Data-Free Universal Attack by Exploiting the Intrinsic Vulnerability of Deep Models

YangTian Yan, Jinyu Tian*

Faculty of Innovation Engineering, Macau University of Science and Technology
naygnahz1128@gmail.com, jytian@must.edu.mo

Abstract

Deep neural networks (DNNs) are susceptible to Universal Adversarial Perturbations (UAPs), which are instance-agnostic perturbations that can deceive a target model across a wide range of samples. Unlike instance-specific adversarial examples, UAPs present a greater challenge as they must generalize across different samples and models. Generating UAPs typically requires access to numerous examples, which is a strong assumption in real-world tasks. In this paper, we propose a novel data-free method called **Intri**nsic **UAP** (**IntriUAP**), by exploiting the intrinsic vulnerabilities of deep models. We analyze a series of popular deep models composed of linear and nonlinear layers with a Lipschitz constant of 1, revealing that the vulnerability of these models is predominantly influenced by their linear components. Based on this observation, we leverage the ill-conditioned nature of the linear components by aligning the UAP with the right singular vectors corresponding to the maximum singular value of each linear layer. Remarkably, our method achieves highly competitive performance in attacking popular image classification deep models without using any image samples. We also evaluate the black-box attack performance of our method, showing that it matches the state-of-the-art baseline for data-free methods on models that conform to our theoretical framework. Beyond the data-free assumption, IntriUAP also operates under a weaker assumption, where the adversary only can access a few of the victim model's layers. Experiments demonstrate that the attack success rate decreases by only 4% when the adversary has access to just 50% of the linear layers in the victim model.

Code — https://github.com/yyt0718/Intri_Attack

1 Introduction

Deep Neural Networks (DNNs) have shown remarkable performance in tasks such as computer vision (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014; He et al. 2016), natural language processing (Bahdanau, Cho, and Bengio 2014; Vaswani et al. 2017; Devlin et al. 2018; Biderman et al. 2023), etc. However, recent research (Goodfellow, Shlens, and Szegedy 2015; Moosavi-Dezfooli, Fawzi, and Frossard 2016; Madry et al. 2019) has highlighted their vulnerability to adversarial attacks. Adversarial scenarios arise when models face inputs specifically designed to deceive or mislead them, often resulting in erroneous outputs. This vulnerability is not merely academic but poses significant practical challenges, especially in critical applications such as autonomous driving and facial recognition systems. As models become more integral to our daily lives, ensuring their reliability against such adversarial attacks is crucial. This motivates ongoing research to understand and mitigate these threats, aiming to develop more reliable and secure DNNs.

Among the spectrum of adversarial strategies, **U**niversal **A**dversarial **P**erturbations (UAPs) represent a particularly important form. UAPs are image-agnostic perturbations, which means a single perturbation, once computed, can be applied to a wide variety of images to consistently mislead a machine learning model. Most of the current work on UAPs requires massive samples (Moosavi-Dezfooli et al. 2017; Mopuri et al. 2018; Poursaeed et al. 2018; Liu et al. 2023). These methods, although efficient, necessitate extensive samples and significant time for iterative training and adjustment of perturbations. They also assume unrestricted access to the original training data or sizable representative datasets. This requirement limits their practicality in scenarios where data privacy or accessibility is constrained. In contrast, a few existing approaches attempt to generate UAPs without direct reliance on extensive large datasets (Mopuri, Garg, and Babu 2017; Mopuri, Uppala, and Babu 2018; Mopuri, Ganeshan, and Babu 2018; Liu et al. 2019a; Zhang et al. 2020b, 2021). They generally follow two lines. The first involves designing batches of proxy datasets, such as using GANs to generate some proxy samples (Mopuri, Uppala, and Babu 2018). Another type of UAPs leverages the characteristic of the model, for instance, exploiting the observation that adversarial samples can occupy a large portion of the label space (Zhang et al. 2020b, 2021).

Recent studies have shown that specific patterns (Mopuri, Ganeshan, and Babu 2018; Liu et al. 2019b) or even random noise (Fawzi, Moosavi-Dezfooli, and Frossard 2016) can successfully attack deep models. This suggests that the existence of UAPs is primarily due to intrinsic flaws within the models themselves, rather than being learned from a large number of samples. Several studies support this view by analyzing the geometric properties of decision bound-

*The corresponding author.

aries (Goodfellow, Shlens, and Szegedy 2015; Zhao et al. 2024; Su et al. 2024; Tao et al. 2023). Along this line, in this paper, we leverage the inherent vulnerabilities of deep models to design UAPs, which we refer to as **Intri**nsic **UAP (IntriUAP)**, without relying on training data. Specifically, we conduct an in-depth analysis of a popular type of deep model that we call the **L**inear and **1**-**L**ipschitz **O**perator **S**ystem **(L1LOS)** and propose a novel data-free UAP method based on its intrinsic vulnerabilities. The L1LOS is composed of linear operators (layers in deep models) and nonlinear operators with a Lipschitz constant of 1. Several popular deep models, such as VGG (Simonyan and Zisserman 2014), ResNet (He et al. 2016), and GoogleNet (Szegedy et al. 2015), fall into this category. We demonstrate that in an L1LOS, its vulnerability is primarily governed by the linear operators. By exploiting their ill-conditioned characteristics, we can amplify the input perturbation by aligning it with the largest right singular vector of each linear operator.

In summary, our contributions can be summarized as follows:

1) We propose a novel data-free universal attack method called IntriUAP by exploiting the intrinsic vulnerability of a popular type of model called L1LOS which includes most deep models.

2) We theoretically reveal that the vulnerability of an L1LOS model is majorly dominated by its linear parts, leading to the interpretability of the proposed IntriUAP.

3) Since our IntriUAP exploits the intrinsic vulnerability of the linear layers in deep models, it does not require full knowledge of the victim model to execute an attack. The IntriUAP can achieve a comparable attack success rate with only partial knowledge of the model's weights, making our method effective under a weak assumption of adversarial knowledge.

4) We benchmarked our method against the latest data-free and data-dependent UAPs on the ImageNet dataset. Our method achieves state-of-the-art performance among data-free approaches and is even comparable to some data-dependent methods in certain aspects.

2 Related Work

Data-dependent Universal attack methods. The pioneering work Universal Adversarial Perturbations (Moosavi-Dezfooli et al. 2017) first revealed the existence of UAPs, generating them through iterative processes using DeepFool. SV-UAP method (Khrulkov and Oseledets 2018) employed the singular vectors of the Jacobian matrices of certain layers' feature maps to generate UAPs, achieving this with a minimal number of samples. Both Poursaeed et al. and Mopuri et al. introduced approaches to synthesize UAPs using generative models, termed GAP (Poursaeed et al. 2018) and NAG (Mopuri et al. 2018), respectively. Zhang et al. proposed a technique for crafting UAPs aimed at specific target classes, dubbed CD-UAP (Zhang et al. 2020a). Furthermore, Zhang's additional works, DF-UAP (Zhang et al. 2020b) and Cos-UAP (Zhang et al. 2021), leveraged dominant features and a cosine similarity loss function to create UAPs. Li et al. proposed an approach that integrates instance-specific and universal attacks from a feature perspective called AT-UAP (Li et al. 2022). Recently, Liu et al. introduced a novel technique employing stochastic gradient aggregation to create a more powerful UAP, termed SGA-UAP (Liu et al. 2023).

Data-free Universal attack methods. To generate powerful UAPs, dependency on samples still remains a pivotal issue. Currently, several data-free UAP methods have been developed. The earliest data-free approach, Fast Feature Fool (Mopuri, Garg, and Babu 2017), generates UAPs by maximizing activations at each layer. Subsequently, GD-UAP (Mopuri, Ganeshan, and Babu 2018) was introduced, enhancing FFF and extending its applicability to a broader range of vision tasks. Following this, the AAA (Mopuri, Uppala, and Babu 2018) method was proposed, employing class-impressions to create UAPs. PD-UA (Liu et al. 2019a), utilized a Monte Carlo sampling approach to increase model uncertainty. Zhang et al. involved the use of proxy datasets for UAP generation, called AT-UAP (Zhang et al. 2020b), but this still relied on real samples. Later, Zhang et al. introduced Cos-UAP (Zhang et al. 2021), which completely replaced proxy datasets with manual samples, thus elevating the development of data-free UAPs to a new level.

3 The Proposed Method

In this section, we provide the motivation and formulate the optimization problem of our IntriUAP. We constrain our discussion to the classification task.

3.1 Problem Statement

Consider a deep model classifier $f(\boldsymbol{x})$ with the input $\boldsymbol{x}$, the generation of a UAP $\boldsymbol{\xi}$ generally can be expressed as the following optimization problem:

$$\arg\max_{\boldsymbol{\xi}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(\boldsymbol{x}_i + \boldsymbol{\xi}\right), \boldsymbol{y}_i\right), \text{s.t. } \|\boldsymbol{\xi}\|_p \leqslant \epsilon, \quad (1)$$

where $f\left(\boldsymbol{x}_i + \boldsymbol{\xi}\right)$ represents the predicted label, $\boldsymbol{y}_i$ is the true label of $\boldsymbol{x}_i$, and $\mathcal{L}(\cdot)$ denotes the adversarial loss, such as cross-entropy. However, such problems have a flaw, namely that adversarial loss often requires samples and their true labels, which lack practicality due to data privacy and accessibility issues.

To eliminate the assumption of access to the training or validation datasets, as well as the ground truth labels, we propose leveraging the intrinsic vulnerabilities of deep models to design UAP. More precisely, consider an ℓ-layer neural network $f(\boldsymbol{x})$ described by the following recursive equations:

$$\begin{aligned} f(\boldsymbol{x}) &= W_\ell \boldsymbol{x}_\ell + \boldsymbol{b}_\ell, \\ \boldsymbol{x}_{k+1} &= \phi\left(W_k \boldsymbol{x}_k + \boldsymbol{b}_k\right), \quad \boldsymbol{x}_1 = \boldsymbol{x}, \end{aligned} \quad (2)$$

where $k = 1, \cdots, \ell - 1$, $\boldsymbol{x}_k$ is the input feature of the k-th linear layer, ϕ represents a nonlinear layer, which can be an activation function, max pooling, etc., W_k and $\boldsymbol{b}_k$ are the layer-wise weight matrix and bias vector respectively. Clearly, the above formula is an abstract representation of

deep models, which consists of linear layers [1] (such as convolutional layers, batch normalization layers, etc.) and nonlinear layers (activation functions, pooling layers, etc.).

Now, suppose the model $f(\boldsymbol{x})$ belongs to the L1LOS category, i.e., the Lipschitz constant of all the nonlinear layers is equal to 1. We have the following important observations about $f(\boldsymbol{x})$. **For an L1LOS model $f(\boldsymbol{x})$, its stability is dominated by its linear parts.** In other words, those nonlinear layers with Lipschitz constant 1 would not amplify the input perturbation to cause a large output error. Thus, we can focus on the linear parts and leverage their ill-conditioned characteristics to design UAPs. As for the linear part, we have another observation. **For a linear operator, the right singular vector corresponding to the maximal singular value could maximize the ℓ_2-norm of the difference between the original and perburbated output of the linear operator**. The above two observations could guide the design of our proposed IntriUAP. That is, the ideal UAP should be aligned with the right singular vector corresponding to the maximal singular value. We will provide formal justification for the above two observations in Section 5. Before this, we first provide the detailed process of our proposed IntriUAP in the upcoming sections.

3.2 The Formulation of Intrinsic UAP

According to the discussion above, IntriUAP aims to design perturbations being aligned with the right singular vector corresponding to the maximal singular value of each linear layer of an L1LOS model. Specifically, let $\boldsymbol{\xi}$ be the UAP to be optimized. Obeying the notations in formula (2), the output error of the first linear layer caused by the input perturbation $\boldsymbol{\xi}$ is

$$\boldsymbol{\delta}_{1+1} = W_1\left(\boldsymbol{x}_1 + \boldsymbol{\xi}\right) - W_1\left(\boldsymbol{x}_1\right) = W_1\boldsymbol{\xi}. \tag{3}$$

Similarly, using a recursive form, the output error of the k-th linear layer is as follows

$$\boldsymbol{\delta}_{k+1} = W_k\left(\boldsymbol{x}_k + \boldsymbol{\delta}_k\right) - W_k\left(\boldsymbol{x}_x\right) = W_k\boldsymbol{\delta}_k \tag{4}$$

where $\boldsymbol{x}_k$ is the input of the k-th linear layer, and $\boldsymbol{\delta}_k$ is the perturbation transmitted from the front linear layers.

Based on our second observation, for the k-th linear operation $\boldsymbol{\delta}_{k+1} = W_k\boldsymbol{\delta}_k$ $(k = 1, ..., \ell)$, the right singular vector corresponding to the maximal singular value of W_k would maximize the ℓ_2-norm of the output of this linear layer (we denote this singular vector by $\boldsymbol{v}_k$). That is the optimal input perturbation $\boldsymbol{\delta}_k$ to maximize the $\|\boldsymbol{\delta}_{k+1}\|_2$ should be align with $\boldsymbol{v}_k$, i.e., $\boldsymbol{\delta}_k/\|\boldsymbol{\delta}_k\|_2 = \boldsymbol{v}_k$. By viewing the input perturbation $\boldsymbol{\delta}_k$ of each layer as a function of the expected UAP $\boldsymbol{\xi}$, the ideal UAP $\boldsymbol{\xi}$ that would cause the maximal model output error thus should be the one, making each $\boldsymbol{\delta}_k$ be align with $\boldsymbol{v}_k$. Therefore, the optimization problem of our IntriUAP could be formally expressed as follows

$$\arg\max_{\boldsymbol{\xi}} \sum_{k=1}^{\ell} \left|\langle \boldsymbol{\delta}_k(\boldsymbol{\xi}), \boldsymbol{v}_k\rangle\right|, \text{ s.t. } \|\boldsymbol{\xi}\|_p \leqslant \epsilon. \tag{5}$$

[1] In this paper, we collectively refer to linear and affine as linear, because the bias term would not introduce additional output error.

Algorithm 1: Algorithm for Our Proposed IntriUAP Method

Require: Target l-layers CNN f, initial data $\boldsymbol{x}$, initial perturbation $\boldsymbol{\xi}$, Scheduler, learning rate γ, maximum perturbation value ℓ_∞-norm ϵ, epoch number T.
Ensure: Perturbation $\boldsymbol{\xi}$

1: Calculate the maximum singular vectors $\boldsymbol{v}_k$ for $k = 1, \ldots, \ell$ of all linear layers of f.
2: **for** $t = 1$ to T **do**
3: **for** $k = 1$ to ℓ **do**
4: $\boldsymbol{\delta}_{k+1} \leftarrow W_k(\boldsymbol{x}_k + \boldsymbol{\delta}_k) - W_k(\boldsymbol{x}_k)$
5: **end for**
6: $\mathcal{L} \leftarrow -\sum_{k=1}^{\ell} |\langle \boldsymbol{\delta}_k(\boldsymbol{\xi}), \boldsymbol{v}_k\rangle|$
7: **Backpropagate** the loss $\mathcal{L}$ to compute gradients $\nabla_{\boldsymbol{\xi}}\mathcal{L}$
8: **Update** the perturbation $\boldsymbol{\xi}$ using gradient descent: $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \gamma\nabla_{\boldsymbol{\xi}}\mathcal{L}$
9: $\boldsymbol{\xi} \leftarrow \text{Clip}^{\epsilon}(\boldsymbol{\xi})$
10: **end for**
11: **return** $\boldsymbol{\xi}$

where the hyper-parameter ϵ controls the perturbation magnitude of the UAP $\boldsymbol{\xi}$.

We can observe several advantages of our IntriUAP from the optimization problem (5). Firstly, this problem does not involve any input example $\boldsymbol{x}$ from the same distribution as the training dataset, which implies that IntriUAP is a data-free approach. Another notable aspect is that the problem does not require any training loss function, such as cross-entropy, which is commonly used in existing UPA methods. This indicates that solving the problem (5) does not necessitate the use of the entire model; even a few linear layers can produce a satisfactory UAP. For instance, we can consider ℓ representing the first few linear layers in a model. This characteristic can partially limit the adversary's capabilities, making the attack scenario more realistic. We will validate these two advantages through our experiments in Section 6.

3.3 The Algorithm for Intrinsic UAP

In this section, we introduce the details of generating Intrinsic UAP. We follow the optimization procedure outlined in Algorithm 1. Firstly, we focus on an L1LOS deep model, which is inherently compatible with our intrinsic attack methodology. We then calculate all of its maximum singular vectors $\boldsymbol{v}_k$ for $k = 1, \ldots, \ell$. The initial data $\boldsymbol{x}$ can be a range prior, Gaussian noise, or uniform noise. We employ the Adam optimizer combined with a StepLR scheduler.

Next, during the optimization phase of Algorithm 1, lines 3 and 4 compute the perturbed outputs $\boldsymbol{\delta_k}$ for each linear layer. Lines 6 to 8 involve calculating the inner product of $\boldsymbol{\delta_k}$ with the corresponding singular vectors $\boldsymbol{v}_k$, performing backpropagation, and optimizing the initial perturbation $\boldsymbol{\xi}$. Finally, a clipping operation is applied to ensure that the ℓ_∞-norm of $\boldsymbol{\xi}$ remains within the bound of 10. The above process is repeated until the loss converges.

4 Deep Models as L1LOS

In this section, we will demonstrate that certain popular networks, including VGG, ResNet, GoogleNet, AlexNet, etc.,

are L1LOS. As a result, our proposed UAP is fully compatible with the task of attacking these widely used deep models. Since our proposed method relies on the singular vectors of the linear layers, we then discuss how to represent the widely adopted linear layers, the convolutional layers, and the BatchNorm layers as matrix operations.

4.1 View Popular Deep Models as L1LOS

For networks such as VGG, GoogleNet, and ResNet, the linear components during the inference stage consist of convolutional layers, batch normalization layers, and fully connected layers. The nonlinear components are primarily composed of ReLU activations and max-pooling layers, both of which have a Lipschitz constant of 1 (Kim, Papamakarios, and Mnih 2021).

For networks like VGG, the L1LOS structure is naturally satisfied. In the case of GoogleNet with its Inception modules, each branch within an Inception module adheres to the L1LOS structure, with a final concatenate operation. For ResNet, with its residual block, Assuming a residual block $\mathcal{F}$ consists of two convolutional layers W_1 and W_2, along with corresponding normalization layers BN_1 and BN_2, as well as their respective nonlinear activations σ, the structure of the residual block can be expressed as:

$$\mathcal{F}(\boldsymbol{x}) = \sigma(\mathrm{BN}_2(W_2 * \sigma(\mathrm{BN}_1(W_1 * \boldsymbol{x})))) + \boldsymbol{x}. \quad (6)$$

Although connecting multiple residual blocks introduces more complex nonlinearity, ResNet still satisfies the L1LOS structure.

4.2 Convolutional Layers as a Linear Operator

There are already existing works that represent convolutional kernels as matrices in different ways (Sedghi, Gupta, and Long 2018; Araujo et al. 2021; Praggastis et al. 2022). We know that a 2-D convolutional kernel can be represented as a doubly-block Toeplitz matrix.

Given any kernel coefficients K, the matrix representation for the convolution by K is represented by the following doubly-block circulant matrix:

$$W = \begin{bmatrix} \mathrm{circ}(K_{0,:}) & \mathrm{circ}(K_{1,:}) & \cdots & \mathrm{circ}(K_{n-1,:}) \\ \mathrm{circ}(K_{1,:}) & \mathrm{circ}(K_{2,:}) & \cdots & \mathrm{circ}(K_{0,:}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{circ}(K_{n-1,:}) & \mathrm{circ}(K_{0,:}) & \cdots & \mathrm{circ}(K_{n-2,:}) \end{bmatrix} \quad (7)$$

where $\mathrm{circ}(K_{i,:})$ is the circulant matrix formed from the i-th row of the matrix K, where each row of the circulant matrix is a right cyclic shift of the previous row, and W is the Toeplitz matrix representing the convolution (Goodfellow, Bengio, and Courville 2016, Page 329)). By vectorizing the input matrix X, we can represent the convolution operation as the matrix multiplication $\mathrm{vec}(Y) = W \cdot \mathrm{vec}(X)$. More details for gain W and presenting a convolutional layer as the matrix operation can be found in the supplementary material.

4.3 BatchNorm Layer as a Linear Operator

The BatchNorm layer is widely used in CNNs to accelerate training and enhance stability by normalizing the input features. During the training phase, BatchNorm includes nonlinear operations, such as the calculation of batch statistics. However, during inference, when batch statistics are replaced by moving averages, the BatchNorm layer can be simplified to a linear transformation. Since the task of UAP is to attack a well-trained model, therefore it is reasonable to treat the BatchNorm layer as a linear operator. Then, we introduce how to represent it as a matrix operation.

Given an input $\mathbf{X} \in \mathbb{R}^{c\times m\times m}$ to the BatchNorm layer, where c is the number of channels and $m \times m$ is the spatial dimension, the output $\mathbf{Y}$ can be expressed as $\mathrm{vec}(\mathbf{Y}) = A \cdot \mathrm{vec}(\mathbf{X}) + \boldsymbol{b}$. Here, A is a diagonal matrix in $\mathbb{R}^{(c\times m\times m)\times(c\times m\times m)}$, which represents the linear transformation, while $\boldsymbol{b}$ accounts for the shift. The diagonal elements of A, denoted as A_{ii}, are defined as:

$$A_{ii} = \frac{\gamma_c}{\sqrt{v_c + \varepsilon}}, \quad c = \left\lfloor \frac{i}{m \times m} \right\rfloor + 1, \quad (8)$$

where γ_c is the scale factor, v_c is the moving variance for channel c, ε is a small constant added for numerical stability, and $\lfloor\cdot\rfloor$ denotes the floor function. The index i ranges over all elements in the input tensor, $i \in [0, 1, \ldots, c \times m \times m - 1]$. The vector $\boldsymbol{b} \in \mathbb{R}^{(c\times m\times m)\times 1}$ is the bias term, with each element $\boldsymbol{b}_i$ given by:

$$\boldsymbol{b}_i = \beta_c - \mu_c \times \frac{\gamma_c}{\sqrt{v_c + \varepsilon}}, \quad (9)$$

where β_c is the shift factor and μ_c is the moving mean for channel c.

5 Theoretical Justification

In this section, we provide the theoretical justification of the two observations inspiring our IntriUAP which we have briefly introduced in Section 3. The detailed proof can be found in the supplementary material. The first observation is that **for a system composed of nonlinear operators with the Lipschitz constant is 1 and linear operators, (i.e., L1LOS), the stability of this system is dominated by its linear parts.** The formal justification of this observation is as follows:

Theorem 1 *For a L1LOS $f(\boldsymbol{x})$ defined in formula (2), we have* $\mathrm{Lip}\,(f) \leq \prod_{k=0}^{\ell-1} \|W_k\|_{op}$, *where* $\mathrm{Lip}\,(f)$ *is the Lipschitz constant of $f(\boldsymbol{x})$.*

As indicated by the above theorem, the upper bound on the Lipschitz constant for a model f is primarily governed by its linear operators. This implies that integrating 1-Lipschitz nonlinear operators into a linear system will not increase the Lipschitz constant of this model. It is easy to show that 1-Lipschitz nonlinear operators will not amplify the perturbation according to the definition of the Lipschitz constant. Thus, we only need to consider linear operators for amplifying the input perturbation.

Upon having the above conclusion, we also need to know what kind of input perturbation would be amplified by the linear operators to cause a large output error. We thus have the second observation discussed in Section 3. 2) **For a linear operator, the right singular vector corresponding to**

the maximal singular value could maximize the output of this operator measured by the L_p-norm, which can be supported by the following theorem

Theorem 2 *For any bounded linear operator, the operator norm $\|A\|_{op}$ is equal to its largest singular value $\sigma_{\max}(A)$, and the corresponding right singular vector $\boldsymbol{v}_{\max}$ achieves this maximum, such that:*

$$\|A\|_{op} = \sigma_{\max}(A) = \sup_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2 = \|A\boldsymbol{v}_{\max}\|_2,$$

where $\boldsymbol{v}_{\max}$ is the right singular vector corresponding to $\sigma_{\max}(A)$.

This theorem shows that the unit vector to cause the maximal output value under the ℓ_2-norm is $\boldsymbol{v}_{\max}$. Therefore, if an input perturbation $\boldsymbol{\xi}$ with a given magnitude ϵ wants to maximize the output value of a linear operator, it should be align with the vector $\boldsymbol{v}_{\max}$, i.e., $\boldsymbol{\xi} = \epsilon\boldsymbol{v}_{\max}$.

6 Experiment

In this section, we present the experimental setup and results to evaluate the effectiveness of our proposed method and our claims. First, we assess the method's effectiveness in the straightforward white-box scenario. Second, to evaluate transferability, we conduct experiments in a black-box setting. Third, we explore the robustness of the method against some image preprocessing methods. Finally, due to the special properties of our approach, we also perform semi-white-box experiments, i.e., only access a few layers in a model.

6.1 Experimental Setup

Initialization of IntriUAP. We consider the following initialization of our IntriUAP $\boldsymbol{\xi}$ describe in Algorithm 1:

1. **ImageNet Mean and Range prior:** Inputs were generated by leveraging the ImageNet mean prior and give it a dynamic range. This involved creating a three-channel RGB image with channel-wise values set to [0.485, 0.456, 0.406] and each elements are then given a dynamic range, aligning with the Range prior introduced in the seminal work of Mopuri et al. (Mopuri, Ganeshan, and Babu 2018).
2. **Gaussian Distribution:** We generated perturbations by sampling from a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. In our experiments, μ was set to 0.45, with σ values of 0.1.
3. **Uniform Distribution:** We generated perturbations by sampling from a uniform distribution $\mathcal{U}(a, b)$. In our experiments, a was set to 0.40, b was set to 0.60.

Victim Models. We consider five popular classification models provided by torchvision, including AlexNet (Krizhevsky, Sutskever, and Hinton 2012), GoogleNet (Szegedy et al. 2015), VGG-16 (Simonyan and Zisserman 2014), VGG-19 (Simonyan and Zisserman 2014) and ResNet152 (He et al. 2016). Note that all the considered models are L1LOS as we discussed in Section 4.

Evaluation metrics. To effectively assess the attack performance of our method, we report the fooling ratio on the 50,000-image validation set from ImageNet ILSVRC2012, a metric widely used in UAP tasks (Moosavi-Dezfooli et al. 2017; Poursaeed et al. 2018; Mopuri et al. 2018; Zhang et al. 2020b, 2021; Li et al. 2022; Liu et al. 2023). The fooling ratio is obtained by calculating the proportion of samples with labels changes when applying UAP. ℓ_∞-norm of the UAP remains within the bound of 10 (rescaled to [0,255]).

Method	AlexNet	GoogleNet	VGG16	VGG19	ResNet152
UAP	93.30	79.90	78.30	77.80	84.00
SV-UAP	– –	– –	52.00	60.00	– –
GAP	– –	82.70	83.70	80.10	– –
NAG	96.44	90.37	77.57	83.78	87.24
AT-UAP	97.01	90.82	97.51	97.56	91.52
SGA-UAP	97.43	92.12	98.36	97.69	94.04
FFF	80.92	56.44	47.10	43.62	– –
AAA	89.04	75.28	71.59	72.84	60.72
GD-UAP	87.02	71.44	63.08	64.67	37.30
PD-UAP	– –	67.12	53.09	48.95	53.51
DF-UAP	89.90	76.80	92.20	91.60	**79.9**
Cos-UAP	91.07	**87.57**	89.48	86.81	65.35
Ours	**94.03**	60.42	**93.40**	**92.28**	65.47

Table 1: Fooling ratio (%) of different UAP generation methods in the white-box attack scenario. The results are divided into universal attacks with access to the original ImageNet training data (upper) and data-free methods (lower).

Baselines. The proposed method is compared with the following UAP methods in the white-box and black-box attack scenario:

1. **Data-dependent UAP methods:** UAP (Moosavi-Dezfooli et al. 2017), SV-UAP (Khrulkov and Oseledets 2018), NAG (Mopuri et al. 2018), GAP (Poursaeed et al. 2018), DF-UAP (Zhang et al. 2020b), Cos-UAP (Zhang et al. 2021), AT-UAP (Li et al. 2022), SPGD-UAP (Liu et al. 2023).
2. **Data-free UAP methods:** FFF-UAP (Mopuri, Garg, and Babu 2017), AAA-UAP (Mopuri, Uppala, and Babu 2018), GD-UAP (Mopuri, Ganeshan, and Babu 2018), PD-UAP (Liu et al. 2019a), DF-UAP (Zhang et al. 2020b), Cos-UAP (Zhang et al. 2021).

6.2 Effectiveness of Intrinsic UAPs

Table 1 provides a summary of nearly all state-of-the-art data-free and data-dependent methods. We compare our Intri-UAP approach with (Mopuri, Garg, and Babu 2017; Mopuri, Uppala, and Babu 2018; Mopuri, Ganeshan, and Babu 2018; Liu et al. 2019a; Zhang et al. 2020b, 2021), as they don't rely on real training samples. In the experimental results table, we use " $--$ " to indicate items for which the other methods did not conduct experiments. As can be seen from Table 1, our method outperforms the competitive data-free methods in most cases. Even for the comparison of the data-driven methods (the upper half table), we still achieve comparable performance. Note, that the DF-UAP method utilizes some images from the COCO dataset.

A comparison of IntriUAP with these existing strictly data-free methods is presented in Table 2. Our results demonstrate that UAPs generated on AlexNet and VGG achieve a high fooling ratio, showing less dependency on the

Initial Data	Method	AlexNet	GoogleNet	VGG16	VGG19	ResNet152
Range Prior	PD-UAP	– –	– –	70.69	64.98	46.39
	GD-UAP	87.02	**71.44**	63.08	64.67	37.30
	Ours	**91.60**	60.42	**93.40**	**92.14**	**65.47**
Uniform	Cos-UAP	82.60	40.30	72.30	64.40	47.20
	Ours	**92.58**	**45.56**	**80.40**	**81.56**	**48.00**
Gaussian	AT-UAP	56.80	24.27	30.56	27.75	19.40
	Cos-UAP	89.50	46.70	76.10	75.40	49.90
	Ours	**92.61**	**47.80**	**86.03**	**82.00**	**50.50**

Table 2: Results for different initialization methods of our IntriUAP in comparison with different data-free methods.

Method	Samples	AlexNet	GoogleNet	VGG16	VGG19	ResNet152
Singular Fool*	49	52.00	60.00	44.00	– –	– –
GD-UAP*	49	72.80	67.60	56.40	– –	– –
UAP*	500	57.33	16.61	25.29	25.04	19.11
GAP*	500	86.89	57.07	70.40	65.89	47.58
SPGD*	500	92.35	41.68	81.70	75.74	23.44
SGA*	500	93.03	68.33	89.83	88.70	52.12
Cos-UAP	0	91.07	**87.57**	89.48	86.81	65.35
Ours	**0**	**94.03**	60.43	**93.40**	**92.28**	**65.47**

Table 3: Fooling ratio (%) for UAPs crafted with limited real samples. * indicates the data-dependent method.

sample data compared to other models. Additionally, ResNet also delivers commendable results.

In Table 3, we compared our approach with the current state-of-the-art data-dependent methods, SPGD and SGA(Liu et al. 2023). Remarkably, even when they utilize 500 real samples, our method still holds an advantage. For example, we achieve that fooling rate of 93.40% on attacking VGG16 while the best competitor is 89.83%

We also provide the visualization of our IntriUAP in Figure 1. As can be seen, the IntriUAPs have not significantly impacted the quality of the cover images.

6.3 Transferability Performance of Intrinsic UAPs

In this section, the transferability results are presented in Table 4. Each row in the table shows the fooling rates for perturbations learned on a specific target model when attacking various other models (columns). These ratios are derived using the IntriUAP objective with range prior. Diagonal values correspond to white-box attack scenarios, while off-diagonal values represent black-box attacks. Our observations indicate that the transferability of our UAPs is effective, particularly when conducting black-box attacks using UAPs generated with L1LOS.

In Table 7, we also present the results of our experiments in black-box scenarios, compared with both data-free and data-dependent approaches, for evaluation of the transferability of those UAPS. Notably, even when compared to data-dependent methods, our approach demonstrates superior performance in black-box scenarios, even surpassing the current data-dependent method SGA in black-box attacks on AlexNet. Furthermore, we found that VGG and AlexNet often exhibit good transferability among themselves. UAPs generated by Intrinsic Attack trained on AlexNet demonstrate the strongest transferability.

Model	AlexNet	GoogleNet	VGG16	VGG19	ResNet152
AlexNet	**94.07**	56.79	71.14	65.74	41.11
GoogleNet	41.44	**60.20**	48.40	45.19	28.17
VGG16	43.39	36.80	**93.40**	82.18	35.67
VGG19	42.03	39.46	85.69	**92.28**	39.96
ResNet152	42.41	38.45	53.05	51.56	**65.47**

Table 4: Fooling ratio (%) of UAPs under Black-box settings.

Model	AlexNet		GoogLeNet		VGG-16		VGG-19		ResNet-152	
Method	GD	Ours	GD	Ours	GD	Ours	GD	Ours	GD	Ours
No Defense	87.10	**94.07**	**71.40**	60.42	63.10	**93.40**	64.70	**92.28**	37.30	**65.47**
75% JPEG	72.10	**90.40**	41.80	**44.51**	49.12	**61.19**	**64.70**	55.47	37.30	**38.30**
50% JPEG	79.20	**85.88**	37.85	**42.54**	39.12	**49.40**	54.07	**54.93**	25.84	**33.23**
Gaussian	65.54	**72.17**	35.62	**46.31**	37.77	**51.48**	35.54	**52.80**	24.56	**32.06**
Median	72.13	**83.98**	46.98	**54.47**	44.68	**78.13**	46.67	**77.23**	28.82	**41.88**
Bilateral	34.80	**67.20**	21.50	**26.74**	35.80	**70.52**	28.20	**66.78**	25.40	**35.07**

Table 5: Fooling rate (%) of UAPs under different defenses.

Model	25% white-box	50% white-box	75% white-box	100% white-box
AlexNet	81.35	90.01	92.77	**94.03**
VGG16	57.95	67.10	91.49	**93.40**
VGG19	73.51	78.62	91.31	**92.28**
GoogleNet	40.65	56.20	59.81	**60.43**

Table 6: Fooling ratio (%) of UAPs for semi-white-box model. Generated with range prior as background. The rows indicate the white-box level of the model, and the columns indicate the target model.

6.4 Robustness of Intrinsic UAPs

Following GD-UAP's approach, we evaluated UAPs against defensive techniques, focusing on input transformations such as JPEG compression, median smoothing, bilateral smoothing, and Gaussian blurring. Results in Table 5 show our method's robustness significantly surpasses GD-UAP.

The findings closely align with the conclusions reached by GD-UAP (Mopuri, Ganeshan, and Babu 2018). While input transformations do reduce the fooling rates of UAPs, they also degrade image quality and significantly lower the model's Top-1 accuracy. This reduction in accuracy is often unacceptable, highlighting the limited adaptability of these defense mechanisms.

6.5 Intrinsic UAPs for Semi-white-box models

Due to the special property of our methond, specifically that our method does not require backpropagation through the entire network, we conducted experiments to demonstrate the feasibility of attacking semi-white-box models. We performed experiments by selecting subsets of linear layers and categorized the degree of white-box access into four scenarios: 25% white-box, 50% white-box, 75% white-box, and

Jigsaw Brain coral Brain coral Brain coral Brain coral Taxi

Figure 1: Examples of perturbed images generated by VGG19 with IntriUAP and their corresponding labels. The perturbaitons were constrainted with ℓ_∞-norm $= 10$. Using range prior.

Target Model	Technique	AlexNet	VGG16	VGG19	ResNet50	ResNet152	GoogleNet
VGG16	UAP*	33.35	76.73	64.14	– –	29.39	33.51
	GAP*	22.33	82.21	76.30	– –	29.46	42.50
	AAA	– –	71.59	65.64	– –	45.33	**60.74**
	GD-UAP	– –	45.47	38.20	27.70	23.80	34.13
	GD-UAP+P	– –	51.63	44.07	32.23	28.78	36.79
	UA	– –	48.46	41.97	29.09	24.90	35.52
	PD-UA	– –	53.09	49.30	33.61	30.31	39.05
	Cos-UAP (Regular UAP)	– –	89.48	76.84	44.11	38.37	48.97
	Cos-UAP (Adaptive UAP)	– –	86.16	77.88	49.30	**44.27**	56.96
	Ours (Range prior)	**43.39**	**93.40**	**82.18**	**50.28**	35.67	35.90
VGG19	UAP*	34.45	65.46	77.79	33.24	28.49	35.21
	GAP*	52.71	75.08	79.11	39.43	35.21	**49.11**
	FFF	42.03	38.19	43.62	28.27	26.34	30.71
	GD-UAP	– –	**55.70**	64.70	– –	35.80	53.50
	UAP-DL	– –	47.50	52.00	– –	30.40	33.70
	DF-UAP (COCO)	– –	83.40	91.70	– –	35.40	39.80
	Ours (Range prior)	42.80	**85.60**	**92.28**	**55.87**	**38.93**	38.65
AlexNet	UAP*	86.53	37.67	35.47	23.45	20.99	27.82
	GAP*	89.06	52.02	48.60	42.54	38.70	33.05
	SGA-logit*	95.23	57.62	53.86	– –	30.39	37.68
	SPGD-logit*	96.60	63.82	59.52	– –	34.95	46.18
	SPGD-cls*	95.97	58.59	54.03	– –	30.52	42.78
	SGA-cls*	**97.23**	66.46	60.60	– –	35.29	48.97
	GD-UAP (Range Prior)	87.02	50.46	49.92	– –	38.58	49.40
	DF-UAP (COCO)	90.45	60.43	58.66	– –	**47.02**	54.77
	Ours (Range prior)	94.07	**71.14**	**65.74**	**50.07**	41.11	**56.79**

Table 7: The fooling ratio (%) on six models in the black-box setting by regular UAP attack methods. The UAPs are crafted on AlexNet, VGG16, and VGG19 respectively. * indicates the data-dependent method.

100% white-box (for example, 50% white-box indicates access to only the first 50% of the model's parameters). We observed that the fooling ratio remained high across these scenarios. The results indicate that partial access to the model's parameters still allows for the creation of effective UAPs.

Table 6 presents the experimental results using VGG16, VGG19, GoogLeNet, and AlexNet. The results show that even with partial white-box access, the success rate remains high. For instance, AlexNet achieves 90% fooling ratio at 50% white-box access and only improves slightly to 94.03% at 100% access. Similarly, VGG16 reaches 91% at 75% white-box access and 93% at 100%, with minimal gains between 75% and 100%. This indicates that semi-white-box scenarios (50% and 75% access) can achieve nearly the same high success rate as full white-box access.

7 Conclusion

In this paper, we explore the intrinsic vulnerability of the popular deep models called L1LOS, which consists of linear layers and 1-Lipschitz nonlinear layers. We point out that the vulnerability of L1LOS is controlled by linear operators. This deepens the theoretical foundation for the existence of UAPs and introduces the method of the Intrinsic UAP. We demonstrate that the vulnerability of L1LOS can be exploited without requiring samples or complete white-box access, achieving high attack success rates and posing the security risks. For L1LOS, which align with our theoretical framework, our fooling ratio achieves SOTA baselines. Compared with the UAP methods, our method not only has the weak assumption of data-free but also works under the weak assumption of access a few layers of the victim models, making our method more practical in real word tasks.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (Grant No. 62202009), the Macau Science and Technology Development Fund (Grant Nos. 0040/2023/ITP1 and 0004/2023/RIB1), and the Basic and Applied Basic Research Foundation of Guangdong Province (Grant No. 2024A1515011755).

References

Araujo, A.; Negrevergne, B.; Chevaleyre, Y.; and Atif, J. 2021. On lipschitz regularization of convolutional layers using toeplitz matrix theory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 6661–6669.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Biderman, S.; Schoelkopf, H.; Anthony, Q. G.; Bradley, H.; O'Brien, K.; Hallahan, E.; Khan, M. A.; Purohit, S.; Prashanth, U. S.; Raff, E.; et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, 2397–2430. PMLR.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fawzi, A.; Moosavi-Dezfooli, S.-M.; and Frossard, P. 2016. Robustness of classifiers: from adversarial to random noise. arXiv:1608.08967.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Khrulkov, V.; and Oseledets, I. 2018. Art of singular vectors and universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8562–8570.

Kim, H.; Papamakarios, G.; and Mnih, A. 2021. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, 5562–5571. PMLR.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Li, M.; Yang, Y.; Wei, K.; Yang, X.; and Huang, H. 2022. Learning universal adversarial perturbation by adversarial example. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 1350–1358.

Liu, H.; Ji, R.; Li, J.; Zhang, B.; Gao, Y.; Wu, Y.; and Huang, F. 2019a. Universal adversarial perturbation via prior driven uncertainty approximation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2941–2949.

Liu, H.; Ji, R.; Li, J.; Zhang, B.; Gao, Y.; Wu, Y.; and Huang, F. 2019b. Universal Adversarial Perturbation via Prior Driven Uncertainty Approximation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2941–2949.

Liu, X.; Zhong, Y.; Zhang, Y.; Qin, L.; and Deng, W. 2023. Enhancing Generalization of Universal Adversarial Perturbation through Gradient Aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4435–4444.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2019. Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1765–1773.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2574–2582.

Mopuri, K. R.; Ganeshan, A.; and Babu, R. V. 2018. Generalizable data-free objective for crafting universal adversarial perturbations. *IEEE transactions on pattern analysis and machine intelligence*, 41(10): 2452–2465.

Mopuri, K. R.; Garg, U.; and Babu, R. V. 2017. Fast feature fool: A data independent approach to universal adversarial perturbations. *arXiv preprint arXiv:1707.05572*.

Mopuri, K. R.; Ojha, U.; Garg, U.; and Babu, R. V. 2018. Nag: Network for adversary generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 742–751.

Mopuri, K. R.; Uppala, P. K.; and Babu, R. V. 2018. Ask, acquire, and attack: Data-free uap generation using class impressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 19–34.

Poursaeed, O.; Katsman, I.; Gao, B.; and Belongie, S. 2018. Generative adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4422–4431.

Praggastis, B.; Brown, D.; Marrero, C. O.; Purvine, E.; Shapiro, M.; and Wang, B. 2022. The SVD of Convolutional Weights: A CNN Interpretability Framework. *arXiv preprint arXiv:2208.06894*.

Sedghi, H.; Gupta, V.; and Long, P. M. 2018. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Su, Y.; Zhang, J.; Xu, T.; Zhang, T.; Zhang, W.; and Yu, N. 2024. Model X-ray: Detect Backdoored Models via Decision Boundary. *arXiv preprint arXiv:2402.17465*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition, 1–9.

Tao, Y.; Liu, D.; Zhou, P.; Xie, Y.; Du, W.; and Hu, W. 2023. 3dhacker: Spectrum-based decision boundary generation for hard-label 3d point cloud attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14340–14350.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Zhang, C.; Benz, P.; Imtiaz, T.; and Kweon, I.-S. 2020a. Cd-uap: Class discriminative universal adversarial perturbation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 6754–6761.

Zhang, C.; Benz, P.; Imtiaz, T.; and Kweon, I. S. 2020b. Understanding adversarial examples from the mutual influence of images and perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14521–14530.

Zhang, C.; Benz, P.; Karjauv, A.; and Kweon, I. S. 2021. Data-free universal adversarial perturbation and black-box attack. In *Proceedings of the IEEE/CVF international conference on computer vision*, 7868–7877.

Zhao, W.; Li, X.; Zhao, S.; Xu, J.; Liu, Y.; and Lu, Z. 2024. Detecting Adversarial Spectrum Attacks via Distance to Decision Boundary Statistics. *arXiv preprint arXiv:2402.08986*.