Probe-Me-Not: Protecting Pre-trained Encoders from Malicious Probing

Ruyi Ding, Tong Zhou, Lili Su, Aidong Adam Ding, Xiaolin Xu, Yunsi Fei
Northeastern University, Boston, MA 02115, USA
{ding.ruy, zhou.tong1, l.su, a.ding, x.xu, y.fei}@northeastern.edu

***Abstract*—Adapting pre-trained deep learning models to customized tasks has become a popular choice for developers to cope with limited computational resources and data volume. More specifically, probing–training a downstream head on a pre-trained encoder–has been widely adopted in transfer learning, which helps to prevent overfitting and catastrophic forgetting. However, such generalizability of pre-trained encoders raises concerns about the potential misuse of probing for harmful intentions, such as discriminatory speculation and warfare applications. In this work, we introduce EncoderLock, a novel applicability authorization method designed to protect pre-trained encoders from malicious probing, i.e., yielding poor performance on specified prohibited domains while maintaining their utility in authorized ones. Achieving this balance is challenging because of the opposite optimization objectives and the variety of downstream heads that adversaries can utilize adaptively. To address these challenges, EncoderLock employs two techniques: *domain-aware weight selection and updating* to restrict applications on prohibited domains/tasks, and *self-challenging training scheme* that iteratively strengthens resistance against any potential downstream classifiers that adversaries may apply. Moreover, recognizing the potential lack of data from prohibited domains in practical scenarios, we introduce three EncoderLock variants with different levels of data accessibility: *supervised* (prohibited domain data with labels), *unsupervised* (prohibited domain data without labels), and *zero-shot* (no data or labels available). Extensive experiments across fifteen domains and three model architectures demonstrate EncoderLock's effectiveness over baseline methods using non-transferable learning. Additionally, we verify EncoderLock's effectiveness and practicality with a real-world pre-trained Vision Transformer (ViT) encoder from Facebook. These results underscore the valuable contributions EncoderLock brings to the development of responsible AI.**

I. Introduction

As the complexity of learning tasks increases, leveraging pre-trained models becomes a popular strategy for developers to train their customized models efficiently. Among various transfer learning methods, model probing has emerged as one of the most common and lightweight strategies to utilize pre-learned knowledge effectively [1]–[3]. It involves freezing the encoder parts of pre-trained models while fine-tuning only the downstream heads. The encoders often include early layers of pre-trained models with more complex structures, which is responsible for extracting useful information from raw data to latent representations, on which downstream heads perform specific tasks such as classification and generation [4], [5].

Probing offers several advantages, including resource efficiency, because of its low requirements on data and computational resources, and semantic consistency, as it helps avoid catastrophic forgetting–the performance reduction due to the encoder's loss of pre-learned knowledge after extensive fine-tuning [6]–[8]. Furthermore, probing allows the pre-trained encoder to be used as a black-box, either as local private models [9]–[11] or cloud services through APIs [12]–[14], ensuring better intellectual property protection [15]. Nowadays, many companies, such as Clarifai [12] and OpenAI [13], offer commercial encoder APIs, allowing users to input data and obtain latent feature vectors, which can then be used for various downstream real-world applications.

However, the general availability of the pre-trained encoder for probing also raises concerns about ***malicious probing***, i.e., users can probe the encoder for unethical or harmful tasks [16]. Examples include building classification heads for discriminatory speculation [17], [18] or autonomous weapons in warfare applications [19]. To address these concerns, model owners have set strict policies regarding the utilization of pre-trained encoders. For instance, OpenAI explicitly prohibits users from employing their encoder services for "any illegal, harmful, or abusive activity". However, relying solely on policies, without concrete technological barriers, is insufficient to prevent model misuse. Considering malicious probing not only poses ethical risks but also represents a serious form of infringement on the intellectual property of model owners, design-time countermeasures are urgently needed for protecting the encoders with applicability authorization [20].

Proactively preventing pre-trained encoders from malicious probing presents three challenges. **Challenge 1: *Integrity of Pre-trained Encoder.*** The protection strategy should maintain the encoder's functionality on *authorized domains* (those for which the encoder is designed), while restricting misuse on *prohibited domains* (those not allowed due to malicious intent). Furthermore, it is advisable to have a small impact on *admissible domains* (those are gray-listed and not explicitly considered during encoder design). **Challenge 2: *Robustness to Malicious Probing.*** Malicious users can customize downstream heads with various configurations (e.g., hyper-parameters and classifier architectures). The protection method must be robust against these diverse setups. **Challenge 3: *Accessibility to***

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.24928
www.ndss-symposium.org

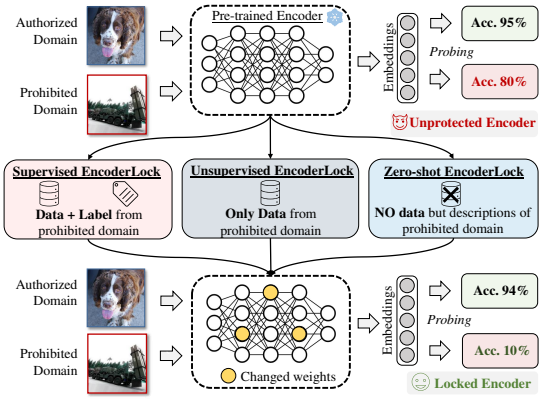

Fig. 1. **Applicability Authorization with EncoderLock**: Fixed pre-trained encoders accept user inputs and return representations. Users can utilize them for various customized tasks by probing with downstream heads. EncoderLock aims to prevent malicious probing to pre-defined prohibited domains, which may have different levels of data accessibility, marked by different colors.

Prohibited Domains. Effective protection requires pre-defined prohibited domains, while a lack of samples from these domains can significantly impact its performance. A few studies design protection against direct applicability on prohibited tasks–malicious users can do inference but no further fine-tuning [20], [21]. They introduced a training strategy considering solely a given prohibited dataset with clear labels and an authorized dataset, called *Non-Transferable Learning (NTL)*. Unfortunately, NTL doesn't apply to pre-trained encoders–as malicious users can further probe encoders with downstream heads using prohibited data. Therefore, we propose **EncoderLock**, a new applicability authorization strategy for pre-trained encoders.

EncoderLock is based on our new three-level threat model for model applicability authorization for pre-trained encoders, following a paradigm akin to that used in representation learning, as illustrated in Fig. 1: a) Level 1 Label-enriched: The provider has a labeled dataset of the prohibited domain, b) Level 2 Label-free: The provider only has an unlabeled dataset, c) Level 3 Theme-only: The provider has no data but knows the theme they wish to exclude the encoder from processing. These levels represent real-world model providers with different data accessibility. Throughout the paper, we will use this color coding to represent these data accessibility levels.

EncoderLock proposes the following solutions to address all three challenges against malicious probing. First, we propose *domain-aware weight selection and updating*, which identifies critical weights to the target domain and adjusts them, successfully restricting the model's transferability to the target domain while minimizing its effect on other authorized domains (addressing Challenge 1). To ensure the robustness of EncoderLock against customized malicious downstream heads (addressing Challenge 2), we introduce a minimax optimization–*self-challenging training*, which refines the encoder's feature space iteratively by continuously adjusting auxiliary downstream heads. Together, these strategies constitute *supervised EncoderLock*, which effectively addresses the Level 1 scenario. To address Challenge 3, we extend two EncoderLock variants for stricter accessibility to the prohibited domain. For Level 2 where only an unlabeled target dataset is available, we introduce *unsupervised EncoderLock*, including a novel regularization term based on contrastive loss in the feature space, which deliberately obfuscates features in the target dataset. For Level 3, we propose *zero-shot EncoderLock*, which leverages an AI agent and a text-to-image generative model to build a reliable pathway from semantic description to an unlabeled synthetic dataset. To ensure the synthetic dataset is representative of the target domain and comprehensive, we propose a prompt refining method utilizing the AI agent.

Our Contributions: We propose **EncoderLock**, a novel and proactive protection on the pre-trained encoder against malicious probing. The contributions of this work include:

1) EncoderLock provides a robust applicability authorization framework to owners of pre-trained encoders. It maintains the encoder's performance on authorized domains with the domain-aware weight selection algorithm and offers robust defense against diverse customized probing through a self-challenging training scheme.
2) We propose a three-level threat model following the practical data availability of representation learning. Correspondingly, we present three variants of EncoderLock with novel techniques to address different levels of target domain data accessibility, tackling realistic comprehensive scenarios.
3) We conduct extensive experiments to evaluate EncoderLock across twelve domains and three encoder architectures, including a large, real-world Vision Transformer [22]. Our results demonstrate the effectiveness of all three EncoderLock variants. Specifically, we assess EncoderLock in a real applicability authorization scenario, preventing a pre-trained encoder from being misused for military purposes while keeping its generalizability to civilian ones.

II. Background

A. Pre-trained Encoders and Model Probing

Pre-trained models are widely used in computer vision [23]–[25], representation learning [26]–[29], and natural language processing [30]–[32], which embed pre-learned knowledge as the model initialization to reduce the complexity in training new tasks. Taking transfer learning in vision tasks as an example, there are three common strategies:

Full Fine-tuning: Full fine-tuning leverages the entire pre-trained model as the training initialization and fine-tunes it with the target dataset. It often has good performance but has the risk of stability and catastrophic forgetting [6], [33].

Prompting: Rather than finetune the model parameters, prompting redirects the pre-trained model via modification on the inputs (i.e., visual prompt). Prompting is efficient but performance experiences a larger degradation [34]–[37].

Model Probing: Probing freezes the early layers of the pre-trained model (e.g., deep convolutional layers or self-attention layers [38], [39]) as the fixed pre-trained encoder and fine-tunes the downstream classifier. It has a small training cost and high stability of the training process [40]–[42].

In this work, we focus on model probing as it is more efficient and stable than fully fine-tuning and has better performance than prompting. Probing also supports pre-trained encoders from different training schemes, which can be categorized into three types: *supervised*, *unsupervised*, and *self-supervised*. For supervised learning, the model (i.e., encoder and downstream head) is trained directly using labeled training data and a loss function (e.g., cross-entropy loss) [43]. Unsupervised learning aims to learn from unlabeled data, using methods such as Gaussian Mixtures Model (GMM) [44], Variational Autoencoder (VAE) [45], and Generative Adversarial Network (GAN) [46]. Self-supervised learning aims to train an encoder to predict one part of data given another part of the input [47]–[49]. It leverages the inherent data characteristics and shows increasing robustness and generalizability of the encoder [50]. Specifically, given the input (e.g., an image), one will use data augmentation operations (e.g., cropping, color jitter, and adding random noise) to build augmented images. The training objective is to make the encoder generate similar embeddings for augmented images from the same input, denoted as positive pairs; while ensuring the discrepancy of embeddings from different images, denoted as negative pairs. The training of a self-supervised encoder utilizes contrastive loss [47], [48], which increases the similarity between positive pairs but decreases those of negative pairs. Our design of EncoderLock considers various data accessibility of prohibited domains, which aligns with the training process of pre-trained encoders–with or without labeled datasets.

B. Applicability Authorization

Recently, applicability authorization, a new IP protection scheme, has been proposed to address the rising concerns of IP infringement on DNN models [20], [21], [51], [52]. Traditional model IP protection aims to protect the rights of owners of DNN models with two typical defense strategies: ownership verification and usage authorization. Ownership verification is designed to trace the illegal behavior of IP infringement using methods such as embedding watermarks during the training procedure or recording fingerprints of the model owner [53]–[55]. In contrast, usage authorization aims to restrict user access to the model, ensuring that only verified, trusted users can access with assigned authorization keys [56], [57]. Instead of protecting the model parameters or hyper-parameters directly like traditional methods, applicability authorization focuses on the unauthorized transfer of the pre-trained models [51]. Specifically, it aims to prevent malicious transfer learning through which an attacker can abuse the pre-trained model for prohibited data or tasks, i.e., non-transfer-learning. In this work, we further propose EncoderLock to address the challenges of applicability authorization of pre-trained *encoders* to safeguard them from unauthorized probing.

C. Non-Transferable Learning (NTL)

Wang et al. [20] introduced NTL for applicability authorization of an entire model without any fine-tuning. In particular, NTL leverages a negative regularization term on the model's target domain performance:

$$L_{NTL} = L_{\mathcal{S}} + R_{\mathcal{T}} \tag{1}$$

where $L_{\mathcal{S}}$ is the Kullback–Leibler (KL) divergence/loss on the source dataset, aiming to retain the model's performance on the source domain. Model non-transferability comes from the regularization term, defined as $R_{\mathcal{T}} = -\min(\beta, \alpha \cdot L_{\mathcal{T}} \cdot L_{dis})$ [20], where $L_{\mathcal{T}}$ is the KL loss on the target dataset, L_{dis} measures the feature space distance between the source and target domains (using Maximum Mean Discrepancy), and α and β are scaling factors. Another prior work [21] proposes an additional CUTI-domain for regularization on private style features with the $R_{\mathcal{T}}$ as $-L_{\mathcal{T}}$. In addition, previous works also proposed 'source-only' NTL for cases when there is no target data available. As the term 'source-only' indicates, this strategy leverages generative models (i.e., GAN) to create a synthetic dataset, which serves as the boundary from the source domain to prohibit the model's transferability to all other domains.

D. Limitations of Prior Works

Prior works focus on the case when the attacker uses the trained model directly but cannot fine-tune it [20], [21]. However, with the increasing popularity and low cost of probing the pre-trained (fixed) encoder, the applicability authorization (model non-transferability) can be bypassed in a few probing epochs. Moreover, previous methods predominantly add a regularization term solely based on the model outputs. Although [20] introduces a feature space distance as a regularization between the source and target domains, they cannot ensure restriction as the class discrepancy might still be large on the feature space. Furthermore, previous methods only consider the case of supervised NTL, i.e., the defender has access to one labeled target dataset and one labeled source dataset. Our Challenge 3 is closer to the practical scenario where the defender lacks knowledge about the prohibited target. Pre-trained encoders and model probing bring new challenges in data availability for applicability authorization. Our work EncoderLock aims to address them accordingly.

III. Threat Model

In this work, we tackle applicability authorization for pre-trained encoders, aiming to prevent malicious users from probing the encoder for harmful tasks (i.e., unethical, illegal, or abusive activities). In this paper, we focus on vision encoders and image classification as the downstream task.

A. Malicious Users

The attackers are users with malicious intent to breach the usage policy of fixed pre-trained encoders with probing [58].
Objective. Their objective is to exploit pre-trained encoders for tasks that are not allowed, specifically, accurately classifying samples from prohibited domains. Other forms of DNN IP infringement of the pre-trained encoder, e.g., model stealing attacks, are out of the scope of this paper.
Capabilities. Capabilities of the malicious users include:

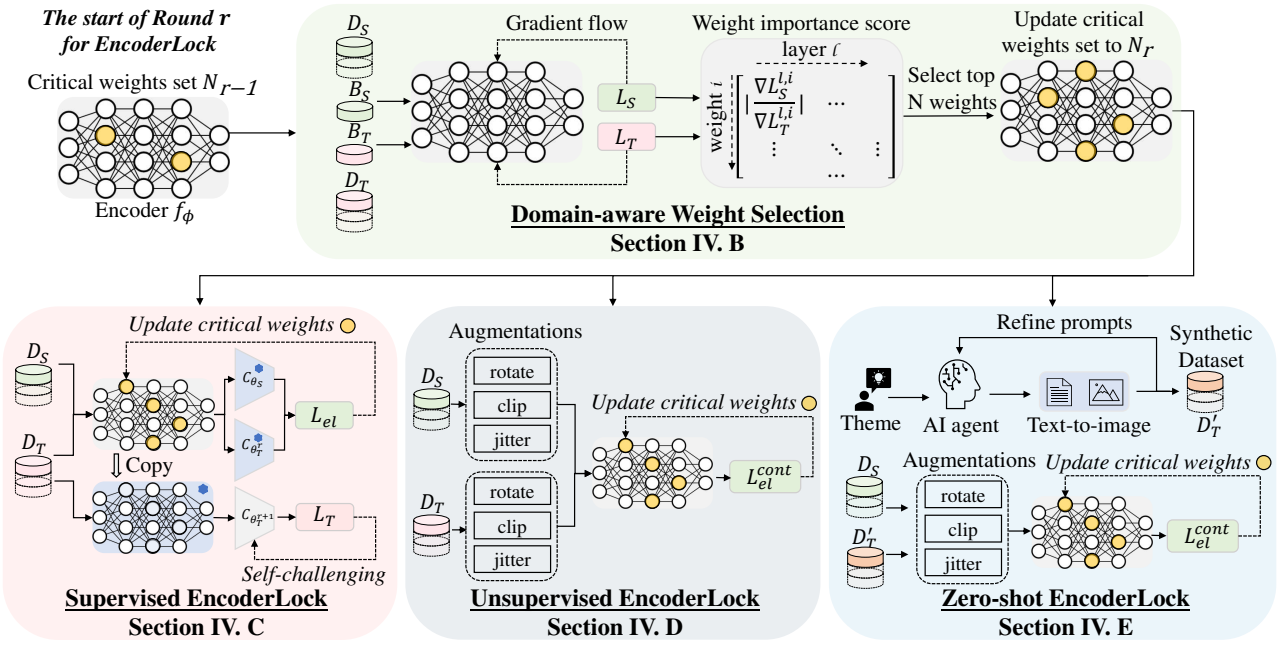

Fig. 2. **Overview of the proposed EncoderLock framework and paper organization**. The procedure in Round r includes: 1.*domain-aware critical weight selection algorithm*: take data batches $\mathcal{B}_\mathcal{S}$ and $\mathcal{B}_\mathcal{T}$ from the authorized source dataset $\mathcal{D}_\mathcal{S}$ and the prohibited target dataset $\mathcal{D}_\mathcal{T}$, respectively, and calculate the weight importance with gradients of loss $L_\mathcal{S}$ and $L_\mathcal{T}$ and choose critical weights to update for the round r as N_r, note here specific losses depend on different levels of accessibility of the target domain; 2. *EncoderLock weight update algorithm* (with three variants for the three levels of target domain dataset), utilizing the supervised EncoderLock loss L_{el}, unsupervised contrastive loss $L_{\text{el}}^{\text{cont}}$ and the generated synthetic dataset $D'_\mathcal{T}$, respectively.

- They can probe a pre-trained encoder using inputs from prohibited target domains and utilize the representations to train a local downstream classifier for inference. Although they query the encoder (as a service or local private model), it is a black-box with both structure and parameters unknown.
- Users can build their own downstream classifiers, customizing the classifier's hyper-parameters and fine-tuning the parameters with any learning rates and optimizers.
- Following the common setting of probing, we assume that the attacker has a small amount of data from the prohibited domain for fine-tuning (e.g., 10% from the target domain).

B. Model Owner

Model owners aim to safeguard the pre-trained encoder against malicious probing proactively. Following the common definition of transfer learning, the dataset on which the encoder is trained is defined as the **source domain (authorized domain)**, and the dataset that the (malicious) downstream classifier is trained for is the **target domain (prohibited domain)**. Moreover, we define data domains other than these two as **admissible domains**, indicating that the usage of the encoder on these domains is allowed but their performance is not guaranteed like the source domain.

Protection Objective. The major goal includes: restricting the pre-trained encoder from being probed for the prohibited domain; preserving its performance on the authorized domain.

Capabilities. The capabilities of the model owner include:

- The model owner has full control of the encoder to adjust the architecture, hyper-parameters, and parameters, and manage training strategies for the encoder before deploying it.
- The owner has no access to the user's dataset after deployment to detect if the probing samples belong to the prohibited domain, and has no knowledge about the probing process or downstream heads.
- The owner may have different levels of accessibility to the prohibited domain, from high to low: a) Level 1: The owner has a labeled target dataset; b) Level 2: The owner obtains the target dataset, but it is unlabeled; c) Level 3: The owner only has an abstract concept about the prohibited target domain (i.e., a text description), which is called a 'theme'. We propose three variants of EncoderLock to address different levels of accessibility, respectively.

IV. Proposed Framework: EncoderLock

In this work, we propose a new applicability authorization strategy for a pre-trained encoder against malicious probing, which we call *EncoderLock*. Fig. 2 depicts an overview of the framework, which consists of two major steps - domain-aware weight selection algorithm and specific weight updating algorithms catering to the different levels of accessibility of the target domain. By managing the weights, EncoderLock restricts the encoder from being probed on the prohibited target domain to extract useful information, while ensuring that the encoder correctly responds to authorized (source) inputs. Existing literature mostly focuses on protecting pre-trained models from being transferred [20], [21], while our EncoderLock targets pre-trained encoders, with several unprecedented challenges outlined in Section IV-A.

A. Design Objectives and Challenges for EncoderLock

In addition to the traditional design objective of controlling the transferability of pre-trained models to prohibited target domains [20], EncoderLock faces three additional challenges that need to be effectively addressed.

Challenge 1. Preservation of Integrity: The integrity of the encoder lies in maintaining the pre-learned knowledge about the authorized domains. One question is raised: how can EncoderLock make minimal modifications to the encoder to restrict it on the prohibited domain while preserving the integrity on the source domain?

Challenge 2. Robustness to malicious probing: When malicious users adjust the downstream heads for the prohibited domain with any learning rate and optimizer, how to successfully 'lock' the encoder against malicious probing?

Challenge 3. Different target domain data accessibility: In reality, as the defender (model owner) may have various levels of knowledge about the target domain, how should EncoderLock be designed for practical scenarios including unlabeled datasets or even no samples from prohibited domains?

B. Domain-aware Weight Selection

To address **Challenge 1**, we propose *domain-aware weight selection* to selectively update weights that are critical only for the target domain, thereby minimizing EncoderLock's negative impact on the integrity of the pre-trained encoder. Our strategy is motivated by two observations of DNN models: 1) *Weight importance varies across different domains*. Different sets of critical weights in the same model may respond to different domains, which can be measured by the weight's gradient magnitude. This notion has been exploited in achieving domain-specific pruning [59], effective fault injections on DNN parameters [60]–[62], and watermarking embedding [54]. Fig. 3 demonstrates one example of different weight importance for different domains. The model is a multi-layer perceptron network trained on MNIST (source domain), and the distribution of the weight gradients is shown in red color. When this model runs inference for another dataset USPS (target domain), the weights gradient profile is shown in blue color, distinctly different from that on MNIST. 2) *Over-parametrization*—DNNs often have more weights than required, with a large portion being insignificant. This characteristic is widely utilized in model compression for efficiency [63]–[65], where only critical weights are retained while others are pruned.

Our *Domain-aware Weight Selection* (DWS) algorithm is described in Function 1. The search process runs iteratively. In the r^{th} round, it uses datasets from the source and target domains, $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$, to search for and update the critical weight set $\mathcal{N}_r$. It is important to note that the composition of $\mathcal{D}_\mathcal{T}$ depends on the level of data accessibility. For supervised EncoderLock (Level-1), inputs from the target dataset have labels, and the loss $L_\mathcal{T}$ is calculated using cross-entropy, similar to $L_\mathcal{S}$. In the unsupervised and zero-shot EncoderLock (Level-2 and Level-3), the target dataset consists of unlabeled or synthetic images, and we propose using a contrastive loss for these unlabeled inputs, presented in Eq.(6). The weight

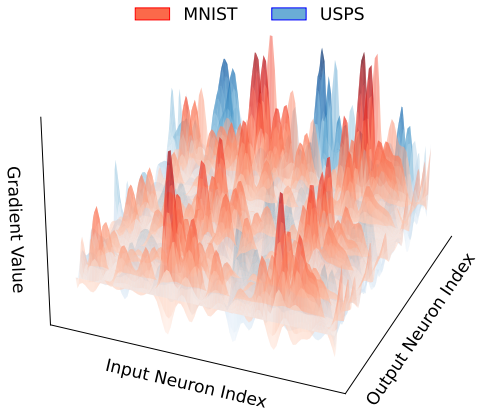

Fig. 3. **Visualization of weight importance in a pre-trained model—**The X-Y plane represents the weight matrix of a selected dense layer in a model trained on MNIST and probed for USPS. The color and height indicate each weight's importance to the output (the higher and darker, the more important).

Function 1 Domain-aware Weight Selection

Input: Source domain $\mathcal{D}_\mathcal{S}$, Target domain $\mathcal{D}_\mathcal{T}$, Pre-trained encoder parameters ϕ, Number of new critical weights N, Set of critical weights for the previous round $\mathcal{N}_{r-1}$
Output: Set of critical weights $\mathcal{N}_r$

function DWS($\mathcal{D}_\mathcal{S}$, $\mathcal{D}_\mathcal{T}$, ϕ, $\mathcal{N}_{r-1}$, N)
  Sample a training batch $\mathcal{B}_\mathcal{T}$ from $\mathcal{D}_\mathcal{T}$
  Sample a training batch $\mathcal{B}_\mathcal{S}$ from $\mathcal{D}_\mathcal{S}$
  /* Compute gradients for both batches */
  $\nabla L_\mathcal{T} \leftarrow$ ComputeGradients(ϕ, $\mathcal{B}_\mathcal{T}$)
  $\nabla L_\mathcal{S} \leftarrow$ ComputeGradients(ϕ, $\mathcal{B}_\mathcal{S}$)
  /* Compute scores and select critical weights */
  Compute score for i^{th} weight in l^{th} layer: $\left|\frac{\nabla L_\mathcal{T}^{l,i}}{\nabla L_\mathcal{S}^{l,i}}\right|$
  Select top N weights: $\text{argmax}_N \left|\frac{\nabla L_\mathcal{T}^{l,i}}{\nabla L_\mathcal{S}^{l,i}}\right|$
  $\mathcal{N}_r \leftarrow \mathcal{N}_{r-1} \cup \{\text{selected weights}\}$
  return $\mathcal{N}_r$
end function

importance score is defined as the magnitude ratio of gradients for the i^{th} weight in layer l between the target and source domains $|\nabla L_\mathcal{T}^{l,i}/\nabla L_\mathcal{S}^{l,i}|$. This score is used to identify weights that are critical to target domains but less crucial to the source.

The search process is iterated across R rounds, with N weights selected in each round, resulting in a total of $N \times R$ weights to update. The values of N and R are two hyperparameters that control the number of altered weights and will be discussed further in Section VI-A. Such design allows us to process the datasets in batches and implement the self-challenging training scheme (see Section IV-C2).

In Section IV-C to IV-E, we further discuss more details about how to update the selected weights and how to achieve robustness against downstream fine-tuning (**Challenge 2**). The method varies across different levels of accessibility to the target domain, as shown in three branches of Fig. 2.

C. Supervised EncoderLock

Level 1 EncoderLock is supervised, with a labeled target domain dataset. Specifically, given the source domain $\mathcal{D}_\mathcal{S}$ and target domain $\mathcal{D}_\mathcal{T}$, with $(x_\mathcal{S}, y_\mathcal{S}) \in \mathcal{D}_\mathcal{S}$ and $(x_\mathcal{T}, y_\mathcal{T}) \in \mathcal{D}_\mathcal{T}$

as the corresponding datasets, let f_ϕ denote the pre-trained encoder, and $C_{\theta_\mathcal{S}}$ and $C_{\theta_\mathcal{T}}$ denote the auxiliary downstream task classifiers for the source and target domains, respectively. Our objective is to find an optimal encoder ϕ^* that minimizes $L_\mathcal{S}$ but maximizes $L_\mathcal{T}$, which are expressed as:

$$L_\mathcal{S} = L(C_{\theta_\mathcal{S}}(f_\phi(x_\mathcal{S})), y_\mathcal{S}), \quad L_\mathcal{T} = L(C_{\theta_\mathcal{T}}(f_\phi(x_\mathcal{T})), y_\mathcal{T}) \quad (2)$$

where L is the classification loss function (i.e., cross-entropy loss) and is used to compute gradient in Algorithm 1 for weight selection. To restrict the impact on the encoder's generalizability, we require $\|\phi^* - \phi\|_0 \leq M$ (:=$N \times R$), where $\|\cdot\|_0$ is ℓ_0 norm and M signifies the weight change budget.

The fundamental supervised EncoderLock consists of three steps: 1) Domain-aware weight selection, 2) Non-transferability updating, and 3) Self-challenging downstream model training. We run these three steps iteratively for R rounds or until the accuracy of the auxiliary downstream classifier reaches the early stopping criterion. For the three different levels of target domain data accessibility, the weight search and update algorithms are similar, but with different loss functions. But the supervised EncoderLock, with its loss design for the output space, requires an additional self-challenging training step to ensure its robustness. We next discuss the other two design steps for supervised EncoderLock in detail.

1) Weight Updating for Non-transferability: With critical weights selected to update, we design a loss function in the form of Equation (1), focusing on the regularization term $R_\mathcal{T}$ to mitigate the malicious probing for the target domain. Previous regularization terms [20], [21] only consider the target domain, which leads to unstable performance especially when $L_\mathcal{S}$ and $L_\mathcal{T}$ are at different orders of magnitude. In particular, when $L_\mathcal{S}$ is very small (i.e., near zero), the introduction of $R_\mathcal{T}$ will cause a strong impact on $L_\mathcal{S}$. Therefore, we propose a new log-ratio regularization term considering both $L_\mathcal{S}$ and $L_\mathcal{T}$:

$$L_{el} = L_\mathcal{S} + R_\mathcal{T}, \text{ where } R_\mathcal{T} = \log(1 + \alpha \frac{L_\mathcal{S}}{L_\mathcal{T}}) \quad (3)$$

Such logarithmic regularization term gently penalizes the loss ratio between the source and target, with α moderating the balance between preserving the source domain accuracy and enforcing the target domain non-transferability. Consequently, the optimization objective for the encoder is defined as:

$$\phi^* = \arg\min_\phi L_{el}(\phi, \theta_\mathcal{S}, \theta_\mathcal{T}) \quad \text{s.t. } \|\phi^* - \phi\|_0 \leq M \ \ \forall \theta_\mathcal{S}, \theta_\mathcal{T} \quad (4)$$

2) Self-challenging Training Scheme: To update the encoder weights in supervised EncoderLock following Equation (4), we consider the auxiliary downstream classifier to compute $L_\mathcal{T}$. However, malicious users have full control of the downstream classifier, including adjusting the architecture and choosing the optimization method, and can fine-tune the model parameters based on the extracted features. Consequently, the performance of the pre-trained encoder and the classifier on the target domain can be improved. Relying solely on a fixed-weight target classifier could lead to vulnerability, where supervised EncoderLock may only be non-transferable for given auxiliary classifiers but not for others the malicious user opts for.

To improve robustness against any potential malicious probing for supervised EncoderLock, we propose a self-challenging training scheme with a minimax problem formulation as:

$$\phi^* = \arg\min_\phi \max_{\theta_\mathcal{T}} L_{el}(\phi, \theta_\mathcal{S}, \theta_\mathcal{T}) \quad \text{s.t.} \quad \|\phi^* - \phi\|_0 \leq M \quad (5)$$

Algorithm 1 Self-challenging Training Scheme

Input: Pre-trained encoder with ϕ, Source domain $\mathcal{D}_\mathcal{S}$, Target training dataset $\mathcal{D}_\mathcal{T}^{\text{train}}$, Target validation dataset $\mathcal{D}_\mathcal{T}^{\text{valid}}$, Number of critical weights N, Number of rounds R, Desired target accuracy α_{goal}.
Output: Encoder with supervised EncoderLock ϕ^*

```
for r = 1 to R do
    /* Initialize critical weights set for the first round*/
    if r == 1 then
        Initialize set N_{r-1} ← ∅
    end if
    /* Begin Domain-aware Weight Selection */
    N_r = DWS(D_S, D_T, φ, N_{r-1}, N)
    /* Minimax optimization for enhancing robustness */
    φ* ← optimize weights in N_r to minimize L_el (4)
    Initialize an auxiliary downstream C_T(·; θ_T)
    Fine-tune C_T using D_T^train with encoder φ*
    Compute accuracy α_T of C_T on D_T^valid
    /* Stop Criterion */
    if α_T < α_goal or ||φ* − φ||_0 > N × R then
        return φ*
    end if
end for
```

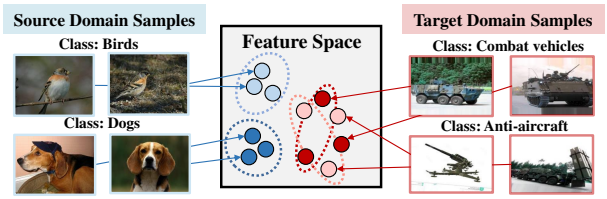

Fig. 4. **Design motivation of unsupervised EncoderLock**

Specifically, during the iterations of updating critical weights in the encoder part, we also adjust the target downstream models iteratively. It is noted that the training objective of the target downstream models will be adversarial to EncoderLock's applicability objective—the fine-tuning aims to extract useful features in the embeddings to enhance the target domain performance. The retrained downstream model adjusts itself frequently to create a *challenging* target downstream classifier that will increase L_{el} (decreasing $L_\mathcal{T}$), prompting the supervised EncoderLock to adjust more critical weights on the encoder part. Algorithm 1 outlines this self-challenging training process. To ensure the randomness of the target downstream classifiers, every iteration we retrain it from scratch (with a random initialization). The iterative training proceeds until the target downstream classifier's accuracy drops below a predefined threshold or reaches the maximum number of altered weights M. The self-challenging training scheme ensures a gradual and smooth reduction in the encoder's transferability, forcing the encoder part to extract features that are less useful for the target domain, thereby leading to more robust performance even when the attacker probes the downstream model adaptively.

D. Unsupervised EncoderLock

In this section, we address Level-2 accessibility of the target domain via *unsupervised EncoderLock*. This scenario is practically relevant when the goal is to prevent transferring to arbitrary sets of images while getting their labels is either

infeasible or expensive. Our method leverages the technique from self-supervised representation learning [47], [48], which builds a highly distinguishable feature space without labeling.

The design idea for unsupervised EncoderLock is as follows: for the latent embeddings of samples from the source domain, we aim to ensure their high discrepancy between classes; while for those from the prohibited target domain, our objective is to obfuscate the latent clusters boundary so that the embeddings would not contain much information about the class. As shown in Fig. 4, an expected encoder will automatically cluster the samples from the source domain but blur the class boundaries of the target domain. Due to such direct manipulation towards the encoder's feature space, the unsupervised EncoderLock is always robust to different downstream heads and doesn't require further self-challenging training.

Towards this goal, we introduce a self-supervised regularization term $R_{\mathcal{T}}$ to be used in Equation (3). Specifically, given a batch of samples from the target domain, we leverage data augmentation, including random crop, color jitter, or Gaussian blur [66], [67], to create a set of positive pairs and a set of negative pairs. Any pair with a sample and an augmented sample from the same original image is defined as positive, and we denote their feature space as $(z_i, \tilde{z}_i)$, where z_i is defined as the normalized embedding of the sample x_i using the encoder f. Any pair with augmented samples from different original images is defined as negative, denoted as $(z_i, \tilde{z}_j)_{i \neq j}$. We define the contrastive loss function L^{cont} as:

$$L^{\text{cont}} := -\frac{1}{N_B} \sum_{i=1}^{N_B} \log\left(\frac{\text{sim}(z_i, \tilde{z}_i)}{\sum_{j=1}^{N_B} \text{sim}(z_i, \tilde{z}_j)}\right) \tag{6}$$

where N_B is the batch size, and $\text{sim}(\cdot, \cdot)$ computes the cosine similarity between the normalized embeddings. We select pairs from $\mathcal{S}$ to compute $L_{\mathcal{S}}^{\text{cont}}$, and from $\mathcal{T}$ to compute $L_{\mathcal{T}}^{\text{cont}}$. They are used to compute gradients in Algorithm1.

The presented loss function aims to increase the similarity between any positive pairs but reduce what between negative pairs, effectively pushing the encoder to learn representations that clearly distinguish similar samples from dissimilar ones within feature space. We follow the regularization framework in Eq. (3) and penalize ratios between contrastive losses:

$$R_{\mathcal{T}}^{\text{cont}} = \log(1 + \alpha \frac{L_{\mathcal{S}}^{\text{cont}}}{L_{\mathcal{T}}^{\text{cont}}})$$

For the unsupervised EncoderLock, self-challenging training is not necessary because this loss function directly penalizes the discrepancy of the feature space for the target dataset. Therefore, as shown in Fig. 2, the procedure of unsupervised EncoderLock in one round includes: 1) Domain-aware weight selection with L^{cont}; 2) Update Encoder's Non-transferability with $R_{\mathcal{T}}^{\text{cont}}$, without retraining the *challenging* classifier.

E. Zero-shot EncoderLock

In this section, we address Level-3 accessibility of the target domain for EncoderLock, where the model owner even has no target samples. This represents the most practical and relevant scenario, as the definition of harmful content is often vague in real-world applications. For instance, in most cases, a DNN product's user guidelines regulate prohibited content using text descriptions of unethical or sensitive material. How to turn such vague scope description into representative and comprehensive target domain dataset is a challenge. We define the basic knowledge about the target domain as a *theme*, which can be in the form of a text description, keywords, or reference figures. Using the target theme, zero-shot EncoderLock aims to generate a synthetic dataset for applicability authorization without relying on real-world samples or labels.

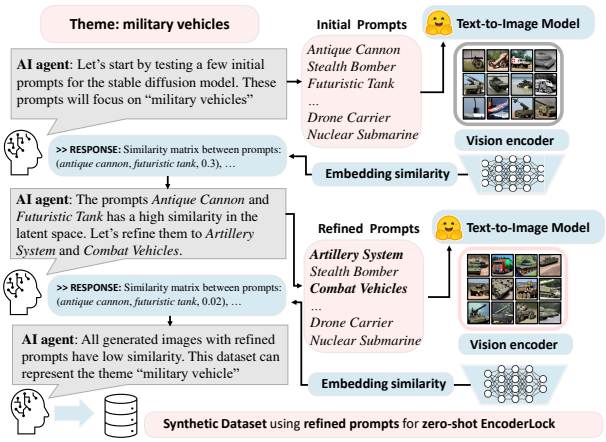

Fig. 5. **Building synthetic datasets for zero-shot EncoderLock**

Fig. 5 presents the framework of zero-shot EncoderLock, illustrating the process of generating a synthetic dataset for "*military vehicles*". Section V-D will showcase the full results. First, we employ a large language model (e.g., GPT-4 [68]) as an AI agent to generate text inputs, known as prompts, for the given theme. These prompts are then fed into pre-trained text-to-image models (e.g., CLIP [69] and Stable Diffusion [70]) to generate the synthetic dataset. To ensure the synthetic dataset comprehensively covers the target domain, we introduce a prompt refining framework. Using a pre-trained vision encoder, we extract latent features from the synthetic images and compute pairwise similarity scores between the initial prompts. This similarity matrix serves as feedback to the AI agent, enabling it to analyze the scores, identify similar prompts, and refine them accordingly. For example, as shown in Fig. 5, *Antique Cannon* and *Futuristic Tank* exhibit high similarity due to their shared barrel feature. Consequently, the AI agent revises these prompts to be *Artillery System* and *Combat Vehicles*. The refinement process continues until all prompt pairs demonstrate low similarity or the similarity stops decreasing. Finally, we employ the synthetic dataset generated in the last round for unsupervised EncoderLock training, resulting in an encoder with restricted transferability to the target "theme."

V. Experiments

A. Experiment Setup

Baselines: As the first of its kind work addressing malicious probing of pre-trained encoders, there is no prior work for direct comparison with our EncoderLock. The closest related work is the SOTA non-transferable learning, including *NTL* [20] and *CUTI* [21]. For such baseline work, we adopt the pre-trained

TABLE I
DATASETS USED IN EVALUATION OF ENCODERLOCK

Dataset	Abbr.	Type	Feat.	Supervised	Unsupervised	Zero-shot
MNIST [71]	MT	digits	Datasets that are used in the baselines [20], [21]. All samples are resized into $(32, 32, 3)$ and the label space is 10.	√	√	-
USPS [72]	UP	digits		√	√	-
SVHN [73]	SN	digits		√	√	-
MNIST-M [74]	MM	digits		√	√	-
Synthetic Digits [75]	SD	digits		√	√	-
CIFAR-10 [76]	CF	image		√	-	-
STL-10 [77]	ST	image		√	-	-
EMNIST [78]	EM	char.	47-class characters	√	-	-
CIFAR-100 [76]	CF100	image	100-class images	√	-	-
ImageNette [79]	-	image	High resolution images with the shape of $(224, 224, 3)$.	√	√	√
ImageWoof [79]	-	image		√	√	√
Military Vehicle [80][1]	-	image		√	√	√

model, freeze the encoder part, and train a new downstream head on the prohibited target domain to evaluate the baseline model's resistance against malicious probing.

Datasets: Table I lists the twelve datasets for our evaluation. In addition to the five digits datasets used in the previous work [20], [21], we also assess datasets with larger label spaces, i.e., EMNIST (47 classes) and CIFAR-100 (100 classes). By utilizing text-to-image generators, zero-shot EncoderLock is evaluated with more complex datasets. We test ImageNette [79] as the source dataset and the military vehicle dataset [80] as the prohibited target[2] following a practical scenario. Further, we evaluate the influence of EncoderLock on three admissible domains–ordinary vehicles[3], weapons[4], and animals[5].

Models: Our method is assessed using three prevalent DNN architectures: VGG-11 [81], ResNet-18 [82], and Vision Transformer (ViT) [22]. We leverage the supervised pre-trained models for VGG-11 and ResNet-18[6] and fine-tune them on various authorized domains (source). The early convolutional layers (residual blocks) of VGG-11 and ResNet-18 are considered encoders, while the output dense layer(s) are used as downstream heads for probing. ViTs utilize a vision encoder structure that is trained with self-supervised learning.

Hyperparameters: Hyperparameters for supervised and unsupervised EncoderLock can be found in Appendix A.

Metric: The metric to quantify the encoder's resistance to malicious probing is the *relative* accuracy drop in both the target and the source domains, defined as $\frac{acc_o - acc_m}{acc_o}$, where acc_o is the probing accuracy of the original encoder, and acc_m is the one when modified with protection methods. A higher accuracy drop indicates a strong restriction on given domains. We expect the accuracy drop in the source domain to be low for preserving the model integrity, while the accuracy drop in the target domain to be high for robustness to malicious probing. In Section V-E, we further introduce the Performance Protection Index (PPI) to evaluate the restriction on prohibited domains and the influence on authorized or admissible domains simultaneously for comparison.

Platform: Our implementation uses PyTorch 1.5.0 on Ubuntu 18.04.6 with NVIDIA TITAN RTX.

Training cost: With this experimental setup, training EncoderLock requires between 0.1 and 6 GPU hours, depending on the encoder architecture and dataset size. Detailed training costs for various configurations are provided in Appendix D.

[2]https://www.kaggle.com/datasets/amanrajbose/millitary-vechiles
[3]https://www.kaggle.com/datasets/marquis03/vehicle-classification
[4]https://huggingface.co/datasets/Kaludi/data-csgo-weapon-classification
[5]https://www.kaggle.com/datasets/alessiocorrado99/animals10/code
[6]https://pytorch.org/vision/stable/models.html

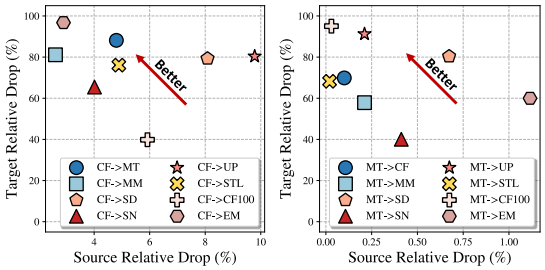

Fig. 6. **Accuracy drop across distinct source and target domains—**It assesses the transferability of VGG-11 encoder with CF (**Left**) and MT (**Right**) as the source. Each data point illustrates the simultaneous impact on accuracies on the source and target domain with supervised EncoderLock.

B. Evaluating the Supervised EncoderLock

Table II demonstrates the performance of supervised EncoderLock across the datasets of digits for VGG-11 and ResNet-18. We compare the accuracy after probing the encoder with corresponding default downstream classification heads, before and after applying supervised EncoderLock, report the relative accuracy drop on the source domain (columns $Drop_{\mathcal{S}}$) and the average accuracy drop on the target domains ($Drop_{\mathcal{T}}$), and also present the average percentage of weight change (column ΔW). The experimental results of VGG-11 reveal that EncoderLock exhibits a steep reduction (up to 78.70%) in performance on the target domains while ensuring minimal degradation on the source domain (highlighted in bold, up to 3.53%). Moreover, the accuracy degradation on ResNet-18 shows even a better restriction on the prohibited domain, from 71.73% to 86.02%. In contrast, the degradation of ResNet-18 encoder with EncoderLock on the authorized domain is minimized from 0.28% to 1.65%. Moreover, with less than 0.08% of the weights changed on average, the supervised EncoderLock preserves a higher generalizability of the pre-trained encoder to the admissible domains and avoids the catastrophic forgetting of the encoder's pre-learned knowledge. It will be further discussed in Section V-E as a comparison between different EncoderLock and baseline methods.

In addition to the accuracy drop, we find that the complexity of the datasets (domains) affects the EncoderLock's performance. For instance, on VGG-11, we observe that restricting transferring from a complex domain (e.g., SN, MM, and SD, comprising RGB-colored digit images) to a simple domain (e.g., MT and UP, including grayscale images) is more challenging. Specifically, the supervised EncoderLock requires more weights to be changed, and yields a smaller target accuracy drop and a larger source accuracy drop, when the source domain is SN which is a more realistic, 3-channeled digits dataset (the Street View House Number) A similar phenomenon is also observed in ReseNet-18 supervised EncoderLock, with the highest average weight modification at 0.041%. Moreover, the similarity between the source and target domains also

TABLE II

SUPERVISED ENCODERLOCK PERFORMANCE: THE ENCODER TRANSFERABILITY—PRE AND POST-ENCODERLOCK ACCURACY ARE REPORTED, DESIGNATED AS 'BEFORE(%) ⇒ AFTER (%)'. BOLD VALUES SHOW ACCURACIES ON THE SOURCE DOMAIN

Source \ Target	MT	UP	SN	MM	SD	ΔW	$Drop_{\mathcal{S}}$	$Drop_{\mathcal{T}}$
	VGG-11: 133M Parameters							
MT	**99.53 ⇒ 99.32**	96.35 ⇒ 8.47	43.74 ⇒ 18.98	68.24 ⇒ 18.05	69.65 ⇒ 13.67	0.63‰	0.21% ↓	78.70% ↓
UP	97.70 ⇒ 11.35	**97.91 ⇒ 94.94**	58.23 ⇒ 16.86	65.24 ⇒ 15.43	87.10 ⇒ 16.93	0.43‰	2.97% ↓	76.16% ↓
SN	95.30 ⇒ 19.72	92.68 ⇒ 15.89	**94.04 ⇒ 90.51**	71.51 ⇒ 32.17	96.96 ⇒ 15.66	2.50‰	3.53% ↓	76.62% ↓
MM	98.85 ⇒ 12.71	94.67 ⇒ 17.99	53.80 ⇒ 23.80	**94.24 ⇒ 92.71**	85.89 ⇒ 28.20	0.99‰	1.53%↓	75.28% ↓
SD	97.13 ⇒ 20.19	93.57 ⇒ 18.68	90.40 ⇒ 41.42	71.53 ⇒ 40.91	**99.83 ⇒ 98.89**	1.78‰	0.94% ↓	64.13% ↓
	ResNet-18: 11.4M Parameters							
MT	**99.48 ⇒ 99.12**	93.77 ⇒ 13.16	41.47 ⇒ 19.67	70.02 ⇒ 20.77	72.48 ⇒ 16.69	0.31 ‰	0.28% ↓	71.73% ↓
UP	95.10 ⇒ 13.89	**96.11 ⇒ 95.69**	33.72 ⇒ 11.36	55.79 ⇒ 9.04	61.76 ⇒ 16.53	0.29 ‰	0.44% ↓	76.98% ↓
SN	94.65 ⇒ 9.53	88.04 ⇒ 15.65	**91.06 ⇒ 90.12**	66.52 ⇒ 11.36	95.08 ⇒ 10.45	0.41 ‰	1.03% ↓	86.02% ↓
MM	98.82 ⇒ 24.05	92.33 ⇒ 12.81	48.18 ⇒ 7.01	**91.49 ⇒ 90.39**	78.03 ⇒ 18.53	0.13 ‰	1.20% ↓	80.87% ↓
SD	96.76 ⇒ 10.39	91.68 ⇒ 8.47	86.74 ⇒ 30.70	68.56 ⇒ 9.95	**99.42 ⇒ 97.77**	0.18 ‰	1.65% ↓	82.53% ↓

TABLE III

UNSUPERVISED ENCODERLOCK PERFORMANCE ON ENCODER (VGG-11) TRANSFERABILITY

Source \ Target	MT	UP	SN	MM	SD	ΔW	$Drop_{\mathcal{S}}$	$Drop_{\mathcal{T}}$
MT	**99.53 ⇒ 99.22**	96.35 ⇒ 16.84	43.74 ⇒ 19.61	68.24 ⇒ 12.26	69.65 ⇒ 17.90	0.12‰	0.31% ↓	73.51% ↓
UP	97.70 ⇒ 45.02	**97.91 ⇒ 96.44**	58.23 ⇒ 9.59	65.24 ⇒ 13.88	87.10 ⇒ 12.66	0.22‰	1.50% ↓	75.41% ↓
SN	95.30 ⇒ 20.74	92.68 ⇒ 17.09	**94.04 ⇒ 94.33**	71.51 ⇒ 50.85	96.96 ⇒ 94.01	0.21‰	0.31% ↑	47.93% ↓
MM	98.85 ⇒ 40.26	94.67 ⇒ 30.74	53.80 ⇒ 33.97	**94.24 ⇒ 93.30**	85.89 ⇒ 55.79	0.15‰	0.99%↓	49.68% ↓
SD	97.13 ⇒ 76.68	93.57 ⇒ 86.75	90.40 ⇒ 75.31	71.53 ⇒ 27.23	**99.83 ⇒ 99.44**	0.20‰	0.39% ↓	26.74% ↓

affects the supervised EncoderLock's performance, e.g., the non-transferability of SD→SN is the worst as they are similar.

Unlike prior research [20], [21] that only examines the applicability authorization between domains that share the same label space (e.g., 0∼9 digits), during malicious probing, one could aspire to transfer the queried features from the encoder to domains with distinctly different label spaces. Thus, we also evaluate the performance of supervised EncoderLock on VGG-11 under various circumstances: a) transition between distinct task types (CF to MT); b) variation in the size of the label space for similar tasks (CF to CF100); c) changes in both the task type and the label space size (CF to EM). The results are presented in Fig. 6. Overall, supervised EncoderLock achieves good performance across all these transfer tasks, manifesting a much higher drop in the target domain than the source. Similar to experiments on digit datasets, the more distinct the target dataset is from the source dataset, the better performance. For instance, supervised EncoderLock performs well when transferring across different tasks, i.e., transitioning from digits to images (CF) and vice versa; while it results in a lower performance when transferring from CF to CF100 as these two share a large number of similar features. Notably, when MT is the source domain and CF100 is the target domain, EncoderLock can significantly reduce the accuracy on CF100 to 1.19%, closely resembling random guessing across the 100 classes, i.e., the encoder fails to extract features. While for transferring between MT and EM (both digits), the accuracy only drops to 32.04%. We posit that a pre-trained encoder tends to capture more intricate features when applied on similar domains, while there will be more distinct features when the source and target domains diverge significantly. We characterize the similarity between datasets using MMD in Appendix B.

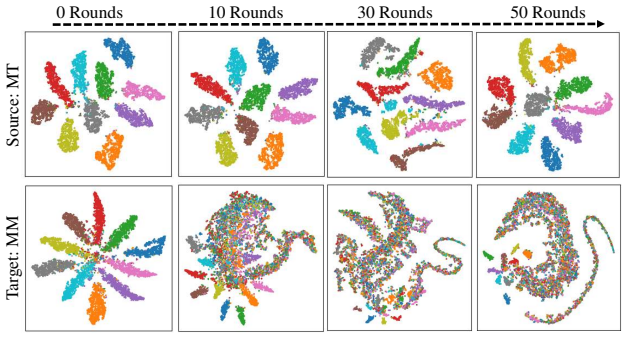

Fig. 7. **Unsupervised EncoderLock**-Latent Space Change via Rounds

C. Evaluating the Unsupervised EncoderLock

The unsupervised EncoderLock addresses the challenge when the model owner has no access to the true labels of samples from the target domain. To demonstrate the effectiveness of the proposed contrastive loss in Eq. (6), we conduct the same evaluation on digits datasets with results on VGG-11 shown in Table III and ResNet-18 in Appendix C Table VII. The unsupervised EncoderLock also successfully restricts the encoder's transferability to the target domain and maintains the encoder's integrity on the source domain, with a small number of weight changes.

Compared with supervised EncoderLock, the unsupervised EncoderLock demonstrates better performance in preserving accuracy within the source domain, but worse performance in reducing the target accuracy. This is due to the proposed $R_{\mathcal{T}}^{\text{cont}}$, utilizing contrastive loss, necessitates a more discriminative latent space for the source domain by reducing $L_{\mathcal{S}}^{\text{cont}}$. To further illustrate, in Fig. 7 we visualize the change of the latent space

from both the source (MT) and target (MM) domains using t-SNE [83]. With more rounds of unsupervised EncoderLock (indicating more training epochs and more weight updates), the source domain remains class-discriminative, while the target domain becomes less discriminative. However, some small clusters of classes can still be observed, which leads to a higher accuracy when we fine-tune a downstream classifier on such a latent space on the target domain.

D. Evaluating the Zero-shot EncoderLock

For zero-shot EncoderLock, we utilize the GPT-4 API as the AI agent to interpret the semantic meaning of the prohibited theme[7] with a pre-trained stable diffusion model for text-to-image generation[8] to create synthetic datasets. To show the effectiveness of the AI agent and prompts refining (Fig. 5), we test the zero-shot EncoderLock under three scenarios: 1) The prompts are created *manually* (without AI agents); 2) An AI agent generates prompts *randomly*; 3) The AI agent *generates* prompts and *refines* them. We utilize 10 prompts to the generative model for 1,000 fake images as the prohibited dataset. Prompts and synthetic images can be found in Appendix K.

We evaluate one-shot EncoderLock's performance on the source domain (ImageNette [79]) and the target domain (a real military vehicle dataset [80]) with a pre-trained encoder using ResNet-18. Furthermore, we select three admissible domains to evaluate EncoderLock's impact on other domains, neither source nor target. Considering the semantics of 'military vehicles', we utilize two admissible datasets with closer semantic meaning and one unrelated. 1). **Ordinary vehicles**: A 10-class normal vehicle classification dataset related to the prohibited domain in 'vehicle', i.e., sedan, SUV, and bus. 2). **Weapons**: An 11-class game-based weapon classification dataset, related to the prohibited domain in 'military', i.e., AK-47, Famas, and UPS. 3). **Animals**: A 10-class animal classification dataset with no obvious semantic meaning with the prohibited domain, i.e., chicken, cat, and butterfly.

Fig. 8 shows the performance of different EncoderLock variants on authorized, prohibited, and three admissible domains. We present the testing accuracy degradation on each domain versus the percentage of critical weights modified. The original pre-trained encoder has an average accuracy 96.59% on the authorized domain and 60.55% on the prohibited domain. Supervised EncoderLock shows the strongest performance restriction on the prohibited domain (11.48%) and meanwhile preserving a high accuracy on the authorized domain (94.17%). The primary reason behind the efficacy of the supervised EncoderLock is its well-defined prohibited domain. This clarity allows for targeted modifications of critical weights. As a result, EncoderLock can enhance specific aspects of the encoder's performance while maintaining its overall generalization capability. Unsupervised EncoderLock demonstrates the second highest performance with a label-free prohibited domain by restricting the target performance to 18.34% and keeping the accuracy on the source at 93.43%. The introduction of contrastive loss penalizes more general features than the supervised loss, causing a larger degradation on the authorized and admissible domains. Zero-shot EncoderLock works on the most challenging accessibility to the prohibited domain. Without any data from it, using prompts from military fans (manual prompts in Appendix K) shows similar to the one-time initial prompt from the AI agent. The prompts and generated synthetic images are not general enough to describe the entire prohibited domain, thereby leading to the smallest restriction on the military vehicle dataset. (Manual: 41.81%, AI initial: 38.61%) When we refine the AI prompts for more general features of the prohibited domain with the presented refinement algorithm, there is a noticeable increase in the EncoderLock restriction ability (23.69%). However, such prompts cause the largest accuracy degradation on the authorized domain (92.86%), as its wide restriction on the encoder's features. In addition to the relevance of prompts, the synthetic dataset's quality also affects the strength of EncoderLock. Specifically, a high-quality synthetic dataset, generated after large number of inference iterations and bearing small noise, enhances the defense capability of EncoderLock. More detailed results are shown in Appendix F.

Other than the performance of authorized and prohibited domains, our evaluation on the admissible domains indicates the impact of EncoderLock on the generalizability to unknown data. In particular, all five EncoderLock's restrictions on the prohibited 'military vehicles' show higher penalties on the 'weapon' dataset. On the other hand, the impact on the encoders' performance on ordinary vehicles is small, similar to what from the animal dataset. We will further present a deep analysis of this phenomenon in Section V-F, where we show the encoder's attention shifting from the attack module of the military vehicles, i.e., the barrel. As a result, it has a lower effect on distinguishing between various types of vehicles but has a significant impact on different types of firearms.

In Fig. 8, we also present the probing performance of EncoderLock when the portion of critical weights to change varies. In particular, the pre-trained encoder's performance keeps dropping on the prohibited domain but its performance degradation on the authorized and admissible domain shows a 'degradation peak' at the point of changing half of critical weights. This phenomenon is likely due to the integrity of EncoderLock's critical weights updating process affected by the random sampling. A complete updating process with either supervised loss (4) or unsupervised loss (6) ensures the encoder cannot perform well on the target domain but still effective on the source domain, as shown the good performance when changing 100% critical weights. However, only updating half of them will break the encoder's integrity on the source domain and the admissible domain as it breaks the network connectivity between critical weights, leading to unstable EncoderLock. Such phenomenon also shows in the standard deviation on the curves in Fig. 8, where when only changing half of critical weights, the EncoderLock performance has the largest variance.

[7]https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4

[8]https://huggingface.co/CompVis/stable-diffusion-v1-4

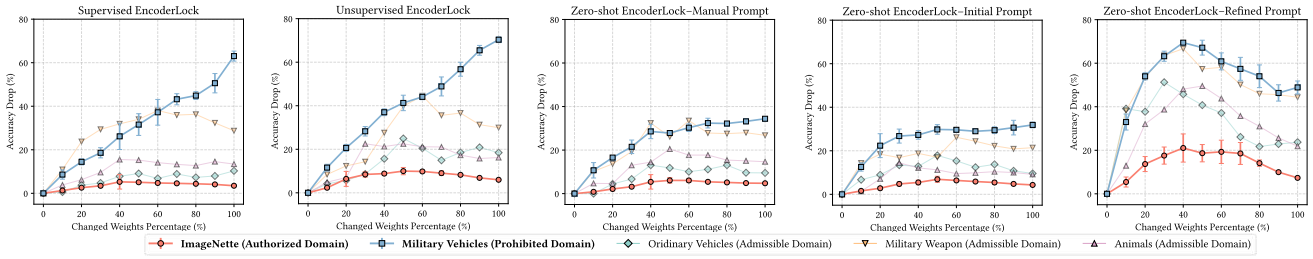

Fig. 8. **Comparison among different EncoderLock**–on authorized, prohibited, and admissible domains. We randomly select a percentage of domain-aware weights as the X-axis, as the optimization process of domain-aware weights. The authorized and prohibited domains are run 5 times with error bars plotted.

E. Comparison with Prior Methods

We compare EncoderLock with SOTA baselines: NTL [20] and CUTI [21] in protection against malicious probing. Specifically, we probe the encoder on the target dataset with fine-tuned downstream heads. Moreover, we propose a new stable metric–*Protection Performance Index (PPI)*–to measure the performance in restricting on the target domain ($\mathcal{D}_\mathcal{T}$) relative to what in retention on the base domain ($\mathcal{D}_\mathcal{B}$), defined as:

$$\text{PPI}(\mathcal{D}_\mathcal{T}, \mathcal{D}_\mathcal{B}) = \frac{acc_o^\mathcal{T}/acc_m^\mathcal{T}}{acc_o^\mathcal{B}/acc_m^\mathcal{B}} \tag{7}$$

PPI measures the ratio of change in performance before and after applying protection for a prohibited target domain with an authorized or admissible base domain. A higher PPI indicates better non-transferability and resistance to malicious probing.

In Fig. 9, we compare the PPI of supervised and unsupervised EncoderLock with the baselines in five pairs of authorized and prohibited domains when probing the downstream heads by epochs. Our proposed supervised EncoderLock shows the best non-transferability performance, while the unsupervised EncoderLock also demonstrates good performance overall. In addition, we also present the PPI between admissible domains and prohibited domains in Fig. 10. The PPI of supervised and unsupervised EncoderLock is still better than the baselines. However, the advantage is not obvious due to the prohibited and admissible domains are highly similar as they are all digital datasets. Therefore, critical weights on prohibited domains often overlap with those in admissible ones, causing a higher accuracy drop as the weights are not optimized during the updating process of the EncoderLock. A good example can be found in Fig. 8, where we analyze the admissible domain with distinct semantic meaning (i.e., animals versus military vehicle), EncoderLock can preserve the encoder generalizability to semantic unrelated domains. Additional numerical results can be found in Appendix H.

When we fine-tune the downstream classifier on the encoder trained using NTL and CUTI, the encoder performance on the source domain may decrease considerably, or its performance on the target domain may still be high. This indicates that previous methods are not suitable for the scenario of malicious probing. To explain this phenomenon, we visualize the source (MM) and target (UP) domains on the latent space with different trained encoders in Fig. 11, where the colors represent the true labels of the testing dataset. The self-challenging scheme employed in supervised EncoderLock effectively preserves a higher discrepancy within the source domain, while very little class-related information is discernible within the target domain. The unsupervised EncoderLock, despite lacking class information, still successfully reduces class distinguishability on the target domain. In contrast, NTL [20] penalizes the Maximum Mean Discrepancy between the source and target latent spaces, showcasing the distribution difference between these domains, but samples from the same class still cluster together in the target domain. Therefore, its transferability can be resumed via fine-tuning. CUTI [21] only considers target performance during training and shows the worst non-transferability–its target domain is clearly clustered by classes.

F. Interpretation of EncoderLock

To further understand the changes in the encoders generated by different variants of EncoderLock, we use the encoders trained from Section V-D and visualize their decision-making process with Gradient-weighted Class Activation Mapping (GradCAM) [84], as illustrated in Fig. 12. Specifically, the GradCAM attribution is computed for the last convolutional layer in the encoder (ResNet-18) and is upsampled to act as a mask added to the original input (the cool-warm heatmap in Fig. 12). The highlighted red part indicates the feature that the encoder focuses on to make its prediction. We observe that the original pre-trained encoder focuses on the English Springer correctly, and it performs well on the military dataset (tanks) as its focus is moved to the main gun barrel of the tank. However, the supervised EncoderLock, which has less effect on the source domain data, switches the encoder's focus to the tank track, leading the fine-tuned downstream classifier to make a wrong prediction as 'Armored combat support vehicles'. The unsupervised EncoderLock, aiming to blur the entire feature space's class-discrepancy, generates a more vague interpretation of the decision process—the focus is mainly on the vehicle but not on specific features of tank. In Appendix J, we also visualize the GradCAM results on three different admissible domains. The findings indicate that the attack module of military weapons also experiences a loss of focus in the model, resulting in a significant impact of EncoderLock on the weapon dataset.

We also visualize the GradCAM of zero-shot EncoderLock with different types of prompts. The manual prompts show less effectiveness in the model's non-transferability, still focusing on some useful features such as the tank roof and gun barrel. The AI-agent-generated prompts (with or without refinement)

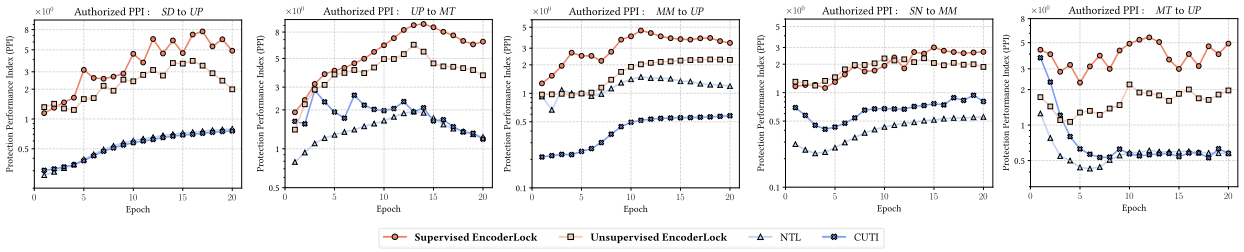

Fig. 9. **Comparison on Authorized PPI**–between EncoderLock with NTL [20] and CUTI [21] on different pairs of authorized and prohibited domains. **The higher the better.** Probing for multiple epochs can't increase the performance on prohibited domains but keeps its performance on authorized domains.

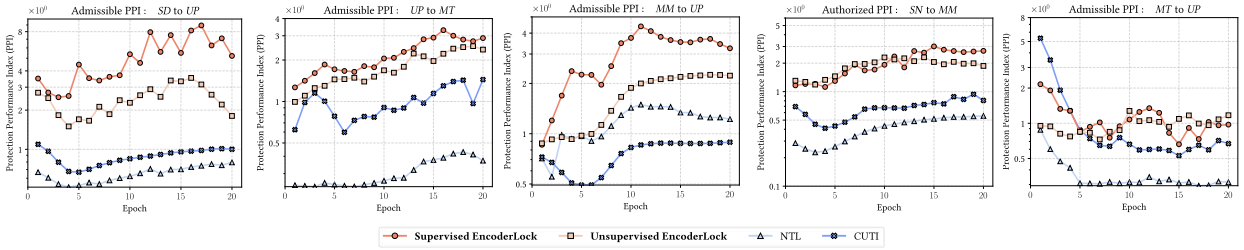

Fig. 10. **Comparison on Admissible PPI**–between EncoderLock with NTL [20] and CUTI [21] on different pairs of admissible and prohibited domains. **The higher the better.** EncoderLock shows the minimal impact on admissible domains while restricting the encoder's performance on prohibited domains.

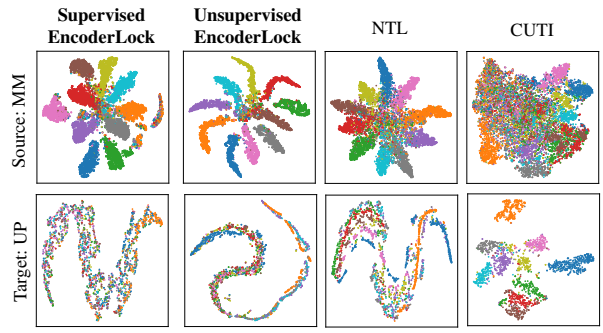

Fig. 11. **Latent Feature Visualization**–Feature Space of Different Methods

have a strong effect on the source domain, especially the green part (bash in the source figure), considering 'green' is likely related to the camouflage. With the proposed prompt refinement process, the synthetic dataset obfuscates most of the useful features of the military theme, leading to a very vague focus on the target domain. The interpretation of EncoderLock further reinforces its effectiveness: the supervised EncoderLock has the highest performance as it has the ground truth labels, and therefore able to move away the encoder's focus on specific features; while the unsupervised and zero-shot EncoderLock directly cause the latent feature space to be less informative, leading to all input features having similar importance.

G. Real-world Case Study

Previous evaluations mainly focus on small encoders extracted from supervised-trained DNN models with basic architectures (e.g., VGG-11 and ResNet-18). To demonstrate that EncoderLock is practical in protecting real-world encoders, we apply it to a public encoder based on Vision Transformer (ViT) released by Facebook [22]. This encoder is a transformer pre-trained on a large collection of images (ImageNet-1k [85] with a resolution of 224×224) in a self-supervised fashion. The input images are presented to the model as a sequence of fixed-size patches (the patch size is 16×16). ViT learns an inner representation of images that can extract features useful for downstream tasks with probing heads (downstream models). As reported, the training process requires approximately 2.6 days with a computational power of 16 GPUs. The pre-trained encoder represents a valid IP, and our proposed EncoderLock for applicability authorization aims to protect the IP.

We first evaluate the performance of the supervised EncoderLock, considering ImageNette [79] and CIFAR-100 [69] as the source domains and the military dataset and Imagewoof dataset as the target domains. The ImageWoof dataset is a selected subset of ImageNet with different types of dogs. Note that to fit the input resolution of ViT, the CIFAR-100 inputs are resized to 224×224. We also evaluate the ViT with EncoderLock on some simple datasets, as shown in Appendix E. In Fig. 13, we visualize the accuracy after fine-tuning the downstream classifier on the source and target domains, respectively, versus the number of weight changes. We observe that for the pre-trained ViT, the supervised EncoderLock still works effectively in restricting the target domain performance and preserving the source domain performance. In addition, the largest accuracy drop often happens in the early epochs (early sets of domain-specific weights). This demonstrates the effectiveness of the proposed domain-aware weight selection algorithm when the most important (target-domain sensitive) weights are selected.

Similarly, we apply unsupervised and zero-shot EncoderLock on the same ViT encoder between the ImageNette (i.e., authorized) and military (i.e., prohibited) datasets, and the results

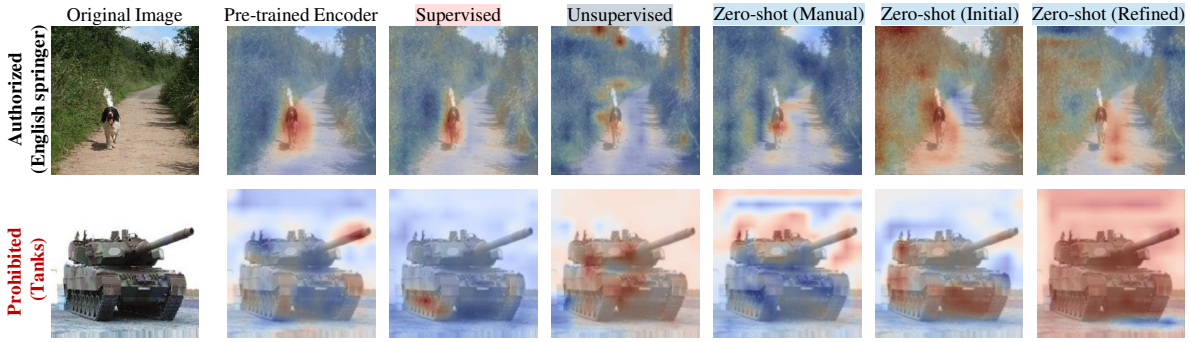

Fig. 12. **Interpretation of Different EncoderLock using GradCAM [84]**–the red parts highlight the focus of encoder to make decisions.

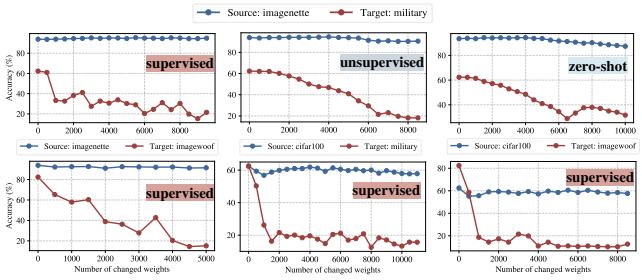

Fig. 13. **Evaluation on a Pre-trained ViT**: **Top row** - source (target) accuracy versus number of changed weights for different levels of EncoderLock; **Bottom row**: performance of supervised EncoderLock on different datasets.

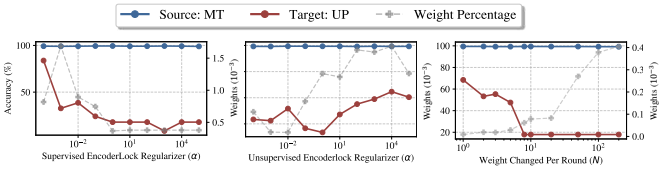

Fig. 14. **Ablation studies–**a) regularizer weights for supervised EncoderLock α; b) regularizer weights for unsupervised EncoderLock ;c) changed weights number per round(N); The probing accuracy on the source and target domains are reported on the **left** y-axis and the changed weight percentage is on the **right** y-axis. Original accuracy on source and target are 99.53% and 94.91%.

are shown in Fig. 13. All three variations of EncoderLock effectively limit the encoder's performance on the prohibited domain. Notably, supervised (21.56%) and unsupervised (18.15%) EncoderLock provide stronger protection than the zero-shot version (29.26%), due to their higher accessibility during training. And supervised EncoderLock outperforms in maintaining the accuracy in the authorized domain, as the more accurate penalization on the labeled prohibited domain. In conclusion, all three variants of EncoderLock demonstrate effectiveness for safeguarding real-world pre-trained encoders.

VI. Discussions

A. Ablation Studies of EncoderLock

1) EncoderLock on Probing with Various Architectures: As shown in Challenge 2, the attacker takes full control of the downstream classifier and can optimize it for malicious probing. In this section, we further evaluate the protection performance of EncoderLock with various downstream head structures. Taking supervised EncoderLock on VGG-11 as an example, we delve into diverse depths and widths for the linear probing heads, including varying numbers of layers and hidden dimensions (no hidden layer when there is only one layer in the classifier). The detailed results of these experiments are presented in Table IV, which shows that one can leverage the features from our pre-trained encoders to achieve impressive performance ($> 99\%$) on the source task but only achieve up to 17.89% on the target task, across a comprehensive collection of configurations of the downstream classifiers. This demonstrates the effectiveness of the self-challenging training scheme.

2) EncoderLock on Probing with More Prohibited Data: We consider a real-world attacker who can be adaptive in attacking the encoders. In our initial malicious probing scenario, we assume an attacker uses a small amount of data (10%) to attempt redirecting the pre-trained encoder. However, once the attacker becomes aware of the defense mechanism, s/he can probe the downstream head with a larger amount of prohibited data. We assess the resistance of EncoderLock against such possible attacks in Appendix G. Thanks to self-challenging training in supervised EncoderLock and the manipulation of the latent feature space in unsupervised EncoderLock, our approach demonstrates strong resistance to malicious probing, even when the attacker gains access to entire prohibited datasets.

3) EncoderLock with Various Hyperparameters: we employ the transfer scenario from MT to UP as an example.

Regularization Term (α): The regularization term in Eqn. (3) plays an important role in balancing the model integrity on the source domain and preventing malicious probing on the target domain. We vary the weight α and Fig. 14 (a) (b) show the effect of the regularization term for supervised and unsupervised EncoderLock, respectively. It is worth noting that the regularization term ensures that all α values sustain the original performance on the source domain, due to the log-scale term. Comparing different α, we choose $\alpha = 10^3$ for supervised EncoderLock and $\alpha = 1$ for unsupervised EncoderLock.

Number of Critical Weights to Update per Round (N): Fig. 14 (c) inspects the repercussions of varying the number of critical weights (N) selected in each round, in the range of

TABLE IV
PERFORMANCE OF THE SUPERVISED ENCODERLOCK ON VARIOUS CLASSIFIER CONFIGURATIONS—SOURCE(MT) TO TARGET(UP).

# Layers	1	2					3					4				
Hidden dim.	/	256	512	1024	2048	4096	256	512	1024	2048	4096	256	512	1024	2048	4096
size (M)	0.25	6.42	12.85	25.7	51.4	102.8	6.49	13.11	26.75	55.59	119.59	6.56	13.38	27.8	59.8	136.37
Acc_S^{el} (%)	99.12	99.09	99.21	99.20	99.29	99.20	99.32	99.29	99.24	99.27	99.36	99.29	99.29	99.26	99.19	99.30
Acc_T^{el} (%)	17.89	17.89	17.89	13.15	17.89	17.89	9.87	17.89	17.89	17.89	17.89	13.15	17.89	8.47	17.89	17.89

1 to 200. When N is too small, the training of EncoderLock is slow and can not converge even with the maximum number of rounds ($R = 100$). A larger N will cause more weight change for EncoderLock, reducing the encoder's generalizability. Our selection of N can be found in Table V.

B. Security Analysis of EncoderLock

The security of an encoder protected with EncoderLock can be assessed by setting an accuracy threshold (acc_{th}) on the prohibited domain, with the accuracy drop of it on the authorized domain below a certain constraint (ϵ). For example, acc_{th} can be defined by the accuracy of a *train-from-scratch* model, and the ϵ is set at 2%, as outlined in Appendix I. In this case, the encoder is deemed secure because an attacker lacks incentive for malicious probing, with its performance no better than direct training.

C. Future Works

In this work, we have demonstrated the effectiveness of EncoderLock for different levels of domain data accessibility. However, EncoderLock still requires the EaaS provider to clearly specify the prohibited domain, meaning that the provider should know what the encoder is allowed to do and what should be prohibited. This causes inconvenience in automatic detection and restriction on any unknown 'theme' of harmful tasks. One potential avenue for future work is to incorporate EncoderLock with toxicity content detection utilizing large language models [86], [87] to automatically identify and restrict prohibited domains without relying on explicit specifications from the model owner. This approach would further enhance the flexibility and adaptability of EncoderLock. Furthermore, while we have showcased the effectiveness of EncoderLock in image classification tasks, pre-trained encoders are useful for many other applications, such as image generation and semantic segmentation. Extending EncoderLock to these different tasks may unlock additional potential in controlling the encoder's transferability. The main challenge will be how to combine different forms of loss terms (i.e., generation loss or segmentation loss) with the loss for a pre-trained encoder.

VII. CONCLUSIONS

In this work, we address a new security issue arising during the probing process of pre-trained encoders: restricting the applicability to harmful prohibited domains. We recognize a realistic challenge about the data accessibility to prohibited domains and propose supervised, unsupervised, and zero-shot EncoderLock for different levels of knowledge of prohibited domains. We propose novel and effective domain-aware weight selection and self-challenging training to maintain the encoder's integrity while protecting it against malicious probing. Our evaluation has validated the efficacy of EncoderLock in resisting malicious probing across various domains and encoders.

ACKNOWLEDGMENT

This work was supported in part by National Science Foundation under grants CNS-2212010, SaTC-1929300, IUCRC-1916762, CNS-2239672, CNS-2326597, and CCF-2340482.

REFERENCES

[1] Y. Belinkov, "Probing classifiers: Promises, shortcomings, and advances," *Computational Linguistics*, 2022.
[2] R. Cao, "Putting representations to use," *Synthese*, p. 151, 2022.
[3] J. C. White, T. Pimentel, N. Saphra, and R. Cotterell, "A non-linear structural probe," *arXiv preprint arXiv:2105.10185*, 2021.
[4] K. He, X. Chen *et al.*, "Masked autoencoders are scalable vision learners," in *PCVPR*, 2022, pp. 16 000–16 009.
[5] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *Ieee Access*, vol. 8, 2020.
[6] X. Chen, S. Wang *et al.*, "Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning," *NeurIPS*, 2019.
[7] A. Chronopoulou, C. Baziotis, and A. Potamianos, "An embarrassingly simple approach for transfer learning from pretrained language models," *arXiv preprint arXiv:1902.10547*, 2019.
[8] M. Iman, H. R. Arabnia, and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technologies*, 2023.
[9] Y. Dong, W.-j. Lu *et al.*, "Puma: Secure inference of llama-7b in five minutes," *arXiv preprint arXiv:2307.12533*, 2023.
[10] K. Gupta, N. Jawalkar *et al.*, "Sigma: secure gpt inference with function secret sharing," *Cryptology ePrint Archive*, 2023.
[11] R. Liu *et al.*, "Secdeep: Secure and performant on-device deep learning inference framework for mobile and iot devices," in *IoTDI*, 2021.
[12] Clarifai, "general-image-embedding3," 2020. [Online]. Available: https://clarifai.com/clarifai/main/models/general-image-embedding
[13] OpenAI, "Openai's embeddings api," 2020, accessed: 4 October 2024. [Online]. Available: https://platform.openai.com/docs/guides/embeddings
[14] W. Qu, J. Jia, and N. Z. Gong, "Reaas: Enabling adversarially robust downstream classifiers via robust encoder as a service," in *NDSS*, 2023.
[15] Y. Tan, G. Long *et al.*, "Federated learning from pre-trained models: A contrastive learning approach," *NeurIPS*, vol. 35, 2022.
[16] B. Cheatham, K. Javanmardian, and H. Samandari, "Confronting the risks of artificial intelligence," *McKinsey Quarterly*, pp. 1–9, 2019.
[17] International Women's Day, "Gender and ai: Addressing bias in artificial intelligence," https://www.internationalwomensday.com/Missions/14458/Gender-and-AI-Addressing-bias-in-artificial-intelligence, 2024.
[18] The White House Office of Science and Technology Policy, "Ai bill of rights: Algorithmic discrimination protections," www.whitehouse.gov/ostp/ai-bill-of-rights/algorithmic-discrimination-protections-2/, 2024.
[19] B. Marr, "Is artificial intelligence dangerous? 6 ai risks everyone should know about," *Forbes. Retrieved May*, vol. 13, p. 2022, 2018.
[20] L. Wang *et al.*, "Non-transferable learning: A new approach for model ownership verification and applicability authorization," in *ICLR*, 2022.
[21] L. Wang, M. Wang, D. Zhang, and H. Fu, "Model barrier: A compact un-transferable isolation domain for model intellectual property protection," in *CVPR*, 2023, pp. 20 475–20 484.
[22] M. Caron *et al.*, "Emerging properties in self-supervised vision transformers," in *ICCV*, 2021, pp. 9650–9660.
[23] P. Marcelino, "Transfer learning from pre-trained models," *Towards data science*, vol. 10, no. 330, p. 23, 2018.
[24] S. Parisi *et al.*, "The unsurprising effectiveness of pre-trained vision models for control," in *ICML*. PMLR, 2022, pp. 17 359–17 371.

[25] L. Yuan, D. Chen *et al.*, "Florence: A new foundation model for computer vision," *arXiv preprint arXiv:2111.11432*, 2021.

[26] Y. Bengio *et al.*, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, 2013.

[27] Y. Shen, C. Guo *et al.*, "Financial feature embedding with knowledge representation learning for financial statement fraud detection," *Procedia Computer Science*, vol. 187, pp. 420–425, 2021.

[28] H.-C. Yi *et al.*, "Graph representation learning in bioinformatics: trends, methods and applications," *Briefings in Bioinformatics*, 2022.

[29] G. Zhong, L.-N. Wang, X. Ling, and J. Dong, "An overview on data representation learning: From traditional feature learning to recent deep learning," *The Journal of Finance and Data Science*, 2016.

[30] X. Han, Z. Zhang *et al.*, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.

[31] X. Qiu, T. Sun *et al.*, "Pre-trained models for natural language processing: A survey," *Science China technological sciences*, 2020.

[32] H. Wang, J. Li *et al.*, "Pre-trained language models and their applications," *Engineering*, 2022.

[33] N. Tajbakhsh, J. Y. Shin *et al.*, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[34] H. Bahng *et al.*, "Exploring visual prompts for adapting large-scale models," *arXiv preprint arXiv:2203.17274*, 2022.

[35] Y. Gan, Y. Bai *et al.*, "Decorate the newcomers: Visual domain prompt for continual test time adaptation," in *AAAI*, 2023.

[36] M. Jia, L. Tang *et al.*, "Visual prompt tuning," in *ECCV*. Springer, 2022, pp. 709–727.

[37] L. Yang, Y. Wang *et al.*, "Fine-grained visual prompting," *NeurIPS*, vol. 36, 2024.

[38] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "On the relationship between self-attention and convolutional layers," *arXiv:1911.03584*, 2019.

[39] K. Han, Y. Wang *et al.*, "A survey on vision transformer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 1, pp. 87–110, 2022.

[40] M. Gao, Q. Wang *et al.*, "Tuning pre-trained model via moment probing," in *ICCV*, 2023, pp. 11 803–11 813.

[41] A. Ravichander, Y. Belinkov, and E. Hovy, "Probing the probing paradigm: Does probing accuracy entail task relevance?" *arXiv preprint arXiv:2005.00719*, 2020.

[42] Z.-F. Wu, C. Mao *et al.*, "Structured model probing: Empowering efficient transfer learning by structured regularization," in *CVPR*, 2024.

[43] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia*. Springer, 2008, pp. 21–49.

[44] D. A. Reynolds *et al.*, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[45] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[46] I. Goodfellow, J. Pouget-Abadie *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[47] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.

[48] K. He, H. Fan *et al.*, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020, pp. 9729–9738.

[49] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2020.

[50] D. Hendrycks *et al.*, "Using self-supervised learning can improve model robustness and uncertainty," *NeurIPS*, vol. 32, 2019.

[51] T. Zhou, S. Ren, and X. Xu, "Archlock: Locking dnn transferability at the architecture level with a zero-cost binary predictor," in *The Twelfth International Conference on Learning Representations*, 2024.

[52] R. Ding, L. Su, A. A. Ding, and Y. Fei, "Non-transferable pruning," in *European Conference on Computer Vision*. Springer, 2025, pp. 375–393.

[53] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *USENIX Security 21*, 2021, pp. 1937–1954.

[54] P. Yang, Y. Lao, and P. Li, "Robust watermarking for deep neural networks via bi-level optimization," in *ICCV*, 2021, pp. 14 841–14 850.

[55] J. Zhang, Z. Gu *et al.*, "Protecting intellectual property of deep neural networks with watermarking," in *ASIACCS*, 2018, pp. 159–172.

[56] M. Alam *et al.*, "Deep-lock: Secure authorization for deep neural networks," *arXiv preprint arXiv:2008.05966*, 2020.

[57] A. Chakraborty, A. Mondai, and A. Srivastava, "Hardware-assisted intellectual property protection of deep learning models," in *DAC*, 2020.

[58] J. Gu, J. Kuen *et al.*, "Self-supervised relationship probing," *NeurIPS*, vol. 33, pp. 1841–1853, 2020.

[59] B. Liu *et al.*, "Transtailor: Pruning the pre-trained model for improved transfer learning," in *AAAI*, vol. 35, no. 10, 2021.

[60] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *ICCAD*. IEEE, 2017, pp. 131–138.

[61] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *ICCV*, 2019, pp. 1211–1220.

[62] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," in *DAC*, 2019, pp. 1–6.

[63] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[64] Z. Liu, M. Sun *et al.*, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[65] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[66] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, 2019.

[67] A. Mikołajczyk *et al.*, "Data augmentation for improving deep learning in image classification problem," in *IIPhDW*. IEEE, 2018, pp. 117–122.

[68] J. Achiam *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[69] A. Radford, J. W. Kim *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*. PMLR, 2021, pp. 8748–8763.

[70] R. Rombach, A. Blattmann *et al.*, "High-resolution image synthesis with latent diffusion models," 2021.

[71] Y. LeCun, L. Bottou *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998.

[72] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[73] Y. Netzer, T. Wang *et al.*, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, no. 2. Granada, 2011.

[74] Y. Ganin, E. Ustinova *et al.*, "Domain-adversarial training of neural networks," *The journal of machine learning research*, 2016.

[75] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*. PMLR, 2015, pp. 1180–1189.

[76] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[77] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*, 2011, pp. 215–223.

[78] G. Cohen, S. Afshar *et al.*, "Emnist: Extending mnist to handwritten letters," in *IJCNN*. IEEE, 2017, pp. 2921–2926.

[79] J. Howard, "imagenette." [Online]. Available: https://github.com/fastai/imagenette/

[80] A. Bose, "Military vehicles dataset," https://www.kaggle.com/datasets/amanrajbose/millitary-vechiles, n.d., accessed: 2023-04-23.

[81] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2019.

[82] K. He, X. Zhang *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[83] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[84] R. R. Selvaraju, M. Cogswell *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017.

[85] J. Deng, W. Dong *et al.*, "Imagenet: A large-scale hierarchical image database," in *CVPR*. IEEE, 2009, pp. 248–255.

[86] M. Phute, A. Helbling *et al.*, "Llm self defense: By self examination, llms know they are being tricked," in *ICLR*, 2023.

[87] Y.-S. Wang and Y. Chang, "Toxicity detection with generative prompt-based inference," *arXiv preprint arXiv:2205.12390*, 2022.

[88] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[89] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[90] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69.

[91] Y. Li, Z. Zhang *et al.*, "Modeldiff: Testing-based dnn similarity comparison for model reuse detection," in *ISSTA*, 2021, pp. 139–151.

[92] J. Howard and S. Gugger, "Fastai: A layered api for deep learning," *Information*, vol. 11, no. 2, p. 108, 2020.

APPENDIX A
HYPERPARAMETER CONFIGURATION OF ENCODERLOCK

TABLE V
HYPERPARAMETERS USED IN THE EXPERIMENT

Hyperparameters	N	R	α	LR
Supervised EncoderLock	100	100	$1,000$	0.01
Unsupervised EncoderLock	200	100	10	0.01

In this section, we provide the hyper-parameters configuration in this work. During training EncoderLock, we use $1,000$ samples from the training dataset. During fine-tuning of the downstream classifiers, we use 10% of the training data and evaluate with the entire testing data (usually 20% of the data). The fine-tuning process uses the Adam optimizer [88] with adaptive learning rate scheduling [89] and an early stopping criterion [90] (patience=10). For the default supervised EncoderLock configuration, we use $N = 100$, $R = 100$, and $\alpha = 10^3$. For the default unsupervised EncoderLock configuration, we use $N = 200$, $R = 100$, and $\alpha = 10^1$. Hyperparameters are evaluated in Section VI-A.

APPENDIX B
DATASET SIMILARITY

TABLE VI
DATASETS' FEATURE SPACE COSINE SIMILARITY

	MT	UP	MM	SN	SD	EM	CF10	STL10	CF100
MT	0.999	**0.707**	0.577	0.452	0.706	**0.942**	0.404	0.303	0.448
UP	**0.712**	0.999	0.551	0.652	0.892	0.786	0.395	0.276	0.425
MM	0.579	0.570	0.993	0.570	0.606	0.590	0.505	0.549	0.5151
SN	0.467	0.650	0.570	0.998	0.777	0.521	0.522	0.455	0.548
SD	0.712	0.891	0.587	0.768	0.999	0.791	0.452	0.345	0.483
EM	**0.938**	0.780	0.553	0.514	0.794	0.999	0.404	0.296	0.432
CF10	0.408	0.398	0.499	0.519	0.454	0.405	0.996	0.831	0.972
STL10	0.302	0.280	0.566	0.459	0.347	0.301	0.836	0.993	0.788
CF100	0.450	0.428	0.507	0.557	0.470	0.438	0.971	0.801	0.995

Evaluating the transferability of our EncoderLock inherently involves understanding the similarities between datasets, as this not only influences encoder transfer learning performance but also reflects the intrinsic characteristics of the data domains. Quantifying domain similarity is challenging, yet crucial for a comprehensive evaluation. To address this, we adopt an approach inspired by previous work [91], utilizing cosine similarity as a metric to compare the features of input samples across different domains. We employ a widely-used feature extractor, a PyTorch pre-trained VGG-16 model, to extract latent features from pairs of data domains and calculate their cosine similarity. The results, presented in Table VI, corroborate the observations made in Section V-B regarding the high similarity between the MT domain and the UP and EM domains—highlighted in bold within the table. This supports the notion that similarities in feature space significantly impact the transferability of EncoderLock.

APPENDIX C
UNSUPERVISED ENCODERLOCK ON RESNET-18

In this section, we present the additional results using ResNet-18 for the unsupervised EncoderLock in Table VII.

TABLE VII
UNSUPERVISED ENCODERLOCK'S PERFORMANCE ON ENCODER (RESNET-18) TRANSFERABILITY

Source \ Target	MT	UP	SN	MM	SD	ΔW	$Drop_S$	$Drop_T$
MT	**99.48 ⇒ 98.87**	93.77 ⇒ 10.26	41.47 ⇒ 11.27	70.02 ⇒ 32.04	72.48 ⇒ 16.17	3.27 ‰	0.61% ↓	73.45% ↓
UP	95.10 ⇒ 37.66	**96.11 ⇒ 94.95**	33.72 ⇒ 18.15	55.79 ⇒ 18.67	61.76 ⇒ 15.18	2.91 ‰	1.21% ↓	62.13% ↓
SN	94.65 ⇒ 22.62	88.04 ⇒ 6.98	**91.06 ⇒ 89.55**	66.52 ⇒ 13.92	95.08 ⇒ 88.9	1.41 ‰	1.66% ↓	63.43% ↓
MM	98.82 ⇒ 17.94	92.33 ⇒ 17.14	48.18 ⇒ 18.18	**91.49 ⇒ 90.97**	78.03 ⇒ 43.83	1.19 ‰	0.57% ↓	67.34% ↓
SD	96.76 ⇒ 50.06	91.68 ⇒ 21.72	86.74 ⇒ 30.86	68.56 ⇒ 17.68	**99.42 ⇒ 97.75**	2.58 ‰	1.67% ↓	65.80% ↓

The unsupervised EncoderLock also shows promising result in applicability authorization.

APPENDIX D
TRAINING COST OF ENCODERLOCK

Architecture	Level	Source	Target	SIZE	DWS (s)	DWU (s)	SC (s)
ResNet-18	sup.	MT	UP	32	0.771 ± 0.147	5.85 ± 0.021	2.33 ± 1.59
ResNet-18	unsup./zero-shot	MT	UP	32	1.03 ± 0.086	45.8 ± 0.126	-
VGG-11	sup.	MT	UP	32	0.821 ± 0.275	8.66 ± 0.039	8.75 ± 9.11
VGG-11	unsup./zero-shot	MT	UP	32	1.34 ± 0.498	45.1 ± 0.122	-
ResNet-18	sup.	ImageNette	Military	224	1.45 ± 0.217	59.2 ± 2.23	53.1 ± 10.2
ResNet-18	unsup./zero-shot	ImageNette	Military	224	1.82 ± 0.286	455.2 ± 1.03	-

We present the training time costs of supervised, unsupervised, and zero-shot EncoderLock on our platform equipped with an RTX TITAN GPU. Note the zero-shot EncoderLock has similar computational complexity as the unsupervised version, because they employ the same training strategy. We choose to examine two representative source-target domain pairs: a smaller digits pair (MT and UP) with an input size of $32 \times 32 \times 3$, and a more complex image pair (ImageNette and Military) with an input size of $224 \times 224 \times 3$. Additionally, we assess the training costs associated with two different encoder architectures, ResNet-18 and VGG-11. We break down the training complexity into the three key steps described in EncoderLock: domain-aware weight search (DWS), domain-aware weight updating (DWU), and self-challenging training (SC). The results demonstrate that the primary computational burden lies in the DWU step, and for supervised EncoderLock, the self-challenging phase is also costly, where the classifier must be retrained to enhance robustness against adversarial attacks. Furthermore, we find that DWS and DWU in unsupervised and zero-shot EncoderLock incur higher computational costs compared to the supervised version. This is attributed to the contrastive learning-based loss function (6). When comparing different input sizes, higher-resolution images lead to substantially longer training time because of the larger feature maps to compute during the forward pass of the encoder, especially in the DWU process. Large models also require more training time. It is worth noting that such training cost is a one-time expense, and there is no performance impact on the protected encoder during inference.

APPENDIX E
NUMERICAL RESULTS – SUPERVISED ENCODERLOCK (VIT)

Following a similar setting in Section V-G, we consider that the model provider aims to prevent the pre-trained ViT encoder on ImageNette from being transferred to a specified simple unauthorized domain. In particular, we evaluate EncoderLock on four target datasets: CF, ST, MT, and CF100. To fit the input size of ViT, we resize the target input to $224 \times 224 \times 3$ and monitor the accuracy degradation on both the target datasets and

ImageNet [92], the source dataset. During training, we follow the ViT fine-tuning instructions and use the last hidden state as the extracted feature space and connect to a single-dense-layer output classifier. From results shown in Table VIII, we find that EncoderLock reaches the non-transferability design goal — reducing the ViT performance on the target domain by 65.8% but keeping the accuracy on the source domain with a small accuracy drop of 2.15%. Our experimental results demonstrate that EncoderLock is effective when applied to a large encoder pre-trained on a large amount of samples.

TABLE VIII
ENCODERLOCK ON VIT–SOURCE TASK (IMAGENETTE [79])

Target	$Acc_{\mathcal{S}}^{org}$	$Acc_{\mathcal{T}}^{org}$	$Acc_{\mathcal{S}}^{el}$	$Acc_{\mathcal{T}}^{el}$	$Drop_{\mathcal{S}}$	$Drop_{\mathcal{T}}$
CF	87.26	74.99	86.83	36.70	0.49% ↓	51.9% ↓
ST		73.50	86.80	10.53	0.53% ↓	85.7% ↓
MT		92.06	85.20	47.56	2.36% ↓	48.3% ↓
CF100		53.31	82.62	12.07	5.32% ↓	77.4% ↓

APPENDIX F
IMPACT OF THE SYNTHETIC DATASET QUALITY ON ZERO-SHOT ENCODERLOCK

In addition to prompt relevance, the generation quality of the synthetic dataset significantly affects the performance of zero-shot EncoderLock. We measure the quality with two metrics: the noise level of the synthetic images and the generation quality of the diffusion model. To ensure a fair comparison, we use the same set of prompts (refined prompts from the prohibited domain) for zero-shot EncoderLock and select an encoder with the same degradation level on the authorized domain as reported in Section V-D (greater than 92%). We report the EncoderLock performance in the format of $(acc_m^{\mathcal{S}}, acc_m^{\mathcal{T}})$. Note that the defense goal of EncoderLock is a higher $acc_m^{\mathcal{S}}$ and lower $acc_m^{\mathcal{T}}$.

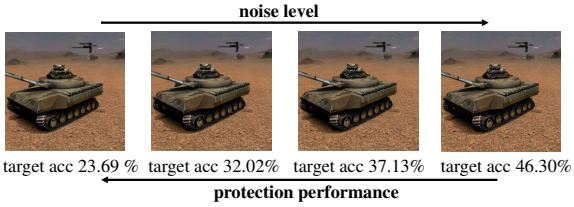

Fig. 15. Zero-shot EncoderLock performance with different noise levels

Fig. 15 shows an example image with different levels of Gaussian noise. Introducing random noise into the images reduces EncoderLock's ability to restrict the prohibited domain. We assess the impact of different noise levels, at $\sigma = 1$, $\sigma = 5$, and $\sigma = 10$, respectively. The zero-shot EncoderLock performance degrades to (91.90%, 32.02%), (91.85%, 37.13%), and (92.20%, 46.30%), where the performance on the noise-free synthetic dataset is (92.86%, 23.69%). High level of noise significantly reduces the synthetic quality, leading to poorer performance of the zero-shot EncoderLock.

We regenerate the synthetic datasets with varying number of inference iterations in the stable diffusion model. The

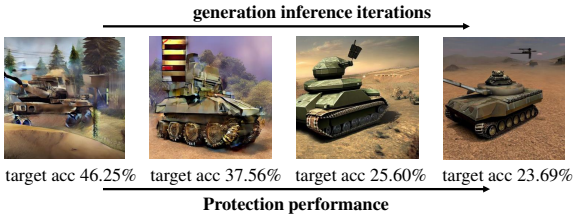

Fig. 16. Zero-shot EncoderLock performance with different diffusion qualities

generation quality improves as we increase the iteration count from 5, 10, 20, to 50 (the value used in the original setting), as shown in Fig. 16. Consequently, the protection performance improves from (92.71%, 46.25%) to (92.25%, 37.56%), (92.31%, 25.60%) and (92.86%, 23.69%). A higher-quality synthetic dataset provides clearer potential features of the prohibited domain, thus enhancing the restriction performance. However, increasing the number of inference iterations in the generator leads to higher computational costs. Running 5 inference iterations achieves a speed of 8.2 items/s, whereas 50 inference iterations reduce the speed to 0.8 items/s.

APPENDIX G
ENCODERLOCK PERFORMANCE ON VARIOUS DATA

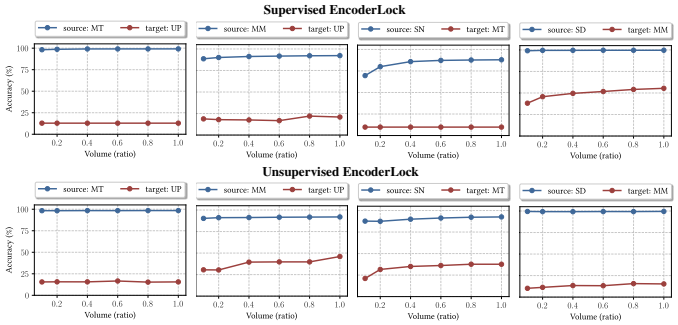

Fig. 17. EncoderLock performance for various volumes of probing data.

We evaluate the scenario where an attacker probes the encoder using varying amounts of probing data, ranging from 10% to the entire dataset, in both supervised and unsupervised settings. The results are illustrated in Fig. 17. Notably, for the prohibited target domain, the accuracy remains low even when the attacker utilizes the full prohibited dataset, while the protection slightly degrades with more data. For the authorized source domain, the accuracy remains consistently high even with a small amount of probing data. These results demonstrate the robustness of both supervised and unsupervised versions of EncoderLock against malicious probing attempts.

APPENDIX H
COMPARE WITH PREVIOUS WORK

In this section, we present more comparison results with baselines. From Table IX, we compare the baseline methods and the proposed supervised EncoderLock and unsupervised

EncoderLock between different pairs of digit datasets. The observation is similar to our conclusion in Section V-E. Specifically, under the condition of fine-tuning the downstream model fine-tuning, the proposed methods outperform baselines.

TABLE IX
COMPARISON THE EFFECTIVENESS OF ENCODERLOCK ON THE TARGET DOMAIN AND ITS PERFORMANCE ON OTHER DOMAINS WITH BASELINES [20], [21]. IN THIS TABLE, **BOLD** TEXT INDICATES THE BEST PERFORMANCE, UNDERLINED DENOTES THE SECOND-BEST PERFORMANCE. THE GRAY ROW DENOTES ORIGINAL TRANSFER ACCURACY.

Methods \ Domain	Source	Target	Other Domains		
	MM	UP	MT	SN	SD
Original Accuracy	94.2%	94.7%	98.9%	53.8%	85.9%
NTL [20]	81.6%	77.3%	97.9%	30.4%	62.1%
CUTI [21]	66.6%	90.6%	96.0%	**52.3%**	**82.0%**
Supervised EncoderLock	**93.5%**	**17.8%**	98.8%	39.3%	69.1%
Unsupervised EncoderLock	93.3%	30.7%	**98.9%**	49.0%	78.4%

APPENDIX I
SECURITY ANALYSIS–TRAIN-FROM-SCRATCH ACCURACY

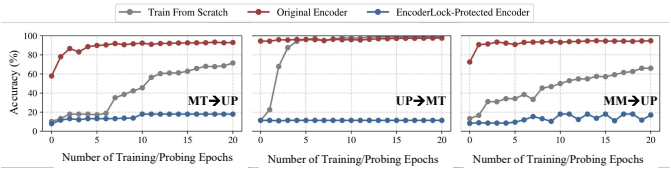

Fig. 18. EncoderLock's Performance Versus Train-from-scratch

Here we apply the security assessment definition in Section VI-B on three example pairs of domains: *MT to UP*, *UP to MT*, and *MM to UP* for the supervised EncoderLock on VGG-11. Their accuracy drops on the authorized domain are 0.07%, 0.25%, and 0.17%, respectively, all below the accuracy drop constraint (ϵ = 2%). Fig. 18 shows the probing performance of the EncoderLock-protected and unprotected encoders on the prohibited domain, compared to the accuracy of the "train-from-scratch" model. It demonstrates that starting from the EncoderLock-protected encoder allows the model to achieve lower accuracy and faster convergence on the prohibited domain, than a model trained from scratch. Therefore, the protected encoder can be considered **SECURE**, as an attacker would have no motivation to perform malicious probing. By contrast, the original encoder is **NOT SECURE**, as the accuracy (red) is always higher than train-from-scratch accuracy (gray) as shown in Fig. 18.

APPENDIX J
ENCODERLOCK GRADCAM ON ADMISSIBLE DOMAINS

In this section, we present the additional results to visualize the admissible domains with GradCAM in Fig. 19.

APPENDIX K
GENERATED PROMPTS & IMAGES

The (refined) prompts for generating synthetic datasets in zero-shot EncoderLock. The theme is **military vehicles**.

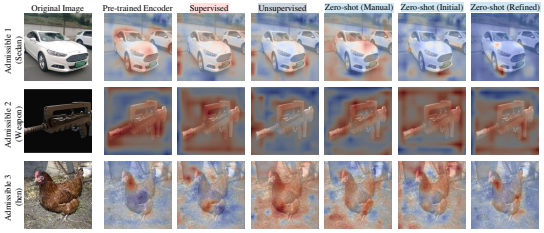

Fig. 19. **Interpretation of Different EncoderLock using GradCAM [84]**– the red parts highlight the focus of encoder to make decisions.

Manual Prompts. See Figure 20

- Armored Personnel Carrier
- Anti-tank Combat Vehicle
- Tactical Missile Vehicle
- Forward Command Vehicle
- Communication Support Vehicle
- Artillery Tractor
- Logistic Support Transport Vehicle
- Tank
- Self-propelled Artillery
- Multi-functional Infantry Vehicle

Initial Prompts: The synthetic images shown in Figure 21

- futuristic tank, stealth design
- antique cannon, ceremonial use
- amphibious assault vehicle, coastal operations
- drone carrier truck, mobile base
- armored medical evacuation vehicle, red cross
- cyberpunk hoverbike, scout unit
- nuclear-powered submarine, deep-sea exploration
- stealth bomber, night operation
- battlefield command and control center, high-tech
- anti-aircraft missile system, mobile defense

Refined Prompts: The synthetic image shown in Figure 22

- Armored Ground Vehicle, Modern Combat
- Artillery System, Classic Aesthetics
- Amphibious Assault Transport
- Drone Carrier, Tactical
- Field Support Unit, Healthcare
- Reconnaissance Craft, Urban Aerial
- Deep Sea Explorer, Nuclear Propulsion
- Stealth Surveillance Plane
- Command Center, High-Tech
- Missile Defense Network, Mobile

Fig. 20. **Manual Prompts** and Generated Synthetic Dataset

Fig. 21. **AI agent Initial Prompts** and Generated Synthetic Dataset

Fig. 22. **AI agent Refined Prompts** and Generated Synthetic Dataset

APPENDIX L
ARTIFACT APPENDIX

Artifact DOI 10.5281/zenodo.14248909.

A. Description and Requirements

1) How to Access: Our artifacts include the source code to run the training and evaluation process of EncoderLock.

2) Hardware Dependencies: A GPU is highly recommended for testing the source code to ensure faster training and evaluation.

3) Software Dependencies: The required software includes Python 3.9 and PyTorch 1.12.

4) Datasets: Most of the datasets can be downloaded via `torchvision`. We also provide links to download the real-world datasets (i.e., ImageNette and Military Vehicle) in the README file and the main manuscript.

B. Artifact Installation

The artifacts run in a Python 3.9 environment. We recommend installing the required packages in an Anaconda environment using the `environment.yml` file.

C. Experiment Workflow

The experiment consists of three parts:

- **Data and Model Preparation**: To run the experiment, we first need to prepare data from both the source domain and the target domain (synthetic dataset for the zero-shot EncoderLock). Additionally, we need a victim pre-trained encoder that performs well on the source domain but is vulnerable to malicious probing on the target domain.
- **EncoderLock Training**: In this stage, the user should fine-tune the encoder using the proposed method to obtain a modified encoder that resists malicious probing on the target domain.
- **EncoderLock Testing**: In this stage, the user needs to test the probing performance of the modified encoder on the target domain to demonstrate that the model is protected against probing.

D. Major Claims

We demonstrate the effectiveness of EncoderLock in protecting against malicious probing. The main results are presented in Table II and Table III. Specifically, you should observe that while the source domain experiences a slight degradation in testing accuracy, the target domain accuracy drops significantly after implementing EncoderLock.

E. Evaluations

1) Experiment Cost: Training EncoderLock usually takes from 0.1 to 6 GPU hours, varying depending on the GPU platform and the dataset.

2) How To: Please follow the steps in the `README.md` file included with the artifacts to run the evaluations.

3) Execution: We provide scripts to directly run all evaluations; you can find them under the `/tests/` directory.

4) Customization: Hyperparameters can be directly adjusted in the execution scripts.