Pre-trained Adversarial Perturbations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Self-supervised pre-training has drawn increasing attention in recent years due to its superior performance on numerous downstream tasks after fine-tuning. However, it is well-known that deep learning models lack the robustness to adversarial examples, which can also invoke security issues to pre-trained models, despite being less explored. In this paper, we delve into the robustness of pre-trained models by introducing Pre-trained Adversarial Perturbations (PAPs), which are universal perturbations crafted for the pre-trained models to maintain the effectiveness when attacking fine-tuned ones without any knowledge of the downstream tasks. To this end, we propose a Low-Level Layer Lifting Attack (L4A) method to generate effective PAPs by lifting the neuron activations of low-level layers of the pre-trained models. Equipped with an enhanced noise augmentation strategy, L4A is effective at generating more transferable PAPs against the fine-tuned models. Extensive experiments on typical pre-trained vision models and ten downstream tasks demonstrate that our method improves the attack success rate by a large margin compared to the state-of-the-art methods.

1 Introduction

Large-scale pre-trained models [39, 14] have recently achieved unprecedented success in a variety of fields, e.g., natural language processing [6, 26, 1], computer vision [2, 16, 17], etc. A large amount of work proposes sophisticated self-supervised learning algorithms, enabling the pre-trained models to extract useful knowledge from large-scale unlabeled datasets. The pre-trained models consequently facilitate downstream tasks through transfer learning or fine-tuning [36, 48, 13]. Nowadays, more practitioners without sufficient computational resources or training data tend to fine-tune the publicly available pre-trained models on their own datasets. Therefore, it has become an emerging trend to adopt the paradigm of pre-training to fine-tuning rather than training from scratch [14].

Despite the excellent performance of deep learning models, they are incredibly vulnerable to adversarial examples [42, 12], which are generated by adding small, human-imperceptible perturbations to natural examples, but can make the target model output erroneous predictions. Adversarial examples also exhibit an intriguing property called *transferability* [42, 25, 31], which means that the adversarial perturbations generated for one model or a set of images can remain adversarial for others. For example, a universal adversarial perturbation (UAP) [31] can be generated for the entire distribution of data samples, demonstrating an excellent cross-data transferability. Other works [25, 8, 45, 9, 33] have revealed that adversarial examples have high cross-model and cross-domain transferability, making *black-box attacks* practical without any knowledge of the target model or even the training data. However, much less effort has been devoted to exploring the adversarial robustness of pre-trained models. As these models have been broadly studied and deployed in various real-world applications, it is of significant importance to identify their weaknesses and evaluate their robustness, especially concerning the pre-training to the fine-tuning procedure.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

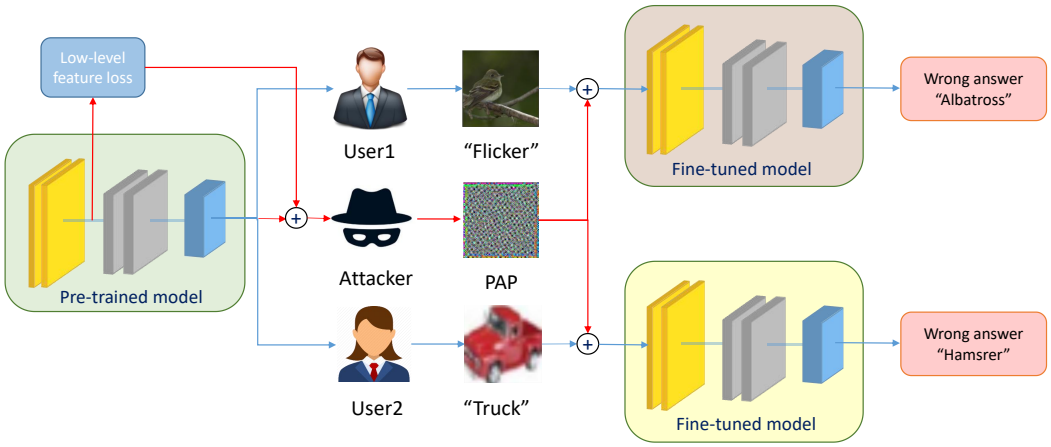


Figure 1: A demonstration of Pre-trained Adversarial Perturbations.

In this paper, we introduce **Pre-trained Adversarial Perturbations (PAPs)**, a new kind of universal adversarial perturbations designed for pre-trained models. Specifically, a PAP is generated for a pre-trained model with the purpose of effectively fooling any downstream model obtained by fine-tuning the pre-trained one, as shown in Fig. 1. It works under a quasi-black-box setting where the downstream task, dataset, and fine-tuned model parameters are not available. This attack setting is more suitable for the pre-training to fine-tuning procedure, since many pre-trained models are publicly available and the adversary may generate PAPs before the pre-trained model has been fine-tuned. Although there are many methods [8, 45] proposed for improving the transferability, they do not consider the specific characteristics of the pre-training to fine-tuning procedure, limiting their *cross-finetuning transferability* in our setting.

To generate more effective PAPs, we propose a **Low-Level Layer Lifting Attack (L4A)** method, which aims to lift the feature activations of low-level layers. Motivated by the finding that the lower the level of the model's layer is, the less its parameters change during fine-tuning, we generate PAPs to damage the low-level feature representations of pre-trained models, making them better reserved after fine-tuning. To further alleviate the overfitting of PAPs to the source domain data, we improve L4A with a noise augmentation technique. We conduct extensive experiments on typical pre-trained vision models [2, 17] and ten downstream tasks. The evaluation results demonstrate that our method achieves a higher attack success rate on average compared to the alternative baselines.

2 Related work

Self-supervised learning. Self-supervised learning methods enable learning from unlabeled data. To achieve this, early approaches utilize handcrafted pretext tasks, like colorization [52], rotation prediction [11], position prediction [35], and Selfie [43]. Another approach for SSL is contrastive learning [24, 38, 2, 21], which aims to map the input image to the feature space and minimize the distance between similar ones while keeping dissimilar ones far away from each other. In particular, a similar sample is retrieved by applying appropriate data augmentation techniques to the original one, and the versions of different samples are viewed as dissimilar pairs.

Adversarial examples. With the knowledge of the structure and parameters of a given model, many algorithms [23, 30, 29, 37] successfully fool the target model in a white-box manner. An intriguing property of adversarial examples is their good transferability [25, 31]. The universal adversarial perturbations [31] demonstrate good cross-data transferability by optimizing under a distribution of data samples. The cross-model transferability has also been extensively studied [8, 45, 9], enabling the attack on black-box models without any knowledge of their internal working mechanisms.

Robustness of the pre-training to fine-tuned procedure. Due to the popularity of pre-trained models, a lot of works [41, 47, 4] study the robustness of this setting. Among them, Dong et al. [7] propose a novel adversarial fine-tuning method in an information-theoretical way to retain robust features learned from the pre-trained model. Jiang et al. [20] integrate adversarial samples into the

pre-training procedure to defend against attacks. Fan et at. [10] adopt Clusterfit [46] to generate pseudo-label data and later use them for training the model in a supervised way, which improves the robustness of the pre-trained model. The main difference between our work and theirs is that we consider the problem from an attacker's perspective and craft adversarial samples in a cross-finetuning way.

3 Methodology

In this section, we first introduce the notations and the problem formulation of the Pre-trained Adversarial Perturbations (PAPs). Then, we detail the Low-Level Layer Lifting Attack (L4A) method.

3.1 Notations and problem formulation

Let $f_{\boldsymbol{\theta}}$ denote a pre-trained model for feature extraction with parameters $\boldsymbol{\theta}$. It takes an image $\mathbf{x} \in \mathcal{D}_p$ as input and outputs a feature vector $\mathbf{v} \in \mathcal{X}$, where $\mathcal{D}_p$ and $\mathcal{X}$ refer to the pre-training dataset and feature space, respectively. We denote $f^k_{\boldsymbol{\theta}}(\mathbf{x})$ as the k-th layer's feature map of $f_{\boldsymbol{\theta}}$ for an input image $\mathbf{x}$. In the pre-training to fine-tuning paradigm, a user fine-tunes the pre-trained model $f_{\boldsymbol{\theta}}$ using a new dataset $\mathcal{D}_t$ of the downstream task and finally get a fine-tuned model $f_{\boldsymbol{\theta'}}$ with updated parameters $\boldsymbol{\theta'}$. Then, let $f_{\boldsymbol{\theta'}}(\mathbf{x})$ be the predicted probability distribution of an image $\mathbf{x}$ over the classes of $\mathcal{D}_t$, and $F_{\boldsymbol{\theta'}}(\mathbf{x}) = \arg\max f_{\boldsymbol{\theta'}}(\mathbf{x})$ be the final classification prediction.

In this paper, we introduce **Pre-trained Adversarial Perturbations (PAPs)** which are generated for the pre-trained model $f_{\boldsymbol{\theta}}$, but can effectively fool fine-tuned models $f_{\boldsymbol{\theta'}}$ on downstream tasks. Formally, a PAP is a universal perturbation $\boldsymbol{\delta}$ within a small budget ϵ crafted by $f_{\boldsymbol{\theta}}$ and $\mathcal{D}_p$, such that $F_{\boldsymbol{\theta'}}(\mathbf{x}+\boldsymbol{\delta}) \neq F_{\boldsymbol{\theta'}}(\mathbf{x})$ for most of the instances belonging to the fine-tuning dataset $\mathcal{D}_t$. This can be formulated as the following optimization problem:

$$\max_{\boldsymbol{\delta}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_t}[F_{\boldsymbol{\theta'}}(\mathbf{x}) \neq F_{\boldsymbol{\theta'}}(\mathbf{x}+\boldsymbol{\delta})], \text{ with } ||\boldsymbol{\delta}||_p \leq \epsilon \text{ and } \mathbf{x}+\boldsymbol{\delta} \in [0,1] \tag{1}$$

where $||\cdot||_p$ denotes the l_p norm and we take the l_∞ norm in this work. There exist some works related to the universal perturbations, such as the universal adversarial perturbation (UAP) [31] and the fast feature fool (FFF) [32], as detailed below.

UAP: Given a classifier f and its dataset $\mathcal{D}$, the UAP method tries to find a perturbation $\boldsymbol{\delta}$ that can fool the model on most of the instances from $\mathcal{D}$, which is usually solved by an iterative method. Every time sampling an image $\mathbf{x}$ from the dataset $\mathcal{D}$, the attacker computes the minimal perturbation $\boldsymbol{\zeta}$ that sends $\mathbf{x}+\boldsymbol{\delta}$ to the decision boundary by Eq. (2) and then adds it into $\boldsymbol{\delta}$.

$$\boldsymbol{\zeta} \leftarrow \arg\min_{\mathbf{r}} ||\mathbf{r}||_2, \text{ with } F(\mathbf{x}+\boldsymbol{\delta}+\mathbf{r}) \neq F(\mathbf{x}). \tag{2}$$

FFF: Fast Feature Fool is an attacking algorithm aiming to produce maximal spurious activations at each layer. To achieve such a goal, one starts with a random $\boldsymbol{\delta}$ and solves the following equation:

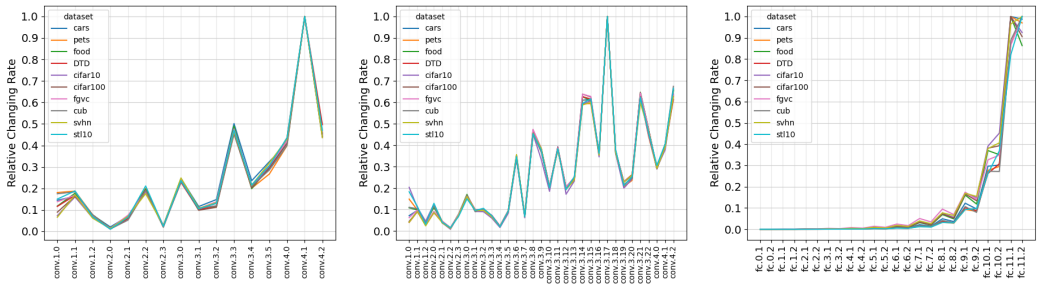
$$\min_{\boldsymbol{\delta}} -\log\left(\prod_{i=0}^{K} \bar{l}_i(\boldsymbol{\delta})\right), \text{ with } ||\boldsymbol{\delta}||_p \leq \epsilon. \tag{3}$$

where $\bar{l}_i(\boldsymbol{\delta})$ is the mean activation of the output tensor at layer i.

3.2 Our design

However, these attacks show limited cross-finetuning transferability in our problem setting due to ignorance of the fine-tuning procedure. Two challenges are degenerating the performance.

- **Fine-tuning Deviation.** The parameters of the model could change a lot during fine-tuning. As a result, the generated adversarial samples may perform well in the feature space of the pre-trained model but fail in the fine-tuned ones.
- **Datasets Deviation.** The statistics (i.e., mean and standard deviation) of different datasets can vary a lot. Only using the pre-training dataset with the fixed statistics to generate adversarial samples may suffer a performance drop.



(a) Resnet50 (b) Resnet101 (c) ViT16

Figure 2: The ordinate represents the Frobenius norm of the difference between the parameters of the finetuned model and its corresponding pretrained model, which is scaled into a range from 0 to 1 for easy comparison. The abscissa represents the level of the layer. Note that Resnet50 and Resnet101 [15] are pretrained in the method of SimCLRv2 [2], and ViT16 [44] is pretrained by MAE [17].

To alleviate the lousy effect of the above negative factors, we proposed **Low-Level Layer Lifting Attack (L4A)** method and Uniform Gaussian Sampling.

Low-Level Layer Lifting Attack. Low-Level Layer Lifting Attack is based on the findings in Fig. 2 that the higher the level of the layers are, the more their parameters change during fine-tuning. This is also consistent with the knowledge that the low-level convolutional layer acts as an edge detector that extracts low-level features like edges and textures and has little high-level semantic information [36, 48]. Since images from different datasets share the same low-level features, the parameters of these layers can be preserved. In contrast, the algorithms based on the high-level layers or the scores predicted by the model may not transfer well in such a cross-finetuning setting, as the feature spaces of high-level layers are easily distorted during fine-tuning. The basic method of L4A can be solved by the following equation:

$$\min_{\boldsymbol{\delta}} L_{base}(f_{\boldsymbol{\theta}}, \mathbf{x}, \boldsymbol{\delta}) = -\mathbb{E}_{\mathbf{x}\sim D_p}\left[||f^k_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta})||^2_F\right], \tag{4}$$

where $||\cdot||_F$ denotes the Frobenius norm of the input tensor. In our experiments, we find the lower the layer is, the better it performs, so we choose the first layer as default, such that $k = 1$. As the Eq. (4) is usually a sophisticated non-convex optimization problem, we solve it using stochastic gradient descent method.

We also find that fusing the adversarial loss of the consecutive low-level layers can boost the performance, which gives L4A_{fuse} method:

$$\min_{\boldsymbol{\delta}} L_{fuse}(f_{\boldsymbol{\theta}}, \mathbf{x}, \boldsymbol{\delta}) = -\mathbb{E}_{\mathbf{x}\sim D_p}\left[||f^{k_1}_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta})||^2_F + ||f^{k_2}_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta})||^2_F\right], \tag{5}$$

where $f^{k_1}_p(\mathbf{x}+\boldsymbol{\delta})$ and $f^{k_2}_p(\mathbf{x}+\boldsymbol{\delta})$ refers to the k_1-th and k_2-th layers' feature maps of f_p respectively. We set $k_1 = 1$ and $k_2 = 2$ as default.

Uniform Gaussian Sampling. Today, most state-of-the-art networks adopt batch normalization [19] for input images for better performance. Thus, the datasets' statistics become an essential factor for training. As can be seen in Fig. 3, the distribution of the downstream datasets can vary a lot compared to that of the pre-training dataset. However, traditional data augmentation techniques [49, 18] are limited to the pre-training domain and cannot alleviate the problem. Thus, we propose sampling Gaussian noises with various means and deviations to avoid overfitting. Combining the base loss using the pre-training dataset and the new loss using uniform Gaussian noises gives the L4A_{ugs} method in Eq. (6):

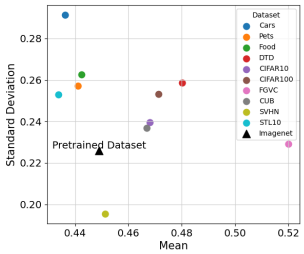


Figure 3: Datasets' statistics.

$$\min_{\boldsymbol{\delta}} L_{ugs}(f_{\boldsymbol{\theta}}, \mathbf{x}, \boldsymbol{\delta}) = -\mathbb{E}_{\boldsymbol{\mu},\boldsymbol{\sigma},\mathbf{n}_0\sim N(\boldsymbol{\mu},\boldsymbol{\sigma})}\left\{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_f}\left[||f_{\boldsymbol{\theta}}^{k_1}(\mathbf{x}+\boldsymbol{\delta})||_F{}^2 + ||f_{\boldsymbol{\theta}}^{k}(\mathbf{n}_0+\boldsymbol{\delta})||_F{}^2\right]\right\} \quad (6)$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are drawn from the uniform distribution $U(\boldsymbol{\mu}_l, \boldsymbol{\mu}_h)$ and $U(\boldsymbol{\sigma}_l, \boldsymbol{\sigma}_h)$, respectively and $\boldsymbol{\mu}_l$, $\boldsymbol{\mu}_h$, $\boldsymbol{\sigma}_l$, $\boldsymbol{\sigma}_r$ are four hyper-parameters.

4 Experiments

4.1 Settings

Pretraining methods. We follow SimCLR [2, 3] to obtain pre-trained models, which uses a backbone of Resnet [15] and pre-trains the model by contrastive learning. We download pre-trained parameters of Resnet50 and Resnet101 from here [1] to evaluate the generalization ability of our algorithm on different model architectures. Besides convolutional neural networks, transformers [44] attract much attention nowadays for their comparable performance. Based on transformers and the self-predicting pre-text task, MAE [17] becomes a well choice to provide other supporting results. Pre-trained parameters of ViT-base-patch16 are downloaded from here[2] for our research.

Datasets and Preprocessing. We adopt the ImageNet ILSVRC-2012 dataset [40] to generate PAPs, which is also used to pre-train the models. We mainly evaluate PAPs in image classification tasks, the same as the settings of SimCLRv2. Ten datasets, both fine-grained and coarse-grained, are adopted to test the cross-finetuning transferability of these adversarial samples. We load these datasets from torchvision(Details in Appendix A.1). Before feeding the images to the model, we first resize them into the size of 256*256 and then centercrop them into 224*224.

Competitors: We choose UAP [31] to test whether image-agnostic attacks also bear good cross-finetuning transferability. Since UAP needs final classification predictions of the inputs, we fit a linear head on the pre-trained feature extractor. Furthermore, by integrating the moment term into the iterative method, UAPEPGD [5] is believed to enhance cross-model transferability. Thus, we adopt UAPEPGD to study the connection between cross-model and cross-finetuning transferability. As our algorithm is based on the feature level, other feature attacks (FFF [32], ASV [22], DR [27], SSP [34]) are chosen for comparison.

Default settings and Metric. Unless otherwise specified, we choose a batch size of 16 and a step size of 0.0002. All the perturbations should be within the bound of 0.05 in the l_∞ norm. We evaluate the perturbations at the iterations of 1000, 5000, 30000 and 60000 and report the best performance. We show the results in the measure of attack success rates(ASK), which represent the classification error rate on the whole testing dataset after adding the perturbations to the legitimate images.

4.2 Main results

In this subsection, we craft pre-trained adversarial samples(PAPs) for ten downstream tasks with three different model architectures. The results are shown in Table 1, Table 2 and Table 3, respectively. Note that the first seven columns of validation datasets are fine-grained, while the next three are coarse-grained ones. We mark the best results for each dataset in **bold**, and the best baseline in **blue**. We highlight the results of L4A_{ugs} in red to emphasize that the Low-Lever Layer Lifting attack equipped with Uniform Gaussian Sampling shows great *cross-finetuning transferability* and performs best.

A quick glimpse shows that our proposed methods outperform all the baselines by a large margin. For example, as can be seen from Table 1, if the target model is Resnet101 pre-trained by SimCLRv2, the best competitor FFF_{mean} achieves an average attack success rate of 59.34%, while the villain L4A_{base} can lift it up to **66.89%** and the UGS technique further boosts the performance up to **72.20%**. Moreover, the STL10 domain is the hardest for PAPs to transfer among these tasks. However, L4A_{ugs} can significantly improve the cross-finetuning transferability, achieving an attack

[1] https://github.com/google-research/simclr
[2] https://github.com/facebookresearch/mae

Table 1: **Resnet101** pretrained by **SimCLRv2**

ASK	Cars	Pets	Food	DTD	FGVC	CUB	SVHN	CIFAR10	CIFAR100	STL10	AVG
FFF_{no}	43.81%	38.62%	49.95%	63.24%	85.57%	48.38%	12.55%	8.53%	77.74%	57.11%	48.55%
$FFFF_{mean}$	33.93%	31.37%	41.77%	52.66%	78.94%	45.00%	**14.85%**	14.42%	72.59%	56.66%	44.22%
FFF_{one}	31.87%	29.74%	39.25%	46.92%	74.17%	43.87%	9.24%	11.77%	65.61%	50.21%	40.26%
DR	36.28%	35.54%	47.43%	47.45%	75.00%	44.15%	12.05%	21.35%	65.39%	41.65%	42.63%
SSP	32.89%	30.50%	43.12%	45.85%	82.57%	45.55%	8.69%	11.66%	65.80%	40.91%	40.75%
ASV	60.75%	19.84%	36.33%	56.22%	84.16%	55.82%	7.11%	7.29%	58.10%	80.89%	46.64%
UAP	48.70%	36.55%	60.80%	63.40%	76.06%	52.64%	8.46%	8.53%	52.35%	31.15%	43.86%
UAPEPGD	94.12%	66.66%	61.30%	72.55%	70.34%	82.72%	13.88%	**61.65%**	20.04%	50.13%	59.34%
$L4A_{base}$	94.07%	61.57%	71.23%	69.20%	96.28%	81.07%	11.70%	12.68%	80.57%	90.49%	72.64%
$L4A_{fuse}$	90.98%	88.53%	**80.65%**	74.31%	93.79%	91.23%	11.40%	17.40%	80.98%	89.69%	67.10%
$L4A_{ugs}$	**94.24%**	**94.99%**	78.28%	**77.23%**	**92.92%**	**91.77%**	**11.40%**	14.60%	**76.50%**	**90.05%**	**72.20%**

Table 2: **Resnet50** pretrained by **SimCLRv2**

ASK	Cars	Pets	FOOD	DTD	CUB	SVHN	FGVC	CIFAR10	CIFAR100	STL10	AVG
FFF_{no}	26.91%	30.83%	43.28%	48.99%	79.00%	41.30%	38.23%	68.50%	44.67%	16.95%	43.86%
FFF_{mean}	36.75%	33.88%	45.26%	50.15%	52.02%	53.13%	77.22%	82.41%	68.10%	22.11%	52.10%
FFF_{one}	37.88%	35.30%	52.79%	59.52%	80.33%	59.62%	57.04%	75.40%	53.58%	18.31%	52.98%
DR	38.64%	34.42%	50.04%	45.53%	47.88%	39.80%	75.67%	76.05%	60.57%	13.98%	48.26%
SSP	41.70%	43.94%	50.83%	48.78%	48.38%	47.67%	82.39%	86.95%	66.23%	19.56%	53.64%
ASV	74.47%	36.93%	45.85%	73.51%	73.16%	64.89%	92.29%	45.14%	53.60%	22.02%	58.19%
UAP	44.86%	46.47%	64.67%	65.53%	52.00%	49.63%	82.32%	79.63%	46.46%	19.99%	55.16%
UAPEPGD	66.29%	66.58%	**81.11%**	69.52%	69.16%	59.07%	87.91%	**87.84%**	68.26%	37.12%	69.28%
LLLL_{base}	94.86%	56.30%	61.31%	75.37%	81.45%	67.61%	94.87%	68.25%	77.04%	34.56%	66.89%
$L4A_{fuse}$	96.00%	59.80%	65.00%	77.93%	85.05%	69.39%	**95.02%**	64.41%	76.29%	37.54%	72.64%
$L4A_{ugs}$	**96.13%**	**79.15%**	74.87%	**82.18%**	**95.29%**	**78.73%**	94.45%	55.03%	**77.10%**	**45.09%**	**77.80%**

Table 3: **ViT16** pretrained by **MAE**

ASK	Cars	Pets	Food	DTD	FGVC	Cub	SVHN	CIFAR10	CIFAR100	STL10	AVG
FFF_{no}	64.31%	88.21%	95.04%	88.18%	81.91%	92.94%	76.10%	49.48%	79.83%	60.91%	77.69%
FFF_{mean}	40.39%	67.54%	54.10%	61.38%	71.47%	73.39%	**92.96%**	86.88%	94.55%	67.90%	71.06%
FFF_{one}	48.36%	77.89%	60.06%	64.04%	75.67%	74.09%	92.33%	86.13%	94.48%	70.40%	74.35%
DR	37.02%	23.84%	59.54%	44.73%	28.01%	10.41%	14.30%	16.66%	14.12%	21.54%	27.02%
SSP	44.15%	73.31%	85.42%	72.82%	52.57%	63.10%	52.45%	27.94%	25.32%	36.90%	53.40%
ASV	38.46%	10.17%	37.49%	48.31%	29.14%	4.97%	8.41%	17.04%	11.34%	21.14%	22.64%
UAP	62.71%	58.90%	89.90%	74.92%	44.69%	39.56%	47.65%	47.77%	33.80%	51.70%	55.16%
UAPEPGD	63.67%	73.09%	96.22%	76.69%	57.78%	73.37%	79.84%	45.89%	47.21%	55.79%	66.95%
$L4A_{base}$	87.66%	89.98%	98.96%	**99.10%**	99.33%	84.06%	86.99%	98.62%	97.08%	98.25%	94.00%
$L4A_{fuse}$	83.24%	89.57%	98.87%	98.77%	98.36%	**93.60%**	89.85%	98.64%	95.72%	97.53%	94.42%
$L4A_{ugs}$	**96.49%**	**90.00%**	**98.97%**	98.89%	**99.48%**	84.01%	89.56%	**99.43%**	**97.27%**	**98.96%**	**95.30%**

success rate of 90.05% and 98.96% for Resnet101 and ViT16 in STL10, respectively. Another intriguing finding is that ViT16 with a transformer backbone shows severe vulnerabilities to PAPs. Although performing best on legitimate samples, they bear an attack success rate of **95.30%** under the $L4A_{ugs}$ attack, closing to random output. These results reveal the serious safety problem of the pre-training to fine-tuning paradigm and demonstrate the effectiveness of our method in such a problem setting.

4.3 Ablation studies

4.3.1 Effect of the targeted layer

Here, we analyze the influence of attacking *different intermediate layers* of the networks on the performance of our proposed $L4A_{base}$ in the pre-training domain (Imagenent) and the fine-tuning domain (ten downstream tasks). To this end, we divide Resnet50, Resnet101 and ViT into five layers(Details in Appendix A.2) and operate our algorithm on them. Note that for the fine-tuning domain, the average attack success rates on the ten datasets are reported.

As can be seen from Fig. 4, the lower the level we choose to attack, the better our algorithm performs in the fine-tuning domain. Moreover, for the pre-training domain, attacking the middle layers of the networks results in a higher attack success rate compared to the top and bottom layers, which is also reported in the existing works [28, 34, 51]. These results reveal the intrinsic property of the pre-training to fine-tuning paradigm. As the lower-level layers change less during the fine-tuning procedure, attacking the low-level layer becomes better when generating adversarial samples in the pre-training domain rather than the middle-level layers.

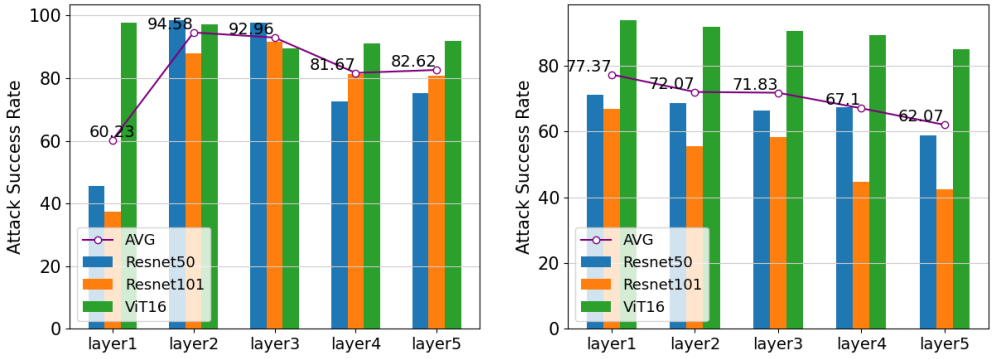
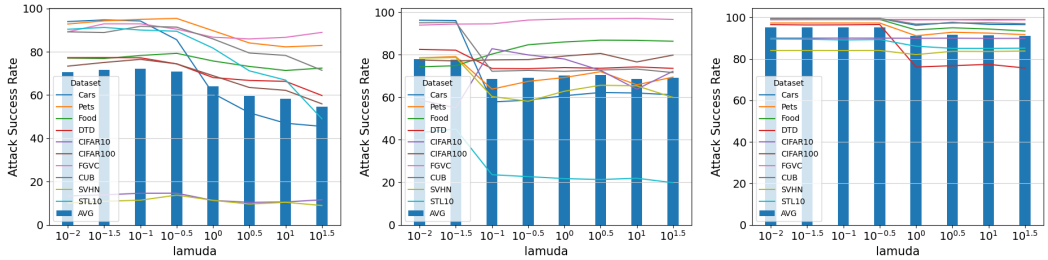


Figure 4: **Left:** Pre-training domain, **Right:** Fine-tuning domain

4.3.2 Effect of Uniform Gaussian Sampling

We set $\boldsymbol{\mu}_l, \boldsymbol{\mu}_h, \boldsymbol{\sigma}_l, \boldsymbol{\sigma}_h$ as 0.4, 0.6, 0.05, 0.1 for all the three models as they perform best. To discuss the effect of the scale factor $\boldsymbol{\lambda}$ in Eq. (6) fusing the base loss and the UGS loss, we select the values with a grid of 8 logarithmically spaced learning rates between 10^{-2} and 10^{2}. Results are shown in Fig. 5. As can be seen from Fig. (a), the best attack success rate is achieved when $\boldsymbol{\lambda}$ is set as $10^{-0.5}$ on Resnet101, boosting the preformance by %1.52 compared to only using the Gaussian noises as inputs.



(a) Resnet101 (b) Resnet50 (c) ViT16

Figure 5: Effect of the scale factor in L4A_{ugs} on different models.

Furthermore, we study whether adopting the fixed statistics of the pre-training dataset (i.e., the mean and standard deviation of Imagenet) can help. The results are shown in the Table 4, where **None** refers to using no data augmentation, **Imagenet** adopts the mean and standard deviation of Imagenet and **Uniform** samples a pair of mean and standard deviation from the uniform distribution first. As can be seen from the table, **Uniform** outperforms **None** by 4.39% on average, while **Imagenet** does not help, which means that our proposed UGS helps avoid overfitting the pre-training domain.

Table 4: Fixed datasets' statistics

Model	R101	R50	MAE
None	66.95%	71.16%	94.00%
Imagenet	68.39%	69.50%	95.30%
Uniform	72.20%	77.80%	95.30%

4.4 Visualization of feature maps

We show the feature maps before and after L4A_{fuse} attack in Fig.6. The **Left** column shows the inputs of the model, while the **Middle** and the **Right** show the feature map of the pre-trained model and the fine-tuned respectively. The **Upper** row represents the pipeline of a clean input in Cars, and the **Lower** shows that of adversarial ones. We can see from the upper row that fine-tuning the model can make it sensitive to the defining features related to the specific domain, such as tires and lamps. However, adding an adversarial perturbation to the image can significantly lift up all the activations

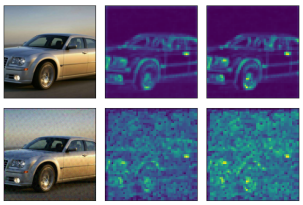
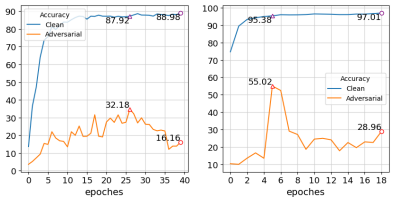


Figure 6: Visualization

and finally mask the useful features. Moreover, the effect of our attack could be well preserved during fine-tuning and cheat the fine-tuned model into misclassification, stressing the safety problem of pre-trained models.

4.5 Trade-off between the clean accuracy and robustness

In this subsection, we study the effect of fine-tuning epochs on the performance of our attack. To that end, we fine-tune the model till it reports the best result on the testing dataset, and then we plot the classification accuracy on Pets and STL10 before and after the attack in Fig. 7. A trade-off between the clean accuracy and robustness of the fine-tuned model can be seen in Fig. For example, the model shows the best robustness at epoch 5 in STL10, achieving an accuracy rate of 95.38% and 55.02% on clean and adversarial samples, respectively. However, the model does not converge until epoch 19. Though the process boosts the clean accuracy by **1.63%**, it suffers a significant drop in robustness, as the accuracy on adversarial samples is lowered to **28.96%**. Such a finding shows agreement with the trade-off between robustness and accuracy reported by Zhang et al. [50]. However, we adopt the pre-training to fine-tuning paradigm where adversarial samples are generated using pre-trained models and evaluated on fine-tuned ones while they operate on the same model.



(a) pets (b) stl10

Figure 7: Trade-off during fine-tuning

5 Discussion

In this section, we first introduce the gradient alignment and then use it to explain the effectiveness of our methods. In particular, we show why our algorithms fall behind UAPs in the pre-training domain but have better cross-finetuning transferability when evaluated on the downstream tasks.

5.1 Preliminaries

Gradient sequences. Given a network $f_{\boldsymbol{\theta}_0}$ and a sample sequence $\{\mathbf{x}_1\, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_N\}_{\mathcal{D}}$ drawn from the dataset $\mathcal{D}$, let $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathcal{D}} = \{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_1}, \Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_2}, \Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_3}, ..., \Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,xN}\}_{\mathcal{D}}$ be the sequence of gradients obtained when generating adversarial samples by the following equation:

$$\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_i} = \nabla_{\boldsymbol{\delta}} L(f_{\boldsymbol{\theta}_0}, \mathbf{x}_{i-1}, \boldsymbol{\delta}_i), \text{ with } \boldsymbol{\delta}_i = P_{\infty,\boldsymbol{\epsilon}}(\boldsymbol{\delta}_{i-1} + \Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_{i-1}}) \tag{7}$$

where L denotes the loss function of iterative attack methods like UAP, FFF and L4A.

Gradient alignment. Given a dataset $\mathcal{D}$ and a model $f_{\boldsymbol{\theta}_0}$, the gradient alignment $\mathcal{GA}$ of an attack algorithm is defined as the expectation over the cosine similarity of $\Delta_{\boldsymbol{\theta}_0,\mathbf{x}_1}$ and $\Delta_{\boldsymbol{\theta}_0,\mathbf{x}_2}$, which can be formulated as Eq. (8)

$$\mathcal{GA} = \mathbb{E}_{\mathbf{x1}\sim\mathcal{D},\mathbf{x2}\sim\mathcal{D}}\Big[\frac{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_1} \cdot \Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_2}}{||\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_1}||_2||\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_2}||_2}\Big], \tag{8}$$

where $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_1}$ and $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta}_0,\mathbf{x}_2}$ are two consecutive elements obtained by Eq. (7).

Then *the L4A algorithm bears a high gradient alignment*(A strict definition and proof in a weaker form in Appendix A.3). In addition, we provide the results of a simulation experiment to justify it in Table 5. As we can see, there is a negative correlation between the gradient alignment and the attack success rate on Imagenet. In contrast, a positive correlation exists between the gradient alignment and ASK in the fine-tuning domain.

Effectiveness of the algorithm. Provided a pre-trained model f_θ and the pre-training dataset $\mathcal{D}_p$, then the generation of PAPs can be reformulated from Eq. (1) to the following equation:

$$\max_{\boldsymbol{\delta}\in\, Span\{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}\}} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_t}[F_{\boldsymbol{\theta'}}(\mathbf{x}) \neq F_{\boldsymbol{\theta'}}(\mathbf{x}+\boldsymbol{\delta})], \text{ with } ||\boldsymbol{\delta}||_p \leq \boldsymbol{\epsilon} \text{ and } \mathbf{x}+\boldsymbol{\delta} \in [0,1] \tag{9}$$

where $Span\{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}\}$ denotes the subspace spanned by the elements of $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}$. The optimal value δ can be viewed as a linear combination of the elements in $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}$, so it is the *feasible region* of the equation. In particular, the feasible region of Eq. (9) is smaller than that of Eq. (1), which reflects

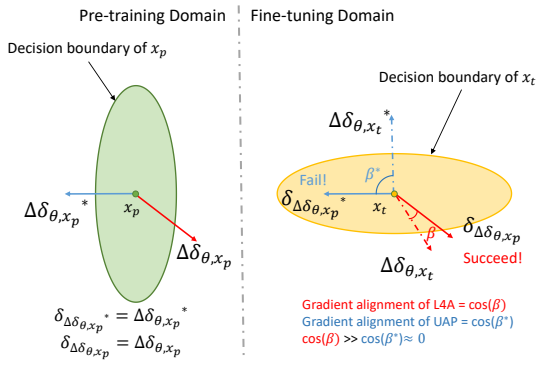


Figure 8: Illustration

Resnet101	$\mathcal{GA}$	Imagenet	AVG
FFF_{no}	0.1221	38.64%	48.55%
FFF_{mean}	0.0194	48.37%	44.22%
FFF_{one}	0.1222	34.15%	40.26%
DR	0.0165	41.06%	42.62%
UAP	0.0018	88.14%	43.86%
UAPEPGD	0.0008	**94.62%**	59.39%
SSP	0.0274	41.81%	40.75%
L4A_{base}	**0.6125**	37.44%	**66.95%**

Table 5: **Left:**Gradient alignments, **Middle:**Attack success rates on the pre-training dataset, **Right:**Average ASK on downstream tasks. Details in Appendix A.4

that the stochastic gradient descent methods may not converge to the global maximum when it is not included in $Span\{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}\}$ and the local maximum of Eq. (9) in $Span\{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}\}$ represents the effectiveness of the algorithm in the fine-tuning domain.

5.2 Explanation

We aim to explain why the effectiveness of our algorithm is higher in the fine-tuning domain while lower in the pre-training domain, as can be seen from Table 5. Let $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}$ and $\Delta\boldsymbol{\delta}^*_{\boldsymbol{\theta},\mathcal{D}_p}$ be the feasible zones of L4A and UAP obtained by feeding instances from D_p into the pre-trained model $f_{\boldsymbol{\theta}}$, respectively. Similarly, we can define $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_t}$ and $\Delta\boldsymbol{\delta}^*_{\boldsymbol{\theta},\mathcal{D}_t}$. Meanwhile, denote $\boldsymbol{\delta}_{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}}$ and $\boldsymbol{\delta}^*_{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}}$ as the maxima in the pre-training domain obtained by L4A and UAP respectively. An illustration is shown in Fig. 8 supposing there is only one step in the iterative method.

Pre-training domain: Because the gradients of UAP obtained by Eq. (2) represent the directions to the closest points on the decision boundary in the pre-training domain. Thus, limiting the feasible zone to $\Delta\boldsymbol{\delta}^*_{\boldsymbol{\theta},\mathcal{D}_p}$ does little harm to the performance when evaluated in the pre-training domain. Meanwhile, according to the optimal objective, L4A finds the next best directions which are worse than those of UAPs. Thus, $\boldsymbol{\delta}^*_{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}}$ performs better than $\boldsymbol{\delta}_{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}}$ in the pre-training domain.

Fine-tuning domain: According to the fact that UAP bears a low gradient alignment near to 0, the subspace spanned by the tensors in $\Delta\boldsymbol{\delta}^*_{\boldsymbol{\Delta}_{\boldsymbol{\theta},\mathcal{D}_p}}$ is almost orthogonal to that spanned by the tensors in $\Delta\boldsymbol{\delta}^*_{\boldsymbol{\Delta}_{\boldsymbol{\theta},\mathcal{D}_t}}$ which represent the best directions that send the sample to the decision boundary in the fine-tuning domain. Thus, limiting the feasible zone of Eq. (1) to $\Delta\boldsymbol{\delta}^*_{\boldsymbol{\theta},\mathcal{D}_p}$ suffers a great drop in ASK when evaluated on the downstream tasks in Eq. (9). However, as shown in Table 5, our algorithm can achieve a gradient alignment of up to 0.6125, which means that there is considerable overlap in the next best feasible region of $\mathcal{D}_p$ in the pre-training domain obtained by L4A and that of $\mathcal{D}_f$ in the fine-tuning domain. Thus the performance of the best solution in $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}$ is close to that of $\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_f}$ which represent the next best solution in the fine-tuning domain. Finally we have $\boldsymbol{\delta}_{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}}$ performs better than $\boldsymbol{\delta}^*_{\Delta\boldsymbol{\delta}_{\boldsymbol{\theta},\mathcal{D}_p}}$ in the fine-tuning domain.
In conclusion, *the high gradient alignment guarantees high cross-finetuning transferability*.

6 Conclusion

In this paper, we address the safety problem of pre-trained models. In particular, an attacker can use them to generate so-called pre-trained adversarial perturbations, which can achieve a high success rate on the fine-tuned models without knowing the victim model and the specific downstream tasks. Considering the inner qualities of the pre-training to fine-tuning paradigm, we propose a novel algorithm, L4A, which performs well in such problem settings. A potential negative societal impact of L4A is that it could be used by malicious adversaries to cause security/safety issues in real-world applications. Furthermore, L4A only utilizes the information in the pre-training domain. When the attacker manages to obtain some information about the targeted tasks, like several unlabeled instances in the fine-tuning dataset, he may be able to enhance his PAPs using the knowledge and further exacerbate the situation, which we leave for future work. Thus, we hope our work can draw attention to the safety problem of pre-trained models to guarantee security.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901, 2020.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[3] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[4] Tong Chen and Zhan Ma. Towards robust neural image compression: Adversarial attack and model finetuning. *arXiv preprint arXiv:2112.08691*, 2021.

[5] Yingpeng Deng and Lina J Karam. Universal adversarial attack via enhanced projected gradient descent. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1241–1245. IEEE, 2020.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Xinshuai Dong, Anh Tuan Luu, Min Lin, Shuicheng Yan, and Hanwang Zhang. How should pre-trained language models be fine-tuned towards adversarial robustness? *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[9] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4312–4321, 2019.

[10] Lijie Fan, Sijia Liu, Pin-Yu Chen, Gaoyuan Zhang, and Chuang Gan. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? *Advances in Neural Information Processing Systems*, 34, 2021.

[11] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[13] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4805–4814, 2019.

[14] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[18] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[20] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. *Advances in Neural Information Processing Systems*, 33:16199–16210, 2020.

[21] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[22] Valentin Khrulkov and Ivan Oseledets. Art of singular vectors and universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8562–8570, 2018.

[23] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[24] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[25] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[27] Yantao Lu, Yunhan Jia, Jianyu Wang, Bai Li, Weiheng Chai, Lawrence Carin, and Senem Velipasalar. Enhancing cross-task black-box transferability of adversarial examples with dispersion reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 940–949, 2020.

[28] Yantao Lu, Yunhan Jia, Jianyu Wang, Bai Li, Weiheng Chai, Lawrence Carin, and Senem Velipasalar. Enhancing cross-task black-box transferability of adversarial examples with dispersion reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 940–949, 2020.

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[30] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[32] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. *arXiv preprint arXiv:1707.05572*, 2017.

[33] Muhammad Muzammal Naseer, Salman H Khan, Muhammad Haris Khan, Fahad Shahbaz Khan, and Fatih Porikli. Cross-domain transferability of adversarial perturbations. *Advances in Neural Information Processing Systems*, 32, 2019.

[34] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 262–271, 2020.

[35] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[36] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[37] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[38] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer, 2020.

[39] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[41] Ananya B Sai, Akash Kumar Mohankumar, Siddhartha Arora, and Mitesh M Khapra. Improving dialog evaluation with a multi-reference adversarial dataset and large scale pretraining. *Transactions of the Association for Computational Linguistics*, 8:810–827, 2020.

[42] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[43] Trieu H Trinh, Minh-Thang Luong, and Quoc V Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[45] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2730–2739, 2019.

[46] Xueting Yan, Ishan Misra, Abhinav Gupta, Deepti Ghadiyaram, and Dhruv Mahajan. Clusterfit: Improving generalization of visual representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6509–6518, 2020.

[47] Zonghan Yang and Yang Liu. On robust prefix-tuning for text classification. *arXiv preprint arXiv:2203.10378*, 2022.

[48] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[50] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[51] Qilong Zhang, Xiaodan Li, Yuefeng Chen, Jingkuan Song, Lianli Gao, Yuan He, and Hui Xue. Beyond imagenet attack: Towards crafting adversarial examples for black-box domains. *arXiv preprint arXiv:2201.11528*, 2022.

[52] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *Springer International Publishing*, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix A.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Since running L4A on all the ten dataset and the three models is time-consuming, we only ran one time.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [Yes] See Appendix C.

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplementary material.

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix C.

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix C.

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]