathcal{D}_{\text{meta\_train}}$. Each task $\mathbb{T}_t$ includes a support set $S^t = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{2s}, y_{2s})\}$ and a query set $Q^t = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{2q}, y_{2q})\}$. The variable y represents category labels, and S^t_j denotes the set of samples with label j in the support set, which contains s samples per class, while the query set has q samples per class.

Denote the backbone network as f_θ, extracting feature $\mathbf{f}$ from input sample $\mathbf{x}$ as $\mathbf{f} = f_\theta(\mathbf{x}), \mathbf{f} \in \mathbb{R}^D$ with learnable parameter θ. The extracted feature is obtained after applying the ReLU function in the backbone network, ensuring non-negative feature components. Denote $\mathbf{o}_{j, j\in\{0,1\}}$ as the class prototype of task $\mathbb{T}_t$, with $\mathbf{o}_j \in \mathbb{R}^D$. Each class prototype is computed by averaging the support set sample features as follows:

$$\mathbf{o}_j = \frac{1}{s} \sum_{(\mathbf{x}, y) \in S^t_j} f_\theta(\mathbf{x}) \times \mathbb{1}\{y = j\} \tag{2}$$

Given a distance function $d : \mathbb{R}^D \times \mathbb{R}^D \to [0, +\infty)$, a distribution over classes for a query point $\mathbf{x}$ is obtained by applying a softmax over distances between the features and class prototypes as follows:

$$p_\theta(y = j \mid \mathbf{x}) = \frac{\exp\left(-d\left(f_\theta(\mathbf{x}), \mathbf{o}_j\right)\right)}{\sum_{j'} \exp\left(-d\left(f_\theta(\mathbf{x}), \mathbf{o}_{j'}\right)\right)} \tag{3}$$

Learning proceeds by minimizing the negative log-probability $J(\theta) = -\log p_\theta(y = j \mid \mathbf{x})$ of the true class label. The metric employed for measuring distance is the squared Euclidean distance. By adopting a negative value for this metric, the distance is transformed into a measure of similarity. This means that the smaller the distance, the higher the similarity score between the sample and the prototype. The meta-training algorithm is illustrated in Algorithm 1. The similarity scores corresponding to $j = 0$ and $j = 1$ are represented as

"scores". The cross-entropy function is then calculated between these scores and the true label.

Algorithm 1 PAD meta-training procedure

Require: The multitask format dataset D_{task}, total task K_{train}, backbone network f_θ and its parameters θ, max iterations N, cross entropy loss $\mathcal{L}$, learning rate λ.
Ensure: the learned network f_θ.

```
sample K_train tasks T_{t∈{1,2,···,K_train}} from D_task
for iter ← 1 to N do
    for t ← 1 to K_train do
        S^t and Q^t ← support set and query set of T_t
        o_j ← (1/s) Σ_{(x,y)∈S_j^t} f_θ(x)                    ▷ compute prototype
        L ← 0
        for each (x, y) ∈ Q^t do
            scores ← [−d(f_θ(x), o_0), −d(f_θ(x), o_1)]       ▷ compute scores
            L ← L + (1/2q) CELoss(scores, y)                  ▷ compute loss
        end for
        θ ← θ − λ∇L                                          ▷ update
    end for
end for
return f_θ
```

3.5 Meta-testing Stage

Unknown adversarial example detection tasks $\mathcal{D}'_{\text{task}}$ are randomly sampled from $\mathcal{D}_{\text{meta\_test}}$. The backbone network f_θ, obtained in the meta-training stage, is employed to extract features from the support and query sets. The support set contains labeled samples with extracted features denoted as $\mathbf{F}_S$, while the query set contains unlabeled samples with extracted features denoted as $\mathbf{F}_Q$. For each $\mathbf{f} \in \mathbf{F}_S \cup \mathbf{F}_Q$, $l(\mathbf{f})$ represents the label of the corresponding sample, while $\mathbf{o}_{j,j\in\{0,1\}}$ signifies the class center corresponding to class j. Given that feature transformations have a positive impact on few-shot learning tasks [93], the feature vector $\mathbf{f}$ undergoes a CL2N feature transformation as per Eq. 4, resulting in $\hat{\mathbf{f}}$.

$$\begin{aligned} &\bar{\mathbf{f}}_S = \frac{1}{s}\sum_{(\mathbf{x},y)\in S} f_\theta(\mathbf{x}),\ \bar{\mathbf{f}}_Q = \frac{1}{q}\sum_{(\mathbf{x},y)\in Q} f_\theta(\mathbf{x}), \\ &\hat{\mathbf{f}} \leftarrow \frac{\mathbf{f}-\bar{\mathbf{f}}}{\parallel \mathbf{f}-\bar{\mathbf{f}} \parallel_2}, \forall \mathbf{f} \in \mathbf{F}_s, \bar{\mathbf{f}} = \bar{\mathbf{f}}_S; \forall \mathbf{f} \in \mathbf{F}_Q, \bar{\mathbf{f}} = \bar{\mathbf{f}}_Q \end{aligned} \tag{4}$$

The prototype network approach assumes the existence of an embedding space where samples cluster around a single class center, i.e., the prototype. Therefore, clustering can be used to classify the query set. Let $\mathbf{P} = \left(p_{ij}\right)_{2q\times 2}$ represent the probability matrix, with each element p_{ij} signifying the likelihood that an unlabeled sample i is assigned to class center $\mathbf{o}_j$. Define $\mathbf{M} = \left(m_{ij}\right)_{2q\times 2}$ as the cost matrix, where $m_{ij} = \left\|\mathbf{f}_i - \mathbf{o}_j\right\|_2^2$ denotes the Euclidean distance between sample i and class center $\mathbf{o}_j$. The optimization objective of the clustering problem is formulated as:

$$\min_{\tilde{p}_{i,j},\tilde{\mathbf{o}}_j} \sum_{i=1}^{2q}\sum_{j=0}^{1} p_{ij} m_{ij} \tag{5}$$

Different approaches to solving Eq. 5 can be derived by applying various assumptions to the probabilities p_{ij} and class center $\mathbf{o}_j$ in the above optimization objective. Three such methods will be introduced.

(1) Nearest Class Mean (NCM) method. The original prototype network classifies query set samples with the NCM method, where distances between query set samples and the prototype are calculated to predict class membership. In Eq. 5, the class center $\mathbf{o}_j$ remains constant, computed solely from the support set samples. Given the distance function $d : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}^+$, the optimal solution to Eq. 5, taking into account the normalization of probabilities, results in:

$$p_{ij^*} = \frac{1}{2q}, j^* = \arg\min_j d(\mathbf{f}_i, \mathbf{o}_j) \tag{6}$$

However, few-shot learning's inherent sample scarcity leads to biases in class center computation, as only a few samples are used. Reference [54] identifies two bias types affecting class center estimation and theorizes that using a larger sample set can improve the lower bound of the expected performance. Consequently, incorporating more unlabeled samples within the unsupervised clustering framework can diminish bias in class center estimation.

(2) K-means method. If Eq. 5 satisfies the condition specified in Eq. 7, then it represents the optimization objective of the K-means algorithm, an unsupervised clustering method. The samples in the query set are uniformly distributed, inversely related to the total number of samples in the set, as indicated by the equation $\sum_{j=0}^{1} p_{ij} = \frac{1}{2q}$. Notably, since $p_{ij} \in \{0, 1/2q\}$, each sample is assigned to only one class center, reflecting a hard assignment approach. In optimizing the objective Eq. 5, the support set is employed to initialize the class center. The K-means algorithm then iteratively refines the class center estimation. Upon updating the class center, Eq. 6 is applied for prediction.

$$\sum_{j=0}^{1} p_{ij} = \frac{1}{2q}, p_{ij} \in \left\{0, \frac{1}{2q}\right\}, \forall i \in \{1, ..., 2q\}, j \in \{0, 1\} \tag{7}$$

(3) MAP method. The hard assignment approach employed by the K-means method renders the inference process non-differentiable [75]. Furthermore, adversarial examples derived from normal samples show varying distances from the original class after neural network transformations, depending on the type of perturbation. This variance necessitates a soft assignment method, shifting the constraint from $p_{ij} \in \{0, 1/(2q)\}$ to $p_{ij} \geq 0$. This adjustment ensures all samples contribute to the current class center calculation, reducing detection errors in scenarios with high intra-class variance and low inter-class variance.

Our approach to detecting adversarial examples aligns with the typical few-shot learning setup, where all the classes within the query sets are equally likely. This is known as the uniform prior on the class distribution, which represents an equal probability scenario of encountering either an attack or normal environment. The equation $\sum_{i=1}^{2q} p_{ij} = \frac{1}{2}$ symbolizes this uniform class prior, indicating equal likelihood of both adversarial and normal classes appearing in the query set. Defining $\mathbf{r}$ and $\mathbf{c}$ as $\mathbf{r} = [\frac{1}{2q}, ..., \frac{1}{2q}]^\mathsf{T}$ and $\mathbf{c} = [\frac{1}{2}, ..., \frac{1}{2}]^\mathsf{T}$, and coupled with the soft allocation condition $p_{ij} \geq 0$, we reformulate the optimization objective as follows:

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{r}, \mathbf{c}) &= \min_{\mathbf{P} \in \mathbb{U}(\mathbf{r}, \mathbf{c})} < \mathbf{P}, \mathbf{M} >= \min_{\mathbf{P} \in \mathbb{U}(\mathbf{r}, \mathbf{c})} \sum_{ij} p_{ij} m_{ij}, \text{where} \\ \mathbb{U}(\mathbf{r}, \mathbf{c}) &= \left\{ \mathbf{P} \in \mathbb{R}_+^{2q \times 2} \mid \mathbf{P}\mathbf{1}_2 = \mathbf{r}, \mathbf{P}^\mathsf{T}\mathbf{1}_{2q} = \mathbf{c} \right\} \end{aligned} \tag{8}$$

The symbol $\mathbf{1}_d$ denotes a d-dimensional vector of ones. The Eq. 8 quantifies the cost of transitioning from the marginal distribution $\mathbf{r}$ to $\mathbf{c}$ via a transport matrix $\mathbf{P}$. Here, $\mathbb{U}(\mathbf{r}, \mathbf{c})$

denotes all possible transport schemes. This process results in the distance $d_{\mathbf{M}}(\mathbf{r}, \mathbf{c})$, leading to the optimal transport scheme $\mathbf{P}^*$ as per optimal transport theory. The Sinkhorn iterative algorithm, noted for its efficiency, is particularly effective in solving this problem. The optimal transport matrix $\mathbf{P}^*$ can be obtained as follows:

$$\begin{aligned} \mathbf{P}^* &= Sinkhorn\left(\mathbf{M}, \mathbf{r}, \mathbf{c}, \lambda\right) \\ &= \underset{\mathbf{P} \in \mathbb{U}(\mathbf{r}, \mathbf{c})}{\arg\max} \sum\nolimits_{ij} p_{ij} m_{ij} - \frac{1}{\lambda} h\left(\mathbf{P}\right) \end{aligned} \tag{9}$$

The entropy of $\mathbf{P}$, denoted as $h\left(\mathbf{P}\right)$, can be expressed as:

$$h\left(\mathbf{P}\right) = -\sum_{ij} p_{ij} \log p_{ij} \tag{10}$$

As λ increases, the influence of information entropy diminishes, and the cost matrix exerts a stronger impact on the final result. Based on Theorem 2 of [20], the solution of Eq. 9 is unique and assumes the following form:

$$\mathbf{P}^* = \operatorname{diag}\left(\mathbf{u}\right) \exp\left(-\lambda \mathbf{M}\right) \operatorname{diag}\left(\mathbf{v}\right) \tag{11}$$

The vectors $\mathbf{u}$ and $\mathbf{v}$ can be determined using Sinkhorn's fixed point iteration algorithm, which can then be substituted into Eq. 11 to obtain $\mathbf{P}^*$. In calculating the cost matrix $\mathbf{M}$ in Eq. 11, the class centers are first initialized using the samples from the support set. After obtaining $\mathbf{P}^*$, the class centers are re-estimated according to Eq. 12 [39] using the MAP method.

$$\mathbf{o}_j \leftarrow \mathbf{o}_j + \alpha \left(\boldsymbol{\mu}_j - \mathbf{o}_j\right) \tag{12}$$

Where $\boldsymbol{\mu}_j$ is calculated by Eq. 13.

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{2q} p_{ij}^* \mathbf{f}_i + \sum_{\mathbf{f} \in \mathbf{f}_S, l(\mathbf{f})=j} \mathbf{f}}{s + \sum_{i=1}^{2q} p_{ij}^*} \tag{13}$$

After N iterations, the rows of $\mathbf{P}^*$ indicate class probabilities, with the maximum value determining the class label assigned to unlabeled samples, denoted as $\hat{\mathbf{Y}}$.

Transductive learning typically outperforms inductive learning in few-shot learning tasks, as evidenced by several leading methods cited in references [39, 40, 77, 100] that leverage the Sinkhorn algorithm, a transductive method. Eq. 8 imposes a priori constraints on the number of samples in each class. In comparison, the NCM and K-means classifiers do not adequately integrate prior knowledge and constraints necessary for adversarial example detection. Therefore, the MAP algorithm, informed by the Sinkhorn algorithm and enriched with a priori knowledge, stands out as a more effective option for detecting adversarial examples during the meta-testing phase (refer to Sect. 4.5 for ablation experiments with different class-center calculation methods). Additionally, the inclusion of the entropy regularization term in the optimization objective facilitates faster convergence.

For the query set of K_{test} tasks, labels are predicted and evaluation metrics are computed. Subsequently, the evaluation metrics for the entire task are averaged to obtain the final evaluation result of the detector. The meta-testing algorithm is shown in Algorithm 2.

3.6 Threat Model

In the weakest defense setting, the defender is aware of the attack and can use the generated adversarial examples for training, i.e., the same type of adversarial examples are employed

Algorithm 2 PAD meta-testing procedure

Require: the multitask format dataset D'_{task}, total tasks K_{test}, the backbone network f_θ, regularization hyper-parameter λ, learning rate α, max iterations N, ground truth $\mathbf{Y}_{t,t\in\{1,\cdots,K_{\text{test}}\}}$.
Ensure: average F1 score over all tasks

for $t \leftarrow 1$ to K_{test} **do**
  S^t and $Q^t \leftarrow$ support set and query set of $\mathbb{T}_t$
  extract features $\mathbf{f}_S$ and $\mathbf{f}_Q$ from S^t and Q^t
  $\hat{\mathbf{f}}_S \leftarrow CL2N\left(\mathbf{f}_S\right), \hat{\mathbf{f}}_Q \leftarrow CL2N\left(\mathbf{f}_Q\right)$ ▷ CL2N transformation
  $\mathbf{o}_j \leftarrow \frac{1}{s}\sum_{\hat{\mathbf{f}}\in\hat{\mathbf{f}}_S, l\left(\hat{\mathbf{f}}\right)=j}\hat{\mathbf{f}}$ ▷ initializing class centers
  for $iter \leftarrow 1$ to N **do**
    $m_{ij} = \left\|\mathbf{f}_i - \mathbf{o}_j\right\|^2$ ▷ compute cost matrix
    $\mathbf{P}^* = Sinkhorn\left(\mathbf{M}, \mathbf{r}, \mathbf{c}, \lambda\right)$ ▷ compute mapping matrix
    calculate $\boldsymbol{\mu}_j$ according to Equation 13
    $\mathbf{o}_j \leftarrow \mathbf{o}_j + \alpha\left(\boldsymbol{\mu}_j - \mathbf{o}_j\right)$ ▷ estimate class center
  end for
  $\hat{\mathbf{Y}}_t \leftarrow \arg\max \mathbf{P}^*$ ▷ get prediction
  $\text{score}_t \leftarrow \text{F1}\left(\hat{\mathbf{Y}}_t, \mathbf{Y}_t\right)$
end for
F1 score $\leftarrow \frac{1}{K_{\text{test}}}\sum_{t=1}^{K_{\text{test}}} score_t$
return F1 score

for both training and testing. This is referred to as attack-aware black-box detection [61]. Traditional supervised adversarial example detection algorithms are evaluated under this setting. In real attack scenarios, it is challenging for the defender to know the adversary's strategy. However, a limited number of adversarial examples can be labeled based on the system's actual response or manual means. Consequently, this paper assesses the detector's capability under this setting.

We consider two different threat models according to the adversary's knowledge of the defender [11, 14, 68]: oblivious adversaries and adaptive white-box adversaries.

Under the oblivious attack, adversaries are unaware of the detector and generates adversarial examples using an unprotected target model. This paper evaluates the detector's performance in two cases: the cross-adversary benchmark and new adversarial attack scenarios. The benchmark employs the setting of [57] and includes less common attack types, such as EAD [15], SA [37], and STA [24], simulating unknown adversarial scenarios. Additionally, two new adversarial attacks, AA [19] and CAA [62], are incorporated to assess the detector's detection capability. This paper adopts the searched strategies of [62], CAA-L_∞, CAA-L_2, and CAA-unrestricted, as shown in Table 1.

Under the adaptive white-box attack, adversaries possess full knowledge of the original classification model's parameters, training strategy, and the detector. Consequently, the adversary can target both the original classification model and the detector, leading to misclassification and detector evasion. Building upon the concept from [11], we combine the original classification model and the detector into a single model. We then employ C&W attack to generate white-box adversarial examples, with the combined model represented as follows:

$$G(\mathbf{x})_i = \begin{cases} Z_C(\mathbf{x})_i & \text{if } i \le N \\ 2 \times Z_{D_K}(\mathbf{x}) \times \max_j Z_C(\mathbf{x})_j & \text{if } i = N+1 \end{cases} \tag{14}$$

$Z_C\ (\mathbf{x})_i$ represents the logits output of the classification layer in the original classification model. Diverging from [57], which only attacks the master network of the meta-learner in MetaAdvDet before fine-tuning, this paper employs a more potent attack setting, i.e.,

Table 1 Types of adversaries under the oblivious attack

Stage	Attack strategy		Types of adversarial examples
Meta-training stage	Known adversarial attacks		FGSM [31], MI-FGSM-L_∞ [23], BIM-L_∞ [47], PGD-L_∞ [60], C&W-L_2 [12], JSMA [70], SPSA [91], VAT [65], MaxConfidence [30]
	Validation adversarial attacks		LBFGS [98]
Meta-testing stage	Cross-adversary benchmark		SA [43], STA [94], DFA(DeepFool Attack) [66], NFA(NewtonFool Attack) [42]
	New adversarial attack	AA [19]	APGD_{CE}, APGD_{DLR}, FAB [18], SquareAttack [4]
		CAA-L_∞ [62]	[('MT-L_∞', e = 8/255, t = 50), ('MT-L_∞', e = 8/255, t = 25), ('C&W-L_∞', e = 8/255, t = 125)]
		CAA-L_2 [62]	[('MT-L_2', e = 0.5, t = 100), ('PGD-L_2', e = 0.4375, t = 125), ('DDNAttack', t = 1000)]
		CAA-unrestricted [62]	[('FogAttack', e = 1, t = 1), ('FogAttack', e = 1, t = 1), ('SPSAAttack', e = 16/255, t = 100)]

dynamically generating adversarial examples for the fine-tuned model on each task. D_K denotes the fine-tuned detection model for the Kth task, and $Z_{D_K}(\mathbf{x})$ signifies the logits of input examples classified as adversarial by this model. If $Z_{D_K}(\mathbf{x})$ exceeds 0.5, the output class of $G(\mathbf{x})_i$ will be $N+1$; otherwise, it will match the original classifier output.

4 Experiment

4.1 Datasets and Settings

4.1.1 Datasets

In the experiments, we utilized three widely-used datasets: MNIST [48], CIFAR-10 [46], and ImageNet [21]. Due to computational resource limitations and time constraints, we used a subset of ImageNet, the ImageNette dataset [38], which consists of 10 categories with 9469 training images and 3925 test images. The MNIST and CIFAR-10 retain their default image sizes, while ImageNet images are uniformly scaled to $224\times224\times3$.

4.1.2 Target Models

For MNIST and CIFAR-10, we employed a 4-layer convolutional network (Conv-4)[1] as the target model. It is trained for 100 epochs on the training set using the Adam optimizer with a learning rate of 0.001, resulting in test set accuracies of 98.87% and 82.83%, respectively. For the ImageNet dataset, we utilized the ResNet-50 network with the same training parameters as in the MNIST and CIFAR-10 datasets, achieving a test set accuracy of 80.36%. Target models are trained without data augmentation techniques, employing only label smoothing with a smoothing factor of 0.1.

4.1.3 Attack Parameter Settings

We employed 15 attack methods from the CleverHans library [69] for the meta-training stage and the cross-adversary benchmark, using the default settings from [57]. The JSMA attack on ImageNet and AA are implemented using the ART toolbox [67], while CAA employed the algorithm provided by [62]. For AA on CIFAR-10 and ImageNet, $\epsilon = 0.03$, while $\epsilon = 0.05$ on MNIST, maintaining a consistent maximum of 100 iterations for both. Based on the attack strategies searched by [62], CAA-L_∞ and CAA-L_2 are conducted on CIFAR-10 and ImageNet, while CAA-unrestricted is conducted on ImageNet with the parameters detailed in Table 1. For the adaptive white-box attack, a confidence level of 0.3 and a maximum of 100 iterations are used.

4.1.4 Detector Parameter Settings

Our detector employs a prototypical network approach suitable for deeper residual networks, such as ResNet-10, while MetaAdvDet employs a 3-layer convolutional network (Conv-3),

[1] Adversarial examples, as created for the cross-adversary benchmark in [57], are generated against the conv-4 architecture. As elucidated in [79], adversarial examples tailored for different architectures, manifest distinct traits. In crafting our adversarial example datasets on MNIST and CIFAR-10, we retained the conv-4 architecture as the target to ensure dataset consistency and comparability in detection outcomes, thereby eliminating any potential bias introduced by the target model's architecture.

where a deeper residual network might degrade performance, as observed in our experiments. The adversarial example detection task applies a two-way setting for the detector to distinguish adversarial examples from normal ones, with one or five samples (s) per category in the support set and 35 and 15 samples (q) in the query set during the meta-training and meta-testing stages, respectively. The total number of tasks is set to $K_{\text{train}} = 20{,}000$ for meta-training and $K_{\text{test}} = 1000$ for meta-testing. The number of max iterations in the meta-training stage is set to 10. The prototypical network training utilizes the Adam optimizer, a default learning rate of 0.001, and a cross-entropy loss function. Following the settings in [39], the hyperparameters for the meta-testing stage are set to $\lambda = 0.1, \alpha = 0.2$, and $N = 20$, respectively.

4.1.5 Evaluation Indicators

In our experiments, we employ three evaluation metrics–detection accuracy, F1 score, and AUC–to comprehensively assess the detector's performance, rather than relying on a single metric. Contrary to [57], which designates normal samples as positive (label 1 for normal, label 0 for adversarial), we define positive samples as adversarial examples when calculating AUC, while other metrics follow the reverse. The final metrics, including detection accuracy, F1 score, and AUC, are obtained by averaging the results across all tasks in the meta-testing stage.

4.1.6 Methods of Comparison

To exhibit the efficacy of our proposed approach, a fair comparison is essential using the adversarial example dataset constructed in Sect. 4.1.1. The compared methods must detect new adversarial examples with limited samples. Consequently, they should be end-to-end, data-driven learning methods amenable to fine-tuning during the testing stage.

The Baseline and Baseline++ methods [16] use the backbone network structure from Sect. 3.2 and are trained on a balanced adversarial example dataset randomly sampled from $\mathcal{D}_{\text{meta\_train}}$ with equal numbers of adversarial and normal examples. Baseline and Baseline++ differ in that Baseline uses a linear layer after the backbone network as the classifier, while Baseline++ uses cosine distance. Both methods are trained for 50 epochs, the parameters of the backbone network are fixed during fine-tuning, and only the parameters of the classifier are trained. An SGD optimizer is used, with a learning rate of 0.01, and fine-tuning is performed with 20 iterations on the support set during the meta-testing stage. For MetaAdvDet, the default settings from [57] are employed for CIFAR-10 and ImageNet. A decaying learning rate setting is applied to MNIST, where the inner layer update learning rate is initialized to 1 and the outer layer update learning rate is set to 0.1 for the 1-shot setting, and to 0.1 and 0.01, respectively, for the 5-shot setting. The learning rate decays every 700 iteration steps to 1/10 of the original. PACA utilizes a two-stream architecture that leverages pixel artifacts and confidence artifacts for detecting adversarial examples subjected to both few-perturbation and large-perturbation attacks [14]. We further improve the PACA method to detect adversarial examples under few-shot conditions, using a pre-trained model provided by the authors with the default settings from [14]. During the testing stage, the entire network is fine-tuned on the support set with an update number of 20.

Table 2 Attack success rate of unknown adversarial attacks on different datasets

Adversarial attack	Attack success rate (%)		
	MNIST	CIFAR-10	ImageNet
EAD [15]	100.0	100.0	100.0
NFA [42]	100.0	100.0	99.3
DFA [66]	100.0	100.0	99.9
SA [43]	87.9	57.1	68.5
STA [94]	94.8	98.8	80.1
AA [19]	40.2	100.0	99.3
CAA-L_∞ [62]	–	100.0	99.9
CAA-L_2 [62]	–	100.0	27.0
CAA-unrestricted [62]	–	–	99.9
LBFGS [98]	96.2	95.96	86.9

4.2 Attack Results Against Target Models

The construction of the adversarial example datasets involves only those samples that are correctly classified by the target model and successfully attacked by the adversary. The attack success rates on the validation and test sets are presented in Table 2. The results suggest that the attacks included in the cross-adversary benchmark achieve high attack success rates on all datasets, except for STA and SA. It is observed that AA is less effective on MNIST. Moreover, L_2-norm attacks, such as LBFGS and CAA-L_2, have low attack success rates on ImageNet.

4.3 Performance Under the Oblivious Attack

Table 3 presents the results of various methods in the cross-adversary benchmark. Our proposed methods achieve optimal detection results across all datasets and settings. On MNIST, all methods achieve high detection performance under the 5-shot setting, while MetaAdvDet lags behind the Baseline method under the 1-shot setting. Baseline and Baseline++ demonstrate poor detection results on the complex CIFAR-10 and ImageNet datasets, with detection rates hovering around 50%, equivalent to a random detector. After transitioning from the 1-shot to the 5-shot setting, MetaAdvDet achieved a substantial improvement in detection performance by over 10%, slightly surpassing PACA on ImageNet, but significantly lower than both PACA and PAD on CIFAR-10.

The AUC results for the detection of the five attacks in the cross-adversary benchmark and the two new adversarial attacks AA, CAA are shown in Figs. 3, 4 and 5. The results in Fig. 3 demonstrate that PAD outperforms all other methods on MNIST. Additionally, MetaAdvDet exhibits a significant gap with the other methods in detecting STA and AA.

The detection results on CIFAR-10 are presented in Fig. 4. PAD surpasses other methods in detecting the five attacks in the cross-adversary benchmark, though its performance on AA and CAA-L_∞ is weaker than Baseline++, MetaAdvDet, and PACA. This weaker performance on L_∞ attacks is likely due to the detector's awareness of the attacks, suggesting that the three detection methods may be overfitting. Additionally, PAD outperforms MetaAdvDet in detecting CAA-L_2 attacks, with a gap of around 5% compared to PACA.

Table 3 Detection results (%) of various methods in the cross-adversary benchmark in [57]

Dataset	Method	F1 (%)		Accuracy (%)		AUC (%)	
		1-Shot	5-Shot	1-Shot	5-Shot	1-Shot	5-Shot
MNIST	Baseline	97.9	98.6	97.8	98.5	99.4	99.8
	Baseline++	91.8	94.7	92.9	95.4	98.2	99.3
	MetaAdvDet	92.9	98.1	93.1	98.1	95.7	99.8
	PACA	98.3	99.3	98.4	99.3	99.7	99.9
	PAD(ours)	**99.6**	**99.9**	**99.6**	**99.9**	**99.7**	**100.0**
CIFAR-10	Baseline	50.0	51.4	52.9	55.5	55.0	58.8
	Baseline++	48.5	48.9	53.4	55.4	55.4	59.5
	MetaAdvDet	60.5	72.2	60.3	71.9	67.0	78.6
	PACA	74.1	79.0	75.3	79.0	82.1	85.9
	PAD(ours)	**85.3**	**88.9**	**85.2**	**88.9**	**88.7**	**92.8**
ImageNet	Baseline	54.0	61.3	57.3	61.6	61.3	67.8
	Baseline++	43.1	45.2	52.9	53.5	58.4	60.3
	MetaAdvDet	57.2	76.8	61.7	77.0	70.8	84.5
	PACA	66.8	76.0	65.0	75.4	69.4	81.2
	PAD(ours)	**79.1**	**88.4**	**79.1**	**88.4**	**83.2**	**93.4**

The best results are indicated in bold

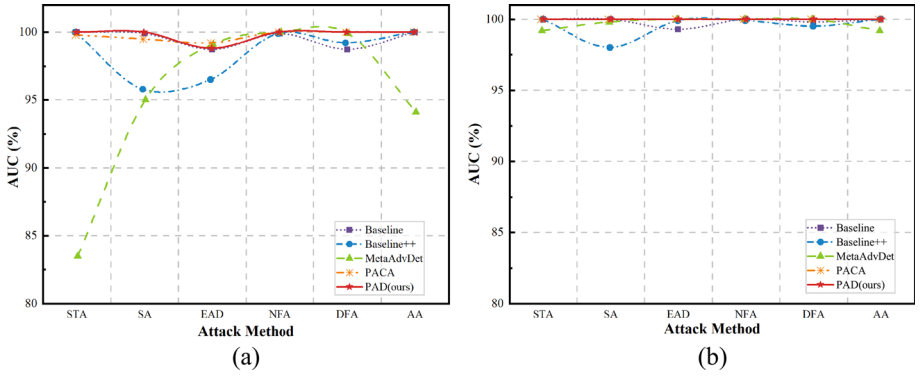

Fig. 3 AUC (%) score of unknown adversaries from the cross-adversary benchmark and novel adversaries under 1-shot and 5-shot settings on MNIST. **a** 1-shot; **b** 5-shot

Figure 5 presents the detection results on ImageNet. PAD demonstrates inferior performance compared to MetaAdvDet in detecting SA; however, it surpasses all other methods in detecting a variety of unknown and new attacks. Although PACA outperforms state-of-the-art detection approaches [14] in detecting EAD with few-perturbation and low confidence levels, PAD and MetaAdvDet are superior in the few-shot detection task presented in this study, with PAD exceeding PACA by over 20%.

The results from three datasets reveal that the presented method enhances robust generalization, accurately characterizes the adversarial subspace [61], and excels in detecting unknown attacks.

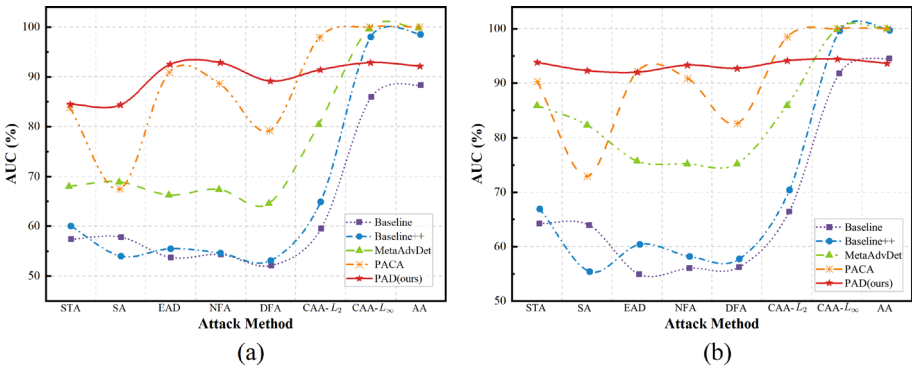

Fig. 4 AUC (%) score of unknown adversaries from the cross-adversary benchmark and novel adversaries under 1-shot and 5-shot settings on CIFAR-10. **a** 1-shot; **b** 5-shot

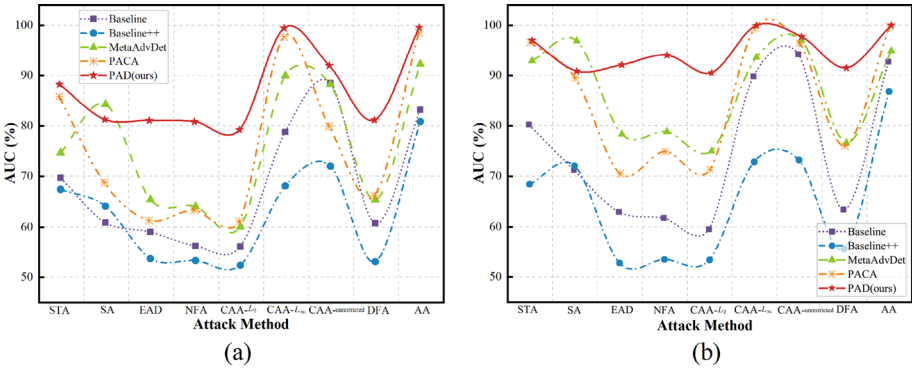

Fig. 5 AUC (%) score of unknown adversaries from the cross-adversary benchmark and novel adversaries under 1-shot and 5-shot settings on ImageNet. **a** 1-shot; **b** 5-shot

4.4 Performance Under the Adaptive White-Box Attack

A targeted white-box C&W attack is executed on a set of tasks, constructed using the methodology described in Sect. 3.3. These tasks involve random samples from the C&W adversarial example dataset combined with normal test samples. During the attack against each task, normal samples from the query set serve as initial inputs, with target labels differing from both the ground truth and the target model's predicted label.

Samples successfully attacked against the combined model are utilized as adversarial examples for task construction, with the detection model fine-tuned on each task to detect these samples. The detection results are presented in Table 4. The table illustrates that the attacked PAD model, when provided with a small number of labeled attacked adversarial examples, improves detection performance across all datasets, achieving optimal detection results.

Table 4 Detection results (%) of various methods under the adaptive white-box attack scenario

Dataset	Method	F1 (%)		Accuracy (%)		AUC (%)	
		1-Shot	5-Shot	1-Shot	5-Shot	1-Shot	5-Shot
MNIST	Baseline	99.3	99.2	99.3	99.2	100.0	100.0
	Baseline++	70.9	82.0	71.4	78.3	81.1	90.0
	MetaAdvDet	99.5	99.8	99.6	99.8	100.0	100.0
	PACA	95.0	94.1	94.9	93.6	99.6	100.0
	PAD(ours)	**99.9**	**100.0**	**99.9**	**100.0**	**100.0**	**100.0**
CIFAR-10	Baseline	50.3	52.5	53.5	54.9	54.8	58.6
	Baseline++	47.1	48.5	52.2	53.6	53.2	56.7
	MetaAdvDet	60.9	69.9	61.4	69.9	68.2	76.6
	PACA	77.5	82.4	77.9	82.3	86.1	90.5
	PAD(ours)	**90.3**	**91.5**	**90.3**	**91.5**	**92.4**	**94.1**
ImageNet	Baseline	51.6	56.5	54.8	59.0	57.0	63.0
	Baseline++	50.2	54.0	52.8	54.2	57.5	60.6
	MetaAdvDet	57.4	70.9	58.1	71.0	64.9	78.3
	PACA	64.5	73.6	62.2	73.0	67.5	79.8
	PAD(ours)	**78.1**	**88.5**	**78.1**	**88.5**	**82.5**	**93.1**

The best results are indicated in bold

Table 5 Ablation study results of PAD modules in the cross-adversary benchmark on CIFAR-10 and ImageNet

Modules			CIFAR-10 (F1 (%))		ImageNet (F1 (%))	
Network's first layer	CL2N	MAP	1-Shot	5-Shot	1-Shot	5-Shot
(3×3 conv)×3+avgpool	×	×	66.8	61.8	65.7	82.1
(3×3 conv)×3	×	×	53.1	64.5	68.0	81.0
(7×7 conv)×1+avgpool	×	×	75.0	72.2	67.0	85.2
(7×7 conv)×1	×	×	80.8	87.4	71.6	85.3
(7×7 conv)×1	✓	×	84.2	88.3	76.9	87.2
(7×7 conv)×1	×	✓	85.2	88.5	78.2	87.5
(7×7 conv)×1	✓	✓	**85.3**	**88.9**	**79.1**	**88.4**

Best results are indicated in bold

4.5 Ablation Study

To assess the effectiveness of each PAD module, ablation experiments are conducted on CIFAR-10 and ImageNet using F1 scores (Table 5) under the cross-adversary benchmark. The table illustrates the impact of the first convolutional layer's structure, CL2N, and MAP usage. One convolution with a kernel size of 7×7 in the first layer outperforms three 3×3 convolutions, regardless of average pooling. Notably, removing average pooling significantly improves CIFAR-10 detection results, indicating larger kernel sizes and unpooled structures benefit the detector's generalization. MAP enhances the detector's performance under the 1-shot setting, with CL2N providing a slight additional improvement.

In Table 6, we present a comparative analysis of the impacts of various class-center calculation methods on the CIFAR-10 and ImageNet datasets. The NCM approach directly

Table 6 Ablation study results with different class-center calculation methods in the cross-adversary benchmark on CIFAR-10 and ImageNet

Class-center calculation methods	CIFAR-10 (F1 (%))		ImageNet (F1 (%))	
	1-Shot	5-Shot	1-Shot	5-Shot
NCM	84.2	88.3	76.9	87.2
K-means	84.7	88.4	78.2	88.1
MAP	**85.3**	**88.9**	**79.1**	**88.4**

Best results are indicated in bold

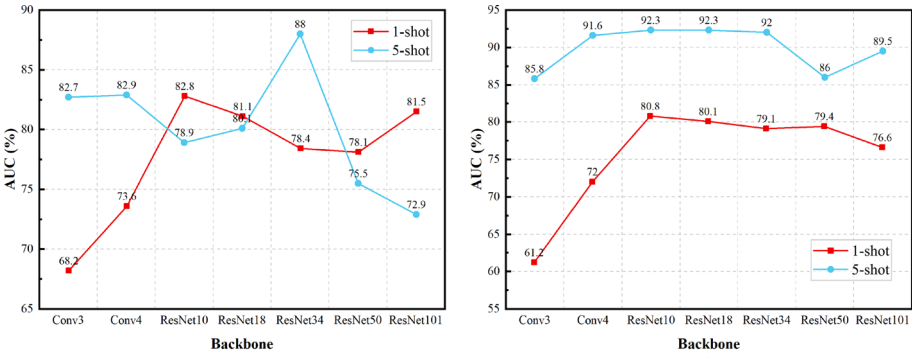

Fig. 6 Ablation study results of backbone networks with different depths. **a** CIFAR-10; **b** ImageNet

leverages support set samples for the computation of class centers. Conversely, the K-means method employs the assignment probability computation method described in Eqs. 5–7. This method initializes the class centers using the support set and subsequently updates them through an iterative process resembling the expectation–maximization (EM) algorithm. On the other hand, the MAP method utilizes Sinkhorn's algorithm to compute assignment probabilities and updates class centers through Eqs. 12 and 13. An examination of the table reveals that the MAP algorithm consistently yields the most optimal detection performance.

To investigate the impact of backbone network depth on detection performance, we incrementally increase the feature backbone depth from Conv-3 to Conv-4, ResNet-10, 18, 34, 50, and 101. AUC scores for CIFAR-10 and ImageNet are presented in Fig. 6. The detection performance exhibits a clear trend with increasing network depth, where it plateaus at ResNet-10 and subsequently declines, except in the 5-shot CIFAR-10 setting. Experiments reveal that deeper networks in MetaAdvDet do not improve performance but increase training costs; thus, they are omitted. PAD is suitable for deeper structures, and ResNet-10 outperforms other architectures across various datasets and settings.

4.6 Time Cost

Regarding the complexity of implementation, we focus on the time costs involved in three phases: the preparation of adversarial examples, model training, and model inference. For the inference phase, our method consists of two primary processes: feature extraction and MAP. The complexity of the feature extraction is equivalent to that of a standard CNN. In

contrast, the complexity of the MAP process is closely related to the Sinkhorn algorithm and depends on both the feature dimension and the number of samples. All timing measurements were conducted using a single Nvidia Tesla V-100 GPU. Given that the principal objective of our research is the enhancement of detection performance, we did not engage in comparative analyses regarding the time consumption of our method against other approaches.

1. Adversarial Examples Generation Time: The generation of adversarial example datasets is generally completed within a few hours. Exceptions include EAD, JSMA, and LBFGS types due to their higher time demands. We employed identical parameters for EAD and JSMA as those in the MetaAdvDet source code for consistency. To reduce computational demands, we adjusted only the attack strength for generating LBFGS adversarial examples on ImageNet. On CIFAR-10, adversarial examples took 20.5 h (EAD), 15.5 h (JSMA), and 57.3 h (LBFGS) to generate. On ImageNet, these times were 89.5 h (EAD), 32.2 h (JSMA), and 22.7 h (LBFGS).
2. Detector Training Time: In the meta-training stage, each task in the 1-shot setting includes 72 samples from both support and query sets, increasing to 80 samples in the 5-shot setting. The training times per epoch for MNIST, CIFAR-10, and ImageNet adversarial example datasets were 5.41, 7.32, and 74.43 min (1-shot) and 6.08, 7.51, and 81.64 min (5-shot), respectively.
3. Inference Time: At the meta-testing stage, each task consists of 32 samples (1-shot) and 40 samples (5-shot). Inference times per task for MNIST, CIFAR-10, and ImageNet were 9.22 ms, 9.47 ms, and 10.14 ms (1-shot) and 23.69 ms, 24.92 ms, and 27.63 ms (5-shot), respectively. The MAP algorithm primarily dictates the inference time.

5 Conclusions, Limitations, and Future Work

While much work focuses on detecting known adversarial examples by treating them as noise, this paper considers adversarial examples as features, proposing an end-to-end detection method called PAD to enhance the detector's generalization capability across various scenarios. PAD employs a convolution with a larger kernel size and omits the pooling module in the first layer, maximizing feature extraction across varying resolutions. The prototypical network is trained on a set of tasks containing known adversarial examples. After applying the CL2N feature transformation, the features of the support set, which come from the unknown adversarial example detection tasks, are used as initial class centers. The optimal-transport inspired MAP algorithm is then used to update the class centers and calculate probabilities, significantly improving few-shot detection performance. Extensive comparative experiments demonstrate PAD's improved generalization to unknown attacks and robustness against adaptive white-box attacks, given a limited number of labeled adversarial examples.

During meta-testing stage, our method detects adversarial examples through feature extraction and the MAP process. Feature extraction, akin to an ordinary DNN, requires one forward propagation. The MAP process, more time-intensive, computes the optimal match, impacting inference efficiency. The MAP method assumes a uniform class prior, which may hinder performance on class-imbalanced tasks, as shown in few-shot learning experiments [103]. Furthermore, our adversarial example detection tasks, containing only one adversarial type per task, may struggle when multiple types are mixed in a single task, potentially affecting class center computation and detection effectiveness. The proposed method, which depends exclusively on image information, demonstrates diminished effectiveness against known attack types, such as L-infinity. In contrast, the PACA method, leveraging both image

data and confidence scores from the target classifier, exhibits superior performance in detecting known adversarial examples on CIFAR-10 dataset.

Future research will explore incorporating additional information beyond the image to enhance the detector's generalizability, without compromising known adversarial example detection. For the purposes of consistency in benchmark testing, adversarial examples tailored to CNN architectures were employed. The rising prominence of transformer architectures in robustness research presents an intriguing direction, especially considering the distinct perturbations against transformers compared to CNNs [79]. By integrating the transformer architecture into our detection backbone, we anticipate improvements in feature extraction capabilities and the provision of adaptive defenses against white-box attacks. In addition, it is important for the detector to be able to detect adversarial examples across different architectures. This requires using adversarial examples generated against one architecture for meta-training and those generated against another architecture for meta-testing.

Acknowledgements This work is supported by the National Natural Science Foundation of China (No. 62172434), Natural Science Foundation of Henan (No. 232300421099), and China Postdoctoral Science Foundation (No. 2021T140531).

Author Contributions All authors contributed to the study's conception and design. WL served as the first author, actively contributed to the study's conception, design, and execution. WZ contributed to data collection, analysis, and figure creation, collaborating closely with the first author on the manuscript. KY participated in study conception and design, providing supervision and feedback. YC conducted manuscript review, ensuring its quality and rigor. KG contributed to data collection and manuscript review, while JW reviewed and edited the manuscript. All authors reviewed and approved the final manuscript.

Data Availability The datasets and code used during this study are available upon reasonable request to the authors.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Consent to participate All individual participants in this study provided informed consent to participate voluntarily.

Consent for publication The authors confirm that participants have granted informed consent for the publication of this manuscript.

Ethical approval This article does not involve studies with human participants or animals conducted by any of the authors, and therefore, ethical considerations and informed consent for data usage are not applicable.

References

1. Akhtar N, Mian A (2018) Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access 6:14410–14430

2. Aldahdooh A, Hamidouche W, Fezza SA et al (2022) Adversarial example detection for DNN models: a review and experimental comparison. Artif Intell Rev 55(6):4403–4462
3. Aldahdooh A, Hamidouche W, Déforges O (2023) Revisiting model's uncertainty and confidences for adversarial example detection. Appl Intell 53(1):509–531
4. Andriushchenko M, Croce F, Flammarion N, et al (2020) Square attack: a query-efficient black-box adversarial attack via random search. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII. Springer, pp 484–501
5. Antoniou A, Edwards H, Storkey A (2018) How to train your MAML. arXiv:1810.09502
6. Athalye A, Carlini N, Wagner D (2018) Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: International conference on machine learning, PMLR, pp 274–283
7. Bhattad A, Chong MJ, Liang K, et al (2019) Unrestricted adversarial examples via semantic manipulation. arXiv:1904.06347
8. Bojarski M, Del Testa D, Dworakowski D, et al (2016) End to end learning for self-driving cars. arXiv:1604.07316
9. Brown TB, Carlini N, Zhang C, et al (2018) Unrestricted adversarial examples. arXiv:1809.08352
10. Buckman J, Roy A, Raffel C, et al (2018) Thermometer encoding: one hot way to resist adversarial examples. In: International conference on learning representations
11. Carlini N, Wagner D (2017) Adversarial examples are not easily detected: bypassing ten detection methods. In: Proceedings of the 10th ACM workshop on artificial intelligence and security, pp 3–14
12. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 IEEE symposium on security and privacy (SP). IEEE, pp 39–57
13. Carrara F, Becarelli R, Caldelli R, et al (2018) Adversarial examples detection in features distance spaces. In: Proceedings of the European conference on computer vision (ECCV) workshops, pp 313–327
14. Chen K, Chen Y, Zhou H et al (2021) Adversarial examples detection beyond image space. In: ICASSP 2021–2021 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, pp 3850–3854
15. Chen PY, Sharma Y, Zhang H, et al (2018) EAD: Elastic-net attacks to deep neural networks via adversarial examples. In: Proceedings of the AAAI conference on artificial intelligence
16. Chen WY, Liu YC, Kira Z, et al (2019) A closer look at few-shot classification. arXiv:1904.04232
17. Cohen G, Sapiro G, Giryes R (2020) Detecting adversarial samples using influence functions and nearest neighbors. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 14453–14462
18. Croce F, Hein M (2020) Minimally distorted adversarial examples with a fast adaptive boundary attack. In: International conference on machine learning, PMLR, pp 2196–2205
19. Croce F, Hein M (2020) Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: International conference on machine learning, PMLR, pp 2206–2216
20. Cuturi M (2013) Sinkhorn distances: lightspeed computation of optimal transport. In: Advances in neural information processing systems, vol 26
21. Deng J, Dong W, Socher R, et al (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE, pp 248–255
22. Ding X, Zhang X, Han J, et al (2022) Scaling up your kernels to 31x31: revisiting large kernel design in CNNS. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11963–11975
23. Dong Y, Liao F, Pang T, et al (2018) Boosting adversarial attacks with momentum. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 9185–9193
24. Engstrom L, Tran B, Tsipras D, et al (2019) Exploring the landscape of spatial robustness. In: International conference on machine learning, PMLR, pp 1802–1811
25. Eniser HF, Christakis M, Wüstholz V (2020) Raid: randomized adversarial-input detection for neural networks. arXiv:2002.02776
26. Feinman R, Curtin RR, Shintre S, et al (2017) Detecting adversarial samples from artifacts. arXiv:1703.00410
27. 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). IEEE, pp 1–8
28. Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th international conference on machine learning, PMLR, pp 1126–1135
29. Freitas S, Chen ST, Wang ZJ, et al (2020) Unmask: Adversarial detection and defense through robust feature alignment. In: 2020 IEEE international conference on big data (big data). IEEE, pp 1081–1088

30. Goodfellow I, Qin Y, Berthelot D (2019) Evaluation methodology for attacks against confidence thresholding models. https://openreview.net/forum?id=H1g0piA9tQ
31. Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. arXiv:1412.6572
32. Grosse K, Manoharan P, Papernot N, et al (2017) On the (statistical) detection of adversarial examples. arXiv:1702.06280
33. He Y, Meng G, Chen K et al (2020) Towards security threats of deep learning systems: a survey. IEEE Trans Softw Eng 48(5):1743–1770
34. Hendrycks D, Gimpel K (2016) A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv:1610.02136
35. Hendrycks D, Gimpel K (2016) Early methods for detecting adversarial images. arXiv:1608.00530
36. Hosseini H, Poovendran R (2018) Semantic adversarial examples. In: 2018 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), pp 1614–1619
37. Hosseini H, Xiao B, Jaiswal M, et al (2017) On the limitation of convolutional neural networks in recognizing negative images. In: 2017 16th IEEE international conference on machine learning and applications (ICMLA). IEEE, pp 352–358
38. Howard J (2020) A smaller subset of 10 easily classified classes from imagenet, and a little more French. https://github.com/fastai/imagenette/
39. Hu Y, Gripon V, Pateux S (2021) Leveraging the feature distribution in transfer-based few-shot learning. In: Artificial neural networks and machine learning–ICANN 2021: 30th international conference on artificial neural networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30. Springer, pp 487–499
40. Huang G, Larochelle H, Lacoste-Julien S (2019) Are few-shot learning benchmarks too simple? arXiv:1902.08605
41. Ilyas A, Santurkar S, Tsipras D, et al (2019) Adversarial examples are not bugs, they are features. In: Advances in neural information processing systems 32
42. 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
43. Joshi A, Mukherjee A, Sarkar S, et al (2019) Semantic adversarial attacks: parametric transformations that fool deep classifiers. In: 2019 IEEE/CVF international conference on computer vision (ICCV), pp 4772–4782
44. Kherchouche A, Fezza SA, Hamidouche W, et al (2020) Natural scene statistics for detecting adversarial examples in deep neural networks. In: 2020 IEEE 22nd international workshop on multimedia signal processing (MMSP). IEEE, pp 1–6
45. Kong Z, Guo J, Li A, et al (2020) Physgan: generating physical-world-resilient adversarial examples for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 14254–14263
46. Krizhevsky A (2009) Learning multiple layers of features from tiny images. University of Toronto, Toronto
47. Kurakin A, Goodfellow IJ, Bengio S (2018) Adversarial examples in the physical world. In: Artificial intelligence safety and security. Chapman and Hall, Boca Raton, pp 99–112
48. LeCun Y, Bottou L, Bengio Y et al (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
49. Lee K, Lee K, Lee H, et al (2018) A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Advances in neural information processing systems 31
50. 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
51. Li Y, Li Y, Xiao B (2022) A physical-world adversarial attack against 3D face recognition. arXiv:2205.13412
52. Liang B, Li H, Su M et al (2018) Detecting adversarial image examples in deep neural networks with adaptive noise reduction. IEEE Trans Dependable Secure Comput 18(1):72–85
53. Lin WA, Lau CP, Levine A et al (2020) Dual manifold adversarial robustness: defense against lp and non-lp adversarial attacks. Adv Neural Inf Process Syst 33:3487–3498
54. Liu J, Song L, Qin Y (2020) Prototype rectification for few-shot learning. In: Computer vision—ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. Springer, pp 741–756
55. Liu Z, Mao H, Wu CY, et al (2022) A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11976–11986

56. Lu J, Issaranon T, Forsyth D (2017) Safetynet: detecting and rejecting adversarial examples robustly. In: 2017 IEEE international conference on computer vision (ICCV), pp 446–454
57. Ma C, Zhao C, Shi H, et al (2019) MetaAdvDet: towards robust detection of evolving adversarial attacks. In: Proceedings of the 27th ACM international conference on multimedia, pp 692–701
58. Ma S, Liu Y, Tao G, et al (2019) Nic: detecting adversarial samples with neural network invariant checking. In: 26th annual network and distributed system security symposium (NDSS 2019), Internet Soc
59. Ma X, Li B, Wang Y, et al (2018) Characterizing adversarial subspaces using local intrinsic dimensionality. arXiv:1801.02613
60. Madry A, Makelov A, Schmidt L, et al (2017) Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083
61. Mao X, Chen Y, Li Y et al (2020) Learning to characterize adversarial subspaces. Ib: ICASSP 2020–2020 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, pp 2438–2442
62. Mao X, Chen Y, Wang S, et al (2021) Composite adversarial attacks. In: Proceedings of the AAAI conference on artificial intelligence, pp 8884–8892
63. Meng D, Chen H (2017) Magnet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 135–147
64. Metzen JH, Genewein T, Fischer V, et al (2017) On detecting adversarial perturbations. arXiv:1702.04267
65. Miyato T, Maeda S, Koyama M, et al (2015) Distributional smoothing with virtual adversarial training. arXiv:1507.00677
66. Moosavi-Dezfooli SM, 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
67. Nicolae MI, Sinn M, Tran MN, et al (2018) Adversarial robustness toolbox v1. 0.0. arXiv:1807.01069
68. Pang T, Du C, Dong Y, et al (2018) Towards robust detection of adversarial examples. In: Advances in neural information processing systems 31
69. Papernot N, Faghri F, Carlini N, et al (2016) Technical report on the cleverhans v2. 1.0 adversarial examples library. arXiv:1610.00768
70. Papernot N, McDaniel P, Jha S, et al (2016) The limitations of deep learning in adversarial settings. In: 2016 IEEE European symposium on security and privacy (EuroS&P). IEEE, pp 372–387
71. Papernot N, McDaniel P, Wu X, et al (2016) Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE symposium on security and privacy (SP). IEEE, pp 582–597
72. Pertigkiozoglou S, Maragos P (2018) Detecting adversarial examples in convolutional neural networks. arXiv:1812.03303
73. Qiu H, Xiao C, Yang L, et al (2020) Semanticadv: generating adversarial examples via attribute-conditioned image editing. In: Computer vision–ECCV 2020: 16th European vonference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16. Springer, pp 19–37
74. Raghu A, Raghu M, Bengio S, et al (2019) Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. arXiv:1909.09157
75. Ren M, Triantafillou E, Ravi S, et al (2018) Meta-learning for semi-supervised few-shot classification. In: International conference on learning representations
76. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 815–823
77. Shalam D, Korman S (2022) The self-optimal-transport feature transform. arXiv:2204.03065
78. Shamsabadi AS, Sanchez-Matilla R, Cavallaro A (2020) Colorfool: semantic adversarial colorization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1151–1160
79. Shao R, Shi Z, Yi J, et al (2022) On the adversarial robustness of vision transformers. Trans Mach Learn Res
80. Sharif M, Bauer L, Reiter MK (2018) On the suitability of lp-norms for creating and preventing adversarial examples. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 1605–1613
81. Sharif M, Bhagavatula S, Bauer L et al (2019) A general framework for adversarial examples with objectives. ACM Trans Priv Secur (TOPS) 22(3):1–30
82. Smith L, Gal Y (2018) Understanding measures of uncertainty for adversarial example detection. arXiv:1803.08533
83. Snell J, Swersky K, Zemel R (2017) Prototypical networks for few-shot learning. In: Advances in neural information processing systems 30

84. Song Y, Kim T, Nowozin S, et al (2017) Pixeldefend: leveraging generative models to understand and defend against adversarial examples. arXiv:1710.10766
85. Song Y, Shu R, Kushman N, et al (2018) Constructing unrestricted adversarial examples with generative models. In: Advances in neural information processing systems 31
86. Sotgiu A, Demontis A, Melis M et al (2020) Deep neural rejection against adversarial examples. EURASIP J Inf Secur 2020:1–10
87. Szegedy C, Zaremba W, Sutskever I, et al (2013) Intriguing properties of neural networks. arXiv:1312.6199
88. Taigman Y, Yang M, Ranzato M, et al (2014) Deepface: closing the gap to human-level performance in face verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1701–1708
89. Tramer F, Boneh D (2019) Adversarial training and robustness for multiple perturbations. In: Advances in neural information processing systems 32
90. Tsipras D, Santurkar S, Engstrom L, et al (2018) Robustness may be at odds with accuracy. arXiv:1805.12152
91. Uesato J, O'donoghue B, Kohli P, et al (2018) Adversarial risk and the dangers of evaluating against weak attacks. In: International conference on machine learning, PMLR, pp 5025–5034
92. Wang J, Zhao J, Yin Q et al (2021) Smsnet: a new deep convolutional neural network model for adversarial example detection. IEEE Trans Multim 24:230–244
93. Wang Y, Chao WL, Weinberger KQ, et al (2019) Simpleshot: revisiting nearest-neighbor classification for few-shot learning. arXiv:1911.04623
94. Xiao C, Zhu JY, Li B, et al (2018) Spatially transformed adversarial examples. arXiv:1801.02612
95. Xiao Z, Gao X, Fu C, et al (2021) Improving transferability of adversarial patches on face recognition with generative models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11845–11854
96. Xie C, Wang J, Zhang Z, et al (2017) Mitigating adversarial effects through randomization. arXiv:1711.01991
97. Xu W, Evans D, Qi Y (2017) Feature squeezing: detecting adversarial examples in deep neural networks. arXiv:1704.01155
98. Yuan X, He P, Zhu Q et al (2019) Adversarial examples: attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst 30(9):2805–2824
99. Zhang H, Yu Y, Jiao J, et al (2019) Theoretically principled trade-off between robustness and accuracy. In: International conference on machine learning, PMLR, pp 7472–7482
100. Zhang H, Cao Z, Yan Z, et al (2021) Sill-net: feature augmentation with separated illumination representation. arXiv:2102.03539
101. Zhang J, Lou Y, Wang J et al (2021) Evaluating adversarial attacks on driving safety in vision-based autonomous vehicles. IEEE Internet Things J 9(5):3443–3456
102. Zheng Z, Hong P (2018) Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In: Advances in neural information processing systems 31
103. Zhu H, Koniusz P (2023) Transductive few-shot learning with prototype-based label propagation by iterative graph refinement. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 23996–24006
104. Zuo F, Zeng Q (2021) Exploiting the sensitivity of l2 adversarial examples to erase-and-restore. In: Proceedings of the 2021 ACM Asia conference on computer and communications security, pp 40–51