Abstract

Self-supervised learning usually uses a large amount of unlabeled data to pre-train an encoder which can be used as a general-purpose feature extractor, such that downstream users only need to perform fine-tuning operations to enjoy the benefit of “large model”. Despite this promising prospect, the security of pre-trained encoder has not been thoroughly investigated yet, especially when the pre-trained encoder is publicly available for commercial use.

In this paper, we propose AdvEncoder, the first framework for generating downstream-agnostic universal adversarial examples based on the pre-trained encoder. AdvEncoder aims to construct a universal adversarial perturbation or patch for a set of natural images that can fool all the downstream tasks inheriting the victim pre-trained encoder. Unlike traditional adversarial example works, the pre-trained encoder only outputs feature vectors rather than classification labels. Therefore, we first exploit the high frequency component information of the image to guide the generation of adversarial examples. Then we design a generative attack framework to construct adversarial perturbations/patches by learning the distribution of the attack surrogate dataset to improve their attack success rates and transferability. Our results show that an attacker can successfully attack downstream tasks without knowing either the pre-training dataset or the downstream dataset. We also tailor four defenses for pre-trained encoders, the results of which further prove the attack ability of AdvEncoder. Our codes are available at: https://github.com/CGCL-codes/AdvEncoder.

1. Introduction

Self-supervised learning [8, 10] (SSL) is an emerging machine learning paradigm that seeks to overcome the restrictions of labeled data. It usually uses a large volume of unlabeled data to pre-train a general-purpose encoder, which can be used as a feature extractor for various downstream tasks like image classification, image retrieval, object detection, etc. As a result, any resource-constrained user can enjoy the advantages of “large model” without performing the expensive training from scratch, where only light-weight fine-tuning operations are needed at its request. Driven by this promising prospect, pre-training encoders become popular in industry and many service providers...
publicly release their pre-trained encoders (e.g., SimCLR by Google [8, 9], MoCo by Meta [12, 26]) or deploy them as a commercial service (e.g., OpenAI [52], Clarifai [13]).

Meanwhile, it is well known that deep neural networks (DNNs) are vulnerable to various adversarial attacks [23, 44, 61, 65], which will make pre-trained encoder fragile as well. However, the security of pre-trained encoder has received much less consideration in the literature. Although some recent works studied security threats on pre-trained encoders including backdoor attack [31, 32], poisoning attack [40], and privacy risks [15, 41], none of them paid attention to adversarial examples, another kind of prevalent and destructive attack on DNNs. Constructing adversarial examples against pre-trained encoders is quite different from its traditional attack route due to the fact that the attacker has no knowledge of the downstream tasks. In other words, the attacker needs to attack a DNN without knowing its task type, the pre-training dataset, and the downstream dataset, even when the whole model will get fine-tuned. To the best of our knowledge, how to realize adversarial example attack in the practical scenario of pre-training still remains challenging and unresolved.

In this work, we take a big step towards bridging the gap between adversarial examples and pre-trained encoders. We consider both adversarial perturbation [5, 23, 37, 45] and patch [3, 30, 39, 61]. The former one has a high imperceptibility, while the latter one is visible but confined to a small area of the image and more readily applicable in the physical world. Furthermore, without the knowledge of downstream data, we aim to realize universal adversarial attacks [25, 44, 63] where one adversarial perturbation or patch applies to a set of natural images and can cause model misclassification.

Specifically, we propose AdvEncoder, a novel attack framework for generating downstream-agnostic universal adversarial examples. The most challenging job lies in addressing the limitations and lacking supervised signals and the information about the downstream tasks. Inspired by the fact that deep neural networks are biased towards texture features of images [34, 60], the change of texture information, i.e., the high frequency components (HFC) of the image, is very likely to cause the model decision change. We first exploit a high frequency component filter to get the HFC of benign and adversarial samples, and pull away their Euclidean distance as much as possible to influence the model’s decision. We then design a generative attack framework to construct adversarial perturbations or patches with high attack success rates and transferability by learning the distribution of the data, with a fixed random noise as input. Our main contributions are summarized as follows:

- We propose AdvEncoder, the first attack framework to construct downstream-agnostic universal adversarial examples in self-supervised learning. We reveal that the pre-trained encoder incurs severe security risks for the downstream tasks.

- We design a frequency-based generative network to generate universal adversarial examples by directly alerting the texture features of the image itself. It is a flexible framework that can generate both adversarial perturbations and patches.

- Our extensive experiments on fourteen self-supervised training methods and four image datasets show that our AdvEncoder achieves high attack success rates and transferability against different downstream tasks.

- We tailor four popular defenses to mitigate AdvEncoder. The results further prove the attack ability of AdvEncoder and highlight the needs of new defense mechanism to defend pre-trained encoders.

2. Background and Related Work

2.1. Self-supervised Learning

Self-supervised learning seeks to utilize the oversight signals within the unlabeled data itself to pre-train encoders that can convert complex inputs into generic representations. The pre-trained encoder that learned generally valuable domain knowledge can be used as a universal feature extractor to transfer knowledge to solve different specific downstream tasks. In this paper, we concentrate on image encoders.

Based on [22, 58], self-supervised learning schemes can be divided into the following categories: (1) contrastive learning methods (e.g., MoCo [10, 12], SimCLR [8]) train representations such that dissimilar negative pairs are widely apart and comparable positive pairs are shown to be near to one another. (2) negative-free methods (e.g., BYOL [24], Sim-Siam [11], and ReSSL [64]) achieve better representation without the use of negative samples by maintaining the consistency between positive samples and ignoring negative ones. (3) clustering-based methods (e.g., SwAV [6], DeepCluster v2 [6], and DINO [7]) group similar samples into the same class using conventional clustering methods. (4) redundancy reduction-based methods (e.g., Barlow Twins [62], W-MSE [21], VICReg [2], and VIBCReg [38]) enhance the connection in the same dimension of the representation while attempting to decoupling in distinct dimensions. Concurrently, the use of nearest-neighbor retrieval has been investigated in NNCLR [19]. These approaches start from different motivations, design different loss functions, and use different network structures and tricks, which also make them have different defense abilities against adversarial attacks.
2.2. Attacks on Pre-trained Encoders

Recently, a growing number of works began to investigate the privacy and security issues of the pre-trained encoders in self-supervised learning. Some efforts investigated privacy risks against pre-trained encoders, such as membership inference attacks [27, 41], model extraction [20, 42, 56]. At the same time, backdoor attacks and poisoning attacks, the common security threats that usually occur in the training phase, have been shown to be deleterious to pre-trained encoders [4, 32, 40, 55]. In contrast, adversarial examples, which appear during the testing phase and pose great threat against neural networks, have not been thoroughly investigated yet. A concurrent work, PAP [1], produced a pre-trained perturbation by lifting the feature activations of low-level layers, but the generated adversarial examples lack semantics and rely heavily on the pre-training dataset. On the contrary, our work aims to achieve effective attacks from the perspective of directly changing the intrinsic texture features of the samples under more demanding conditions that better reflect realistic scenarios.

2.3. Universal Adversarial Examples

It is well known that deep neural networks are vulnerable to adversarial examples, where an attacker can fool the model by adding minor noise to the image, usually in the form of perturbation [5, 23, 28, 29] and patch [30, 39, 61]. Universal adversarial attack [44] was proposed to fool the target model by imposing a single adversarial noise vector on all the images. Existing works can be divided into optimization-based universal adversarial attacks [44, 46, 47] and generative universal adversarial attacks [25, 48, 49]. Compared with optimization-based solution, generative universal adversarial attacks can generate more generalized and natural-looking adversarial examples by learning the distribution of samples. However, existing generative universal adversarial attacks in supervised learning can only fool a single model and require the label information of the model output. Since pre-trained encoders can only output the feature vector corresponding to the image, exiting attacks cannot be directly applied to the pre-trained encoders, let alone having no knowledge about the downstream tasks. Some works also proposed different defenses against adversarial examples, such as data preprocessing, adversarial training [43, 59], pruning [66], and fine-tuning [53]. These methods can defend against adversarial examples at different phases.

3. Methodology

3.1. Threat Model

Following existing studies on attacking pre-trained encoders [32, 40, 55], we assume the attacker has access to the pre-trained encoders (e.g., through purchasing or directly downloading from publicly available websites), but has no knowledge of the pre-training datasets and the downstream tasks. The goal of the attacker is to conduct non-targeted adversarial attacks to disable the downstream tasks or damage their accuracy. Specifically, the attacker uses the pre-trained encoder to design a downstream agnostic universal adversarial perturbation or patch that applies to various kinds of the input images from different datasets. Then the adversarial example can mislead all the downstream classifiers that inherit the victim pre-trained encoder. We also assume that the downstream task undertaken (called user hereinafter) is able to fine-tune the linear layer or the pre-trained encoder for their cause and the model provider can adopt common defenses like adversarial training to purify the encoder.

3.2. Problem Definition

Given an input \( x \in D_p \) to a pre-trained encoder \( g_\theta(\cdot) \) that returns a feature vector \( v \in V \), a downstream classifier \( f_\theta(\cdot) \) gives predictions based on the similarity of the feature vectors, where \( \theta \) and \( \theta' \) denote the parameters of the pre-trained encoder and the downstream classifier, \( x \) indicates any image in the dataset, \( D_p \) and \( V \) refer to the pre-training dataset and feature space, respectively. The attacker uses an attacker’s surrogate dataset \( D_a \), which is irrelevant to the pre-training dataset and feature space, respectively. The attacker uses an attacker’s surrogate dataset \( D_a \), and the downstream dataset \( D_d \), to generate a universal adversarial noise against the pre-trained encoder. Additionally, the universal adversarial noise \( \delta \) should be sufficiently small, and modeled through an upper-bound \( \epsilon \) on the \( l_p \)-norm. This problem can be formu-
lated as:
\[
g_\theta(x + \delta) \neq g_\theta(x), \quad \text{s.t.} \|\delta\|_p \leq \epsilon \quad (1)
\]

The attacker’s goal is to implement a universal non-target attack to fool the downstream classifier \(f_{\theta'}\). When a universal adversarial noise \(\delta\) is attached to the downstream dataset sample \(x \in D_d\), it leads to misclassification. Therefore, the attacker’s goal can be formalized as:
\[
f_{\theta'}(g_\theta(x + \delta)) \neq f_{\theta'}(g_\theta(x)), \quad \text{s.t.} \|\delta\|_p \leq \epsilon \quad (2)
\]

### 3.3. Intuition Behind AdvEncoder

The pre-trained encoder outputs similar feature vectors for similar images, which are close together in the feature space and far away from the images of other categories. The downstream tasks will output decisions based on these feature vectors, thus the attacker needs to push the adversarial example away from its initial position as much as possible in the feature space. In order to realize downstream-agnostic adversarial attack, there are two challenges ahead.

**Challenge I: Lack of supervised signals in pre-trained encoder.** When the attacker feeds the image to the pre-trained encoder, it only obtains the corresponding feature vector instead of the label. It is infeasible to effectively attack the pre-trained encoder with the traditional approaches of adversarial examples in supervised learning. An intuitive idea is to add a large budget perturbation to the sample to make the pre-trained encoder misclassify it. However, as seen from Fig. 2(a), a large budget perturbation will not necessarily achieve the above goal, but may simply be an internal movement within the same class, rather than in a direction away from that class. Recent works [17, 43, 60] have revealed that surface-statistical content with high frequency property is essential for DNNs and adversarial perturbations also have this property. Therefore, we propose using a universal adversarial noise to change the high frequency component of the image, i.e., the texture information, to influence the output of the pre-trained encoder. It plays the role of label guidance in the supervised learning, and it is easier to push the target samples out of the original decision boundaries from the perspective of directly altering the semantics of the image itself.

**Challenge II: Lack of information about the downstream tasks.** In the pre-trained encoder to downstream task paradigm, where fine-tuning affects original feature boundaries of the model, the above approach that simply fools the pre-trained encoder can barely influence downstream task decisions. As seen in Fig. 2(b), the adversarial examples that have left the original class are again correctly classified by the downstream model after the change of decision boundaries caused by fine-tuning. We thus hope to make the adversarial examples far enough away from the original class by a universal adversarial noise under a small perturbation bound, as depicted in Fig. 2(c). Consequently, the downstream classifier will be misled based on the apparent similarity of the feature vectors. Given the remarkable capability of generative networks at generating features with fixed patterns, we further design a generative attack framework to improve the generalization of universal adversarial noise. As shown in Fig. 2(d), all target samples will be clustered together in the feature space and get away from all the normal samples, making it difficult for downstream tasks to correctly classify target samples.

### 3.4. Frequency-based Generative Attack Framework

In this section, we present AdvEncoder, a novel generative attack against pre-trained encoder in self-supervised learning. The pipeline of AdvEncoder is depicted in Fig. 3. It consists of an adversarial generator \(G\), a high frequency filter \(H\), and a victim encoder \(E\). Specifically, we design a frequency-based generative attack framework to generate a universal adversarial noise. By feeding a fixed noise \(z\) into the adversarial generator, we obtain a universal adversarial noise and paste it onto target image of the attacker’s surrogate dataset \(D_a\) to get an adversarial example \(x^{adv}\).
The objective function of the adversarial generator $G$ is:

$$L_G = \alpha L_{adv} + \beta L_{hfc} + \lambda L_q$$ (3)

where $L_{adv}$ is the adversarial loss function, $L_{hfc}$ is the high frequency component loss function, $L_q$ is the quality loss function, $\alpha$, $\beta$, $\lambda$ are pre-defined hyper-parameters.

$L_{adv}$ enhances the attack strength of universal adversarial noise by maximizing the feature vector distance between the normal and adversarial samples of the encoder output. We adopt InfoNCE [51] loss to measure the similarity between the output feature vectors of the pre-trained encoder $g(\cdot)$. Specifically, we treat the benign sample $x_i \in D_a$ and the adversarial sample $x_i^{adv}$ as negative pairs, pulling away their feature distance. Thus $L_{adv}$ is expressed as:

$$L_{adv} = \log \left[ \frac{\exp \left( S \left( g(x_i^{adv}), g(x_i) / \tau \right) \right)}{\sum_{j=0}^{K} \exp \left( S \left( g(x_i^{adv}), g(x_j) / \tau \right) \right)} \right]$$ (4)

where $S(\cdot)$ denotes the cosine similarity measure function, $j$ is not equal to $i$, and $\tau$ indicates a temperature parameter.

Due to the lack of the guidance of label information, pushing away the locations of output embeddings in the feature space by adding noises alone requires large perturbation budget. $L_{hfc}$ changes the original semantic features of the image by modifying the high frequency components to further separate the location of the target sample.

We can obtain the HFC of an image through the high frequency component filter $H$. The high frequency component loss $L_{hfc}$ can be formalized as:

$$L_{hfc} = -\| H(x^{adv}) - H(x) \|_2$$ (5)

To achieve better stealthiness, we use $L_q$ to control the magnitude of the adversarial noises output by the generator and crop $\delta$ after each optimisation to ensure it meets the constraints $\epsilon$. Formally, we have:

$$L_q = \| x^{adv} - x \|_2$$ (6)

Without changing the framework of AdvEncoder, we can convert a universal adversarial noise into two common forms of attacks, universal adversarial perturbation (AdvEncoder-Perturbation, abbreviated as Adv-PER) and universal adversarial patch (AdvEncoder-Patch, abbreviated as Adv-PAT).

**Adv-PER.** The attacker directly adds the universal adversarial perturbation generated by the generator to the image, which has better stealthiness. The perturbation-based adversarial example can be represented as:

$$x^{adv} = x + G(z)$$ (7)

**Adv-PAT.** The attacker can apply the adversarial patch to the image with a randomly chosen hidden location to obtain the adversarial example. It is easier to be realized in the physical world. The patch-based adversarial example can be represented as:

$$x^{adv} = x \odot (1 - m) + G(z) \odot m$$ (8)

where $\odot$ denotes the element-wise product, $m$ is a binary matrix that contains the position information of the universal adversarial patch.

4. **Experiments**

4.1. **Experimental Setting**

**Datasets and Models.** We use the publicly available pre-trained encoders from solo-learn [16], an established SSL library, as victim encoders. All the encoders are pretrained on ImageNet [54] or CIFAR10 [36] with ResNet18 backbone. For a comprehensive study, we select fourteen SSL methods (Barlow Twins [62], BYOL [24], DeepCluster v2 [6], DINO [7], MoCo v2+ [10], MoCo v3 [12], NNCLR [19], ReSSL [64], SimCLR [8], SupCon [35], SwAV [6], VlbCReg [38], VICReg [2], W-MSE [21]). We make no strong assumptions about the attacker’s knowledge, so we set the attacker’s surrogate dataset to be CIFAR-10 as the default setting. For different downstream tasks, we use the following four image datasets: STL10 [14], GT-SRB [57], CIFAR10, and ImageNet.

**Evaluation Metrics.** We use Attack Success Rate (ASR) to evaluate the attack performance of our AdvEncoder, which refers to the success rate of generated adversarial examples in deceiving the classifier. A higher value indicates stronger attack capability.

4.2. **Attack Performance**

**Implementation Details.** To demonstrate the effectiveness of AdvEncoder when the downstream task is unknowable, two types of downstream tasks, *image classification* and *image retrieval*, are chosen for testing. Following [30, 44, 48], we set $\epsilon$ (i.e., the perturbation budget of Adv-PER) to...
Table 1: The ASR (%) of Adv-PER under different settings. $S_1 - S_4$ denote the settings where the downstream datasets are CIFAR10, STL10, GTSRB, ImageNet, respectively, and all the attacker’s surrogate dataset is CIFAR10. $S_5 - S_8$ use ImageNet as the attacker’s surrogate dataset. In all datasets, the settings with the same as $S_1 - S_4$ are abbreviated as Barlow and DeepC2, respectively.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Setting</th>
<th>Barlow</th>
<th>BYOL</th>
<th>DeepC2</th>
<th>DINO</th>
<th>MoCoV2+</th>
<th>MoCoV3</th>
<th>NNCLR</th>
<th>ReSSL</th>
<th>SimCLR</th>
<th>SupCon</th>
<th>SwAV</th>
<th>VbReg</th>
<th>VReg</th>
<th>W-MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>$S_1$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_2$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_3$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_4$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td>ImageNet</td>
<td>$S_5$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_6$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_7$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_8$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
</tbody>
</table>

Table 2: The ASR (%) of Adv-PAT under different settings. $S_1 - S_8$ represent the same settings as mentioned in Tab. 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Setting</th>
<th>Barlow</th>
<th>BYOL</th>
<th>DeepC2</th>
<th>DINO</th>
<th>MoCoV2+</th>
<th>MoCoV3</th>
<th>NNCLR</th>
<th>ReSSL</th>
<th>SimCLR</th>
<th>SupCon</th>
<th>SwAV</th>
<th>VbReg</th>
<th>VReg</th>
<th>W-MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>$S_1$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_2$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_3$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_4$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td>ImageNet</td>
<td>$S_5$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_6$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_7$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
<tr>
<td></td>
<td>$S_8$</td>
<td>90.02</td>
<td>88.37</td>
<td>89.37</td>
<td>84.92</td>
<td>84.79</td>
<td>88.48</td>
<td>87.85</td>
<td>57.07</td>
<td>92.37</td>
<td>92.25</td>
<td>37.91</td>
<td>54.25</td>
<td>52.25</td>
<td>68.38</td>
</tr>
</tbody>
</table>

Table 3: Top-10 retrieval attack results. “per-mAP” represents the retrieval accuracy of the Adv-PER samples corresponding to clean samples, while “pat-mAP” denotes the accuracy of Adv-PAT samples.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Setting</th>
<th>Barlow</th>
<th>BYOL</th>
<th>DeepC2</th>
<th>DINO</th>
<th>MoCoV2+</th>
<th>MoCoV3</th>
<th>NNCLR</th>
<th>ReSSL</th>
<th>SimCLR</th>
<th>SupCon</th>
<th>SwAV</th>
<th>VbReg</th>
<th>VReg</th>
<th>W-MSE</th>
</tr>
</thead>
</table>

SSL pre-trained encoders. We evaluate AdvEncoder on each victim pre-trained encoder over four downstream tasks using two attacker’s surrogate datasets, respectively. As for the retrieval task, we attack six types of SSL encoders trained on CIFAR10 corresponding to the retrieval tasks of GTSRB and STL10. We use mAP (mean average precision) [67] to measure the retrieval accuracy, and adapt per-mAP and pat-mAP to measure the retrieval accuracy for adversarial examples. Adversarial examples generated by AdvEncoder are shown in Fig. 4.

Analysis. Our experimental results on classification tasks reveal the severe vulnerability of downstream tasks based on pre-trained encoders. Firstly, from Tab. 1 and Tab. 2, we can see that among the 224 attack settings, both Adv-PER and Adv-PAT perform well on all downstream tasks. In particular, Adv-PAT has a consistently high attack performance under different settings, with an average ASR of over 90%. Secondly, the attacker’s surrogate dataset has an impact on

10/255 and the patch size (i.e., noise percentage of each sample) of Adv-PAT to 0.03. We choose the bottom right corner of the image, which is not easily visible, to apply the patch. We set the hyper-parameters $\alpha = 1$, $\beta = 5$, $\lambda = 1$ and the training epoch to 20 with batch size of 256. The generator network is trained by Adam optimizer with the initial learning rate 0.0002.

For the classification task, we attack fourteen types of
Figure 5: The ablation study results. (a) - (d) examine the effect of the number of the surrogate data and different modules. (e) - (f) explore the effect of different attack strengths.

Figure 6: The results (%) of transferability study.

4.4. Transferability Study

In this section, we choose Adv-PER as a representative to analyze the transferability of our scheme from two perspectives, namely, crossing pre-training datasets and SSL methods. To this end, we conduct experiments using CIFAR10 as the attacker’s surrogate dataset and GTSRB as the downstream dataset. In Fig. 6(a) - (b), CIFAR10-ImageNet represents we use CIFAR10 and ImageNet to train two encoders based on which adversarial examples and downstream tasks are made, respectively. ImageNet-ImageNet has the same definition. Each column represents different downstream models attacked with the same adversarial examples. The results indicate that the Adv-PER method can effectively transfer attacks across different pre-training datasets and SSL methods, even without any prior knowledge of the pre-training and downstream datasets.

Table 4: The ASR (%) of comparison study

<table>
<thead>
<tr>
<th>Method</th>
<th>Barlow</th>
<th>BYOL</th>
<th>DINo</th>
<th>MoCo v2+</th>
<th>NNCLR</th>
<th>SimCLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAP [44]</td>
<td>48.95</td>
<td>45.97</td>
<td>43.01</td>
<td>42.24</td>
<td>46.41</td>
<td>48.41</td>
</tr>
<tr>
<td>UPGD [18]</td>
<td>18.31</td>
<td>23.89</td>
<td>18.59</td>
<td>18.31</td>
<td>17.57</td>
<td>20.83</td>
</tr>
<tr>
<td>FFF [47]</td>
<td>47.71</td>
<td>50.26</td>
<td>45.53</td>
<td>50.33</td>
<td>46.64</td>
<td>52.33</td>
</tr>
<tr>
<td>SSP [59]</td>
<td>50.40</td>
<td>46.30</td>
<td>47.12</td>
<td>50.87</td>
<td>49.51</td>
<td>48.06</td>
</tr>
<tr>
<td>PAP-base [1]</td>
<td>37.89</td>
<td>27.29</td>
<td>25.52</td>
<td>30.47</td>
<td>24.65</td>
<td>46.89</td>
</tr>
<tr>
<td>PAP-fuse [1]</td>
<td>46.61</td>
<td>45.47</td>
<td>30.26</td>
<td>45.14</td>
<td>40.05</td>
<td>46.30</td>
</tr>
<tr>
<td>PAP-ugs [1]</td>
<td>38.20</td>
<td>36.89</td>
<td>30.55</td>
<td>37.71</td>
<td>42.99</td>
<td>53.56</td>
</tr>
<tr>
<td>Adv-PER</td>
<td>55.49</td>
<td>58.67</td>
<td>61.67</td>
<td>53.22</td>
<td>53.00</td>
<td>55.87</td>
</tr>
<tr>
<td>UAP [3]</td>
<td>49.20</td>
<td>49.60</td>
<td>46.12</td>
<td>47.04</td>
<td>49.74</td>
<td>50.35</td>
</tr>
<tr>
<td>Adv-PAT</td>
<td>82.35</td>
<td>88.60</td>
<td>89.07</td>
<td>90.70</td>
<td>88.86</td>
<td>89.42</td>
</tr>
</tbody>
</table>

4.5. Comparison Study

Implementation Details. In this section, we compare Adv-Encoder with state-of-the-art adversarial attacks. PAP [1] is the most similar work with ours as it produces pre-trained perturbations from the perspective of model feature activation values. In contrast, we address the attack inheritance of adversarial samples by directly changing the important texture features of the samples themselves. To further demonstrate our superiority, for perturbation we compare with classic optimized-based UAP.
In this section, we tailor four defensive measures trying to mitigate AdvEncoder from the perspective of the user and the model provider, respectively. For users using pre-trained encoders, we adopt pre-processing the input data, fine-tuning the entire model using a small amount of data, and pruning the parameters to defend against adversarial attacks. As for the model providers, the defender can perform adversarial training on pre-trained encoders. In the following experiments, we use the default settings from Sec. 4.3.

5.1. Corruption

We defend against adversarial examples by corrupting the images through adding different degree of Gaussian noise to the samples. As illustrated in Fig. 7(a), the accuracy of the model decreases significantly as the degree of Gaussian noise added increases. In particular, Adv-PER only experiences a slight decrease in accuracy when the degree of Gaussian noise is increased to 0.03, while Adv-PAT is almost unaffected. These findings indicate that Adv-Encoder can effectively resist the corruption-based pre-processing defense.

5.2. Fine-tuning & Pruning

Fine-tuning [53] and pruning [66] are two commonly used methods for downstream models to inherit pre-trained encoders, providing better adaptability to downstream tasks. We first fully fine-tune pre-trained encoders based on MoCo v2+, using ten classes in CIFAR10, STL10, GTSRB, and ImageNet, respectively. The results in Fig. 7(b) - (c) show that AdvEncoder still has excellent attack performance even after the encoder is fully fine-tuned. Furthermore, we choose pruning rate in [0.1, 0.7], the results in Fig. 7(d) show that AdvEncoder is able to resist the defenses based on model parameter pruning.

5.3. Adversarial Training

Adversarial training improves the robustness of the pre-trained encoder and poses a greater challenge to the adversarial examples. Following [33], we use the ImageNet dataset for adversarial training of the pre-trained encoder and choose CIFAR10 as downstream dataset. As demonstrated in Fig. 7(e) - (f), we explore the degree of resistance to adversarial training for AdvEncoder of different attack strengths. Adversarial training slightly affects Adv-PER, but our attack can succeed after improving the attack strength. Adv-PAT has not been affected at all.

6. Conclusion

In this paper, we propose the first generative attack to construct downstream-agnostic adversarial examples in self-supervised learning. It is a flexible framework that can generate both universal adversarial perturbations and patches. We verify the excellent attack performance of AdvEncoder on four downstream tasks corresponding to fourteen publicly available SSL encoders over two pre-training datasets. We tailor four popular defenses to mitigate Adv-Encoder. The results further prove the attack ability of Adv-Encoder and highlight the needs of new defense mechanism to defend pre-trained encoders.
Acknowledgments. Shengshan’s work is supported in part by the National Natural Science Foundation of China (Grant No.U20A20177) and Hubei Province Key R&D Technology Special Innovation Project under Grant No.2021BA032. Qian’s work is supported in part by the National Natural Science Foundation of China under Grants U20B2049 and U21B2018. Shengshan Hu is the corresponding author.

References


