BESA: Boosting Encoder Stealing Attack with Perturbation Recovery

Xuhao Ren, Haotian Liang, Yajie Wang, Chuan Zhang, *Member, IEEE,* Zehui Xiong, *Senior Member, IEEE,* Liehuang Zhu, *Senior Member, IEEE*

***Abstract*—To boost the encoder stealing attack under the perturbation-based defense that hinders the attack performance, we propose a boosting encoder stealing attack with perturbation recovery named BESA. It aims to overcome perturbation-based defenses. The core of BESA consists of two modules: perturbation detection and perturbation recovery, which can be combined with canonical encoder stealing attacks. The perturbation detection module utilizes the feature vectors obtained from the target encoder to infer the defense mechanism employed by the service provider. Once the defense mechanism is detected, the perturbation recovery module leverages the well-designed generative model to restore a clean feature vector from the perturbed one. Through extensive evaluations based on various datasets, we demonstrate that BESA significantly enhances the surrogate encoder accuracy of existing encoder stealing attacks by up to 24.63% when facing state-of-the-art defenses and combinations of multiple defenses.**

***Index Terms*—Encoder stealing attack, perturbation recovery, perturbation detection, generative model.**

I. Introduction

Pre-trained encoders are extensively utilized across various domains in real-world scenarios [1]. However, training well-performing pre-trained encoders is a time-consuming, resource-intensive, and costly process [2]. Hence, encoder owners are highly motivated to safeguard the privacy of their pre-trained encoders.

Unfortunately, recent works have shown that pre-trained encoders are susceptible to encoder stealing attacks [3]. These attacks allow an attacker to create a surrogate encoder that closely mimics the functionality of a targeted encoder by simply querying it through the APIs. The consequences of such attacks can be quite severe. On the one hand, many service providers, such as OpenAI, Google, and Meta, offer cloud-based Encoder as a Service (EaaS) solutions to monetize their pre-trained encoders [4]. Users compensate these service providers for accessing the encoder through Application Programming Interfaces (APIs). However, an attacker can acquire the cloud-based encoder at a significantly reduced expense compared to the investment in data collection and training, which not only infringes on intellectual property but also results in financial setbacks for the original service provider [5]–[7]. On the other hand, encoder stealing attack can also act as a launchpad for various types of attacks like adversarial sample attacks [8], membership inference attacks [9], [10], and backdoor injection attacks [11]–[13]. For example, certain adversarial example attacks rely on having access to the gradient of the target encoder, which is inaccessible in black-box scenarios. In such cases, an adversary can create a surrogate encoder through model encoder stealing attacks and generate adversarial examples using the white-box surrogate encoder. Additionally, researchers have demonstrated that a surrogate encoder obtained through encoder stealing attacks can enable membership inference attacks as well as more damaging backdoor injection attacks.

Extensive research has been conducted to mitigate encoder stealing attacks in various ways. Common defense strategies include detection methods [14], [15], watermarking techniques [16]–[18], and perturbation-based approaches [19], [20]. Among these methods, the perturbation-based approaches have been widely adopted by many EaaS providers based on their performance in degrading existing encoder stealing attacks. Based on their good performance in real-world scenarios, these defense strategies have been adopted by real-world EaaS providers to safeguard against encoder stealing attacks. For example, Liu et al. [20] have demonstrated that using perturbation-based defense mechanisms can cause a decrease in the accuracy of the substitute encoder on downstream classification tasks, from 78.12% to 42.07%. Therefore, it is necessary for the attackers to explore practical ways to bypass or penetrate such defense methods for more effective stealing attacks.

To the best of our knowledge, previous works have widely ignored the possibility of performing encoder stealing attacks against defended target encoders. Motivated by this research gap, we propose a boosting encoder stealing attack with perturbation recovery, called BESA. The core idea behind BESA is to detect and recover perturbed feature vectors, which is illustrated in Fig. 1. To be specific, canonical encoder stealing attacks typically involve three steps. Initially, the surrogate encoder is either initialized as empty or pre-trained. Next,

This work was financially supported by the National Natural Science Foundation of China (Grant Nos. 62472032, 62202051, and 62232002); the Open Project Funding of Key Laboratory of Mobile Application Innovation and Governance Technology, Ministry of Industry and Information Technology (Grant No. 2023IFS080601-K); the Beijing Institute of Technology Research Fund Program for Young Scholars; the Young Elite Scientists Sponsorship Program by CAST (Grant No. 2023QNRC001); Xiaomi Research Fund for Young Scholars, and the Fundamental Research Funds for the Central Universities (Grant No. 2024CX06034); and the BIT Research and Innovation Promoting Project (Grant No. 2024YCXZ022).

Xuhao Ren, Haotian Liang, Yajie Wang, Chuan Zhang, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: xuhaor@bit.edu.cn; haotianl@bit.edu.cn; wangyajie0312@foxmail.com; chuanz@bit.edu.cn; liehuangz@bit.edu.cn).

Zehui Xiong is with Singapore University of Technology and Design, Singapore (email: zehui_xiong@sutd.edu.sg).

Corresponding author: Chuan Zhang.

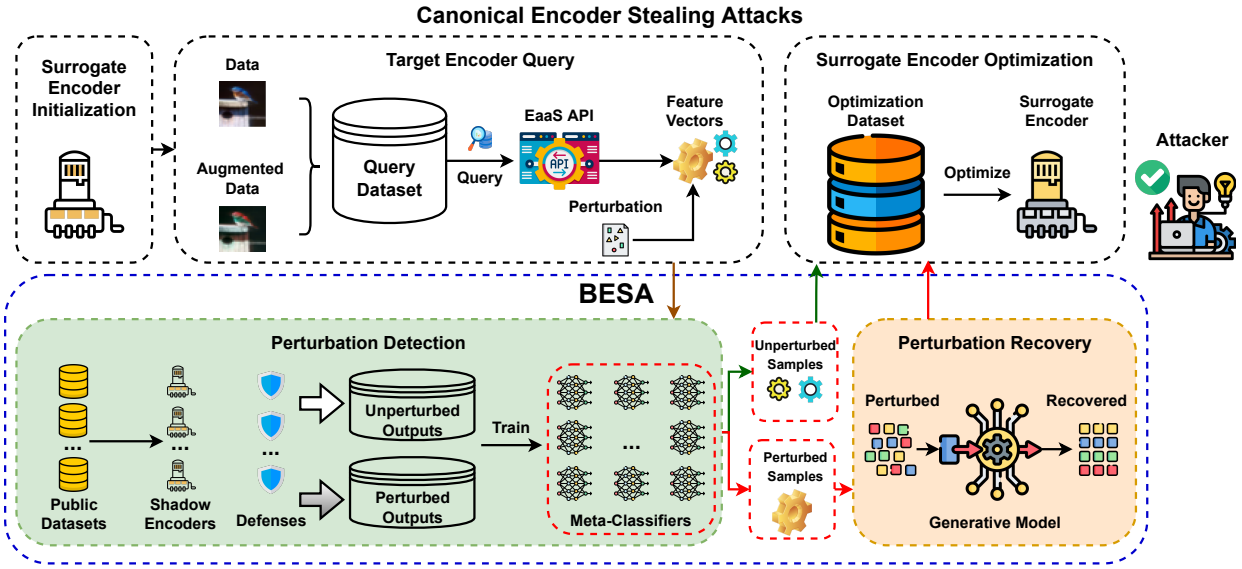

Fig. 1. The architecture of BESA.

original or augmented samples [20] are chosen to query the target encoder for feature vectors. Subsequently, the surrogate encoder is optimized using these feature vectors. This query and optimization process is iterated until optimal performance is achieved. However, if the feature vectors are perturbed by the service provider, the optimization efforts may prove futile. As shown in Section V, the performances of existing works are seriously degraded if the feature vectors are perturbed by the service provider. To solve this problem, we introduce two general modules in BESA, namely perturbation detection and perturbation recovery, which can be incorporated into canonical encoder stealing attacks before the optimization phase for the surrogate encoder.

To achieve our objective of boosting encoder stealing attacks, we encounter several challenges as follows.

- **Detecting the adopted defense method is challenging.**

Generally, the defense method adopted by the service provider is often unknown. Although some detecting methods have been proposed recently [21], they mainly aim at logits with small sizes and simple distribution characteristics. However, existing methods fall short as the feature vectors from pre-trained encoders are always large and with complex distribution characteristics. To tackle this problem, we first train a collection of shadow encoders applied with multiple defenses. Here, the data augmentation technique is adopted to overcome the lack of enough public training data. Subsequently, we train a binary meta-classifier with multiple layers for each defense method using perturbed and unperturbed feature vectors. The trained meta-classifier can accurately predict if a particular feature vector has been altered by the respective defense technique. The results from the experiments in Section V-B2 demonstrate its ability to differentiate the defense method with an accuracy of over 99%.

- **Recovering perturbed feature vectors is challenging.**

Even if we can detect the defense method, the precise amount of noise added remains uncertain. Although generative models have been used for perturbation recovery [21], their architectures are designed for logits with small sizes and simple distribution. However, these models are inefficient and ineffective for feature vectors with large sizes and complex distribution. To tackle this challenge, we construct a generative model inspired by the MagNet [22] based on its outstanding performance of perturbation recovery on samples with large size and complex distribution. Next, we proceed to train the model using perturbed feature vectors as inputs and unperturbed ones as outputs. The experiments detailed in Section V-B1 illustrate that this method can enhance the accuracy of current attacks by as much as 24.63%.

The contributions of this paper are as follows.

- We have developed a boosting encoder stealing attack with perturbation recovery called BESA, illustrated in Fig. 1, to enhance the effectiveness of canonical attacks against perturbation-based defenses.
- We have devised algorithms for perturbation detection and recovery, enabling the identification of the defense methods and the recovery of perturbed feature vectors for the optimization of the surrogate encoder.
- We have conducted experiments across different defense methods on three state-of-the-art attacks to evaluate the performance of BESA. The results show that it can improve the accuracy of existing attacks by up to 24.63%.

The rest of the paper is structured as follows. Section II presents the related works. In Section III, we give the threat model of our scheme, including the object, capabilities, and limitations. We give the scheme details in Section IV, followed by experiments in Section V. Finally, we conclude this paper in Section VII.

II. Related Works

A. Model Stealing Attacks

Most model stealing attacks mainly focus on the classifiers instead of the emerging pre-trained encoders [23], [24]. For

instance, Tramer et al. [25] were the first to investigate model stealing attacks in supervised learning. They demonstrated the feasibility of extracting the functionality of high-performing machine learning models deployed online through APIs. Subsequently, extensive research has been conducted on various aspects [26]–[34]. For example, if the attacker manages to obtain similar or in-distribution data that resemble the surrogate dataset used for attacks, they can leverage data augmentation or active learning techniques by combining the datasets to query the API [35], [36]. Furthermore, Truong et al. [30] and Kariyappa et al. [29] have also explored the possibility of stealing the model in data-free [37]. However, this approach may prove insufficient in practical scenarios where the tasks involve significant commercial value, and the associated training dataset is considered highly confidential and hard to access [27], [28].

Numerous researchers have proposed data-free methods for extracting models to tackle this challenge [33]. In such scenarios, the adversary lacks any information about the dataset used to train the target models. Two recently developed techniques, namely DFME [30] and MAZE [29], have been specifically designed to extract the functionality of target models under this challenge. However, these studies impose significant query budgets on the adversary, rendering them impractical in real-world scenarios. Lin et al. [34] tackled this problem by integrating Generative Adversarial Networks (GAN) to make use of weak image priors. They also employed deep reinforcement learning methods to enhance the query efficiency of data-free model extraction attacks.

In the aspect of emerging pre-trained encoders, Liu et al. [20] introduced a pioneering attack named "StolenEncoder" with the objective of extracting the functionality of pre-trained encoders. Their method formulates the stealing attack as an optimization problem and utilizes the standard stochastic gradient descent paradigm to solve it. To optimize the query budget, the attack incorporates data augmentations to enhance its effectiveness. Additionally, a method called "Cont-Steal" is introduced by Sha et al. [3], which employs the concept of contrastive learning to effectively utilize the rich information in the feature vectors. However, existing works on encoder stealing attacks have disregarded the exploration of stealing pre-trained encoders under perturbation-based defensive approaches. In contrast, our work focuses on encoder stealing attacks against defended pre-trained target encoders, which is more applicable in real-world scenarios.

B. Defensive Approaches

When protecting the pre-trained encoders against encoder stealing attacks, the service provider encounters two opposing objectives: impeding malicious and enhancing benign queries. In other words, the service provider aims to hinder the attacker's efforts in carrying out encoder stealing attempts while improving the performance of legitimate queries. Generally, existing defensive approaches in defending against encoder stealing attacks can be classified into three main categories: detection, watermarking, and perturbation.

1) Detection-based Methods.: Detection-based methods focus on determining if a query sequence is malicious, without modifying the feature vectors. In cases where a query sequence is flagged as malicious, the service provider might opt to adjust the feature vector or refuse service to the user. For the detection-based method, Dubiński et al. [38] proposed an active defense method to counter encoder stealing attacks. This method prevents stealing in real-time, without compromising the quality of the representations for legitimate API users. Specifically, their defense approach is based on the observation that the representations provided to adversaries attempting to steal the encoder's functionality cover a significantly larger portion of the embedding space compared to representations of normal users using the encoder for a specific downstream task. However, how to accurately define the concept of an "anomalous query" still becomes challenging in practice.

2) Watermarking-based Methods.: Watermarking-based methods involve injecting a carefully designed watermark backdoor into the target encoder. This allows for effective transfer to the surrogate encoder of the attacker, aiding in copyright verification. For the watermarking-based method, Cong et al. [39] introduced the first robust watermarking for pre-trained encoders called SSLGuard. This algorithm serves as a defense against model extraction and other watermark recovery attacks like input noising, output perturbing, overwriting, model pruning, and fine-tuning. Additionally, Peng et al. [40] introduced EmbMarker, a backdoor-based watermarking scheme to protect pre-trained encoders in the large language aspect. However, existing watermarking-based methods still face serious threats to extraction-based attacks due to their functional-irrelevant characteristic. known for its ability to protect the copyright of pre-trained encoders through the use of Backdoor Watermark.

3) Perturbation-based Methods: The perturbation-based methods perturb the feature vectors of some or all queries. For the perturbation-based method, Liu et al. [20] first discuss perturbation-based methods for defending against their proposed *StolenEncoder* attack on pre-trained encoders. According to their categorization, there are three main approaches adopted in defending the pre-trained encoders. Firstly, the top-k features approach resets the contents that are not among the top k largest absolute values to 0. Secondly, the feature rounding approach rounded feature vectors. Lastly, the feature poisoning approach adds carefully crafted perturbations to the feature vector. Nevertheless, the trade-off between the security under perturbation-based defenses and the usability for regular users still warrants further discussion. According to their categorization, three main approaches have been adopted to defend the pre-trained encoders. Firstly, in the top-k features approach, the EaaS API resets the contents of features that are not among the top k largest absolute values to 0 before returning a feature vector to a customer. Secondly, in the feature rounding approach, the EaaS API returns rounded feature vectors to a customer. Lastly, the feature poisoning approach involves the EaaS API adding carefully crafted perturbations to a feature vector in order to manipulate the optimization of the surrogate encoder.

III. Threat Model

In this section, we establish the threat model based on the objective, capabilities, and the attack's limited knowledge.

A. Objective

In BESA, the attacker's goal is to create a surrogate encoder that imitates the actions of the target encoder, which is safeguarded by specific defense mechanisms. Additionally, the attacker aims to achieve this objective while working within restricted query budgets.

B. Capabilities

In BESA, we consider the attacker to possess three specific capabilities during the attack.

- **The attacker has access to publicly available datasets.** The attacker has the ability to gather datasets that are either consistent or inconsistent with the original training distribution of the target encoder. For instance, if the target encoder is pre-trained on the CIFAR-10 [41] dataset for downstream tasks, the CIFAR-100 dataset could be considered an in-distribution dataset, while the SVHN [42] dataset, which consists of Google Street View house numbers, could be considered an out-of-distribution dataset.
- **The attacker has the capability to make queries to the target encoder.** The attacker interacts with the target encoder by making queries through the EaaS API or other interfaces to retrieve the relevant query results. It is assumed that the query results consist of feature vectors, which is a commonly found feature in most EaaS systems. However, due to the limited query budget, the attacker is only able to make a restricted number of queries to the target encoder. In the case of BESA, it is assumed that service providers utilize a defense strategy based on perturbation to modify some or all of the feature vector outcomes while preserving accuracy or adhering to accuracy constraints.
- **The attacker has the ability to reconstruct the defensive strategy.** The attacker has the ability to reconstruct a set of defense strategies labeled as $f_{De} = \{f^1, f^2, \cdots, f^K\}$, where each $f^k,\ k \in \{1, 2, \cdots, K\}$ represents a specific defense tactic. It is assumed that the strategy employed by the service provider is included in this reconstructed set f_{De}. Importantly, we demonstrate the effectiveness of BESA through our experiments by showing its effectiveness even when the service provider adopts defense strategies that are not included in the attacker's reconstructed set.

C. Limitations

In BESA, we make assumptions about the attacker's limitations. First, the attacker lacks knowledge about the target encoder, including its architecture, parameters, and hyperparameters. Moreover, it cannot access the original pre-trained samples of the target encoder due to their unavailability, difficulty in obtaining them, or prohibitively high cost. Finally, it lacks knowledge of the specific defense strategies employed by the service provider. The service provider is capable of selectively altering a subset of feature vectors, for which the attacker lacks detailed information. The details are as follows.

- **No information about the Target Encoder.** The attacker lacks knowledge about the target encoder, including its architecture, parameters, and hyperparameters.
- **No Access to Original Pre-train Dataset.** The attacker cannot access the original pre-trained samples of the target encoder due to their unavailability, difficulty in obtaining them, or prohibitively high cost.
- **Inadequate Defense Strategy Knowledge.** The attacker lacks knowledge of the specific defense strategies employed by the service provider. The service provider is capable of selectively altering a subset of feature vectors, for which the attacker lacks detailed information.

IV. Detailed Construction

In contrast to traditional encoder stealing attacks, BESA is improved by incorporating two additional modules positioned between the target encoder query module and the surrogate encoder optimizing module. These additional modules are intended to identify and restore perturbation feature vectors, as demonstrated in Fig. 1. The first module, the perturbation detection module, utilizes meta-classifiers to recognize the defense mechanisms implemented through the service provider. Another module, the perturbation recovery module, focuses on recovering pristine feature vectors from the perturbed ones. These two modules can be seamlessly integrated into existing canonical encoder-stealing attack frameworks. This section provides a comprehensive overview of both the perturbation detection module and the perturbation recovery module and we present the detailed procedure of BESA in Algorithm 1.

A. Perturbation Detection

Our main objective is to identify the defense strategies employed by the service provider by leveraging the distinct characteristics of perturbed feature vectors associated with different defense strategies. To achieve this, we start by constructing a pool of unprotected pre-trained encoders. These encoders, denoted as $\{E_1, E_2, \cdots, E_M\}$, are trained on various publicly available datasets $\{D_1, D_2, \cdots, D_M\}$ using different architectures and contrastive learning algorithms. To construct more training data samples for shadow encoders with limited public datasets, the attacker can adopt the data augmentation technique utilized in various works [3], [20]. Note that $E_m,\ m \in [1, M]$ refers to a set of unprotected encoders pre-trained on D_m since we utilize different architectures and contrastive learning algorithms to train multiple models for each public dataset. Although it would be ideal for these shadow encoders to have architectures similar to those of the target encoder, this is not possible in black-box scenarios. [20]. Thus, we incorporate diversity into the designs of the shadow encoders to enhance overall generalization. Following this, we implement a range of defense strategies for these shadow encoders. With K defense techniques available and M groups of shadow encoders, we have the flexibility to assign

Algorithm 1 BESA: Boosting encoder stealing attack with perturbation recovery

Require: API of the target encoder $\mathcal{T}$, public datasets with data augmentation $D_1, D_2, \cdots D_M$, the series of defense tattics $f^1, f^2, \cdots f^K$.

Ensure: Trained surrogate encoder $\mathcal{S}$

// Preparation.
Train shadow encoders $E_1, E_2, \cdots E_M$ using the datasets $D_1, D_2, \cdots D_M$.
for $k \in [1, K]$ **do**
  Leverage f^k to $E_m,\ m \in [1, M]$.
  Use the inputs $(x_m, E_m^k(x_m)),\ x_m \in D_m,\ \forall m$ as positive samples and $(x_m, E_m(x_m)),\ \forall m$ as negative samples to train a binary meta-classifier B_k, where $E_m^k(x_m)$ is the output of encoder E_m under defense f^k.
  Train a generative model G_k using $E_m^k(x_m)$ as inputs and $E_m(x_m)$ as outputs for all m.
end for
// Surrogate encoder initialization.
Initialize a raw or a pre-trained surrogate encoder $\mathcal{S}$.
while The query budget is not exhausted or $\mathcal{S}$ is not converged as desired **do**
  // Target encoder query.
  Construct a query sample x_q.
  Use x_q to query the target encoder $\mathcal{T}$ and receive the feature vector $\mathcal{T}(x_q)$.
  // Perturbation detection.
  Input $(x_q, \mathcal{T}(x_q))$ into $B_1, B_2, \cdots B_K$.
  if There exist available prediction results **then**
    Choose the prediction result with the highest confidence score, which is represented as k^*.
    // Perturbation recovery.
    Input $\mathcal{T}(x_q)$ into G_{k^*} to get $\mathcal{T}(x_q) \leftarrow G_{k^*}(\mathcal{T}(x_q))$.
  end if
  // Surrogate encoder optimizing.
  Use x_q and the recovered $\mathcal{T}(x_q)$ to optimize the surrogate encoder $\mathcal{S}$.
end while

a particular defense approach to multiple shadow encoders or opt for employing multiple defense methods for a single group of shadow encoders. Our goal is to secure an ample number of pairs comprising protected and unprotected shadow encoders for each defense tactic.

Here, we define E_m^k as the protected version of E_m under the defense strategy f^k. $E_m^k(x_m)$ denotes the output feature vector, while $E_m(x_m)$ is the corresponding clean output from the same encoder without any defense. These notations help distinguish between perturbed and unperturbed features, which are used to train the meta-classifiers. We expect that $E_m(x_m),\ x_m \in D_m$ captures the characteristics of unperturbed feature vectors, while $E_m^k,\ x_m \in D_m$ embodies the characteristics of feature vectors using defense strategy f^k. Building on this observation, we train K binary meta-classifiers, $B_1, \cdots, B_K$, where B_k is responsible for detecting whether the k-th defense approach safeguards the feature vectors. Positive samples $(x_m, E_m^k(x_m)),\ \forall m$ (with the label is 1) and negative samples $(x_m, E_m(x_m)),\ \forall m$ (with the label is 0) are utilized in the training of the corresponding meta-classifier B_k. Since B_k is being trained with samples from various classification tasks, it is anticipated that the model will grasp the inherent distinctions in characteristics between modified and unaltered feature vectors resulting from the defense mechanism f^k. This enables the meta-classifiers to generalize to the target encoder, even when the training dataset of the target encoder is unknown to the adversary.

Trained meta-classifiers are employed to determine the defense tactics used by the target encoder. Set $x_q \in D_q$ represent a query sample from the query dataset, and $\mathcal{T}(x_q)$ represents the feature vector. The pair $(x_q, \mathcal{T}(x_q))$ is input into the meta-classifiers $B_1, \cdots B_K$ to obtain the corresponding prediction results. If all predictions are below a specified threshold (e.g., 0.5), the sample is unperturbed. However, if any prediction result exceeds the threshold T_h, we compare the confidence scores of these predictions and select the one with the highest confidence as the predicted defense tactic. Our experiments demonstrate that in the majority of cases, the meta-classifier associated with the actual defense strategy generates a confidence score exceeding 90%.

B. Perturbation Recovery

If the feature vector of x_q is predicted to be intact, we will optimize the surrogate encoder directly using this feature vector. However, if it is disrupted, we will retrieve the original feature vector from the altered one using the specified defense strategy. It is crucial to highlight that defense detection can be conducted at various intervals, including one-time, regular intervals, or for each query. In the most precise scenario, if the service provider alternates between different defenses randomly, the attacker can perform defense detection on every query outcome and subsequently restore each altered feature vector based on the detection outcomes.

To retrieve the feature vector safeguarded through a particular defense strategy f^k, we construct a generative model G_k and compute the perturbed feature vector $\mathcal{T}(x_q)$ as follows:

$$\mathcal{T}(x_q) \leftarrow G_k(\mathcal{T}(x_q)) \tag{1}$$

This model takes $\mathcal{T}(x_q)$ as input and produces the corresponding unperturbed feature vector. In order to train the generative model G_k, we utilize the shadow encoders initially set up to detect perturbations in the previous step. To be specific, each training sample follows the format $(E_m^k(x_m), E_m(x_m)),\ \forall m$. Here, $E_m^k(x_m)$ represents the protected feature vector by defense strategy f^k for shadow encoder E_m, while $E_m(x_m)$ refers to the corresponding clean feature vector. By using this approach, G_k acquires the ability to map perturbed feature vectors to unperturbed ones, thereby ensuring that $G_k(E_m^k)$ exhibits the same characteristics as E_m.

Two major families of generative models are commonly used in the field: Variational Autoencoders [43] and Generative Adversarial Networks (GANs) [44]. Autoencoders are primarily employed for dimensionality reduction, compressing data samples into informative representations with smaller

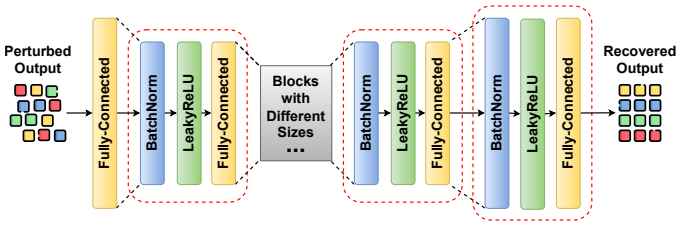

Fig. 2. The design of the generator in BESA

dimensions than the input. On the other hand, GANs are typically used for inverse transformations, generating samples of more complex distributions based on random noise with larger dimensions than the input. However, in the case of perturbation discovery, the dimensionality of the clean feature vector remains the same as that of the perturbed feature vector. Therefore, neither VAEs nor GANs are suitable for achieving the objective of perturbation recovery.

In Fig. 2, we have developed a generator that draws inspiration from MagNet [22]. This generator denoted as G, is composed of multiple blocks comprising BatchNorm, LeakyReLU, and fully connected layers. Within each block, the BatchNorm layer initializes by normalizing the feature vectors to address convergence difficulties and mode collapse. Subsequently, the LeakyReLU activation function is employed rather than ReLU to alleviate the issue of a vanishing gradient [45]. Finally, the perturbed feature vectors are processed by the fully connected layer to eliminate the influence of defense strategies and reconstruct the corresponding clean feature vectors. Our proposed generator is designed to be lightweight but contains all essential components for effective perturbation recovery.

The generator is trained to minimize the difference between the recovered feature vector $G_k(E_m^k(x_m))$ and the actual clean feature vector $E_m(x_m)$. The minimization optimization problem can be formulated as follows:

$$\arg\min_{\theta_{G_k}} \mathbb{E}_{x_m \sim \mathcal{D}_m}[\mathcal{L}(G_k(E_m^k(x_m)), E_m(x_m))], \quad (2)$$

where θ_{G_k} are the parameters of the generative model G_k, and $\mathcal{L}$ represents the loss function. Commonly employed loss functions in this context are cosine similarity, L_2-norm loss, and L_1-norm loss.

Cosine Similarity. Cosine similarity is a commonly used loss function in various encoder training algorithms and encoder stealing attacks. It provides an effective measure of the dissimilarity between two distinct feature vectors produced by their respective encoders.

$$L_C = -\frac{1}{|\sum_m D_m|} \sum_{x_m \in D_m} \frac{G_k(E_m^k(x_m)) \cdot E_m(x_m)}{||G_k(E_m^k(x_m))|| \times ||E_m(x_m)||}. \quad (3)$$

L_2-norm loss. The L_2-norm loss is a straightforward loss function known for its rapid convergence.

$$L_2 = \frac{1}{|\sum_m D_m|} \sum_{x_m \in D_m} [G_k(E_m^k(x_m)) - E_m(x_m)]^2. \quad (4)$$

L_1-norm loss. The L_1-norm loss exhibits a slower and less substantial gradient decay compared to the L_2-norm loss, but the derivative of the L_1-norm loss is not unique at the point 0, potentially leading to non-convergence.

$$L_1 = \frac{1}{|\sum_m D_m|} \sum_{x_m \in D_m} [G_k(E_m^k(x_m)) - E_m(x_m)]. \quad (5)$$

V. Experiments

A. Experimental Settings

1) Datasets and Target Encoders: We conducted experiments on four commonly used datasets: MNIST, Fashion-MNIST, CIFAR-10, SVHN, and ImageNette. We trained target encoders on these datasets using VGG16 and ResNet-34 respectively. For the contrastive learning algorithm, we employed three widely adopted methods: SimCLR, MoCo, and BYOL.

2) State-of-the-Art (SOTA) Attacks: We list three SOTA encoder stealing attacks improved by the use of BESA.

SSLGuard [39]. The encoder stealing approach in *SSL-Guard* leverages the samples in the shadow dataset to simply query the target encoder for optimizing the surrogate encoder.

StolenEncoder [20]. *StolenEncoder* employs data augmentation to enhance the loss function for optimizing the surrogate encoder. Moreover, they leverage the inner characteristics of the pre-trained encoder to decrease the query budget.

Cont-Steal [3]. Inspired by the concept of contrastive learning, *Cont-Steal* ensures that the surrogate feature vector of an image is closely aligned with its target feature vector, while also creating a distinction between feature vectors of different images.

3) Settings of BESA: In this study, we implement BESA within three established encoder stealing attack frameworks, resulting in their enhanced variations. The experimental settings for existing attacks are selected based on the ones in their original paper for a fairer comparison, and the results are the average based on ten repeated experiments to avoid the occasional situation.

In order to ensure that the meta-classifiers can generalize effectively in accurately detecting the specific defense method used by the service provider, a substantial number of shadow encoders with various architectures (e.g., ResNet, VGG, etc.) are trained for the surrogate encoder. In our experiments, the default number of shadow encoders is set to 128. All encoders undergo training utilizing contrastive algorithms that can be found on GitHub. For training the surrogate encoders, we use the following default hyperparameters across all experiments unless otherwise specified: a learning rate of 0.1, a batch size of 128, and the SGD optimizer with momentum 0.9. Each training run lasted for 100 epochs, and learning rate decay was applied with a factor of 0.1 every 60 epochs. Finally, we initialized the surrogate encoder with ResNet-50 as the architecture and used SimCLR for pre-training. For the experimental hardware platform, we utilized two NVIDIA 4090 GPUs, each equipped with 24 GB of memory.

TABLE I
ACCURACY OF THE SURROGATE ENCODER FOR DIFFERENT DATASETS. THE HIGHER LEVEL OF ACCURACY IS EMPHASIZED IN BOLD.

Dataset	Method	Top-K	RD	NP	Hybrid1	Hybrid2	Hybrid3
MNIST	SSLGuard	79.65%	72.67%	75.38%	71.24%	70.38%	70.99%
	SSLGuard + BESA	**97.24%**	**96.23%**	**97.09%**	**95.87%**	**94.66%**	**95.03%**
	StolenEncoder	87.39%	85.46%	83.59%	81.33%	81.32%	82.01%
	StolenEncoder + BESA	**98.24%**	**97.68%**	**97.17%**	**93.99%**	**93.56%**	**92.29%**
	Cont-Steal	90.71%	88.44%	91.25%	87.54%	88.11%	88.26%
	Cont-Steal + BESA	**99.11%**	**98.54%**	**99.43%**	**97.51%**	**97.15%**	**96.13%**
Fashion-MNIST	SSLGuard	77.21%	73.24%	72.54%	68.45%	67.66%	68.88%
	SSLGuard + BESA	**96.43%**	**94.58%**	**93.44%**	**89.22%**	**90.00%**	**89.68%**
	StolenEncoder	85.49%	84.25%	83.98%	77.45%	77.50%	77.63%
	StolenEncoder + BESA	**97.55%**	**94.18%**	**95.46%**	**91.32%**	**91.89%**	**91.05%**
	Cont-Steal	89.25%	90.16%	88.29%	83.23%	84.33%	85.01%
	Cont-Steal + BESA	**97.58%**	**96.84%**	**93.43%**	**93.98%**	**93.27%**	**93.51%**
CIFAR-10	SSLGuard	62.98%	62.49%	63.45%	61.11%	60.91%	61.51%
	SSLGuard + BESA	**74.53%**	**71.34%**	**72.88%**	**70.10%**	**71.33%**	**69.35%**
	StolenEncoder	63.55%	64.57%	62.33%	61.91%	61.99%	61.86%
	StolenEncoder + BESA	**76.12%**	**77.23%**	**78.59%**	**77.05%**	**76.84%**	**77.05%**
	Cont-Steal	64.54%	66.15%	65.94%	63.41%	62.87%	63.24%
	Cont-Steal + BESA	**78.92%**	**76.33%**	**75.20%**	**73.22%**	**73.59%**	**73.26%**
SVHN	SSLGuard	60.35%	60.21%	59.34%	58.22%	59.36%	59.01%
	SSLGuard + BESA	**68.24%**	**66.98%**	**65.54%**	**66.29%**	**67.26%**	**66.81%**
	StolenEncoder	61.24%	60.87%	62.45%	60.98%	61.54%	61.37%
	StolenEncoder + BESA	**69.25%**	**70.11%**	**68.82%**	**66.45%**	**67.35%**	**67.19%**
	Cont-Steal	62.47%	63.11%	62.73%	61.04%	62.05%	61.99%
	Cont-Steal + BESA	**70.47%**	**70.98%**	**71.47%**	**72.08%**	**73.18%**	**72.98%**
ImageNette	SSLGuard	70.68%	68.32%	69.34%	55.20%	55.96%	56.32%
	SSLGuard + BESA	**78.41%**	**72.82%**	**76.53%**	**60.23%**	**61.93%**	**62.07%**
	StolenEncoder	69.42%	67.21%	68.11%	63.01%	63.68%	64.08%
	StolenEncoder + BESA	**74.25%**	**72.13%**	**75.66%**	**70.15%**	**71.16%**	**71.82%**
	Cont-Steal	76.84%	73.12%	68.35%	70.01%	71.13%	71.96%
	Cont-Steal + BESA	**80.72%**	**78.67%**	**72.67%**	**76.77%**	**77.25%**	**78.33%**

4) State-of-the-Art Defenses: Based on the discussion in state-of-the-art works [3], [20], we here mainly concentrate on three commonly adopted perturbation-based defense methods: Top-K, rounding, and noise poisoning.

- **Top-K.** The top-K algorithm selects and preserves the K largest elements in the feature vector while setting all other elements to zero.
- **Rounding (RD).** Rounding is performed on the feature vector, preserving a specific number of digits after the decimal point for each element.
- **Noise Poisoning (NP).** NP adding Gaussian noise to the feature vector. The default setting for $z \in \mathcal{N}(0, \sigma^2)$ is $\sigma^2 = 0.2$.

B. Experimental Results

1) Performance of BESA: Table I illustrates the experimental results of BESA for state-of-the-art encoder stealing attacks under three perturbation-based defenses: Top-k, rounding, and noise poisoning. The analysis of experimental results is explained in detail below.

Single Defense: Our proposed BESA effectively enhances existing canonical encoder stealing attacks for all three cases of the single defense listed. We have observed that the defenses significantly reduce the accuracy of the surrogate encoder, particularly in the earlier attack in *SSLGuard*. When employing perturbation-based defense methods, BESA improves the performance by up to 24.63% compared to state-of-the-art encoder stealing attacks. While the improvement on existing attacks may be decreased for more complex datasets (e.g., ImageNette, SVHN), the surrogate encoder is still able to enhance their accuracy by up to 16.26%.

When employing hybrid defenses, such as the combination of random dropout (RD), noise perturbation (NP), and top-k suppression, service providers typically achieve stronger protection than using a single defense alone. To reflect realistic scenarios where the defense strategy may change dynamically across queries, we adopt a per-query perturbation detection approach in BESA. We evaluate BESA under three hybrid defense settings—RD+NP (Hybrid1), RD+Top-K (Hybrid2), and NP+Top-K (Hybrid3)—as summarized in Table I. Although these hybrid strategies significantly increase the difficulty of encoder stealing, BESA consistently improves the surrogate encoder performance across all datasets and attack baselines. For instance, under Hybrid3 on MNIST, BESA enhances the surrogate accuracy of *SSLGuard* from 70.99% to 92.29%. These results demonstrate that the perturbation detection module in BESA can reliably identify and handle complex, mixed defenses with high accuracy, maintaining its effectiveness even under more aggressive protection schemes.

Unknown Defense: To evaluate the robustness of BESA under unknown defense strategies, we simulate scenarios where the attacker trains BESA using a subset of known defenses, then applies it to encoders protected by unseen defenses. Previously, we considered a setting where BESA is trained on Top-K and rounding, and tested on NP (denoted as Un-NP). We now extend this evaluation to two additional settings: (1) training on NP and RD but testing on Top-K (Un-Top-k), and (2) training on Top-K and NP but testing on RD (Un-RD). As shown in Table II, BESA consistently improves the accuracy of surrogate encoders across all datasets, even under these unseen defenses. For example, under Un-Top-k on MNIST, the surrogate encoder from *SSLGuard + BESA* achieves 64.47%,

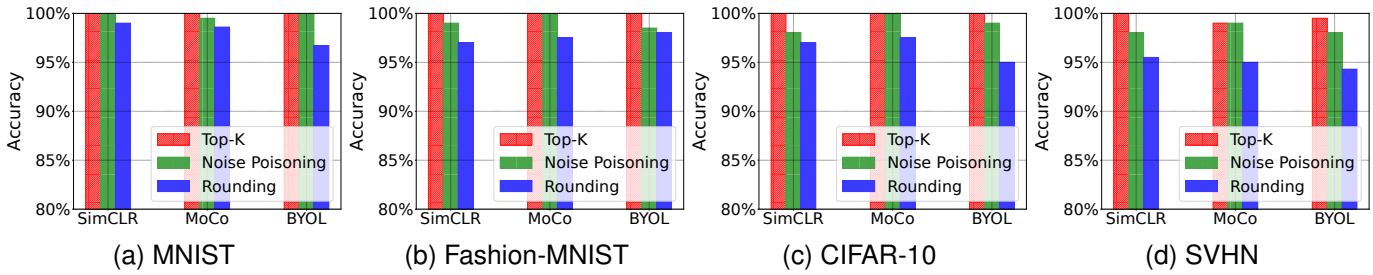

Fig. 3. Detection accuracy of meta-classifiers in BESA.

TABLE II
ACCURACY OF THE SURROGATE ENCODER FOR DIFFERENT DATASETS ON DIFFERENT UNKNOWN DEFENSES.

Dataset	Method	Un-Top-k	Un-RD	Un-NP
MINST	SSLGuard	57.43%	64.94%	69.44%
	SSLGuard + BESA	**64.47%**	**71.32%**	**86.98%**
	StolenEncoder	62.35%	68.14%	78.28%
	StolenEncoder + BESA	**70.88%**	**75.72%**	**85.45%**
	Cont-Steal	65.71%	75.10%	81.23%
	Cont-Steal + BESA	**72.36%**	**79.50%**	**88.60%**
Fashion-MNIST	SSLGuard	62.22%	61.86%	67.34%
	SSLGuard + BESA	**68.57%**	**73.37%**	**84.91%**
	StolenEncoder	59.96%	66.52%	74.83%
	StolenEncoder + BESA	**70.39%**	**71.24%**	**82.11%**
	Cont-Steal	68.04%	73.65%	80.65%
	Cont-Steal + BESA	**75.00%**	**81.41%**	**87.33%**
CIFAR-10	SSLGuard	63.25%	68.29%	60.34%
	SSLGuard + BESA	**72.96%**	**77.34%**	**65.17%**
	StolenEncoder	61.82%	59.33%	60.66%
	StolenEncoder + BESA	**69.13%**	**71.62%**	**71.24%**
	Cont-Steal	58.37%	65.29%	62.93%
	Cont-Steal + BESA	**68.38%**	**73.06%**	**70.45%**
SVHN	SSLGuard	68.25%	57.20%	57.36%
	SSLGuard + BESA	**76.59%**	**63.01%**	**65.77%**
	StolenEncoder	59.63%	58.11%	59.87%
	StolenEncoder + BESA	**65.00%**	**67.26%**	**66.17%**
	Cont-Steal	63.03%	65.73%	60.99%
	Cont-Steal + BESA	**72.90%**	**71.25%**	**68.33%**
ImageNette	SSLGuard	55.21%	54.99%	52.63%
	SSLGuard + BESA	**65.37%**	**67.82%**	**55.78%**
	StolenEncoder	56.61%	52.34%	59.87%
	StolenEncoder + BESA	**62.38%**	**60.84%**	**66.17%**
	Cont-Steal	58.35%	55.60%	60.83%
	Cont-Steal + BESA	**65.39%**	**72.09%**	**68.33%**

significantly higher than the baseline SSLGuard (57.43%). These results demonstrate that BESA retains a strong degree of generalization and transferability, making it effective even in the presence of novel or unseen perturbation-based defenses.

2) Performance on Perturbation Detection: In this subsection, we assess how well the meta-classifier can detect the defensive techniques used by the service provider. Following the training of the meta-classifiers, we assess their prediction accuracy by testing on an additional 128 shadow encoders, each protected by distinct defense strategies.

From Fig. 3, the results indicate that the meta-classifiers achieve an accuracy of over 98% for all three contrastive algorithms utilized in pre-training the target encoder, particularly for typical datasets like MNIST and FashionMNIST. However, the performance of the meta-classifiers exhibits a slight decline when dealing with complex datasets, such as CIFAR-10 and SVHN. Notably, the presence of the *Rounding* defense can be identified by examining the number of digits in the feature vectors. Interestingly, both *Top-K* and *Noise Poisoning* defense approaches can be accurately detected with almost perfect accuracy of nearly 100%, possibly due to distinguishing characteristics present in the feature vectors, such as zero values or added noise.

C. Impact Factors

1) Architecture of the Surrogate Encoder: In our previous experiments, we trained the surrogate encoder using the complex architecture of ResNet-50. This choice is motivated by the belief that a more capable architecture will better emulate the functionality of the target encoder. In this part, we aim to investigate the impact of different architectural choices for surrogate encoders on attack performance. Specifically, when using VGG16 as the target encoder for CIFAR-10 and ResNet-34 for SVH, we select various architectures for the surrogate encoder. In the case of CIFAR-10, the surrogate

encoder options consisted of AlexNet, ResNet-18, ResNet-34, and VGG16, while for SVHN, the options included AlexNet, ResNet-18, ResNet-34, and VGG16.

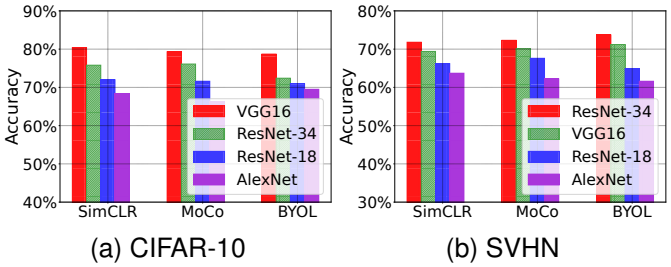

Fig. 4. Impact of architecture choice on BESA accuracy.

As shown in Fig. 4, our results indicate that it is possible to successfully steal the target encoder even when there is a mismatch between its architecture and that of the surrogate encoder, provided that the surrogate encoder's architecture is complex enough. Moreover, our findings suggest that a more intricate architecture for the surrogate encoder enhances its performance by enabling it to effectively mimic the target encoder through utilizing a greater amount of information.

2) Loss functions: The core element of the perturbation recovery in BESA involves minimizing the distance between the perturbed feature vectors and the unperturbed ones. Notably, the main objective of BESA is to bypass the perturbation-based defense mechanisms by providing recovered clean feature vectors. Therefore, selecting a more suitable loss function can significantly enhance BESA performance. In this study, we evaluate the effects of three distinct loss functions: cosine similarity, $\mathcal{L}_2$ norm, and $\mathcal{L}_1$ norm.

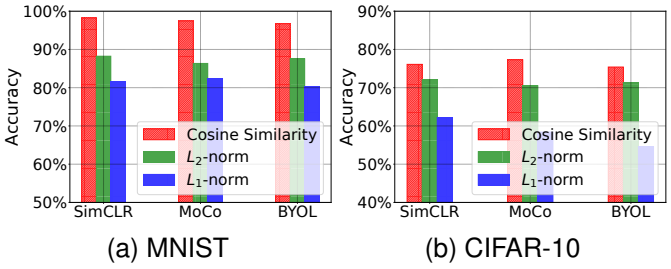

Fig. 5. Impact of loss functions on BESA accuracy.

Fig. 5 illustrates the outcomes associated with each loss function. In both the MNIST and CIFAR-10 datasets, cosine similarity demonstrates superior performance compared to the other two loss functions. One potential rationale behind this is that cosine similarity more effectively captures the resemblance between distinct feature vectors in contrast to the other functions [46]. In comparing the efficacy of $\mathcal{L}_2$ distance with $\mathcal{L}_1$ distance, it was observed that $\mathcal{L}_2$ distance delivers enhanced accuracy. This is likely due to the fact that in $\mathcal{L}_2$ distance, the differences between the feature vectors generated by the target encoder and the surrogate encoder are minimized across all dimensions.

TABLE III
TRAINING TIME OF BESA (SECONDS)

Dataset	Shadow Model	Meta- Classifier	Generator
MINST	13	210	15326
Fashion-MNIST	15	305	17025
CIFAR-10	35	272	20985
SVHN	62	291	31025
ImageNette	52	1650	28650

D. Time Costs

The computational cost of BESA can be separated into two stages: the offline training phase, where shadow models, meta-classifiers, and generators are built; and the online inference phase, where these trained components are used to detect and recover perturbed outputs during encoder stealing. The offline training introduces a one-time cost. As shown in Table III, we report the training time across four benchmark datasets. The time varies depending on the dataset complexity and the number of shadow models involved. Notably, although SVHN requires more shadow encoders, the overall training time remains manageable and is performed only once. In cases where a new perturbation defense is encountered and known to the attacker, the corresponding components can be trained offline and integrated into the existing framework with minimal disruption. Moreover, the trained meta-classifiers and generators are reusable across different target encoders, enabling multiple encoder stealing attempts without retraining.

For unknown or novel defenses not previously modeled, our results show that BESA exhibits a degree of robustness, likely due to shared patterns between new and existing defense strategies. This allows for partial generalization even without retraining. During the online attack phase, BESA only performs lightweight inference using the pre-trained meta-classifier and generator, resulting in negligible overhead when integrated into standard encoder stealing pipelines.

VI. DISCUSSION

In this section, we discuss the limitations and ethical problems of BESA.

A. Ethical Problem

In this paper, we propose the BESA to enhance existing encoder stealing attacks and show the inadequacy of existing perturbation defenses in the face of such attacks. While our work reveals the limitations of current defenses, we emphasize that the BESA is only a further exploration of existing attack methods, rather than advocating malicious use. We call on academia and industry to propose more effective defense strategies to protect pre-trained encoders from such attacks.

One potential mitigation is to limit the effectiveness of such attacks through request detection mechanisms. Since BESA attacks require a large number of query requests, the probability of successful attacks can be reduced by limiting the query frequency of each user and combining anomaly detection algorithms to identify and block malicious query behaviors. In addition, service providers can introduce stronger authentication mechanisms and behavioral analysis to strengthen

defenses. Such measures can effectively prevent malicious use of the BESA and ensure normal use by legitimate users.

B. Compatibility with Other Defense

BESA is designed primarily to bypass perturbation-based defenses by recovering clean feature vectors from intentionally disrupted outputs. However, its design does not directly target detection-based or differential privacy (DP)-based defenses. Detection-based approaches typically analyze the distribution of query inputs to identify potential malicious behavior. Since BESA operates only on the returned feature vectors and does not control the query generation process, it may be affected by detection mechanisms depending on how queries are issued. That said, many detection-based systems return perturbed outputs rather than reject responses altogether, in which case BESA remains effective. To fully bypass such defenses, future extensions could incorporate more advanced query synthesis strategies that generate queries mimicking normal usage patterns. For DP-based defenses, their evaluation remains limited due to the lack of publicly available implementations. Once such mechanisms become accessible, we plan to extend our evaluation accordingly.

C. Generalization

The effectiveness of BESA's perturbation detection module relies on the meta-classifier's ability to recognize defense-specific patterns in feature vectors. This requires training with outputs from shadow encoders equipped with various defenses. Because the attacker does not have access to the target encoder's architecture, these shadow models must be diverse to ensure generalization. While this strategy works well in practice, it does come with additional training overhead. A potential future direction is to adopt meta-learning techniques that improve the adaptability of the meta-classifier using fewer shadow encoders. Additionally, as discussed in Section III, BESA operates under a black-box setting where the attacker has no access to model internals or original training data but can query the encoder and simulate common defenses. These assumptions reflect practical constraints in real-world Encoder-as-a-Service scenarios.

VII. Conclusion

We propose a novel encoder stealing attack called BESA which allows the construction of a well-performing surrogate encoder, even in the presence of perturbation-based defenses protecting the target encoder. We identify the specific defense method utilized by the service provider using meta-classifiers and restore perturbed feature vectors using a generative model. Extensive experimental results demonstrate that BESA significantly enhances the accuracy of surrogate encoders when facing various defensive mechanisms.

References

[1] A. Baevski, A. Babu, W. Hsu, and M. Auli, "Efficient self-supervised learning with contextualized target representations for vision, speech and language," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 1416–1429.

[2] W. Qu, J. Jia, and N. Z. Gong, "Reaas: Enabling adversarially robust downstream classifiers via robust encoder as a service," in *NDSS*. The Internet Society, 2023.

[3] Z. Sha, X. He, N. Yu, M. Backes, and Y. Zhang, "Can't steal? cont-steal! contrastive stealing attacks against image encoders," in *CVPR*. Computer Vision Foundation /IEEE, 2023, pp. 16 373–16 383.

[4] T. Zhang, H. Wu, X. Lu, and G. Sun, "Awencoder: Adversarial watermarking pre-trained encoders in contrastive learning," *CoRR*, vol. abs/2208.03948, 2022.

[5] J. Jia, H. Liu, and N. Z. Gong, "10 security and privacy problems in self-supervised learning," *CoRR*, vol. abs/2110.15444, 2021.

[6] A. Dziedzic, N. Dhawan, M. A. Kaleem, J. Guan, and N. Papernot, "On the difficulty of defending self-supervised learning against model extraction," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 5757–5776.

[7] W. Wang, X. Qian, Y. Fu, and X. Xue, "DST: dynamic substitute training for data-free black-box attack," in *CVPR*. IEEE, 2022, pp. 14 341–14 350.

[8] L. Chen, Y. Zhang, Y. Song, L. Liu, and J. Wang, "Self-supervised learning of adversarial example: Towards good generalizations for deepfake detection," in *CVPR*. IEEE, 2022, pp. 18 689–18 698.

[9] H. Liu, J. Jia, W. Qu, and N. Z. Gong, "Encodermi: Membership inference against pre-trained encoders in contrastive learning," in *CCS*. ACM, 2021, pp. 2081–2095.

[10] X. He, H. Liu, N. Z. Gong, and Y. Zhang, "Semi-leak: Membership inference attacks against semi-supervised learning," in *ECCV (31)*, ser. Lecture Notes in Computer Science, vol. 13691. Springer, 2022, pp. 365–381.

[11] J. Jia, Y. Liu, and N. Z. Gong, "Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning," in *SP*. IEEE, 2022, pp. 2043–2059.

[12] A. Saha, A. Tejankar, S. A. Koohpayegani, and H. Pirsiavash, "Backdoor attacks on self-supervised learning," in *CVPR*. IEEE, 2022, pp. 13 327–13 336.

[13] C. Li, R. Pang, Z. Xi, T. Du, S. Ji, Y. Yao, and T. Wang, "An embarrassingly simple backdoor attack on self-supervised learning," in *CVPR*. Computer Vision Foundation /IEEE, 2023, pp. 4367–4378.

[14] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "PRADA: protecting against DNN model stealing attacks," in *EuroS&P*. IEEE, 2019, pp. 512–527.

[15] M. Tang, A. Dai, L. DiValentin, A. Ding, A. Hass, N. Z. Gong, and Y. Chen, "Modelguard: Information-theoretic defense against model extraction attacks," in *USENIX Security Symposium*. USENIX Association, 2024.

[16] P. Li, P. Cheng, F. Li, W. Du, H. Zhao, and G. Liu, "Plmmark: A secure and robust black-box watermarking framework for pre-trained language models," in *AAAI*. AAAI Press, 2023, pp. 14 991–14 999.

[17] X. Li, C. Yin, L. Fang, R. Wang, and C. Lin, "Ssl-auth: An authentication framework by fragile watermarking for pre-trained encoders in self-supervised learning," *CoRR*, vol. abs/2308.04673, 2023.

[18] Y. Tang, J. Yu, K. Gai, X. Qu, Y. Hu, G. Xiong, and Q. Wu, "Watermarking vision-language pre-trained models for multi-modal embedding as a service," *CoRR*, vol. abs/2311.05863, 2023.

[19] A. Marshall, J. Parikh, E. Kiciman, and R. Kumar, "Threat modeling ai/ml systems and dependencies," *Security documentation*, 2019.

[20] Y. Liu, J. Jia, H. Liu, and N. Z. Gong, "Stolenencoder: Stealing pre-trained encoders in self-supervised learning," in *CCS*. ACM, 2022, pp. 2115–2128.

[21] Y. Chen, R. Guan, X. Gong, J. Dong, and M. Xue, "D-DAE: defense-penetrating model extraction attacks," in *SP*. IEEE, 2023, pp. 382–399.

[22] D. Meng and H. Chen, "Magnet: A two-pronged defense against adversarial examples," in *CCS*. ACM, 2017, pp. 135–147.

[23] X. Gong, Q. Wang, Y. Chen, W. Yang, and X. Jiang, "Model extraction attacks and defenses on cloud-based machine learning models," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 83–89, 2020.

[24] X. Gong, Y. Chen, W. Yang, G. Mei, and Q. Wang, "Inversenet: Augmenting model extraction attacks with training data inversion," in *IJCAI*. ijcai.org, 2021, pp. 2439–2447.

[25] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX Security Symposium*. USENIX Association, 2016, pp. 601–618.

[26] Z. Yue, Z. He, H. Zeng, and J. J. McAuley, "Black-box attacks on sequential recommenders via data-free model extraction," in *RecSys*. ACM, 2021, pp. 44–54.

[27] Z. Ma, X. Liu, Y. Liu, X. Liu, Z. Qin, and K. Ren, "Divtheft: An ensemble model stealing attack by divide-and-conquer," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 6, pp. 4810–4822, 2023.

[28] H. Shah, A. G, P. Kulkarni, Y. Govindarajulu, and M. Parmar, "Data-free model extraction attacks in the context of object detection," *CoRR*, vol. abs/2308.05127, 2023.
[29] S. Kariyappa, A. Prakash, and M. K. Qureshi, "MAZE: data-free model stealing attack using zeroth-order gradient estimation," in *CVPR*. Computer Vision Foundation /IEEE, 2021, pp. 13 814–13 823.
[30] J. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *CVPR*. Computer Vision Foundation / IEEE, 2021, pp. 4771–4780.
[31] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *USENIX Security Symposium*. USENIX Association, 2021, pp. 1937–1954.
[32] S. Szyller, V. Duddu, T. Gröndahl, and N. Asokan, "Good artists copy, great artists steal: Model extraction attacks against image translation generative adversarial networks," *CoRR*, vol. abs/2104.12623, 2021.
[33] M. Yu and S. Sun, "Fe-dast: Fast and effective data-free substitute training for black-box adversarial attacks," *Comput. Secur.*, vol. 113, p. 102555, 2022.
[34] Z. Lin, K. Xu, C. Fang, H. Zheng, A. A. Jaheezuddin, and J. Shi, "QUDA: query-limited data-free model extraction," in *AsiaCCS*. ACM, 2023, pp. 913–924.
[35] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 4954–4963.
[36] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *AsiaCCS*. ACM, 2017, pp. 506–519.
[37] W. Wang, X. Qian, Y. Fu, and X. Xue, "DST: dynamic substitute training for data-free black-box attack," in *CVPR*. IEEE, 2022, pp. 14 341–14 350.
[38] J. Dubinski, S. Pawlak, F. Boenisch, T. Trzcinski, and A. Dziedzic, "Bucks for buckets (B4B): active defenses against stealing encoders," *CoRR*, vol. abs/2310.08571, 2023.
[39] T. Cong, X. He, and Y. Zhang, "Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders," in *CCS*. ACM, 2022, pp. 579–593.
[40] W. Peng, J. Yi, F. Wu, S. Wu, B. Zhu, L. Lyu, B. Jiao, T. Xu, G. Sun, and X. Xie, "Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark," in *ACL (1)*. Association for Computational Linguistics, 2023, pp. 7653–7668.
[41] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images." Toronto, ON, Canada, 2009.
[42] N. Yuval, "Reading digits in natural images with unsupervised feature learning," in *NeurIPS*, 2011.
[43] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.
[44] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014, pp. 2672–2680.
[45] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 30. PMLR, 2013, p. 3.
[46] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," *CoRR*, vol. abs/2209.02299, 2022.

Xuhao Ren received his B.S. degree in the School of Information Engineering of Sichuan Agricultural University, Sichuan, China, in 2022. He is currently a master's student in the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include applied cryptography.

Haotian Liang received the B.S. degree from Lanzhou University in 2022. He is currently pursuing the master's degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include machine learning security, Internet of Things security, and cloud security.

Yajie Wang received his Ph.D. degree in the School of Computer Science, Beijing Institute of Technology, Beijing, China. He is currently a postdoctoral at the School of Cyberspace Science and Technology, Beijing Institute of Technology. His main research interests include the robustness and vulnerability of artificial intelligence, cyberspace security, etc.

Chuan Zhang (Member, IEEE) received his Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2021. From Sept. 2019 to Sept. 2020, he worked as a visiting Ph.D. student with the BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is currently an assistant professor at the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include applied cryptography, machine learning, and blockchain.

Zehui Xiong (Senior Member, IEEE) is currently an Assistant Professor at Singapore University of Technology and Design, and also an Honorary Adjunct Senior Research Scientist with Alibaba-NTU Singapore Joint Research Institute, Singapore. He received the PhD degree in Nanyang Technological University (NTU), Singapore. He was the visiting scholar at Princeton University and University of Waterloo. His research interests include wireless communications, Internet of Things, blockchain, edge intelligence, and Metaverse.

Liehuang Zhu (Senior Member, IEEE) received his Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004. He is currently a professor at the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include security protocol analysis and design, group key exchange protocols, wireless sensor networks, and cloud computing.