On the Adversarial Transferability of Generalized "Skip Connections"

Yisen Wang, Yichuan Mo, Dongxian Wu, Mingjie Li, Xingjun Ma, *Member, IEEE*, and Zhouchen Lin, *Fellow, IEEE*

***Abstract*—Skip connection is an essential ingredient for modern deep models to be deeper and more powerful. Despite their huge success in normal scenarios (state-of-the-art classification performance on natural examples), we investigate and identify an interesting property of skip connections under adversarial scenarios, namely, the use of skip connections allows easier generation of highly transferable adversarial examples. Specifically, in ResNet-like models (with skip connections), we find that biasing backpropagation to favor gradients from skip connections–while suppressing those from residual modules via a decay factor–allows one to craft adversarial examples with high transferability. Based on this insight, we propose the *Skip Gradient Method* (SGM). Although starting from ResNet-like models in vision domains, we further extend SGM to more advanced architectures, including Vision Transformers (ViTs), models with varying-length paths, and other domains such as natural language processing. We conduct comprehensive transfer-based attacks against diverse model families, including ResNets, Transformers, Inceptions, Neural Architecture Search-based models, and Large Language Models (LLMs). The results demonstrate that employing SGM can greatly improve the transferability of crafted attacks in almost all cases. Furthermore, we demonstrate that SGM can still be effective under more challenging settings such as ensemble-based attacks, targeted attacks, and against defense equipped models. At last, we provide theoretical explanations and empirical insights on how SGM works. Our findings not only motivate new adversarial research into the architectural characteristics of models but also open up further challenges for secure model architecture design.**

***Index Terms*—Skip connections, adversarial attacks, transferability, model architectures.**

Received 11 October 2024; revised 26 June 2025; accepted 11 February 2026. Date of publication 18 February 2026; date of current version 5 June 2026. The work of Yisen Wang was supported in part by the National Natural Science Foundation of China under Grant 92370129 and Grant 62376010, in part by Beijing Major Science and Technology Project under Contract Z251100008425006, in part by Beijing Nova Program under Grant 20230484344 and Grant 20240484642, and in part by the State Key Laboratory of General Artificial Intelligence. The work of Zhouchen Lin was supported in part by Beijing Major Science and Technology Project under Contract Z251100008425006 and in part by NSF China under Grant 62276004. Recommended for acceptance by V. B. Radhakrishnan. *(Corresponding author: Zhouchen Lin.)*

Yisen Wang, Yichuan Mo, Mingjie Li, and Zhouchen Lin are with the State Key Lab of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing 100080, China (e-mail: yisen.wang@pku.edu.cn; mo666666@stu.pku.edu.cn; lmjat0111@pku.edu.cn; zlin@pku.edu.cn).

Dongxian Wu is with Tsinghua University, Beijing 100190, China (e-mail: wudx16@gmail.com).

Xingjun Ma is with Fudan University, Shanghai 200438, China (e-mail: xingjunma@fudan.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2026.3666165, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2026.3666165

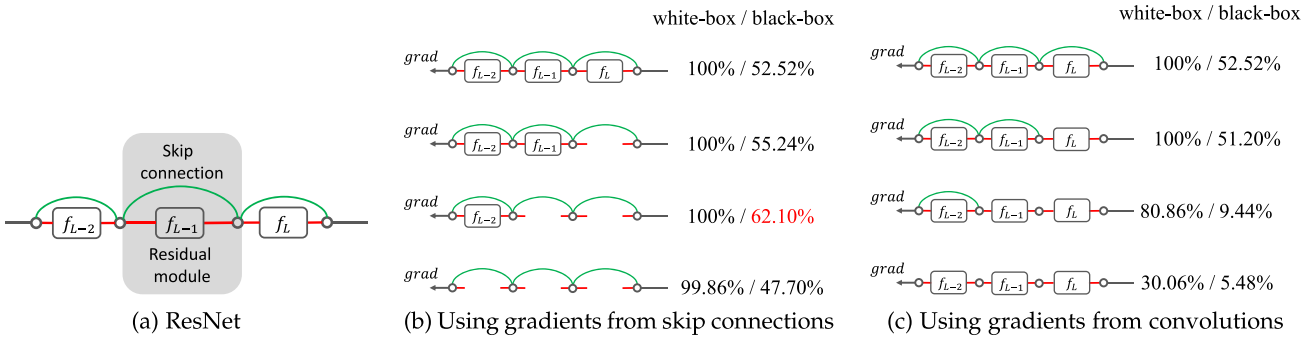
I. Introduction

In DEEP neural networks (DNNs), skip connection builds one kind of short-cut from a shallow layer to a deep layer by connecting the input of a building block including convolution layers or self-attention layers (also known as the residual module) directly to its output. While different layers of neural networks learn different "levels" of features, skip connections can help preserve low-level features and avoid performance degradation when adding more layers. This has been shown to be crucial for building very deep and powerful DNNs such as ResNet [1], WideResNet [2], DenseNet [3], and Vision Transformer [4]. However, despite their superior performance, DNNs have been found to be extremely vulnerable to adversarial examples (or attacks), which are input examples slightly perturbed by small noise to fool the network into making wrong predictions [5], [6]. Adversarial examples are imperceptible to human observers and transferable across different models [7].

Generally, adversarial examples can be crafted following either a white-box setting (the adversary has full access to the target model) or a black-box setting (the adversary has no information of the target model). White-box methods such as Fast Gradient Sign Method (FGSM) [6], Basic Iterative Method (BIM) [8], Projected Gradient Decent (PGD) [9], and Carlini and Wagner (CW) [10] often suffer from low transferability in a black-box setting, thus posing only limited threats to models which are usually kept secret in practice while only APIs are accessible [11], [12]. Several techniques have been proposed to improve the transferability of black-box attacks based on a surrogate model [13], such as momentum boosting [11], diverse input [12], and adversarial tuning [14]. Although these techniques are effective, they (as well as white-box methods) all treat the entire network (either the target model or the surrogate model) as a single component while ignoring its inner architectural characteristics. Therefore, a natural question is raised here:

> *Can the model architecture itself expose more transferability of adversarial attacks?*

In this paper, we identify one such property of the skip connections used by many state-of-the-art DNNs. We first conduct a toy experiment on the ImageNet validation dataset [15] to investigate how skip connections affect the adversarial strength of attacks. Adversarial examples are generated from ResNet-18 by BIM attack and then transfer to attack target model VGG19. For the last 3 skip connections and residual modules



(a) ResNet (b) Using gradients from skip connections (c) Using gradients from convolutions

Fig. 1. Illustration of the last 3 skip connections (green lines) and residual modules (black boxes) of a ImageNet-trained ResNet-18. The success rate ("white-box/black-box") of adversarial attacks crafted using gradients flowing through either a skip connection (b) or a convolution module (c) at each junction point (circle). The attacks are crafted by BIM on 5000 ImageNet validation images under maximum L_∞ perturbation $\epsilon = 16$ (pixel values are in [0,255]). The black-box success rate is tested against a VGG19 target model.

of ResNet-18, we illustrate the success rate of attacks crafted using partial gradients after removing the gradient flow through some modules in Fig. 1. Comparing Fig. 1(b) and (c), we find that, if we remove more gradients that go through the residual modules while keeping the gradients through skip connections, the success rate of the black-box attack increases significantly to a certain degree (the top 3 rows in Fig. 1(b)). For example, the black-box success rate is even improved from 52.52% to 62.10% when skipping the last two residual modules (the third row in Fig. 1(b)). This implies that gradients from the skip connections carry more transferable information which might be exploited by the adversary. Surely, if all the gradients are through the skip connections rather than the residual modules, the attack success rate will decrease as the available information about the input is very limited for the attack.

Motivated by the above observations, in this paper, we propose the *Skip Gradient Method (SGM)* to generate adversarial examples using tuneable gradients more from the skip connections rather than the residual modules. In particular, SGM utilizes a decay factor to reduce gradients from the residual modules. We find that this simple adjustment on the gradient flow can serve as a catalyst to improve the transferability of current state-of-the-art attacks. In summary, our main contributions are:

- We identify one surprising property of skip connections in ResNet-like models, i.e., they allow an easy generation of highly transferable adversarial examples.
- We propose the Skip Gradient Method (SGM) to craft adversarial examples using tuneable gradients more from the skip connections. Using a single decay factor on gradients, SGM is an appealingly simple and generic technique that can be used by any existing gradient-based attack method.
- We provide comprehensive transfer attack experiments, and find SGM improves the state-of-the-art transferability benchmarks by a large margin.

The main results of convolutional neural networks with skip connections (ResNet-like models) were published originally in ICLR as a spotlight paper [16]. In this longer article version, although SGM is motivated from ResNet-like models, we first extend it not only to the currently prevailing transformer architectures [4], [17] in Section III-B, but also to almost all networks as long as they have varying-length paths even without skip connections (e.g., Inception [18], [19] or models from Neural Architecture Search [20], [21]) in Section III-C. Comprehensive experiments on 35 state-of-the-art attacks in Section V demonstrate its effectiveness on various architectures. Furthermore, we provide some theoretical analysis in Section IV to explore how SGM works and a series of experiments in Section VI to illustrate SGM can even improve the transferability in more complex scenarios including the ensembles of models, targeted attacks, and defense equipped models. Lastly, we provide adaptivity and interpretability analysis on SGM in Section VII-B and reveal that SGM can further bring benefits to other domains such as attacking Large Language Models (LLMs) in Section VIII.

II. Related Work

Existing adversarial attacks can be categorized into two groups: 1) white-box attacks and 2) black-box attacks. In the white-box setting, the adversary has full access to the parameters of the target model, while in the black-box setting, the target model is kept secret from the adversary.

A. White-Box Attacks

Given a clean example $\boldsymbol{x}$ with class label y and a target DNN model f, the goal of an adversary is to find an adversarial example $\boldsymbol{x}_{adv}$ that fools the network into making an incorrect prediction (e.g. $f(\boldsymbol{x}_{adv}) \neq y$), while still remaining in the ϵ-ball centered at $\boldsymbol{x}$ (e.g. $\|\boldsymbol{x}_{adv} - \boldsymbol{x}\|_\infty \leq \epsilon$). A wide range of attacking methods have been proposed for the crafting of adversarial examples. Here, we only mention a selection.

Fast Gradient Sign Method (FGSM) [6]: FGSM perturbs clean example $\boldsymbol{x}$ for one step by the amount of ϵ along the gradient direction:

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot \text{sign}\left(\nabla_{\boldsymbol{x}} \ell\left(f\left(\boldsymbol{x}\right), y\right)\right). \tag{1}$$

Projected Gradient Descent (PGD) [9]: PGD is an iterative version of FGSM, which perturbs clean example $\boldsymbol{x}$ for T steps with a smaller step size. After each step of perturbation, PGD projects the adversarial example back onto the ϵ-ball of $\boldsymbol{x}$, if it

goes beyond the ϵ-ball:

$$\boldsymbol{x}_{adv}^{t+1} = \Pi_{\epsilon}\left(\boldsymbol{x}_{adv}^{t} + \alpha \cdot \text{sign}\left(\nabla_{\boldsymbol{x}} \ell\left(f\left(\boldsymbol{x}_{adv}^{t}\right), y\right)\right)\right), \quad (2)$$

where $\Pi_{\epsilon}(\cdot)$ is the projection operation.

There are also other types of white-box attacks including sparsity-based methods such as Jacobian-based Saliency Map Attack (JSMA) [22], one-pixel attack [23], and optimization-based methods such as Carlini and Wagner (CW) [10]. Although these methods are effective in the white-box setting, they often suffer from low transferability in the black-box setting [11].

B. Black-Box Attacks

Black-box attacks can be generated by transferability-based method that transfers from attacking a surrogate model or query-based method that directly generates adversarial examples on the target model via lots of queries to the system. Query-based method estimates the gradient of the target model via a large number of queries, which is then used to generate adversarial examples such as Finite Differences (FD) [24] or Natural Evolution Strategies (NES) [25]. These methods all require a large number of queries to the target model, which not only reduces the efficiency but also potentially exposes the attack.

Alternatively, black-box adversarial examples can be crafted on a surrogate model then applied to attack the target model. Although the white-box methods can be directly applied on the surrogate model, they are far less effective in the black-box setting [11], [12]. Several transfer techniques have been proposed to improve the transferability of black-box attacks and they can be mainly classified into four different categories based on their design principles.

Gradient-related Attacks: Previous works show that the decision boundaries vary across different architectures [7] and falling into the local minima will largely impair the transferability to target models. The gradient-related attacks alleviate it by developing advanced updating strategies such as momentum acceleration [11], [26], [27], neighborhood correlation [28], [29], [30], [31], norm penalization [14], [32] and adaptive step size adjustment [33], [34].

Augumentation-related Attacks: Similar to the generalization of models, data augmentation can also improve the generalization of attacks by undermining the intrinsic features. It ensures the invariance of the attack effect when transforming the images with augmentations and thus maintains its threats to diverse architectures. Earliest work develop simple augmentations such as random resizing or padding [12], [35], frequency-based noises [36], random masking [37] or rotating the split patches [38]. More advanced methods are further exploited such as diverse augmentations for each image block [39], automatic augmentation selection to each image [40], or applying a stylized network for the customized enhancement [41].

Feature-related Attack: Since the aim of the attacks is to induce misclassification, an intuitive approach is to craft adversarial perturbations with the cross-entropy loss. However, previous work [42] reveals that the intermediate features might be a better choice since they share high similarity across models. Motivated by this finding, feature-based attack attempts to craft the adversarial example by increasing its perturbation on a pre-specified layer of the model. Earliest work [43] shows that directly maximizing the differences of feature maps will only obtain unsatisfied performances because only the important features contribute to the classification. Therefore, FIA [44] and NAA [45] introduce the backward gradients and the decomposed integral respectively as the soft mask to filter out unrelated features. Experience from other categories, such as updating the weighted matrix with momentum [46], averaging it on multiple inputs [47], ensembling features across multiple layers [48] and blending benign and adversarial features [49] is illustrated to be successful in further improving the performances of those attacks.

Parameter-related Attack: Noticing that although exploring powerful attack algorithms is crucial, the influence of model parameters on black-box transferability is another intriguing perspective to study. Particularly, in [50], they observe that a little adversarial robustness improves the transferability in a novel margin ($>10\%$). Other techniques are also studied, such as training the surrogate model with knowledge distillation [51], [52], or adversary-centric contrastive learning [53]. All of them can help attackers improve the black-box transferability with a novel margin while maintaining the attack algorithm unchanged.

Although the above transferability techniques are effective, they either 1) treat the network as a single component or 2) utilize intermediate layer outputs without considering the architectural structure of the model. Closely related to our work, ViT-specific attacks [54], [55], [56], [57] aim to enhance transferability by leveraging architectural components unique to Transformers. However, these methods are limited to ViT architectures and are orthogonal to our approach. MUP [58], on the other hand, improves attack performance by masking unimportant components during backpropagation. In contrast, our work explicitly exploits a fundamental architectural perspective to enhance adversarial transferability in a principled and generalizable manner.

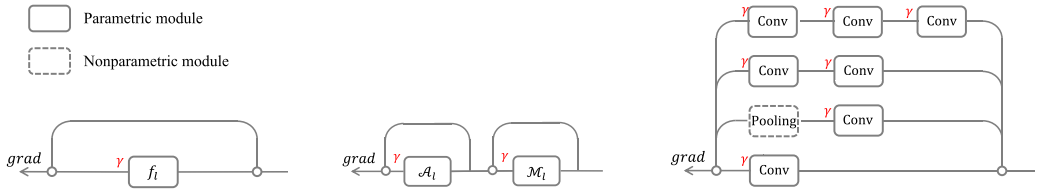
III. Proposed Skip Gradient Method (SGM)

In this section, we first introduce the gradient decomposition of skip connection and residual module. Following that, we propose our Skip Gradient Method (SGM) for ResNet-like architectures, then extend it to transformer architectures and other modern models with varying-length paths.

A. SGM for ResNet-Like Architectures

In ResNet-like neural networks, a skip connection uses identity mapping to bypass residual layers, allowing data to flow from a shallow layer directly to subsequent deep layers. Thus, we can decompose the network into a collection of paths of different lengths [59]. We denote a skip connection together with its associated residual module as a building block (residual block) of a network. Considering three successive building blocks (eg. $\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + f_{i+1}(\boldsymbol{z}_i)$) in a residual network from input $\boldsymbol{z}_0$ to output $\boldsymbol{z}_3$, the output $\boldsymbol{z}_3$ can be expanded as:

$$\begin{aligned}\boldsymbol{z}_3 &= \boldsymbol{z}_2 + f_3(\boldsymbol{z}_2) = [\boldsymbol{z}_1 + f_2(\boldsymbol{z}_1)] + f_3\left(\boldsymbol{z}_1 + f_2\left(\boldsymbol{z}_1\right)\right) \\ &= [\boldsymbol{z}_0 + f_1\left(\boldsymbol{z}_0\right) + f_2\left(\boldsymbol{z}_0 + f_1\left(\boldsymbol{z}_0\right)\right)] \\ &\quad + f_3\left(\left(\boldsymbol{z}_0 + f_1\left(\boldsymbol{z}_0\right)\right) + f_2\left(\boldsymbol{z}_0 + f_1\left(\boldsymbol{z}_0\right)\right)\right). \quad (3)\end{aligned}$$



(a) The gradient through a residual block in ResNet. (b) The gradient through a transformer block in ViT. (c) The gradient through a basic block in Inception.

Fig. 2. Diagrams for performing SGM during the backpropagation on various prevailing architectures.

According to the chain rule in calculus, the gradient of a loss function ℓ with respect to input $\boldsymbol{z}_0$ can then be decomposed as,

$$\begin{aligned}\frac{\partial \ell}{\partial \boldsymbol{z}_0} &= \frac{\partial \ell}{\partial \boldsymbol{z}_3}\frac{\partial \boldsymbol{z}_3}{\partial \boldsymbol{z}_2}\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}_0} \\ &= \frac{\partial \ell}{\partial \boldsymbol{z}_3}\left(1+\frac{\partial f_3}{\partial \boldsymbol{z}_2}\right)\left(1+\frac{\partial f_2}{\partial \boldsymbol{z}_1}\right)\left(1+\frac{\partial f_1}{\partial \boldsymbol{z}_0}\right). \end{aligned} \tag{4}$$

Extending this toy example to a network with L residual blocks, the gradient can be decomposed for all residual blocks as,

$$\frac{\partial \ell}{\partial \boldsymbol{x}} = \frac{\partial \ell}{\partial \boldsymbol{z}_L}\prod_{i=0}^{L-1}\left(\frac{\partial f_{i+1}}{\partial \boldsymbol{z}_i}+1\right)\frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{x}}, \tag{5}$$

where $\boldsymbol{z}_0 = \boldsymbol{x}$ is usually the input to the first residual blocks.

To use more gradient from the skip connections, here, we introduce a decay parameter into the decomposed gradient to reduce the gradient from the residual modules, as shown in Fig. 2(a). Following the decomposition in (5), the "skipped" gradient is,

$$\nabla_{\boldsymbol{x}}\ell = \frac{\partial \ell}{\partial \boldsymbol{z}_L}\prod_{i=0}^{L-1}\left(\gamma\frac{\partial f_{i+1}}{\partial \boldsymbol{z}_i}+1\right)\frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{x}}, \tag{6}$$

where $\gamma \in (0,1]$ is the decay parameter. Accordingly, given a clean example $\boldsymbol{x}$ and a DNN model f, an adversarial example can be crafted iteratively by,

$$\boldsymbol{x}_{adv}^{t+1} = \Pi_\epsilon\left(\boldsymbol{x}_{adv}^{t}+\alpha\cdot\operatorname{sign}\left(\frac{\partial \ell}{\partial \boldsymbol{z}_L}\prod_{i=0}^{L-1}\left(\gamma\frac{\partial f_{i+1}}{\partial \boldsymbol{z}_i}+1\right)\frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{x}}\right)\right). \tag{7}$$

During the backpropagation process, SGM simply multiplies the decay parameter to the gradient whenever it passes a residual module. Therefore, SGM does not require any computation overhead, and works efficiently even on densely connected networks such as DenseNets.

B. Extending SGM to Transformers

Recently, Vision Transformers (ViTs) [4] have achieved competitive performance in vision tasks through the use of self-attention mechanisms. Since the basic building block of ViTs is composed of a self-attention layer, an MLP layer, and some skip connections, extending SGM to ViTs is straightforward. Illustrative experiments demonstrating this extension are provided in Fig. 6 in Appendix A, available online.

Recall that the gradient flow through a basic block of ResNet consists of two parts, i.e., one through the residual module f_l, and the other through the skip connection. Using SGM, we decay the gradient flow through the residual module by multiplying γ while keeping the other gradient flow unchanged as illustrated in Fig. 2(a). Similarly, for the basic block of ViTs, the input $\boldsymbol{z}_0$ is first processed by a multi-head attention module $\mathcal{A}_1$ and an identity function (skip connection) in parallel. Its output $\boldsymbol{z}'_1$ is further processed by a multi-layer perception $\mathcal{M}_1$ and an identity function in parallel. Therefore, the output of the transformer block consists of 4 terms:

$$\begin{aligned}\boldsymbol{z}_1 &= \mathcal{M}_1(\boldsymbol{z}'_1)+\boldsymbol{z}'_1 \\ &= \mathcal{M}_1\left(\mathcal{A}_1(\boldsymbol{z}_0)+\boldsymbol{z}_0\right)+\mathcal{A}_1(\boldsymbol{z}_0)+\boldsymbol{z}_0 \end{aligned} \tag{8}$$

According to the chain rule, the gradient of the loss function ℓ with respect to the input $\boldsymbol{z}_0$ can be formulated as,

$$\frac{\partial \ell}{\partial \boldsymbol{z}_0} = \frac{\partial \ell}{\partial \boldsymbol{z}_1}\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}'_1}\frac{\partial \boldsymbol{z}'_1}{\partial \boldsymbol{z}_0} = \frac{\partial \ell}{\partial \boldsymbol{z}_1}\left(\frac{\partial \mathcal{M}_1}{\partial \boldsymbol{z}'_1}+1\right)\left(\frac{\partial \mathcal{A}_1}{\partial \boldsymbol{z}_0}+1\right). \tag{9}$$

It is almost same as (5), if we set $L=2$, $f_1=\mathcal{A}_1$, and $f_2=\mathcal{M}_1$. Further, we can easily extend SGM to a L-layer vision transformer illustrated in Fig. 2(b):

$$\begin{aligned}\frac{\partial \ell}{\partial \boldsymbol{x}} &= \frac{\partial \ell}{\partial \boldsymbol{z}_L}\prod_{l=0}^{L-1}\left(\gamma\frac{\partial \mathcal{M}_{l+1}}{\partial \boldsymbol{z}'_{l+1}}+1\right)\left(\gamma\frac{\partial \mathcal{A}_{l+1}}{\partial \boldsymbol{z}_l}+1\right)\frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{x}} \\ &= \frac{\partial \ell}{\partial \boldsymbol{z}_L}\prod_{l=0}^{L-1}\left(\gamma^2\frac{\partial \mathcal{M}_{l+1}}{\partial \boldsymbol{z}'_{l+1}}\frac{\partial \mathcal{A}_{l+1}}{\partial \boldsymbol{z}_l}\right. \\ &\qquad \left.+\gamma\frac{\partial \mathcal{M}_{l+1}}{\partial \boldsymbol{z}'_{l+1}}+\gamma\frac{\partial \mathcal{A}_{l+1}}{\partial \boldsymbol{z}_l}+1\right)\frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{x}}, \end{aligned} \tag{10}$$

where $\boldsymbol{x}$ is the raw image before patch embedding, $\boldsymbol{z}_0$ is the input to the 1-st transformer block after patch embedding, and γ is a factor for decaying the gradient.

Note that we have expanded the gradient into 4 parts in (10). The gradient flow through both the attention module $\mathcal{A}_l$ and the perception module $\mathcal{M}_l$ is decayed by γ^2, the gradient flow through only one module ($\mathcal{A}_l$ or $\mathcal{M}_l$) is decayed by γ, and the gradient through only skipping connections is unchanged. In other words, the more modules a gradient flow goes through,

the more we decay it. This motivates us to further extend SGM to models with varying-length paths.

C. Extending SGM to Architectures With Varying-Length Paths

The above model architectures all include skip connections whose role can be regarded as providing different lengths of paths from the input to the output. However, unfortunately, not all modern models have skip connections. Therefore, in this part, we attempt to extend the proposed SGM to architectures without skip connections but with varying-length paths.

Taking Inception-V3 as an example, as shown in Fig. 2(c), there are 4 different parallel processing paths in a basic block, e.g., a single convolutional layer $\mathcal{P}_{1,1}$, a combination of pooling and a convolutional layer $\mathcal{P}_{1,2}$, and two or three successive convolutional layers: $\mathcal{P}_{1,3}$ and $\mathcal{P}_{1,4}$. For the first layer, if we denote the input with $\boldsymbol{z}_0$, the output of this basic block $\boldsymbol{z}_1$ consists of 4 parts:

$$\boldsymbol{z}_1 = \mathcal{P}_{1,1}(\boldsymbol{z}_0) + \mathcal{P}_{1,2}(\boldsymbol{z}_0) + \mathcal{P}_{1,3}(\boldsymbol{z}_0) + \mathcal{P}_{1,4}(\boldsymbol{z}_0), \quad (11)$$

where each part goes through different numbers of convolutional layers, that is, every gradient flow goes through a varying-length path.[1] According to the chain rule, the gradient of the loss function ℓ with respect to the input can be derived as,

$$\frac{\partial \ell}{\partial \boldsymbol{x}} = \frac{\partial \ell}{\partial \boldsymbol{z}_1}\left(\frac{\partial \mathcal{P}_{1,1}}{\partial \boldsymbol{z}_0} + \frac{\partial \mathcal{P}_{1,2}}{\partial \boldsymbol{z}_0} + \frac{\partial \mathcal{P}_{1,3}}{\partial \boldsymbol{z}_0} + \frac{\partial \mathcal{P}_{1,4}}{\partial \boldsymbol{z}_0}\right). \quad (12)$$

Inspired by SGM in ResNet and ViTs, we decay the gradient by multiplying γ every time the gradient flow goes through a parametric module (e.g., convolutional layer). Note that, since pooling is an extremely simple operation, we do not decay the gradient when the gradient flow goes through it. Thus, we obtain:

$$\frac{\partial \ell}{\partial \boldsymbol{x}} = \frac{\partial \ell}{\partial \boldsymbol{z}_1}\left(\gamma\frac{\partial \mathcal{P}_{1,1}}{\partial \boldsymbol{z}_0} + \gamma\frac{\partial \mathcal{P}_{1,2}}{\partial \boldsymbol{z}_0} + \gamma^2\frac{\partial \mathcal{P}_{1,3}}{\partial \boldsymbol{z}_0} + \gamma^3\frac{\partial \mathcal{P}_{1,4}}{\partial \boldsymbol{z}_0}\right). \quad (13)$$

Since paths of varying lengths are almost ubiquitous in modern deep models, SGM is a generic method that can be applied to a wide range of model architectures, including those designed via Neural Architecture Search (NAS). The principle is that the gradient from the shorter path should dominate those from the longer path when crafting adversarial examples. This intuition is empirically validated by the illustrative experiments shown in Fig. 7 in Appendix A, available online.

IV. Theoretical Analysis of SGM

From a data distribution perspective, deep models tend to perform well on samples that are independently and identically distributed (IID) with the training data, but often fail to generalize to out-of-distribution (OOD) inputs. Therefore, adversarial examples crafted in directions that move samples away from the original data distribution tend to achieve higher transferability. This suggests that a perturbation achieves better transferability when it aligns more closely with the direction that shifts the sample distribution away from the training data [60]. Utilizing this property, [60] proposed a metric measuring the similarity between adversarial attack direction $\nabla_{\boldsymbol{x}}\ell$ and the direction of moving sample away from its original data distribution $p_D(\boldsymbol{x}|y)$ (Intrinsic attack) called AAI (Alignment between the Adversarial attack and Intrinsic attack) to measure the transferability of a given attack:

$$\begin{aligned}\mathrm{AAI}\{\nabla_x\ell\} &= \mathbb{E}_{p_D(y)}\mathbb{E}_{p_D(\boldsymbol{x}|y)}\left\langle \frac{\nabla_{\boldsymbol{x}}\ell}{\|\nabla_{\boldsymbol{x}}\ell\|_2}, \nabla_{\boldsymbol{x}}\log p_D(\boldsymbol{x}|y)\right\rangle \\ &= \mathbb{E}_{p_D(y)}\mathbb{E}_{p_D(\boldsymbol{x}|y)}\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^\top\nabla_{\boldsymbol{x}}\frac{\nabla_{\boldsymbol{x}}\ell\boldsymbol{v}}{\|\nabla_{\boldsymbol{x}}\ell\|_2}\right] + C,\end{aligned} \quad (14)$$

where $\boldsymbol{v}$ is the Gaussian random vector $\boldsymbol{v}\sim\mathcal{N}(0,\mathbf{I})$ and C is a constant only related to the data distribution. Although the ground truth data distribution is not known, we can easily estimate its gradient $\nabla_x\log p_D(\boldsymbol{x}|y)$ via Langevin dynamics [61], and the gradient only considers attacks from the view of data distribution, which is supposed to be a general attack that can transfer well among different models. As demonstrated in [60], larger AAI means the attack direction aligns better with the direction away from the data distribution and the generated adversarial examples thus enjoying better transferability.

Revisiting our proposed SGM, it modifies the gradients of generating adversarial examples ($\nabla_{\boldsymbol{x}}\ell$ in (14)) through tuning a hyper-parameter γ to affect the AAI metric. In the following, we formally analyze why SGM can work through introducing the hyper-parameter γ under the view of data distribution. Proofs are in Appendix C, available online.

Proposition 1: Consider the following binary-classification residual model as follows:

$$\hat{\boldsymbol{y}} = \boldsymbol{x} + g(\boldsymbol{x})$$

with $\boldsymbol{x}\in\mathbb{R}^2, \hat{\boldsymbol{y}}\in\mathbb{R}^2$ is the one-hot label vector, and $g(\boldsymbol{x})$ is a residual block with learnable parameters. If the attack is generated with the hinge loss on a certain class:

$$\ell(\hat{y}, y) = \sum_i y_i \max\left(0, 1-\hat{y}_i\right),$$

with y as a label. If $\|\nabla_{\boldsymbol{x}}g(\boldsymbol{x})\|_F \le 1$ and $0 \le \frac{\partial^2 g}{\partial x_i^2} \le \frac{\partial^2 g}{\partial x_i \partial x_j}$ for all x in ground-truth data distribution $p_D(\boldsymbol{x}|y)$ with $i,j\in\{1,2\}$, there exist a $\gamma\in(0,1)$ which makes

$$\mathrm{AAI}_{\mathrm{SGM}} \ge \mathrm{AAI}_{\mathrm{ORI}},$$

where $\mathrm{AAI}_{\mathrm{SGM}}$ denotes the alignment between the SGM attack direction $\frac{\partial \ell}{\partial \hat{y}}(1+\gamma\frac{\partial f}{\partial x})$ and the log ground-truth class-conditional data, while $\mathrm{AAI}_{\mathrm{ORI}}$ denotes the alignment between the vanilla one-step attack direction $\frac{\partial \ell}{\partial \hat{y}}(1+\frac{\partial f}{\partial x})$ and the log ground-truth class-conditional data.

From the above proposition, we can see that our SGM's gradient aligns better with the direction away from the original data distribution compared with the vanilla scheme under a certain loss. Therefore, SGM enjoys better transferability. Note that this is only one type of theoretical explanation for SGM

[1] We regard the length of a path using the number of parametric layers in this path.

under a specific setting. Comprehensive verification experiments on SGM under various settings are shown in the following sections.

V. Experiments Under Different Architectures

In this section, we demonstrate the catalytic effect of SGM on black-box transferability by combining it with the existing methods on the ImageNet dataset [15] under various architectures.

Competing Attacks: We evaluate the transferability of 35 state-of-the-art attacks, both before and after integrating them with SGM: 1) White-box: PGD [9]; 2) Gradient-related: MI [11], PI [26], DTA [34], GRA [31], PC-I [29], PGN [32], GI [27], RAP [14], AI-FGTM [33], VAI [28], and SMI-FRSM [62]; 3) Augmentation-related: DI [12], Admix [63], SIM [41], VT [64], S^2I [36], AITL [40], MaskBlock [37], STM [41], BSR [38], and SIA [39]; 4) Feature-related: FIA [44], NAA [45], RPA [47], TAIG [65], FMAA [46], ILPD [49], and MFAA [48]; 5) Parameter-related: DSM [66] and RFA [67]; 6) ViT-specific: TGR [56], VDC [57], PNA [54] and SAPR [55]. Due to the space constraints, we only select some representative attacks from each category and one representative source model per architecture in the main paper. For comprehensive results across more attacks and architectures, please refer to Appendix D, available online. We select Masking Unimportant Parameters (MUP) attack [58] as our baseline for comparison. All attacks are conducted under standard settings [11], [12] to generate untargeted adversarial examples with a maximum L_∞ perturbation of $\epsilon = 16$ (on pixel values in $[0, 255]$), using a step size $\alpha = 2$ and 10 iterations. For SGM, the decay factor is set to $\gamma = 0.6$. Some examples of generated images are shown in Appendix B, available online. For fairness, all attack-specific hyperparameters are configured as described in their respective original papers.

Threat Model: We adopt a black-box threat model in which adversarial examples are generated by attacking a source model and then applied to attack the target model. The attacks are crafted on 5000 randomly selected ImageNet validation images that are classified correctly by all source models, and are repeated for 5 random runs.

Source and Target Models: Three kinds of architectures are selected as our source models to show the catalytic effect of SGM on attack transferability: 1) ResNet-like CNNs: ResNet-50 [1] and DenseNet-201 [3]; 2) Vision Transformers: ViT-B [4] and Mixer-B [17]; 3) Models with varying-length paths: Inception-V3 [18] and P-DARTS [68]. For target models, we consider both architectures with or without skip connections, covering almost all popular architectures: VGG16 [69], VGG19 [69], ResNet-152 (RN152) [1], DenseNet-201 (DN201) [3], 154 layer Squeeze-and-Excitation network (SE154) [70], ConViT-B [71], TNT-S [72], Visformer-S [73], Inception V4 (IncV4) [19], and Inception-ResNetV2 (IncResV2) [19]. Whenever the input size of the source model does not match the target model, we resize the crafted adversarial images to the input size of the target model. For Inception/Inception-ResNet models, images are cropped and resized to 299×299 while for other architectures, images are cropped and resized to 224×224.

A. SGM in ResNet-Like CNNs

In Section III-A, we propose that SGM can be easily performed on the ResNet-like CNNs considering skip connections are their basic building component. Therefore, in this section, we further conduct experiments by selecting ResNet-50 and Densenet-201 as the source models to demonstrate its effectiveness.

ResNet-50 and DenseNet-201: We summarize the representative results of ResNet-50 in Table I (full results in Table IX of Appendix D, available online). The results on DenseNet-201 can be found in Table X in Appendix D, available online. Across all scenarios, our proposed SGM consistently enhances attack success rate (ASR) when combined with existing attacks, outperforming the performance of MUP. For example, the ASR of the vanilla SMI-FRSM attack on SENet-154 is 47.96%. But after combining with SGM, the ASR increases significantly to 62.09%, representing an improvement of over 10%. In contrast, MUP yields a smaller gain of only 5.31%. Similar observations are also observed for DenseNet-201, except that the source and target models have the same architectures.

B. SGM in Vision Transformers

In Section III-B, we extend SGM to ViTs since they also have skip connections. Here, we conduct a series of experiments to demonstrate its effectiveness. In addition to architecture-free attacks, we also include ViT-specific attacks for integration with SGM.

ViT: Due to space constraints, we present representative results on ViT-B in Table II, and refer readers to Appendix D, available online (Table XI) for the full results. Across all settings, SGM consistently improves transferability. For instance, when PGD is combined with SGM, the ASR on VGG16 increases from 16.34% to 21.04%. Similar gains are observed for ViT-specific attacks: for example, integrating SGM with TGR on TNT-S yields a 1.57% improvement on ASR. When comparing transferability across target models, we observe that black-box attacks tend to succeed more easily on architectures that share higher similarity with the source model. For example, under the PGD attack, ConViT-B and TNT-S exhibit higher ASRs than Visformer-S. This is likely because Visformer-S replaces key ViT components (e.g., MLP layers and layer normalization) with convolutional layers and batch normalization, reducing its structural similarity with ViT-B. Additionally, shallower models are generally more vulnerable than deeper ones: all attacks achieve higher ASRs on VGG16 compared to VGG19.

MLP-Mixer: We also evaluate SGM on another ViT variant, i.e., MLP-Mixer [17], which substitutes the self-attention modules in ViTs with multilayer perceptrons (MLPs), while still achieving competitive performance on standard image benchmarks.[2] Following the same approach as for ViTs, SGM can be directly applied to MLP-Mixer models. Results reported in Table XII (Appendix D, available online) show that SGM not only enhances MLP-to-ViT transferability but also improves transferability from MLPs to CNNs, including ones with skip

[2]https://paperswithcode.com/sota/image-classification-on-imagenet

TABLE I
MULTI-STEP TRANSFERABILITY (%±STD OVER 5 RANDOM RUNS) USING RESNET-50 AS THE SOURCE MODEL: THE ATTACK SUCCESS RATES OF DIFFERENT METHODS AND THE BEST RESULTS ARE IN **BOLD**

	Target	VGG16	VGG19	RN152	DN201	SE154	ConViT-B	TNT-S	Visformer-S	IncV4	IncRes
White-box	PGD	45.00±0.06	43.48±0.12	54.54±0.58	43.20±0.28	17.76±0.20	2.78±0.08	13.26±0.28	8.76±0.08	13.74±0.18	12.43±0.13
	PGD+MUP	50.84±0.30	49.52±0.14	61.41±0.39	49.45±0.45	21.17±0.21	2.96±0.06	14.27±0.13	9.75±0.17	15.51±0.03	14.15±0.35
	PGD+SGM	**58.68±0.26**	**56.78±0.24**	**99.98±0.00**	**64.99±0.07**	**37.86±0.06**	**7.92±0.02**	**23.88±0.12**	**22.09±0.05**	**26.98±0.06**	**25.36±0.02**
Gradient-related	SMI-FRSM	78.11±0.37	77.39±0.11	86.68±0.06	79.46±0.40	47.96±0.04	10.77±0.03	31.16±0.10	28.03±0.27	38.93±0.05	36.89±0.25
	SMI-FRSM+MUP	83.71±0.19	82.93±0.31	91.16±0.14	84.30±0.08	53.27±0.33	11.46±0.06	33.46±0.32	30.31±0.19	42.37±0.13	40.50±0.26
	SMI-FRSM+SGM	**88.07±0.11**	**87.09±0.27**	**92.89±0.17**	**87.85±0.13**	**62.09±0.21**	**18.64±0.36**	**43.28±0.38**	**41.56±0.16**	**50.09±0.19**	**47.51±0.15**
Augmentation-related	SIA	90.88±0.04	87.39±0.33	88.71±0.15	84.88±0.06	57.24±0.40	6.63±0.37	29.19±0.03	27.88±0.20	42.42±0.32	35.27±0.21
	SIA+MUP	90.93±0.05	87.71±0.07	89.30±0.12	84.91±0.07	57.72±0.34	6.48±0.08	29.68±0.00	27.59±0.01	41.92±0.02	35.09±0.09
	SIA+SGM	**95.93±0.03**	**93.09±0.07**	**93.22±0.06**	**90.24±0.30**	**72.44±0.04**	**12.40±0.32**	**43.53±0.01**	**41.90±0.50**	**53.10±0.36**	**46.63±0.13**
Feature-related	MFAA	90.11±0.03	90.29±0.03	95.32±0.04	88.67±0.23	67.88±0.02	10.14±0.04	42.32±0.06	34.60±0.26	57.13±0.13	52.36±0.02
	MFAA+MUP	90.87±0.01	90.94±0.10	95.39±0.05	89.76±0.04	69.91±0.15	10.81±0.09	43.31±0.09	35.94±0.40	58.89±0.41	54.19±0.23
	MFAA+SGM	**93.28±0.08**	**93.47±0.17**	**96.12±0.20**	**91.49±0.03**	**77.91±0.21**	**15.04±0.14**	**56.50±0.00**	**42.25±0.21**	**69.46±0.22**	**66.07±0.41**
Parameter-related	RFA	87.71±0.05	88.74±0.02	95.30±0.12	93.27±0.05	75.16±0.02	36.82±0.02	62.76±0.20	60.68±0.12	66.46±0.18	65.63±0.25
	RFA+MUP	88.93±0.21	90.27±0.01	96.15±0.07	94.37±0.01	77.13±0.11	38.93±0.01	64.95±0.07	63.28±0.20	68.58±0.24	67.69±0.17
	RFA+SGM	**93.52±0.02**	**93.99±0.03**	**97.29±0.03**	**96.50±0.02**	**83.47±0.25**	**51.37±0.15**	**76.16±0.22**	**73.28±0.28**	**75.05±0.13**	**75.40±0.16**

TABLE II
MULTI-STEP TRANSFERABILITY (%±STD OVER 5 RANDOM RUNS) USING VIT-B AS THE SOURCE MODEL: THE ATTACK SUCCESS RATES OF DIFFERENT METHODS AND THE BEST RESULTS ARE IN **BOLD**

	Target	VGG16	VGG19	RN152	DN201	SE154	ConViT-B	TNT-S	Visformer-S	IncV4	IncRes
White-box	PGD	16.34±0.10	14.74±0.22	8.02±0.00	10.04±0.10	7.58±0.24	17.07±0.05	25.99±0.57	7.27±0.07	7.61±0.03	5.81±0.07
	PGD+MUP	17.39±0.17	15.57±0.25	8.56±0.00	11.00±0.18	8.45±0.25	19.22±0.14	28.11±0.15	8.18±0.04	8.22±0.02	6.37±0.05
	PGD+SGM	**21.04±0.14**	**18.87±0.25**	**10.25±0.19**	**13.33±0.03**	**10.73±0.03**	**22.72±0.24**	**32.57±0.37**	**10.48±0.16**	**9.13±0.23**	**7.03±0.01**
Gradient-related	SMI-FRSM	29.76±0.04	27.06±0.24	16.48±0.04	21.05±0.05	16.11±0.15	32.81±0.07	43.76±0.30	15.78±0.26	15.30±0.02	11.83±0.07
	SMI-FRSM+MUP	30.39±0.13	28.94±0.02	17.48±0.02	23.45±0.23	17.17±0.17	34.66±0.06	46.29±0.01	16.89±0.03	16.08±0.38	12.79±0.25
	SMI-FRSM+SGM	**34.21±0.33**	**31.01±0.05**	**19.51±0.31**	**25.21±0.31**	**19.66±0.28**	**37.31±0.27**	**49.21±0.49**	**19.20±0.00**	**16.65±0.03**	**13.67±0.00**
Augmentation-related	SIA	47.42±0.14	42.91±0.01	28.24±0.38	34.13±0.17	36.14±0.32	51.17±0.47	68.39±0.03	34.95±0.25	27.94±0.20	21.71±0.03
	SIA+MUP	47.57±0.11	43.40±0.06	28.19±0.23	34.66±0.42	36.22±0.36	51.26±0.02	69.47±0.37	34.85±0.33	28.27±0.07	21.86±0.06
	SIA+SGM	**56.67±0.33**	**51.47±0.13**	**36.48±0.16**	**44.25±0.23**	**46.98±0.04**	**61.29±0.09**	**78.14±0.02**	**45.86±0.32**	**34.10±0.06**	**27.51±0.51**
Feature-related	MFAA	27.25±0.05	25.09±0.05	10.91±0.13	12.86±0.24	9.12±0.04	8.95±0.07	22.43±0.05	4.57±0.05	9.47±0.03	7.55±0.17
	MFAA+MUP	27.02±0.20	24.72±0.54	10.82±0.16	12.75±0.11	9.17±0.07	9.00±0.02	21.78±0.20	4.50±0.12	9.41±0.15	7.34±0.02
	MFAA+SGM	**40.80±0.44**	**39.27±0.31**	**27.66±0.38**	**33.39±0.17**	**30.16±0.16**	**58.41±0.09**	**63.04±0.32**	**30.95±0.09**	**19.86±0.32**	**17.43±0.05**
Parameter-related	RFA	40.06±0.22	38.23±0.39	26.64±0.12	32.65±0.21	29.86±0.64	57.60±0.44	61.59±0.11	30.14±0.06	19.60±0.42	17.29±0.09
	RFA+MUP	40.80±0.44	39.27±0.31	27.66±0.38	33.39±0.17	30.16±0.16	58.41±0.09	63.04±0.32	30.95±0.09	19.86±0.32	17.43±0.00
	RFA+SGM	**46.69±0.59**	**44.09±0.13**	**31.57±0.09**	**38.59±0.15**	**34.72±0.10**	**59.99±0.27**	**68.09±0.11**	**34.05±0.15**	**22.35±0.17**	**19.61±0.19**
ViT-specifc	TGR	25.99±0.11	23.21±0.09	12.86±0.02	17.07±0.09	13.89±0.03	26.65±0.23	39.45±0.03	12.80±0.08	11.21±0.19	8.68±0.14
	TGR+MUP	27.83±0.13	25.20±0.06	13.46±0.02	17.97±0.27	14.86±0.12	26.89±0.03	40.93±0.01	12.85±0.20	11.78±0.17	8.97±0.07
	TGR+SGM	**30.41±0.19**	**27.41±0.19**	**14.57±0.01**	**19.63±0.29**	**15.60±0.04**	**27.22±0.26**	**41.02±0.16**	**13.05±0.23**	**11.88±0.18**	**9.02±0.10**
	VDC	20.05±0.35	18.11±0.19	10.06±0.20	12.83±0.17	10.22±0.16	22.53±0.23	31.93±0.33	10.11±0.29	8.79±0.13	6.92±0.10
	VDC+MUP	21.00±0.40	18.84±0.10	10.60±0.34	13.54±0.04	10.39±0.09	23.96±0.16	33.50±0.24	10.90±0.04	9.29±0.25	7.11±0.17
	VDC+SGM	**24.08±0.06**	**21.70±0.06**	**11.89±0.43**	**15.67±0.21**	**12.28±0.12**	**24.68±0.36**	**35.99±0.11**	**11.60±0.00**	**9.66±0.16**	**7.49±0.03**

TABLE III
MULTI-STEP TRANSFERABILITY (%±STD OVER 5 RANDOM RUNS) USING INCEPTION-V3 AS THE SOURCE MODEL: THE ATTACK SUCCESS RATES OF DIFFERENT METHODS AND THE BEST RESULTS ARE IN **BOLD**

	Target	VGG16	VGG19	RN152	DN201	SE154	ConViT-B	TNT-S	Visformer-S	IncV4	IncRes
White-box	PGD	29.74±0.02	28.14±0.52	13.76±0.16	14.07±0.07	11.81±0.15	2.12±0.04	10.04±0.04	5.80±0.10	29.63±0.85	26.23±0.27
	PGD+MUP	21.62±0.24	20.57±0.15	8.69±0.33	8.72±0.06	6.34±0.16	1.02±0.10	7.80±0.18	3.06±0.00	16.90±0.18	15.54±0.06
	PGD+SGM	**45.43±0.21**	**41.40±0.04**	**20.77±0.23**	**21.40±0.34**	**20.07±0.01**	**3.60±0.00**	**17.17±0.11**	**11.30±0.14**	**44.90±0.06**	**40.78±0.06**
Gradient-related	SMI-FRSM	53.71±0.33	53.06±0.56	34.75±0.07	37.21±0.09	30.45±0.19	7.41±0.09	23.37±0.07	17.23±0.15	56.58±0.06	52.31±0.43
	SMI-FRSM+MUP	41.74±0.00	40.06±0.18	21.85±0.07	22.09±0.29	19.50±0.16	3.51±0.11	16.83±0.25	8.76±0.16	36.79±0.31	33.60±0.08
	SMI-FRSM+SGM	**69.43±0.03**	**66.14±0.32**	**44.09±0.19**	**45.10±0.20**	**43.32±0.26**	**11.13±0.05**	**35.33±0.35**	**26.94±0.08**	**68.36±0.14**	**64.65±0.35**
Augmentation-related	SIA	44.94±0.06	40.33±0.31	17.07±0.39	16.10±0.18	14.54±0.14	1.93±0.19	11.55±0.01	5.74±0.22	38.25±0.19	32.15±0.35
	SIA+MUP	48.02±0.26	43.72±0.14	19.37±0.11	18.27±0.41	15.96±0.06	1.85±0.05	12.40±0.08	6.53±0.13	42.07±0.41	35.83±0.03
	SIA+SGM	**80.59±0.03**	**75.68±0.10**	**46.02±0.08**	**43.91±0.25**	**43.91±0.05**	**7.14±0.10**	**33.50±0.14**	**25.95±0.15**	**74.26±0.16**	**68.19±0.01**
Feature-related	MFAA	75.11±0.19	74.34±0.22	51.34±0.24	50.29±0.05	48.54±0.02	8.03±0.05	34.02±0.08	24.82±0.00	80.83±0.07	76.15±0.01
	MFAA+MUP	64.96±0.20	64.19±0.31	37.55±0.21	35.40±0.16	36.64±0.10	4.84±0.18	25.12±0.18	15.66±0.44	67.20±0.22	62.17±0.09
	MFAA+SGM	**83.74±0.04**	**82.22±0.34**	**59.23±0.07**	**58.04±0.18**	**58.37±0.19**	**10.82±0.08**	**46.51±0.33**	**32.52±0.00**	**85.05±0.03**	**82.46±0.10**
Parameter-related	RFA	44.17±0.33	42.16±0.16	23.80±0.02	22.55±0.13	19.67±0.05	3.89±0.09	15.86±0.08	10.57±0.05	43.30±0.24	41.43±0.17
	RFA+MUP	40.05±0.03	38.18±0.44	20.86±0.12	19.57±0.07	17.16±0.30	3.11±0.13	14.12±0.06	8.83±0.03	37.57±0.07	36.50±0.16
	RFA+SGM	**57.27±0.29**	**52.71±0.07**	**29.85±0.31**	**27.95±0.05**	**28.68±0.14**	**6.83±0.07**	**25.06±0.24**	**18.55±0.15**	**54.75±0.05**	**52.25±0.25**

connections and the others with varied length. For example, SGM improves the ASR of DI attack by 10.42% on ConViT-B and by 7.30% on VGG16.

C. SGM in Models With Varying-Length Paths

In Section III-C, we present that SGM can be extended into a broader range of architectures with varying-length paths. In this part, taking Inception [18] and models from Neural Architecture Search (NAS) [74] as examples, we evaluate the effectiveness of SGM in models without skip connections.

Inception Models: Inception is a widely used architecture such as Inception-V3 illustrated in Fig. 2(c). Using Inception-V3 as the source model, we evaluate the transferability of adversarial examples generated with and without SGM on a variety of target models. Representative results are reported in Table III, and full results are provided in Table XIII in Appendix D, available online. Similarly, SGM yields notable improvements in transferability and outperforms MUP across most settings. Interestingly, despite the source model lacking skip connections, SGM still enhances transferability to target models both without skip connections (e.g., VGG19) and with skip connections (e.g., ResNet-152). This suggests that the effectiveness of SGM is not limited to models with explicit skip connections.

Models from Neural Architecture Search: While the above models are hand-designed by experts, NAS techniques aim to

TABLE IV
THE ATTACK SUCCESS RATES (%±STD OVER 5 RANDOM RUNS) OF DIFFERENT METHODS ON 10 TARGET MODELS ADOPTING AN ENSEMBLE OF MODELS AS THE SOURCE MODEL. THE BEST RESULTS ARE IN **BOLD**.

Category	Attack	VGG16	VGG19	RN152	DN201	SE154	ConViT-B	TNT-S	Visformer-S	IncV4	IncRes
White-box	PGD	73.74±0.18	73.95±0.11	81.98±0.28	**99.94±0.00**	52.28±0.12	11.35±0.33	29.86±0.26	33.84±0.02	51.06±0.48	45.57±0.11
	PGD+SGM	**85.59±0.13**	**85.12±0.30**	**91.08±0.16**	99.93±0.01	**67.95±0.47**	**19.08±0.02**	**42.07±0.23**	**50.00±0.32**	**59.84±0.48**	**55.45±0.81**
Gradient-related	SMI-FRSM	92.86±0.08	93.00±0.16	95.95±0.05	**99.77±0.01**	82.81±0.09	35.41±0.33	61.07±0.05	67.58±0.22	83.09±0.11	79.98±0.08
	SMI-FRSM+SGM	**96.20±0.04**	**96.28±0.06**	**97.94±0.14**	99.69±0.03	**89.73±0.15**	**45.84±0.26**	**72.51±0.27**	**78.33±0.21**	**86.29±0.19**	**84.58±0.04**
Augmentation-related	SIA	98.35±0.09	97.99±0.11	97.32±0.06	99.89±0.00	91.89±0.03	22.37±0.17	62.98±0.03	72.33±0.27	90.56±0.07	83.68±0.07
	SIA+SGM	**99.07±0.05**	**98.97±0.03**	**98.71±0.07**	**99.98±0.00**	**94.91±0.03**	**32.62±0.06**	**73.20±0.12**	**81.14±0.26**	**90.78±0.02**	**84.12±0.16**
Feature-related	MFAA	83.62±0.10	83.66±0.10	80.61±0.01	98.02±0.08	61.70±0.50	10.54±0.10	42.25±0.27	34.52±0.16	71.12±0.50	64.55±0.65
	MFAA+SGM	**97.77±0.11**	**97.46±0.04**	**97.89±0.05**	**99.96±0.00**	**90.32±0.02**	**30.94±0.04**	**77.27±0.09**	**67.97±0.15**	**89.84±0.06**	**87.19±0.07**
Parameter-related	RFA	90.89±0.27	91.13±0.17	94.44±0.02	99.99±0.01	81.90±0.12	39.56±0.24	64.76±0.28	69.62±0.20	76.98±0.08	74.70±0.10
	RFA+SGM	**93.70±0.10**	**93.79±0.07**	**96.48±0.02**	**100.00±0.00**	**86.78±0.10**	**46.79±0.13**	**72.90±0.06**	**77.20±0.08**	**79.49±0.01**	**77.79±0.05**

design diverse structures automatically. To verify the generality of SGM, we evaluate it on a representative NAS-generated model, P-DARTS [68]. Following the implementation strategy used for Inception, we apply gradient decay by multiplying a decay factor γ at every ReLU activation along the back-propagation path. The results are summarized in Table XIV in Appendix D, available online. Again, SGM consistently enhances transferability, even when combined with advanced techniques such as MaskBlock where the attack success rate on VGG19 increases from 22.64% to 31.94% with the integration of SGM. These results collectively demonstrate that SGM is a general and architecture-agnostic technique that can be effectively integrated with various attack methods—even in models without explicit skip connections.

VI. EXPERIMENTS UNDER COMPLEX SCENARIOS

We further evaluate the effectiveness of SGM under more challenging and realistic settings, including ensemble-based attacks, targeted attacks, and scenarios where target models are equipped with defense mechanisms. Our experiments demonstrate that SGM consistently improves ASR even in these complex settings, highlighting its robustness and practical utility in real-world adversarial scenarios.

A. Evaluation of SGM on Ensemble Models

Model ensemble is a widely adopted technique in machine learning to improve performance by combining multiple individual models [75]. In the adversarial context, [7] demonstrated that aggregating gradients from an ensemble of surrogate models can significantly enhance the transferability of black-box attacks by leveraging more information from different architectures.

Building on this insight, we investigate whether SGM can serve as a complementary component in such ensemble-based attacks. Specifically, we construct a surrogate model by ensembling three architectures: ResNet-50, DenseNet-201, and Inception-V3. The target model configuration and attack settings follow those described in Section V, with the decay parameter for SGM set to $\gamma = 0.8$.

The results, summarized in Table IV, show that integrating SGM consistently improves performance across various target architectures. For instance, against SE154, the success rate of MFAA increases from 61.70% to 90.32% after applying SGM. These results demonstrate that SGM can be effectively combined with ensemble-based attacks to further enhance black-box transferability.

B. Combination of SGM With Target Attacks

In the previous sections, we have demonstrated that SGM acts as a universal catalyst, significantly enhancing the performance of untargeted black-box attacks. However, in real-world scenarios, attackers sometimes also aim for targeted attacks, which are more challenging as they require manipulating the model's prediction toward a specific, pre-defined class.

To evaluate the applicability of SGM in this setting, we consider four recent advanced targeted attacks as baselines: Logit [76], FFT [77], CFM [78], and Logit-Margin [79]. ResNet-50 is selected as the source model, and we assess the transferability against ten different target models. Following the protocol in [78], we increase the total number of iterations to 300 to allow targeted perturbations sufficient convergence. All other hyperparameters are kept consistent with the untargeted setting.

As shown in Table V, integrating SGM improves the performance of all targeted attacks. For example, in the case of CFM, the ASR on ConViT-B increases from 39.02% to 47.10% after applying SGM, with almost no additional computational cost. Similar to the untargeted scenario, the architectural similarity between source and target models also plays a key role in transferability: CNN-based targets consistently yield higher ASRs than ViT-based ones.

C. Robustness of SGM Against Existing Defenses

To alleviate the threat of adversarial attacks, multiple defense approaches have been developed, which can be categorized into four groups: 1) adversarial training (AT) [9], [80], [81], [82], 2) certified robustness [83], [84], 3) denoised models [85], and 4) purified-based defenses [86]. For AT-based defenses, we evaluate the effectiveness of SGM against three representative methods: Fast-AT [87], CFA [81], and Robust Architectures (RA) [82]. For the other categories, we select one representative from each: Random Smoothing (RS) [83] for certified defenses, High-Level Representation Guided Denoiser (HGD) [85] for denoising-based defenses, and Neural Representation Purifier (NRP) [86] for purification-based defenses. Detailed settings are provided in Appendix E, available online.

As shown in Table VI, SGM consistently enhances the ability of existing attacks to bypass various defenses. This improvement arises from SGM's ability to leverage more gradients from the path with shorter lengths, thereby better preserving adversarial perturbations. Among the evaluated defenses, AT-based methods (Fast-AT, CFA, RA) yield the lowest ASRs, even against some advanced attacks like MFAA, consistent with findings in [88].

TABLE V
THE ATTACK SUCCESS RATES (%±STD OVER 5 RANDOM RUNS) OF *TARGETED* ATTACKS USING RESNET-50 AS THE SOURCE MODEL. THE BEST RESULTS ARE IN **BOLD**.

Attack	VGG16	VGG19	RN152	DN201	SE154	ConViT-B	TNT-S	Visformer-S	IncV4	IncRes
Logit	37.90±0.38	38.08±0.50	61.39±0.23	70.03±0.27	47.28±0.48	3.41±0.01	8.03±0.47	19.74±0.16	32.12±0.06	34.26±0.40
Logit+SGM	**63.49±0.57**	**59.95±0.13**	**77.35±0.61**	**84.93±0.01**	**65.92±0.00**	**10.82±0.02**	**20.49±0.05**	**41.54±0.00**	**52.76±0.30**	**55.71±0.19**
FFT	9.72±0.26	9.03±0.05	18.83±0.17	15.70±0.22	6.24±0.22	0.25±0.03	0.98±0.08	1.42±0.02	3.13±0.17	3.22±0.08
FFT+SGM	**16.99±0.29**	**14.88±0.26**	**25.35±0.33**	**22.15±0.07**	**10.87±0.05**	**0.68±0.04**	**2.44±0.12**	**3.08±0.08**	**4.71±0.19**	**5.15±0.19**
CFM	81.86±0.02	81.44±0.24	87.82±0.18	88.06±0.18	76.73±0.53	23.46±0.46	50.65±0.07	61.58±0.08	69.50±0.30	69.89±0.09
CFM+SGM	**84.52±0.16**	**83.41±0.13**	**89.23±0.05**	**89.13±0.13**	**78.42±0.12**	**32.18±0.24**	**58.09±0.01**	**66.62±0.06**	**71.09±0.11**	**71.97±0.03**
Logit-Margin	47.86±0.14	47.07±0.37	72.44±0.32	77.58±0.06	52.77±0.31	4.01±0.13	8.85±0.19	22.31±0.39	34.72±0.86	37.08±0.44
Logit-Margin+SGM	**71.69±0.05**	**68.19±0.65**	**86.74±0.02**	**90.66±0.04**	**70.71±0.25**	**11.51±0.15**	**22.03±0.45**	**44.56±0.68**	**54.65±0.11**	**57.60±0.28**

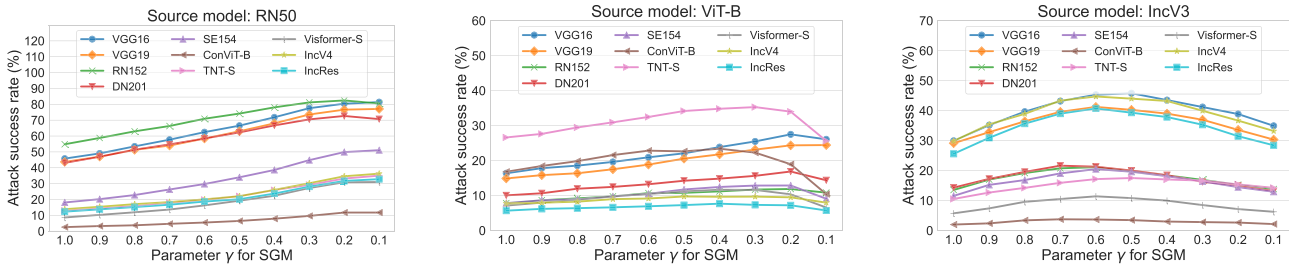


Fig. 3. The attack success rates of 10-step PGD combined with SGM with varying decay parameter γ. Each figure represents different source models: (a) RN50, (b) ViT-B, and (c) IncV3. Curves denote results against different target models.

TABLE VI
RESISTANCE TO DEFENSE (%±STD OVER 5 RANDOM RUNS): THE SUCCESS RATES OF DIFFERENT ATTACKS AGAINST DEFENSE METHODS WITH OR WITHOUT SGM. THE BEST RESULTS ARE IN **BOLD**.

Attack \Defense	Fast-AT	CFA	RA	RS	HGD	NRP
PGD	21.25±0.01	8.77±0.07	4.03±0.03	63.15±0.01	11.78±0.30	57.38±0.10
PGD+SGM	**22.40±0.08**	**10.98±0.36**	**4.51±0.01**	**64.27±0.15**	**29.53±0.27**	**61.61±0.30**
SMI-FRSM	22.98±0.12	8.77±0.07	4.91±0.09	64.19±0.09	49.71±0.01	63.91±0.35
SMI-FRSM+SGM	**24.40±0.00**	**10.98±0.36**	**5.59±0.03**	**64.84±0.08**	**54.39±0.29**	**68.28±0.04**
SIA	22.26±0.02	12.27±0.09	4.52±0.00	65.03±0.01	32.09±0.17	67.04±0.06
SIA+SGM	**23.68±0.02**	**15.32±0.20**	**5.18±0.02**	**65.80±0.02**	**40.62±0.04**	**70.41±0.69**
MFAA	22.46±0.06	10.23±0.03	4.74±0.00	67.42±0.04	31.54±0.16	71.96±0.04
MFAA+SGM	**25.11±0.09**	**14.44±0.04**	**5.54±0.02**	**68.01±0.13**	**36.95±0.31**	**75.95±0.01**
RFA	21.70±0.06	9.40±0.02	4.37±0.01	59.60±0.04	23.48±0.28	64.12±0.00
RFA+SGM	**23.51±0.03**	**13.13±0.15**	**4.98±0.06**	**64.89±0.23**	**40.42±0.08**	**64.74±0.06**

Nonetheless, SGM remains effective even in these challenging settings. For instance, against CFA, SGM improves the ASR of RFA from 9.40% to 13.13% (+3.73%). These results highlight SGM's potential to enhance attack strength even in the presence of defenses.

VII. EMPIRICAL UNDERSTANDINGS OF SGM

Beyond demonstrating the effectiveness of SGM across various scenarios, we conduct a series of comprehensive experiments to gain deeper empirical insights for SGM.

A. The Selection of Hyperparameter γ

Here, we first analyze the influence of the hyperparameter γ and then provide practical guidance for its selection. Specifically, we vary $\gamma \in [0.1, 1.0]$, where $\gamma = 1.0$ indicates no decay on residual gradients. To ensure that our guidance can be applied across different architectures, we select 6 models, i.e., ResNet50, DenseNet201, ViT-B, Mixer-B, Inception-v3, and P-DARTS as source models. The results of ResNet50, ViT-B and Inception-v3 are shown in Fig. 3. For the results of the other three models, please refer to Fig. 9 in Appendix F, available online.

First, we observe that the transferability trends with respect to the decay parameter γ are highly consistent. For example, using ResNet-50 as the source model, decreasing γ (i.e., applying stronger decay) generally improves transferability, with performance peaking around $\gamma = 0.2$. A similar trend is observed on source models without skip connections (e.g., Inception-V3), where transferability improves consistently as γ decreases, with optimal performance typically achieved around $\gamma = 0.6$.

When comparing the optimal γ values across different source models, we find a clear distinction based on architectures. For models with skip connections (e.g., ResNet-50, ViT-B), smaller γ values tend to yield better transferability. This is likely because skip connections inherently provide more transferable gradient pathways, allowing these models to benefit from a stronger decay on the residual gradients. In contrast, for models without skip connections (e.g., Inception-V3), a larger γ is preferable to retain useful low-level signals propagated through shorter paths–stronger decay (i.e., smaller γ) may cause gradient vanishing along these paths. Based on this analysis, we set $\gamma = 0.6$ throughout our main experiments, as a balanced trade-off across different architectures: it is neither too aggressive for models without skip connections nor too conservative for those with them.

Nonetheless, we emphasize that optimal γ values can be easily tuned in practice, as our experiments reveal that the optimal setting is primarily determined by the source model, and is largely invariant across different target models. This significantly simplifies the hyperparameter selection process: one can calibrate γ solely based on the source model architecture, without needing to account for each specific target model. For instance, as shown in Fig. 3(a), even when the actual target model is DN201 (red

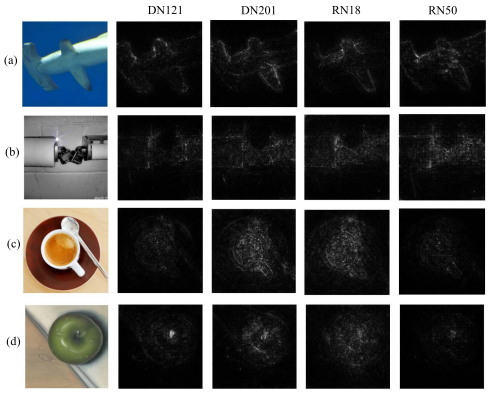


Fig. 4. SmoothGrad of different models on the ImageNet dataset. The average confidence (%) of four architectures on the ground truth class is (a) Hammerhead: 99.47. (b) Toilet tissue: 95.38. (c) Espresso: 87.36. (d) Granny Smith: 99.54.

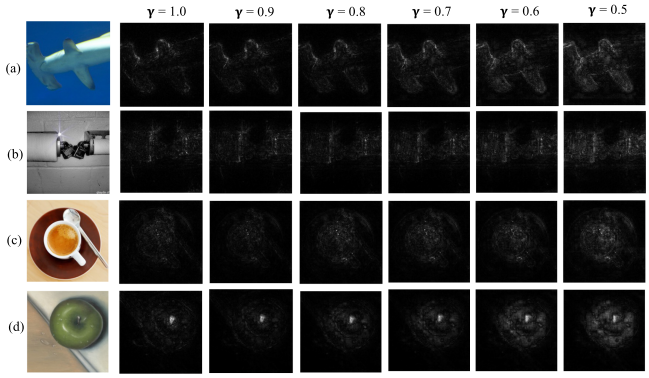


Fig. 5. SmoothGrad of DenseNet-121 with varying γ. When we decay the gradient from skip connections, the features gradually change from local to global.

curve), tuning γ on ResNet-50 to attack RN152 (green curve) still leads to the optimal value $\gamma = 0.2$, which also performs well against DN201. Similar patterns are observed for source models without skip connections—for example, in Fig. 3(c), TNT-S and VGG16 share a similar optimal γ when using Inception-V3 as the source.

B. Adaptivity and Interpretability of SGM

We further examine the adaptability of SGM from two perspectives: varying attack budgets and adaptive decay strategies. As detailed in Appendix H, available online, under different perturbation budgets ϵ, SGM consistently yields substantial relative improvements on transferability. Moreover, in Appendix G, available online, we explore different decay strategies for SGM. Specifically, we consider: 1) module-wise decay, where separate decay factors are assigned to different architectural components—namely, γ_{attn} for self-attention layers and γ_{mlp} for MLP blocks in ViTs. We find that transferability is highly sensitive to the decay of attention gradients. And 2) decay frequency along the path, where the decay is applied either once per residual path or multiple times at each parametric module in Inception. The results consistently show that decaying gradients at each parametric module is critical for achieving stronger transferability.

To gain deeper insight into how SGM influences feature learning, we visualize the perturbed regions using SmoothGrad [89] on ResNet-like models. In Fig. 4, we present SmoothGrad maps for four models (DN121, DN201, RN18, RN50) on high-confidence images from the ImageNet validation set. These visualizations show that different architectures rely on different predictive features and perturb them in distinct ways, which helps explain the low transferability of vanilla adversarial examples. Building on this, we apply PGD+SGM with different γ and visualize the resulting features in Fig. 5. We find that smaller γ values lead to broader feature activation, indicating that SGM encourages perturbations to shift from highly localized patterns to more global, transferable features. To quantify this effect, we evaluate the transferability of models in Fig. 5 against ResNet-50. As shown in Table VII, the confidence of the ground-truth class on the target model significantly decreases as γ decreases. This indicates that applying SGM can effectively introduce more global features into adversarial perturbation, which improves the transferability and enhances threats to target models.

TABLE VII
THE CONFIDENCE (%) OF ADVERSARIAL SAMPLES ON GROUND-TRUTH CLASS (AVERAGE OVER 5 RANDOM SEEDS). LOWER RESULTS MEAN HIGHER BLACK-BOX TRANSFERABILITY. HERE THE ALPHABETS, E.G. (A), REPRESENTS THE SAME IMAGES SHOWN IN FIG. 5. THE BEST RESULTS ARE IN **BOLD**.

γ	1.0	0.9	0.8	0.7	0.6	0.5
(a)	16.84	15.94	3.62	1.93	0.98	**0.56**
(b)	19.71	21.92	10.79	5.86	5.50	5.50
(c)	0.48	0.18	4.31e-02	2e-02	1.06e-02	**1.76e-04**
(d)	41.35	13.09	11.46	0.40	2.93e-03	**5.87e-05**

VIII. EXTENDING SGM TO ATTACK LARGE LANGUAGE MODELS

Adversarial examples are not limited to the vision domain—they also emerge in natural language processing tasks, particularly with Large Language Models (LLMs). Known as "jailbreak attacks" [90], [91], [92], these adversarial inputs involve appending seemingly meaningless suffixes to prompts, which can trigger unintended or harmful model behavior. Similar to black-box attacks in vision, such adversarial suffixes can be crafted using a surrogate LLM and then transferred to attack unseen target LLMs [90].

Given that LLMs are fundamentally composed of Transformer blocks, SGM can be readily extended to this domain by combining it with gradient-based jailbreak attack strategies. In particular, we integrate SGM into the GCG attack framework [90], one of the most effective and widely used methods. Following the setup in the original GCG paper, we consider two scenarios: 1) individual setting where an adversarial suffix is optimized for a single attack prompt; and 2) multiple setting where a universal suffix is jointly optimized over multiple training prompts and then evaluated on new, unseen prompts. Implementation details and hyperparameter configurations are provided in Appendix I, available online. We use Vicuna-7B [93] as the surrogate model to craft adversarial suffixes and evaluate their

TABLE VIII
TRANSFERABILITY (%±STD OVER 2 RANDOM RUNS) OF ADVERSARIAL SUFFIX WITH $\gamma = 0.8$: THE ATTACK SUCCESS RATES OF JAILBREAK ATTACKS FOR LLMS WITH OR WITHOUT SGM AND THE BEST RESULTS ARE IN **BOLD**.

Setting	Attack	MPT-7B	Pythia-12B	Vicuna-13B	Stable-Vicuna-13B
Individual	GCG	6.50±0.50	56.50±2.50	2.00±1.00	31.50±2.50
	GCG+SGM	**8.50±0.50**	**61.50±1.50**	**5.50±0.50**	**42.00±4.00**
Multiple	GCG	8.50±1.50	50.50±2.50	2.00±1.00	10.50±1.50
	GCG+SGM	**15.00±2.00**	**70.00±1.00**	**6.00±1.00**	**48.50±7.50**

transferability on several target models, including MPT-7B [94], Pythia-12B [95], Vicuna-13B [93], and Stable-Vicuna-13B [96].

As shown in Table VIII, incorporating SGM consistently improves the attack success rates across all target models. These results demonstrate the generalizability of SGM beyond the vision domain, highlighting its potential as a transferable gradient modulation framework across modalities.

IX. CONCLUSION

In this paper, we have identified a surprising property of the generalized "skip connections" used by many state-of-the-art deep models, that is, they can be easily used to generate highly transferable adversarial examples. Starting from ResNet-like models in vision domains, we propose the *Skip Gradient Method* (SGM), which enhances transferability by biasing backpropagation to favor gradients flowing through skip connections while attenuating those from residual modules via a decay factor. Further, we generalize SGM to a wide range of architectures, including Vision Transformers, models with varying-length paths (e.g., Inception, NAS-based models), and even large language models. Extensive experiments across these diverse settings consistently demonstrate that SGM leads to a substantial boost in adversarial transferability, including in challenging scenarios such as targeted attacks, ensemble-based attacks, and against defense-equipped models. In addition to empirical evaluations, we provide both theoretical analysis and interpretability-based insights to understand the mechanism behind SGM's effectiveness. Our findings highlight an important link between architectural design and adversarial vulnerability, pointing toward the design of more secure and robust model architectures.

ACKNOWLEDGMENT

Dongxian Wu's main contribution to this work was done during his doctoral studies.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[2] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016.
[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2261–2269.
[4] A. Dosovitskiy et al., "An image is worth 16 × 16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–21.
[5] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2013, pp. 1–10.
[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1–11.
[7] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–24.
[8] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–23.
[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.
[11] Y. Dong et al., "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.
[12] C. Xie et al., "Improving transferability of adversarial examples with input diversity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2725–2734.
[13] Z. Zhao, H. Zhang, R. Li, R. Sicre, L. Amsaleg, and M. Backes, "Towards good practices in evaluating transfer adversarial attacks," 2022, *arXiv:2211.09565*.
[14] Z. Qin et al., "Boosting the transferability of adversarial attacks with reverse adversarial perturbation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 29845–29858.
[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[16] D. Wu, Y. Wang, S.-T. Xia, J. Bailey, and X. Ma, "Skip connections matter: On the transferability of adversarial examples generated with ResNets," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–15.
[17] I. O. Tolstikhin et al., "MLP-mixer: An all-MLP architecture for vision," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 24261–24272.
[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.
[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284.
[20] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.
[21] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019, *arXiv:1905.01392*.
[22] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2016, pp. 372–387.
[23] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[24] A. N. Bhagoji, W. He, B. Li, and D. Song, "Practical black-box attacks on deep neural networks using efficient query mechanisms," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 154–169.
[25] L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang, "Black-box adversarial attacks on video recognition models," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 864–872.
[26] X. Wang, J. Lin, H. Hu, J. Wang, and K. He, "Boosting adversarial transferability through enhanced momentum," 2021, *arXiv:2103.10609*.
[27] J. Wang et al., "Boosting the transferability of adversarial attacks with global momentum initialization," *Expert Syst. Appl.*, vol. 255, 2022, Art. no. 124757.
[28] M. Zhang, X. Kuang, H. Li, Z. Wu, Y. Nie, and G. Zhao, "Improving transferability of adversarial examples with virtual step and auxiliary gradients," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 1629–1635.
[29] C. Wan and F. Huang, "Adversarial attack based on prediction-correction," 2023, *arXiv:2306.01809*.
[30] A. Peng, Z. Lin, H. Zeng, W. Yu, and X. Kang, "Boosting transferability of adversarial example via an enhanced Euler's method," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1–5.
[31] H. Zhu, Y. Ren, X. Sui, L. Yang, and W. Jiang, "Boosting adversarial transferability via gradient relevance attack," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 4718–4727.
[32] Z. Ge, W. Xiaosen, H. Liu, F. Shang, and Y. Liu, "Boosting adversarial transferability by achieving flat local maxima," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 70141–70161.
[33] J. Zou, Y. Duan, B. Li, W. Zhang, Y. Pan, and Z. Pan, "Making adversarial examples more transferable and indistinguishable," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 3662–3670.

[34] X. Yang, J. Lin, H. Zhang, X. Yang, and P. Zhao, "Improving the transferability of adversarial examples via direction tuning," *Inf. Sci.*, vol. 647, 2023, Art. no. 119491.
[35] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2020.
[36] Y. Long et al., "Frequency domain model augmentation for adversarial attack," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 549–566.
[37] M. Fan, C. Chen, X. Liu, and W. Guo, "MaskBlock: Transferable adversarial examples with Bayes approach," 2022, *arXiv:2208.06538*.
[38] K. Wang, X. He, W. Wang, and X. Wang, "Boosting adversarial transferability by block shuffle and rotation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 24336–24346.
[39] X. Wang, Z. Zhang, and J. Zhang, "Structure invariant transformation for better adversarial transferability," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 4584–4596.
[40] Z. Yuan, J. Zhang, and S. Shan, "Adaptive image transformations for transfer-based adversarial attack," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 1–17.
[41] Z. Ge et al., "Improving the transferability of adversarial examples with arbitrary style transfer," in *Proc. ACM Int. Conf. Multimedia*, 2023, pp. 4440–4449.
[42] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3519–3529.
[43] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie, and S.-N. Lim, "Enhancing adversarial example transferability with an intermediate level attack," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4732–4741.
[44] Z. Wang, H. Guo, Z. Zhang, W. Liu, Z. Qin, and K. Ren, "Feature importance-aware transferable adversarial attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 7619–7628.
[45] J. Zhang et al., "Improving adversarial transferability via neuron attribution-based attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2022, pp. 14973–14982.
[46] Y. Li, H. Qu, and J. Dong, "Enhancing the transferability via feature-momentum adversarial attack," 2022, *arXiv:2204.10606*.
[47] Y. Zhang, Y.-A. Tan, T. Chen, X. Liu, Q. Zhang, and Y. Li, "Enhancing the transferability of adversarial examples with random patch," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 1672–1678.
[48] D. Zheng, W. Ke, X. Li, Y. Duan, G. Yin, and F. Min, "Enhancing the transferability of adversarial attacks via multi-feature attention," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 1462–1474, 2025.
[49] Q. Li, Y. Guo, W. Zuo, and H. Chen, "Improving adversarial transferability via intermediate-level perturbation decay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 32900–32912.
[50] Y. Zhang et al., "Why does little robustness help? a further step towards understanding adversarial transferability," in *Proc. IEEE Symp. Secur. Privacy*, 2024, pp. 3365–3384.
[51] X. Wang, K. Tong, and K. He, "Rethinking the backward propagation for adversarial transferability," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 1905–1922.
[52] W. Cui, X. Li, J. Huang, W. Wang, S. Wang, and J. Chen, "Substitute model generation for black-box adversarial attack based on knowledge distillation," in *Proc. IEEE Int. Conf. Image Process.*, 2020, pp. 648–652.
[53] R. Wang, Y. Guo, and Y. Wang, "AGS: Affordable and generalizable substitute training for transferable adversarial attack," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 5553–5562.
[54] Z. Wei, J. Chen, M. Goldblum, Z. Wu, T. Goldstein, and Y.-G. Jiang, "Towards transferable adversarial attacks on vision transformers," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 2668–2676.
[55] H. Zhou, Y.-A. Tan, Y. Wang, H. Lyu, S. Wu, and Y. Li, "Improving the transferability of adversarial examples with restructure embedded patches," 2022, *arXiv:2204.12680*.
[56] J. Zhang, Y. Huang, W. Wu, and M. R. Lyu, "Transferable adversarial attacks on vision transformers with token gradient regularization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16415–16424.
[57] J. Zhang, Y. Huang, Z. Xu, W. Wu, and M. R. Lyu, "Improving the adversarial transferability of vision transformers with virtual dense connection," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 7133–7141.
[58] D. Yang, W. Yu, Z. Xiao, and J. Luo, "Generating adversarial examples with better transferability via masking unimportant parameters of surrogate model," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2023, pp. 01–08.
[59] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 550–558.
[60] Y. Zhu, J. Sun, and Z. Li, "Rethinking adversarial transferability from a data distribution perspective," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–23.
[61] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11918–11930.
[62] X. Han, A. Liu, C. Yao, Y. Fan, and K. He, "Sampling-based fast gradient rescaling method for highly transferable adversarial attacks," 2023, *arXiv:2307.02828*.
[63] X. Wang, X. He, J. Wang, and K. He, "Admix: Enhancing the transferability of adversarial attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 16138–16147.
[64] X. Wang and K. He, "Enhancing the transferability of adversarial attacks through variance tuning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1924–1933.
[65] Y. Huang and A. W.-K. Kong, "Transferable adversarial attack based on integrated gradients," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–26.
[66] D. Yang, Z. Xiao, and W. Yu, "Boosting the adversarial transferability of surrogate models with dark knowledge," in *Proc. IEEE 35th Int. Conf. Tools Artif. Intell.*, 2023, pp. 627–635.
[67] J. Springer, M. Mitchell, and G. Kenyon, "A little robustness goes a long way: Leveraging robust features for targeted transfer attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 9759–9773.
[68] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1294–1303.
[69] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–14.
[70] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
[71] S. d'Ascoli, H. Touvron, M. L. Leavitt, A. S. Morcos, G. Biroli, and L. Sagun, "ConViT: Improving vision transformers with soft convolutional inductive biases," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 2286–2296.
[72] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 15908–15919.
[73] Z. Chen, L. Xie, J. Niu, X. Liu, L. Wei, and Q. Tian, "VisFormer: The vision-friendly transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 569–578.
[74] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–16.
[75] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Front. Comput. Sci.*, vol. 14, pp. 241–258, 2020.
[76] Z. Zhao, Z. Liu, and M. Larson, "On success and simplicity: A second look at transferable targeted attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 6115–6128.
[77] H. Zeng, B. Chen, and A. Peng, "Enhancing targeted transferability VIA feature space fine-tuning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2024, pp. 4475–4479.
[78] J. Byun, M.-J. Kwon, S. Cho, Y. Kim, and C. Kim, "Introducing competition to boost the transferability of targeted adversarial examples through clean feature mixup," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 24648–24657.
[79] J. Weng, Z. Luo, S. Li, N. Sebe, and Z. Zhong, "Logit margin matters: Improving transferable targeted adversarial attack by logit calibration," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 3561–3574, 2023.
[80] Y. Mo, D. Wu, Y. Wang, Y. Guo, and Y. Wang, "When adversarial training meets vision transformers: Recipes from training to architecture," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 18599–18611.
[81] Z. Wei, Y. Wang, Y. Guo, and Y. Wang, "CFA: Class-wise calibrated fair adversarial training," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 8193–8201.
[82] N. D. Singh, F. Croce, and M. Hein, "Revisiting adversarial training for ImageNet: Architectures, training and generalization across threat models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 13931–13955.
[83] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1310–1320.

[84] Z. Yang, L. Li, X. Xu, B. Kailkhura, T. Xie, and B. Li, "On the certified robustness for ensemble models and beyond," in *Proc. Int. Conf. Learn. Representations*, 2022, *arXiv:2107.10873*.
[85] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2018, pp. 1778–1787.
[86] M. Naseer, S. Khan, M. Hayat, F. S. Khan, and F. Porikli, "A self-supervised approach for adversarial robustness," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2020, pp. 259–268.
[87] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–17.
[88] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 274–283.
[89] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: Removing noise by adding noise," 2017, *arXiv:1706.03825*.
[90] A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," 2023, *arXiv:2307.15043*.
[91] Z. Wei, Y. Wang, and Y. Wang, "Jailbreak and guard aligned language models with only few in-context demonstrations," *IEEE Trans. Pattern Anal. Mach. Intell.*, Feb. 02, 2026, doi: 10.1109/TPAMI.2026.3660147.
[92] Y. Mo, Y. Wang, Z. Wei, and Y. Wang, "Fight back against jailbreaking via prompt adversarial tuning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 64242–64272.
[93] W.-L. Chiang et al., "VICUNA: An open-source chatbot impressing GPT-4 with 90% ChatGPT quality," Mar. 2023. [Online]. Available: https://vicuna.lmsys.org
[94] M. N. Team, "Introducing MPT-7B: A new standard for open-source, commercially usable LLMs," 2023. Accessed: May 5, 2023. [Online]. Available: www.mosaicml.com/blog/mpt-7b
[95] S. Biderman et al., "Pythia: A suite for analyzing large language models across training and scaling," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 2397–2430.
[96] "stable-vicuna-13b-delta," 2023. [Online]. Available: https://huggingface.co/CarperAI/stable-vicuna-13b-delta
[97] F. Croce et al., "RobustBench: A standardized adversarial robustness benchmark," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1–17.
[98] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry, "Do adversarially robust imagenet models transfer better?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 3533–3545.

Yisen Wang received the PhD degree from Tsinghua University, in 2018. He is currently an assistant professor with Peking University. His research interests include machine learning and deep learning, such as adversarial learning, graph learning, and weakly/self-supervised learning.

Yichuan Mo received the BE degree from Shanghai Jiao Tong University, in 2022. He is currently working toward the PhD degree with Peking University. His research interests include adversarial learning, model robustness, and trustworthy AI.

Dongxian Wu received the PhD degree from Tsinghua University, in 2021. His research interests include trustworthy machine learning, especially adversarial learning, and data security.

Mingjie Li received the PhD degree from Peking University, in 2023. His research interests include trustworthy machine learning, such as adversarial robustness, privacy, and data security.

Xingjun Ma (Member, IEEE) received the PhD degree from the University of Melbourne, in 2019. He is currently an associate professor with Fudan University. His research focuses on trustworthy machine learning.

Zhouchen Lin (Fellow, IEEE) received the PhD degree from Peking University, in 2000. He is currently a professor with Peking University. His research interests include computer vision, image processing, machine learning, pattern recognition, and numerical optimization. He is an associate editor-in-chief of *IEEE Transactions on Pattern Analysis and Machine Intelligence* and an associate editor of the *International Journal of Computer Vision*. He is a Fellow of IAPR.