Supplementary Material: Boosting Black-Box Attack with Partially Transferred Conditional Adversarial Distribution

Content Outline

This is the supplementary material of CVPR 2022 paper, "Boosting Black-Box Attack with Partially Transferred Conditional Adversarial Distribution". The content of this manuscript will be organized as following:

- In Section 1, we provide a detailed description of c-Glow model.
- In Section 2, we provide the verification and discussion for the assumptions and claims, including: the verification for **Assumption 1** in the main manuscript; the verification that the energy-based model is suitable for modeling the conditional adversarial distribution (CAD); the comparison between CAD and marginal distribution; the comparison between c-Glow model and Gaussian for approximating the CAD.
- In Section 3, we present the proof of **Theorem 1** in the main manuscript.
- In Section 4, a description to the basic algorithm of $\mathcal{CG}$-ATTACK, *i.e.*, CMA-ES, will be given.
- In Section 5, we present the details of the DCT downsampling method.
- In Section 6, we present a detailed description of our experimental settings.
- In Section 7, additional results of targeted attacks on both CIFAR-10 and ImageNet, additional results of **Case 1** and **Case 2** open-set attacks on ImageNet, as well as the ablation studies on the impact of DCT downsampling, the basic search algorithms, the depth of c-Glow and the ratio of transferred parameters are provided.
- In Section 8, We also discuss the the computation complexity, limitations, possible defenses and potential negative social effects of $\mathcal{CG}$-ATTACK.

1. Details of the c-Glow Model

c-Glow [18] is a conditional flow-based generative model based on a multi-scale architecture, which could model the distribution of target variables from coarse to fine scales, and each scale consists of a series of flow steps. As shown in the main manuscript, the c-Glow model [18] plays an important role to capture the distribution of adversarial perturbations in the proposed method. For clarity, here We repeat the generic formulation of the c-Glow model, as follows

$$\boldsymbol{\eta} = g_{\boldsymbol{x},\boldsymbol{\phi}}(\boldsymbol{z}) = g_{\boldsymbol{x},\boldsymbol{\phi}_1}(g^{-1}_{\boldsymbol{x},\boldsymbol{\phi}_2}(...(g_{\boldsymbol{x},\boldsymbol{\phi}_M}(\boldsymbol{z}))...)), \tag{1}$$

$$\boldsymbol{z} = g^{-1}_{\boldsymbol{x},\boldsymbol{\phi}}(\boldsymbol{\eta}) = g^{-1}_{\boldsymbol{x},\boldsymbol{\phi}_M}(g_{\boldsymbol{x},\boldsymbol{\phi}_{M-1}}(...(g^{-1}_{\boldsymbol{x},\boldsymbol{\phi}_1}(\boldsymbol{\eta}))...)). \tag{2}$$

To facilitate the understanding of our method, we repeat the detailed mathematical definition of $g^{-1}_{\boldsymbol{x},\boldsymbol{\phi}}(\cdot)$ and the graphical structure of the whole model from [18].

1.1. Glow Step

The glow step is the basic component in the c-Glow model. As shown in Fig. 1(left), it consists of three layers: a conditional actnorm layer, a conditional 1×1 convolutional layer and a conditional affine coupling layer Here we use $\boldsymbol{x}, \boldsymbol{v}^{in}, \boldsymbol{v}^{out} \in \mathbb{R}^{h \times w \times c}$ to denote the conditional variable (*i.e.* the benign input in the adversarial attack problem), input variable and output variable, respectively, where h, w are spatial dimensions and c is the number of channels. We denote spatial indices of tensors as (i, j). The details about these three layers are presented as follows.

Conditional actnorm layer. The conditional activation normalization (actnorm) layer performs an affine transformation on the input tensor, where the transformation parameters $\boldsymbol{s} \in \mathbb{R}^{1\times c}$ and $\boldsymbol{b} \in \mathbb{R}^{1\times c}$ are the outputs of a simple conditioning network (CN) w.r.t. the conditional variable $\boldsymbol{x}$, *i.e.*

$$\boldsymbol{v}_{i,j}^{out} = \boldsymbol{s} \odot \boldsymbol{v}_{i,j}^{in} + \boldsymbol{b},\ \forall i, j,\ \text{where } \boldsymbol{s}, \boldsymbol{b} = \mathbf{CN}_1(\boldsymbol{x}). \tag{3}$$

Conditional 1×1 **convolutional layer**. This layer aims to permute the input tensor along the channel dimensions, using a weight matrix $\boldsymbol{W} \in \mathbb{R}^{c\times c}$, which is also the output of a CN w.r.t. $\boldsymbol{x}$, *i.e.*

$$\boldsymbol{v}_{i,j}^{out} = \boldsymbol{W}\boldsymbol{v}_{i,j}^{in},\ \forall i, j,\ \text{where } \boldsymbol{W} = \mathbf{CN}_2(\boldsymbol{x}). \tag{4}$$

Conditional affine coupling layer. This powerful and computationally efficient layer was firstly introduced in [5]. It first splits the input tensor along channel dimension into $\boldsymbol{v}_1^{in}$ and $\boldsymbol{v}_2^{in}$. The latent representation $\boldsymbol{x}_l$ is extracted from $\boldsymbol{x}$ with CN. $\boldsymbol{x}_r$ is then concatenated with $\boldsymbol{v}_1^{in}$ to generate scale and bias parameters for the affine transformation of $\boldsymbol{v}_2^{in}$ with a neural network (NN). The affine transformed variable $\boldsymbol{v}_2^{out}$ is then concatenated with $\boldsymbol{v}_1^{in}$ to form the final output. These operations can be formulated as follows

$$\begin{cases} \boldsymbol{v}_1^{in}, \boldsymbol{v}_2^{in} = \mathbf{split}(\boldsymbol{v}^{in}),\ \boldsymbol{x}_l = \mathbf{CN}_3(\boldsymbol{x}), \\ \boldsymbol{s}_2, \boldsymbol{b}_2 = \mathbf{NN}(\boldsymbol{v}_1^{in}, \boldsymbol{x}_l),\ \boldsymbol{v}_2^{out} = \boldsymbol{s}_2 \odot \boldsymbol{v}_2^{in} + \boldsymbol{b}_2 \\ \boldsymbol{v}^{out} = \mathbf{concat}(\boldsymbol{v}_1^{in}, \boldsymbol{v}_2^{out}). \end{cases} \tag{5}$$

Besides, we refer the readers to [18] for the specifications of the simple neural networks $\mathbf{CN}_1(\cdot), \mathbf{CN}_2(\cdot), \mathbf{CN}_3(\cdot)$ and $\mathbf{NN}(\cdot)$ used in above equations.

1.2. Muti-scale Architecture

As shown in Fig. 1(right), multiple glow steps are combined with a multi-scale architecture [5]. The whole model consists of M blocks (corresponding to M functions $g^{-1}_{\boldsymbol{x},\boldsymbol{\phi}_i}(\cdot)$ in Eq. (2)), with different scales. Each block starts with a squeeze operation, followed by N glow steps as defined above, and ends with a split operation. The squeeze operation divide the input tensor into sub-squares of size $2 \times 2 \times c$ and reshape them into size $1 \times 1 \times 4c$, such that a $h \times w \times c$-shaped tensor will be transformed to the shape of $\frac{h}{2} \times \frac{w}{2} \times 4c$. The split operation splits the input tensor into two halves along the channel dimensions. One half serves as the immediate output (*i.e.* $\boldsymbol{z}_i$ in Fig. 1(right)), while the other half is fed into the next block for further transformation. Consequently, the size of input tensors is reduced by half per block, significantly reducing computation and memory costs of the whole model. Note that there is no split operation in the last block. Therefore, the output shape of $\boldsymbol{z}_M$ is $\frac{h}{2^{M-1}} \times \frac{w}{2^{M-1}} \times 2^{M-1}c$. Finally, all immediate outputs $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$ are reshaped as one dimensional vectors and concatenated to the output variable $\boldsymbol{z}$, which is assumed to be Gaussian-distributed. Following [18], M and N in this work are also set as 3 and 8, respectively.

2. Verification and Discussion

2.1. Verification of Assumption 1.

On the basis of Assumption 1, to evaluate the similarity of two mapping parameters, we propose to use the ASR against the target model by the query input, which is sampled from the c-Glow model learned from surrogate models. We conduct four experiments on the CIFAR-10 dataset over four pre-trained DNN models described in Section 4.2 of the main submission, *i.e.*, ResNet, DenseNet, VGG, and PyramidNet. For each experiment, we pick one DNN model as the target while the others as surrogates. Then, for each test image, we sample a single perturbation from the pre-trained c-Glow model and report the ASR against the target model. As shown in Table 1, the high ASR values reveal that the CADs between the target and surrogate models are similar. To further demonstrate the similarity between CADs of target and surrogate models, we calculate the KL divergence between the CADs approximated by the corresponding c-Glow models (with the same base Gaussian distribution). More specifically, we again train four c-Glow models based on single targeted DNN models, and approximate the symmetric KL divergence between the c-Glow model trained on the target model and the c-Glow model trained on other three surrogate

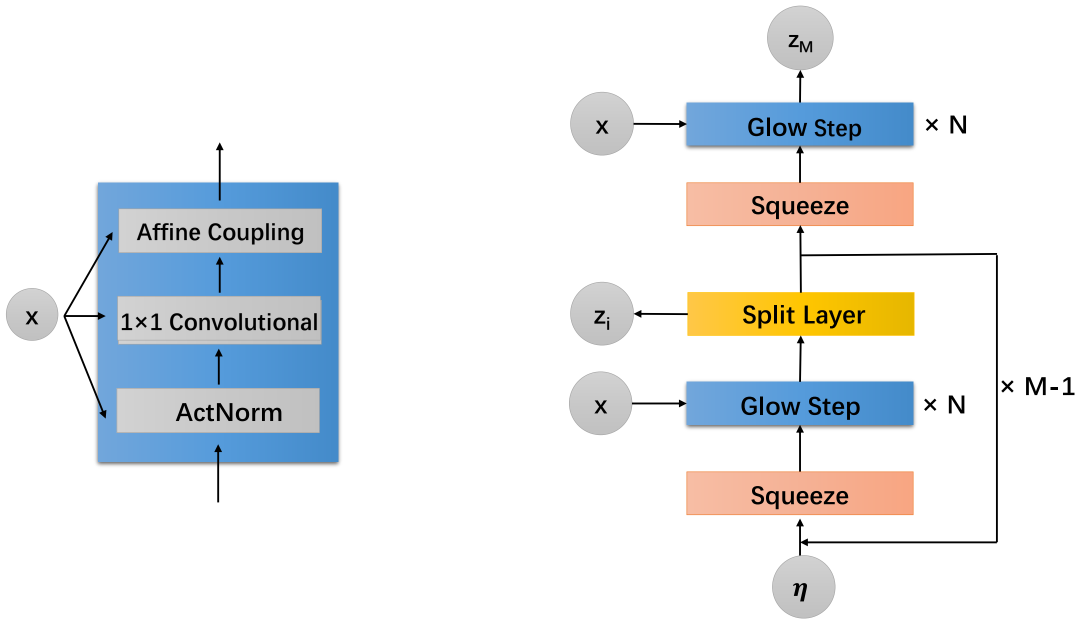

Figure 1. c-Glow architecture. Each glow step consists of conditional actnorm layer, conditional 1×1 convolutional layer and a conditional affine coupling layer **(left)**. The glow steps are combined in a multi-scale architecture **(right)**. $\boldsymbol{z}$ in (right) refers to the latent variable described in section 4.1 of the main manuscript.

Table 1. ASR for queries sampled from pretrained c-Glow models.

Target $\rightarrow$	ResNet	DenseNet	VGG	PyramidNet
ASR %	74.9	84.1	84.5	92.1

Table 2. Symmetric KL divergence for the target and surrogate c-Glow models. The first row shows results for the original c-Glow models and second shows results for c-Glows models with small random noises.

Target $\rightarrow$	ResNet	DenseNet	VGG	PyramidNet
Original	67.3	91.0	48.2	78.7
Noisy	224.5	329.7	293.7	386.5

models. More specifically, we sample $N = 10000$ perturbations $\{\boldsymbol{\eta}_i\}_{i=1}^{N}$ from each of the c-Glow models, and approximate the symmetric KL divergence based on the definition in Eq. (6) of main manuscript [1]. For comparison, we add uniform random noises to the c-Glow model parameters (the upper bound of the random noises is set as 1% of the maximum value of the parameters for each layer), and evaluate how this affect the KL divergence. From Tab. 2, we can see that the KL divergence between target CADs and surrogate CADs are indeed small, and will raise significantly when adding small random noises to the model parameters. The above results implies the mapping parameters ϕ are similar.

2.2. Verification of the Energy-based Model for the Perturbation Distribution

In this section, we try to experimentally verify that the energy-based model of Eq. (5) in Section 3.2.2 of the main submission is a suitable parametric model for the perturbation distribution. There are two main challenges for such a verification. **1) The analytical form of the real ground-truth perturbation distribution $\mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$ is unknown.** To tackle this, we propose to adopt a non-parametric estimation of $\mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$. Specifically, given one benign example $\boldsymbol{x}$, we firstly collect a set of 10^5 adversarial perturbations that successfully fool the surrogate model $\mathcal{F}_s$ [2], and then adopt the kernel density estimation

[1]For distribution P and Q, we defined KL divergence $D_{KL}(P||Q)$ in Eq. (6). The symmetric form of KL divergence is defined as $D_{KL}(P||Q) + D_{KL}(Q||P)$

[2]We uniformly sample perturbations within the ϵ-ball and collect ones that successfully fool the surrogate model, until the number of successful perturbations reached 10^5.

(KDE) [22] to obtain the non-parametric estimation of $\mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$ based on these 10^5 perturbations. **2)The value of λ in Eq. (10) cannot be accurately estimated**. It then makes no sense to check whether the values of $\log \mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$ and $-\lambda \cdot \mathcal{L}_{adv,s}^u(\boldsymbol{\eta}, \boldsymbol{x})$ are equivalent or not. However, Eq. (10) implies that $\log \mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x}) \propto -\lambda \cdot \mathcal{L}_{adv,s}^u(\boldsymbol{\eta}, \boldsymbol{x})$, *i.e.* the proportional relationship. Thus, we propose to adopt the pairwise consistency as a substitute measure of this relationship. Specifically, let $\boldsymbol{\eta_1}$ and $\boldsymbol{\eta_2}$ be two adversarial perturbations sampled randomly from $\mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$, then we define

$$\alpha(\boldsymbol{\eta_1}, \boldsymbol{\eta_2}|\boldsymbol{x}) = \mathbb{I}_{\left\{(\log \mathcal{P}_s^u(\boldsymbol{\eta_1}|\boldsymbol{x}) - \log \mathcal{P}_s^u(\boldsymbol{\eta_2}|\boldsymbol{x}))\cdot(\lambda\cdot\mathcal{L}_{adv,s}^u(\boldsymbol{\eta_2},\boldsymbol{x}) - \lambda\cdot\mathcal{L}_{adv,s}^u(\boldsymbol{\eta_1},\boldsymbol{x}))>0\right\}},$$

where $\mathbb{I}$ is the indicator function. If $\alpha(\boldsymbol{\eta_1}, \boldsymbol{\eta_2}|\boldsymbol{x}) = 1$, then $\boldsymbol{\eta_1}$ and $\boldsymbol{\eta_2}$ are pairwisely consistent between $\log \mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$ and $-\lambda \cdot \mathcal{L}_{adv,s}^u(\boldsymbol{\eta}, \boldsymbol{x})$, and otherwise inconsistent. We then evaluate the global consistency in terms of $a(\boldsymbol{\eta_1}, \boldsymbol{\eta_2}|\boldsymbol{x})$, as follows

1. Randomly sampling d pairs of adversarial perturbations $\{(\boldsymbol{\eta}_1^i, \boldsymbol{\eta}_2^i)\}_{i=1}^d$ from the KDE model;
2. Estimating the values of $\mathcal{P}_s^u(\boldsymbol{\eta}_1^i|\boldsymbol{x})$ and $\mathcal{P}_s^u(\boldsymbol{\eta}_2^i|\boldsymbol{x})$ for each pair by the KDE model;
3. Evaluating the values of $\mathcal{L}_{adv,s}^u(\boldsymbol{\eta}_1^i, \boldsymbol{x})$ and $\mathcal{L}_{adv,s}^u(\boldsymbol{\eta}_2^i, \boldsymbol{x})$ by Eq. (12);
4. Calculating the value of $\alpha^i = \alpha(\boldsymbol{\eta}_1^i, \boldsymbol{\eta}_2^i|\boldsymbol{x})$ for $i = 1, 2, \cdots, d$.

Let $S = \left\{s_i | s_i = \left|\log \mathcal{P}_s^u(\boldsymbol{\eta_1^i}|\boldsymbol{x}) - \log \mathcal{P}_s^u(\boldsymbol{\eta_2^i}|\boldsymbol{x})\right|\right\}_{i=1}^d$. We further rank the $\{\alpha^i\}_{i=1}^d$ in descending order according to their corresponding values in S, leading to $\{\alpha^{[i]}\}_{i=1}^d$ with $\alpha^{[i]}$ being the α w.r.t. the i-th largest value in S. Let $p_l = \frac{1}{l}\sum_{i=1}^l \alpha^{[i]}, l = 1, \cdots, d$. The larger the p_l indicates the higher global consistency between $\log \mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$ and $-\lambda \cdot \mathcal{L}_{adv,s}^u(\boldsymbol{\eta}, \boldsymbol{x})$.

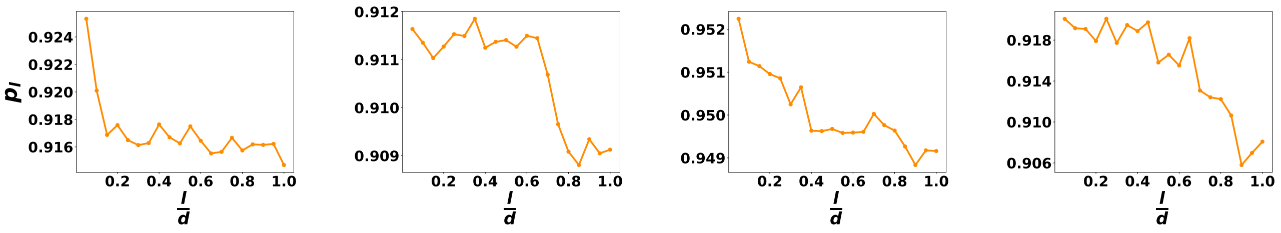

Figure 2. Tendency curves of p_l w.r.t $\frac{l}{d}$ for four randomly sampled benign images from CIFAR-10. The ground-truth classes of the four images (from left plot to right plot) are 0 (plane), 1 (automobile), 2 (bird) and 5 (dog), respectively.

We experimentally set $d = 10^5$ and randomly choose four benign examples from CIFAR-10. Fig. 2 plots the curves of p_l v.s. $\frac{l}{d}$. As shown in the figure, all values of p_l at different $\frac{l}{d}$ are larger than 0.9, which demonstrates the high global consistency. Note that p_l generally decreases along with the increase of $\frac{l}{d}$. The possible reason is that when $\boldsymbol{\eta}_1^i$ and $\boldsymbol{\eta}_2^i$ are too close (*i.e.* s_i is small), the inaccuracy of KDE may lead to the locally pairwise inconsistency between $\log \mathcal{P}_s^u(\boldsymbol{\eta}|\boldsymbol{x})$ and $-\lambda \cdot \mathcal{L}_{adv,s}^u(\boldsymbol{\eta}, \boldsymbol{x})$. **In summary**, above empirical studies demonstrate that the energy-based model is a suitable parametric model for capturing the perturbation distribution.

2.3. The *CAD* $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})$ *vs.* the marginal distribution $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta})$.

In this work, we adopt the c-Glow model to approximate the *CAD* $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})$, rather than the marginal distribution $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta})$. We believe that if there is a marginal adversarial distribution $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta})$ independent of $\boldsymbol{x}$, then the sampled adversarial perturbation for one benign example should be likely to be adversarial for other benign examples.

To verify it, we conduct the transfer attack by adopting ResNet-110 as the target model and randomly selecting 100 test images from the CIFAR-10 dataset. Then, we uniformly draw perturbations from the l_∞ ball with the radius of 0.03125. For each image, we keep 10000 adversarial perturbations. As shown in Fig. 3(**left**), the low success rates of transfer attack for most adversarial perturbations reveal that most adversarial perturbations are specific to benign examples, rather than agnostic. Thus, we conclude that approximating $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})$ is better than approximating $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta})$.

2.4. c-Glow *vs.* Gaussian distribution.

Here we verify that the *CAD* can be better approximated by the c-Glow model than the Gaussian model used in [17], by setting the target model, test images, and perturbations following the same description in the above paragraph. Specifically, we adopt kernel density estimation (KDE) [22] to obtain a non-parametric estimation of the *CAD* for each test image, *i.e.*,

$\mathcal{P}_{KDE}(\boldsymbol{\eta}|\boldsymbol{x})$. We also perform the NES attack over each test image to optimize the Gaussian model for approximating the *CAD*, noted as $\mathcal{P}_G(\boldsymbol{\eta}|\boldsymbol{x})$. We sample $N = 10000$ perturbations $\{\boldsymbol{\eta}_i\}_{i=1}^N$ from $\mathcal{P}_{KDE}(\boldsymbol{\eta}|\boldsymbol{x})$ for each image, and approximate the KL divergence KL$_{KDE-G}$ (refer to the definition in Eq. (6) of main manuscript) based on these perturbations. We also calculate the KL divergence between the KDE model and c-Glow.

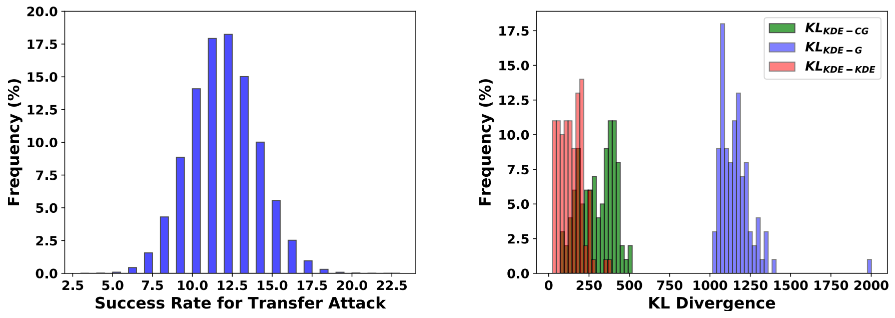

Figure 3. **Left**: The success rate of transfer attacks. **Right**: The KL divergences between two *CADs* approximated by different models.

As shown in Fig. 3(**right**), each KL divergence is represented by a histogram, and KL$_{KDE-CG}$ (green) is much smaller than KL_{KDE-G} (blue), demonstrating that the c-Glow obtains a closer *CAD* to KDE than the Gaussian model. To further measure how close the *CAD*s between c-Glow and KDE, we repeat the perturbation generation process and produce another KDE, then we compute the KL divergence between two KDE models, represented by the red histogram in Fig. 3(**right**). We assume that the KL divergence between these two KDE distributions is small, which can be served as the baseline in the comparison. The green histogram KL$_{KDE-CG}$ is very close to the red one, and the Wasserstein distance [24] between them is 158.12. It demonstrates that the *CAD* gap between KDE and c-Glow is very small, implying that the *CAD* approximated by c-Glow is very close to the *real CAD*. In contrast, the Wasserstein distance between the blue histogram KL$_{KDE-CG}$ and the red one is up to 1019.78. The results shows that c-Glow can not only give a much better approximation to the *real CAD* than the Gaussian model, but also give a very good approximation.

3. Proof of Theorem 1

In this section, we present the proof of Theorem 1 in the main manuscript. Before starting our proof, Definition 1 and Lemma 1 proposed from [26] and [15] should be firstly introduced.

Definition 1 ([15]). *Given a differentiable vector function $h(\boldsymbol{x}) : \mathbb{R}^k \to \mathbb{R}^k$, its divergence is denoted as $\nabla \cdot h(\boldsymbol{x})$, and the definition is*

$$\nabla \cdot h(\boldsymbol{x}) := \sum_{j=1}^{k} \frac{\partial [h(\boldsymbol{x})]_j}{\partial [\boldsymbol{x}]_j}, \tag{6}$$

where $[\boldsymbol{x}]_j$ indicates the j-th entry of $\boldsymbol{x}$. And, it satisfies the following property,

$$\int \nabla \cdot h(\boldsymbol{x}) d\boldsymbol{x} = 0 \tag{7}$$

for any vector function $h(\boldsymbol{x})$ such that $h(\infty) = 0$. Similarly, we can define a differentiable function $w(\boldsymbol{x}) = [w_1(\boldsymbol{x}), \ldots, w_l(\boldsymbol{x})] : \mathbb{R}^k \to \mathbb{R}^{k\times l}$, and its divergence can be represented as $\nabla \cdot w(\boldsymbol{x}) = [\nabla \cdot w_1(\boldsymbol{x}), \ldots, \nabla \cdot w_l(\boldsymbol{x})]$.

Lemma 1 ([26]). *Using the notations and definitions in Theorem 1, we have*

$$\nabla_{\boldsymbol{\theta}} \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) = -\nabla_{\boldsymbol{\eta}} \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) - \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) \nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}), \tag{8}$$

where $h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) = \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z})\big|_{\boldsymbol{z}=g^{-1}_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{\eta})}$.

Proof of Lemma 1. We firstly define a small change $\boldsymbol{\Delta}_i = \delta \boldsymbol{e}_i \in \mathbb{R}^d$, where $d = |\boldsymbol{\theta}|$, $\boldsymbol{e}_i \in \{0,1\}^d$ and only the i-th entry is 1, and δ is a sufficiently small scalar. Recall that $\boldsymbol{\eta} = g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z})$. Accordingly, we can define a new perturbation $\boldsymbol{\eta}' = g_{\boldsymbol{x},\boldsymbol{\theta}+\boldsymbol{\Delta}_i}(\boldsymbol{z})$, and meanwhile, $\boldsymbol{\eta}'$ can be represented by a simple transformation from $\boldsymbol{\eta}$, *i.e.*

$$\boldsymbol{\eta}' = \boldsymbol{\eta} + h_{\boldsymbol{\theta}}(\boldsymbol{\eta})\boldsymbol{\Delta}_i + o(\delta), \tag{9}$$

where $h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) : \mathbb{R}^n \to \mathbb{R}^{n\times d}$ (see Definition 1) will be specified later, and $o(\delta)$ indicates a small constant similar to δ. Then, the probability density function of $\boldsymbol{\eta}'$ is formulated as

$$\begin{aligned}
\mathcal{P}_{\boldsymbol{\theta}+\boldsymbol{\Delta}_i}(\boldsymbol{\eta}'|\boldsymbol{x}) &= \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})|\det(d\boldsymbol{\eta}'/d\boldsymbol{\eta})|^{-1} \\
&= \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})|\det(\mathbf{I} + \frac{d(h_{\boldsymbol{\theta}}(\boldsymbol{\eta}))}{d\boldsymbol{\eta}}\boldsymbol{\Delta}_i + o(\delta))|^{-1} \\
&= \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})(1 + \boldsymbol{\Delta}_i^\top \nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) + o(\delta))^{-1} \\
&= \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})(1 - \boldsymbol{\Delta}_i^\top \nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) + o(\delta)) && (10) \\
&= \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) - \boldsymbol{\Delta}_i^\top \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}'|\boldsymbol{x})\nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}') + o(\delta) && (11) \\
&= \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}'|\boldsymbol{x}) - \boldsymbol{\Delta}_i^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta}')^\top \cdot \nabla_{\boldsymbol{\eta}'}\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}'|\boldsymbol{x}) - \boldsymbol{\Delta}_i^\top \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}'|\boldsymbol{x})\nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}') + o(\delta). && (12)
\end{aligned}$$

The first equality utilizes the change of variables [28] for probability densities, from $\boldsymbol{\eta}$ to $\boldsymbol{\eta}'$. The second equality adopts Eq. (9). The third equality uses the definition of determinant and Definition 1. The fourth equality is derived based on the Taylor expansion that $(1+\xi)^{-1} = 1 - \xi + o(\xi)$ with $\xi = \boldsymbol{\Delta}_i^\top \nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta})$. (11) follows from the fact that $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}'|\boldsymbol{x}) = \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) + o(1)$ and $\nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}') = \nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) + o(1)$. (12) utilizes $\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) = \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}'|\boldsymbol{x}) - (\boldsymbol{\eta}' - \boldsymbol{\eta})^\top \cdot \nabla \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}'|\boldsymbol{x}) + o(\delta)$.

Since $\boldsymbol{\eta}'$ is arbitrary, the above equation implies that

$$\mathcal{P}_{\boldsymbol{\theta}+\boldsymbol{\Delta}_i}(\boldsymbol{\eta}|\boldsymbol{x}) = \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) - \boldsymbol{\Delta}^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta})^\top \cdot \nabla_{\boldsymbol{\eta}}\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) - \boldsymbol{\Delta}^\top \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})\nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) + o(\|\delta\|)$$

for all $\boldsymbol{\eta} \in \mathbb{R}^n$ and $i = 1, \ldots, d$. Further, if we take $\delta \to 0$, then Eq. (8) is proved.

□

Theorem 1. *Utilizing the definition $\boldsymbol{\eta} = g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_0)$ and $\boldsymbol{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and defining the term $D(\boldsymbol{\eta}, \boldsymbol{x}) = \log \frac{\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x})}{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})}$, then the gradient of $\mathcal{L} = \int_{\boldsymbol{\eta}} \mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}) \log \frac{\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x})}{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})} d\boldsymbol{\eta}$ w.r.t. $\boldsymbol{\theta}$ is computed as follows*

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L} &= -\mathbb{E}_{\boldsymbol{z}_0\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\exp^{D(\boldsymbol{\eta},\boldsymbol{x})} \cdot \nabla_{\boldsymbol{\eta}} D(\boldsymbol{\eta}, \boldsymbol{x})^\top\big|_{\boldsymbol{\eta}=g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_0)} \cdot \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_{\mathbf{0}})\right], && (13) \\
&= -\mathbb{E}_{\boldsymbol{z}_0\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\frac{\exp^{-\lambda\cdot\mathcal{L}_{adv,s}(\boldsymbol{\eta},\boldsymbol{x})}}{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})} \cdot \nabla_{\boldsymbol{\eta}} D(\boldsymbol{\eta}, \boldsymbol{x})^\top\big|_{\boldsymbol{\eta}=g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_0)} \cdot \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_{\mathbf{0}})\right],
\end{aligned}$$

where $\nabla_{\boldsymbol{\eta}} D(\boldsymbol{\eta}, \boldsymbol{x}) = \nabla_{\boldsymbol{\eta}}\big[- \lambda \cdot \mathcal{L}_{adv,s}(\boldsymbol{\eta}, \boldsymbol{x}) - \log \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) \big]$.

Proof. We firstly define $\ell(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\theta}(\boldsymbol{\eta}|\boldsymbol{x})) = \mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}) \log \frac{\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x})}{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})}$. Then, the loss function $\mathcal{L}$ (see Eq. (11) in section 4.2.2 of the main manuscript) can be rewritten as

$$\mathcal{L} = \int_{\boldsymbol{\eta}} \ell(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\theta}(\boldsymbol{\eta}|\boldsymbol{x}))d\boldsymbol{\eta}. \tag{14}$$

We also denote the gradient of $\ell(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\theta}(\boldsymbol{\eta}|\boldsymbol{x}))$ as $\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\theta}(\boldsymbol{\eta}|\boldsymbol{x})) = \frac{\partial \ell(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\theta}(\boldsymbol{\eta}|\boldsymbol{x}))}{\partial \mathcal{P}_{\theta}(\boldsymbol{\eta}|\boldsymbol{x})} = -\frac{\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x})}{\mathcal{P}_{\theta}(\boldsymbol{\eta}|\boldsymbol{x})}$. Using the chain rule and Lemma 1, we have

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}}\ell(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})) \\
=&\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))\nabla_{\theta}\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}) \\
=&\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))\left[-\nabla_{\boldsymbol{\eta}}\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) - \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})\nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta})\right] \\
=&\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})\nabla_{\boldsymbol{\eta}}\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) - \nabla_{\boldsymbol{\eta}} \cdot \left[\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})h_{\boldsymbol{\theta}}(\boldsymbol{\eta})\right], && (15)
\end{aligned}$$

where the third equality is obtained by applying the product rule as follows

$$\begin{aligned}\nabla_{\boldsymbol{\eta}} \cdot [\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})h_{\boldsymbol{\theta}}(\boldsymbol{\eta})] &= \ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})\nabla \cdot h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) \\ &+ \ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))\nabla_{\boldsymbol{\eta}}\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) \\ &+ \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})\nabla_{\boldsymbol{\eta}}\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta}). \end{aligned} \tag{16}$$

By integrating (15) over $\boldsymbol{\eta}$, and using the fact that $\int_{\boldsymbol{\eta}} \nabla \cdot f(\boldsymbol{\eta}) d\boldsymbol{\eta} = \mathbf{0}$ with $f(\boldsymbol{\eta}) = \ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})h_{\boldsymbol{\theta}}(\boldsymbol{\eta})$, we have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L} &= \int_{\boldsymbol{\eta}} \nabla_{\boldsymbol{\theta}}\ell(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))d\boldsymbol{\eta} \\ &= \int_{\boldsymbol{\eta}} \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})\nabla_{\boldsymbol{\eta}}\ell_2'(\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x}), \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x}))^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta})d\boldsymbol{\eta} \\ &= \int_{\boldsymbol{\eta}} \mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})\nabla_{\boldsymbol{\eta}}(-\frac{\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x})}{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})})^\top h_{\boldsymbol{\theta}}(\boldsymbol{\eta})d\boldsymbol{\eta}. \end{aligned} \tag{17}$$

Further, through reparametrization, and utilizing $D(\boldsymbol{\eta}, \boldsymbol{x}) = \log \frac{\mathcal{P}_s(\boldsymbol{\eta}|\boldsymbol{x})}{\mathcal{P}_{\boldsymbol{\theta}}(\boldsymbol{\eta}|\boldsymbol{x})}$ and $h_{\boldsymbol{\theta}}(\boldsymbol{\eta}) = \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z})\big|_{\boldsymbol{z}=g^{-1}_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{\eta})}$ (see Lemma 1), we obtain

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L} &= -\mathbb{E}_{\boldsymbol{z}_0\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\nabla_{\boldsymbol{\eta}} \exp^{D(\boldsymbol{\eta},\boldsymbol{x})}\big|^\top_{\boldsymbol{\eta}=g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_0)} \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_0)\right] \\ &= -\mathbb{E}_{\boldsymbol{z}_0\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\exp^{D(\boldsymbol{\eta},\boldsymbol{x})} \cdot \nabla_{\boldsymbol{\eta}} D(\boldsymbol{\eta}, \boldsymbol{x})^\top\big|_{\boldsymbol{\eta}=g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_0)} \cdot \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{x},\boldsymbol{\theta}}(\boldsymbol{z}_0)\right]. \end{aligned} \tag{18}$$

□

4. CMA-ES for Black-box Optimization

One promising approach for the black-box optimization is evolutionary strategies (ES) [23]. The main idea is introducing a search distribution $\boldsymbol{\pi}$ to sample some perturbations $\boldsymbol{\eta}$ to obtain the better values of the black-box objective function, *i.e*. the smaller $\mathcal{L}_{adv}$ in the score-based black-box adversarial attack problem. Many variants of ES have been developed, such as natural ES (NES) [29, 30], co-variance matrix adaptation ES (CMA-ES) [10], self-adaptation ES (SA-ES) [11, 25], *etc*. The main difference among these variants is the update step of the search distribution $\boldsymbol{\pi}$. Among these variants, CMA-ES has been considered as one of the state-of-the-art variants in ES, especially for the optimization problem in high-dimensional space.

The basic idea of CMA-ES is to update the parameters of $\boldsymbol{\pi}$ by maximizing the weighted average of log-likelihoods $\sum_{i=1}^m w_i \log \mathcal{P}_{\boldsymbol{\pi}}(\boldsymbol{\eta}_{i:k})$, where $\log \mathcal{P}_{\boldsymbol{\pi}}(\boldsymbol{\eta})$ denotes the log-likelihood of $\boldsymbol{\eta}$ from the distribution $\boldsymbol{\pi}$, where $m, w_i, \boldsymbol{\eta}_{i:k}$ will be defined soon later. Consequently, it is more likely to sample perturbations of better values of the objective function, *i.e*. lower values of $\mathcal{L}_{adv}(\cdot, \boldsymbol{x})$. The search distribution $\boldsymbol{\pi}$ used in CMA-ES is set to Gaussian, *i.e*. $\boldsymbol{\pi} := \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \cdot \boldsymbol{C})$. Specifically, the *Update* step consists the following sequential parts:

• *Update* $\boldsymbol{\mu}$:

$$\boldsymbol{\mu}' = \boldsymbol{\mu},\ \boldsymbol{\mu} \leftarrow \sum_{i=1}^m w_i \cdot \boldsymbol{\eta}_{i:k}, \tag{19}$$

where $\boldsymbol{\eta}_{i:k}$ indicates the i-th best perturbation out of k sampled perturbations, *i.e*. $\mathcal{L}_{adv}(\boldsymbol{\eta}_{1:k}, \boldsymbol{x}) \leq \mathcal{L}_{adv}(\boldsymbol{\eta}_{2:k}, \boldsymbol{x}) \leq \ldots \mathcal{L}_{adv}(\boldsymbol{\eta}_{k:k}, \boldsymbol{x})$, and $m \leq k, \sum_{i=1}^m w_i = 1$ are hyper-parameters.

• *Update* σ:

$$\begin{cases} \boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\, \boldsymbol{C}^{-\frac{1}{2}}(\frac{\boldsymbol{\mu}-\boldsymbol{\mu}'}{\sigma}), \\ \sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right), \end{cases} \tag{20}$$

where $\mathbb{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| = \sqrt{2}\Gamma(\frac{n+1}{2})/\Gamma(\frac{n}{2})$ with $\Gamma(\cdot)$ being the gamma function [4].

• *Update $\boldsymbol{\mathcal{C}}$*:

$$\begin{cases} \boldsymbol{p}_c \leftarrow (1-c_\sigma)\boldsymbol{p}_c + h_\sigma\sqrt{c_c(2-c_c)\mu_{\text{eff}}}(\frac{\boldsymbol{\mu}-\boldsymbol{\mu}'}{\sigma}), \\ \bar{w}_i = w_i \times (1 \text{ if } w_i \geq 0 \text{ else } k/\|\boldsymbol{\mathcal{C}}^{-\frac{1}{2}}(\frac{\boldsymbol{\mu}-\boldsymbol{\mu}'}{\sigma})\|^2), \\ \boldsymbol{\mathcal{C}} \leftarrow \boldsymbol{\mathcal{C}} + c_1\boldsymbol{p}_c\boldsymbol{p}_c^\top + c_\mu\sum_{i=1}^{m}\bar{w}_i(\frac{\boldsymbol{\mu}-\boldsymbol{\mu}'}{\sigma})(\frac{\boldsymbol{\mu}-\boldsymbol{\mu}'}{\sigma})^\top. \end{cases} \tag{21}$$

We refer the readers to [10] for the detailed meanings of $\boldsymbol{p}_\sigma$, $\boldsymbol{p}_c$, as well as the empirical settings of all hyper-parameters $(m, w_{i=1,\ldots,m}, \mu_{\text{eff}}, d_\sigma, c_\sigma, c_\mu, c_c, c_1)$. Furthermore, to reduce the number of parameters, we simply adopt the diagonal co-variance matrix $\boldsymbol{\mathcal{C}}$, such that the search distribution can be represented as $\boldsymbol{\pi} := \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$ with $\boldsymbol{\sigma}^2 = [\sigma_1^2; \sigma_2^2; \ldots; \sigma_n^2]$.

5. Details on DCT Downsampling

Due to the high dimension of input space, blackbox attacks generally requires excessive amount of queries, thus finding a suitable low-dimensional subspace of the original input space can be the key for improving query efficiency. Recent works [7] have demonstrated that the low-frequency components of adversarial perturbations are more effective for attacking deep models. Therefore we propose to reduce the input dimension via keeping only the low frequency components of inputs. Following [7], we perform discrete cosine transformation (DCT) to represent adversarial perturbations in frequency space. More specifically, for a 2d perturbation $\boldsymbol{\eta} \in \mathbb{R}^{d\times d}$, the basis function ϕ_d is defined as:

$$\phi_d(i,j) = \cos\frac{\pi}{d}(i+\frac{1}{2})j. \tag{22}$$

Then the DCT transform V = DCT(X) is:

$$V_{j_1,j_2} = N_{j_1}N_{j_2}\sum_{i_1=0}^{d-1}\sum_{i_2=0}^{d-1}\boldsymbol{\eta}_{i_1,i_2}\phi_d(i_1,j_1)\phi_d(i_2,j_2). \tag{23}$$

where $N_j = \sqrt{\frac{1}{d}}$ if $j=0$, otherwise $N_j = \sqrt{\frac{2}{d}}$. Here the elements of V corresponds to the magnitude of wave $\phi_d(i,j)$, with lower i, j representing lower frequencies. The process can be inversed by inverse discrete cosine transformation (IDCT), i.e. $\boldsymbol{\eta}$ = IDCT(V),

$$\boldsymbol{\eta}_{i_1,i_2} = \sum_{j_1=0}^{d-1}\sum_{j_2=0}^{d-1}N_{j_1}N_{j_2}V_{j_1,j_2}\phi_d(i_1,j_1)\phi_d(i_2,j_2). \tag{24}$$

DCT and IDCT are performed channel-wise independently for each channel of the input.

In order to restrict the input to the lower frequency subspace, we only keep the top-left $rd \times rd$ entries of V, where r is the ratio parameter $r \in (0,1]$. More specifically, for a give perturbation $\boldsymbol{\eta}$, we first perform DCT to get $V = \text{DCT}(\boldsymbol{\eta})$. Then the input space is restricted to low frequency subspace, dubbed r-DCT subspace, by setting $V_{j_1,j_2} = 0$ for $j_1 > rd$ or $j_2 > rd$. The c-Glow models are learned and tested in this r-DCT subspace. When querying the surrogate and target classification models, the perturbations are first sampled in this r-DCT subspace. Then we perform IDCT as described in Eq. (24) to upsample the perturbations into the original space.

6. Experimental Details

1) *pretraining the c-Glow model* is conducted on the standard training set of CIFAR-10 and 10 randomly selected classes [3] from the training set of ImageNet, respectively. The adversarial loss $\mathcal{L}_{adv,s}(\boldsymbol{\eta},\boldsymbol{x})$ in Eq. (13) is specified as the average of CW-L2 losses [2] w.r.t. three surrogate models, and ξ is set as 20. We adopt the normalized gradient descent (NGD) [21] method to achieve stable training. We set the downsampling ratio $r = 0.125$ for ImageNet. The batch-size is set as 16 and the learning rate is 0.0002. We sample $K = 32$ instantiations of $\boldsymbol{z}_0$ for each iteration of training. For finetuing the hyper-parameter λ, we randomly split 10% of the training set of CIFAR-10 and ImageNet as validation set, and search λ within the range $\{10, 20, ..., 100\}$. The fine-tuned values of λ are 20 for CIFAR-10 and 50 for ImageNet. **2)** *The CMA-ES algorithm* is implemented using PyCMA[4], with the population size set to 20 and the selection size to 10. All other hyper-parameters are set as default values in PyCMA.

[3]The details of the classes can be found in Section 5.2 of Appendix

[4]*https://github.com/CMA-ES/pycma*

Table 3. Attack success rate (ASR, %), mean and median number of queries of targeted attack on CIFAR-10 (targeted class is class 4). The best and second-best value among methods that achieve more than 90% ASR are highlighted in bold and underline, respectively.

Target model →	ResNet			DenseNet			VGG			PyramidNet		
Attack Method ↓	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
Bandits [14]	88.3	3125.43	2678.0	86.2	2678.32	2136.0	86.4	4326.73	3979.0	73.0	3373.04	1886.0
SimBA [8]	**99.8**	945.65	836.0	**100**	762.08	665.0	96.6	1214.74	947.0	**100**	874.18	747.0
Subspace [9]	90.0	1715.83	988.0	94.0	851.21	384.0	87.2	1727.58	1274.0	93.6	926.35	424.0
P-RGF [3]	87.9	844.02	532.0	95.5	712.62	432.0	85.7	952.66	434.0	92.6	648.16	430.0
TREMBA [12]	91.0	952.67	632.0	97.8	852.89	510.0	94.2	1452.12	452.0	96.1	674.26	284.0
MetaAttack [6]	97.8	1777.02	1281.0	**100**	1608.41	1409.0	87.6	2479.31	1794.0	98.2	1625.63	1281.0
Signhunter [1]	**99.8**	869.10	327.0	**100**	779.0	359.0	**99.5**	**1203.05**	**421.0**	**100**	805.10	279.0
$\mathcal{CG}$-ATTACK (ours)	98.0	**718.94**	**261.0**	99.5	**408.10**	**241.0**	92.6	1253.42	461.0	97.1	**494.36**	**181.0**

Table 4. Attack success rate (ASR, %), mean and median number of queries of targeted attack on CIFAR-10 (targeted class is class 9). The best and second-best value among methods that achieve more than 90% ASR are highlighted in bold and underline, respectively.

Target model →	ResNet			DenseNet			VGG			PyramidNet		
Attack Method ↓	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
Bandits [14]	68.0	2307.99	1500.0	88.0	1186.69	744.0	75.0	1869.19	1130.0	82.0	1848.49	1224.0
SimBA [8]	**100**	929.55	813.0	**100**	763.26	702.0	99.4	1273.30	1134.0	**100**	863.15	804.0
Subspace [9]	97.3	2352.62	1378.0	99.2	683.42	317.0	90.1	1969.24	1342.0	98.3	1635.44	898.0
P-RGF [3]	67.3	1149.39	436.0	77.3	1405.73	736.0	51.9	1337.75	**532.0**	78.6	1397.60	736.0
TREMBA [12]	96.9	985.13	652.0	91.2	882.0	362.0	97.2	1362.53	911.0	98.2	1012.24	872.0
MetaAttack [6]	98.6	2281.80	1794.0	99.9	1868.92	1665.0	87.0	3174.96	2435.0	98.6	1785.40	1665.0
Signhunter [1]	**100**	**610.07**	**357.0**	**100**	700.28	329.0	**99.7**	1341.91	961.0	**100**	979.66	597.0
$\mathcal{CG}$-ATTACK (ours)	99.5	936.97	621.0	99.8	**589.34**	**1.0**	97.0	**1150.80**	841.0	**100**	**809.09**	**581.0**

7. Addtional Results

7.1. Targeted Attack on CIFAR-10

We present the results of targeted attack on CIFAR-10 with target class 4 (deer) in Table 3. As shown in the table, the proposed $\mathcal{CG}$-ATTACK is very query-efficient at most cases. Specifically, $\mathcal{CG}$-ATTACK achieves the best values of both mean and median query numbers and the second best values of ASR on both ResNet and DenseNet. When attacking PyramidNet, $\mathcal{CG}$-ATTACK achieves ASR of 97.1% with the lowest mean and median number of queries. Although the ASR of SimBA and Signhunter are slightly higher (2.9% higher) on PyramidNet, their mean and median query numbers are more than 1.5x of ours.

Table 4 tabulates the results when attacking into target class 9 (truck). In terms of PyramidNet, $\mathcal{CG}$-ATTACK obtains the best values of ASR and mean and median query numbers. When attacking ResNet, $\mathcal{CG}$-ATTACK achieves the second best values of both ASR and median number of queries. In terms of DenseNet, $\mathcal{CG}$-ATTACK achieves the lowest mean and median query numbers with ASR of 99.8%. Besides, the median query number is 1, which is significantly lower than the second best value (317 of Subspace). When attacking VGG, $\mathcal{CG}$-ATTACK obtains the best value of mean query number and the second best value of median query number. Above results demonstrate the effectiveness and efficiency of $\mathcal{CG}$-ATTACK.

7.2. Targeted Attack on ImageNet

In this section, we provide more results of targeted attack on ImageNet. As decribed in the main manuscript, we select 10 classes for training and testing, *i.e.* 41 (whiptail, whiptail lizard), 265 (toy poodle), 394 (sturgeon), 430 (basketball), 497 (church, church building), 523 (crutch), 776 (sax, saxophone), 864 (tow truck, tow car, wrecker), 911 (wool, woolen, woollen), 988 (acorn). Table 5 presents the performance when attacking into target class 864. As shown in the table, $\mathcal{CG}$-ATTACK is very effective at most cases. Specifically, for target model GoogleNet, $\mathcal{CG}$-ATTACK obtains the best performance in terms of mean and median number of queries and second best in ASR. Signhunter is 2.6% higher in ASR, but this is achieved with the cost of higher query numbers. For target model ResNet and SqueezeNet, $\mathcal{CG}$-ATTACK achieves over 90% of ASR and the best values of both mean and median number of queries. For target model VGG, $\mathcal{CG}$-ATTACK achieves the best value of median query number and second best mean number of queries.

The performance of attacking into target class 776 is shown in Table 6. Comparing to all methods, $\mathcal{CG}$-ATTACK obtains better or comparable performance at most cases. More specifically, when attacking SqueezeNet, $\mathcal{CG}$-ATTACK obtains the highest ASR with the lowest mean and median number of queries, demonstrating its effectiveness and efficiency. When attacking ResNet, $\mathcal{CG}$-ATTACK obtains the second best ASR of 90.1% with the lowest mean and median number of queries. The ASR of Signhunter is slighly higher than $\mathcal{CG}$-ATTACK on ResNet, however, it is inefficient and it costs approximately 5% more query

Table 5. Attack success rate (ASR, %), mean and median number of queries of targeted attack on ImageNet (targeted class is class 864). The best and second-best value among methods that achieve more than 90% ASR are highlighted in bold and underline, respectively.

Target model →	ResNet			GoogleNet			VGG			SqueezeNet		
Attack Method ↓	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
NES [13]	43.7	6977.4	6682.0	54.1	7263.5	7541.0	47.3	6942.3	6531.0	41.3	6541.7	6732.0
$\mathcal{N}$ATTACK [17]	91.8	4082.7	4125.0	88.7	3822.7	3521.0	85.2	4431.7	3825.0	92.3	5567.3	5128.0
Bandits [14]	55.6	5359.6	4842.0	41.7	4933.7	4573.0	51.7	5372.0	4725.0	48.9	4172.3	3864.0
SimBA [8]	89.2	3473.8	3061.0	87.3	4827.5	5023.0	85.9	7492.3	6845.0	93.7	4952.7	3862.0
Subspace [9]	62.8	4761.7	4533.0	58.9	5273.1	4866.0	55.6	4062.9	3681.0	51.0	5844.3	5237.0
P-RGF [3]	76.3	4926.4	4632.0	68.9	3645.4	3162.0	53.4	2826.4	3125.0	71.9	4132.3	3629.0
TREMBA [12]	92.4	3927.6	3741.0	86.3	3672.3	3861.0	72.1	3128.7	2821.0	93.4	5132.7	4861.0
MetaAttack [6]	72.4	6933.5	6523.0	56.1	7263.4	6892.0	49.3	6297.5	6488.0	41.3	7329.15	7088.0
Signhunter [1]	**93.1**	4274.7	3862.0	**93.2**	4451.8	4036.0	**93.5**	4460.4	**3961.0**	**94.0**	3950.7	3372.0
AdvFlow [20]	87.8	5723.6	5232.0	86.4	6733.9	6068.0	81.3	5529.7	4972.0	90.2	6829.3	6329.0
$\mathcal{CG}$-ATTACK (ours)	91.6	**3692.3**	**3041.0**	90.9	**4109.2**	**3621.0**	91.1	**4292.7**	4061.0	93.2	**3531.9**	**2861.0**

Table 6. Attack success rate (ASR, %), mean and median number of queries of targeted attack on ImageNet (targeted class is class 776). The best and second-best value among methods that achieve more than 90% ASR are highlighted in bold and underline, respectively.

Target model →	ResNet			GoogleNet			VGG			SqueezeNet		
Attack Method ↓	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
NES [13]	46.8	6409.8	6728.0	43.5	5897.8	5122.0	53.4	6482.4	6831.0	47.5	5926.3	5542.0
$\mathcal{N}$ATTACK [17]	77.5	6144.3	5681.0	81.3	6473.5	6581.0	70.4	5823.3	5128.0	85.2	6273.4	5826.0
Bandits [14]	47.8	6299.3	5832.0	56.3	7054.3	6423.0	51.3	6329.7	5722.0	54.8	6824.3	6421.0
SimBA [8]	85.8	4329.3	4121.0	87.7	5732.9	5562.0	91.7	6831.7	7023.0	90.2	6092.5	5861.0
Subspace [9]	61.3	4231.8	3874.0	65.2	5791.3	5421.0	73.4	7132.8	6751.0	59.2	6473.5	6120.0
P-RGF [3]	63.7	4192.7	4273.0	54.1	3961.1	3677.0	61.8	5607.8	5832.0	62.4	5831.7	5672.0
TREMBA [12]	89.2	3942.7	3411.0	86.2	4185.2	3961.5	87.3	5960.2	5463.0	81.7	3961.4	3461.0
MetaAttack [6]	70.3	7365.9	7532.0	61.3	6388.7	5961.0	64.7	6084.7	6238.6	55.7	5842.7	5671.0
Signhunter [1]	**92.7**	4532.9	4128.0	**93.9**	3829.5	**3236.0**	**94.3**	4539.2	**4028.0**	91.9	3894.3	3261.0
AdvFlow [20]	82.9	4982.3	5066.0	87.8	4377.4	4563.0	90.5	5506.7	5162.0	81.7	6973.4	6592.0
$\mathcal{CG}$-ATTACK (ours)	90.5	**4257.3**	**3961.0**	91.3	**3692.4**	3381.0	90.8	**4491.3**	4171.0	**92.5**	**3672.1**	**3041.0**

numbers. For target model GoogleNet and VGG, $\mathcal{CG}$-ATTACK achieves the best value of mean query numbers and second best value in median number of queries. Above results demonstrate that $\mathcal{CG}$-ATTACK is very effective and query-efficient for black-box attack.

7.3. Experiments on Open-set Attack Scenario

In this section, we provide results for **Case 1** and **Case 2** open-set experiments for ImageNet. More specifically, for training and testing, we adopt a sub-set, **Imagenette**[5], which contains 10 classes from ImageNet (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute). As specified in Sec. 4.3.1 of the main manuscript, we evenly split the training images of each class, and train the surrogate models on one half, while the target models on the other for **Case 1** open-set attack. For **Case 2** open-set attack, we split the whole training set by classes evenly, and train surrogate models on one half and target models on the other. We report the results for ImageNet in Tab. 7. As can be seen from the left half of Tab. 7, $\mathcal{CG}$-ATTACK achieves the best results in mean and median query numbers against all the models in **Case 1** open-set scenario with ASR of at least 94.3%. Signhunter obtains slightly higher ASR (2.9% higher on average), but it also requires 1.4x mean and 2.3x median of query numbers. For **Case 2** open-set scenarios, $\mathcal{CG}$-ATTACK achieves 6 best results and 6 second-best results out of all 12 results, which is the most among all the attack methods. The results on ImageNet dataset again demonstrates that $\mathcal{CG}$-ATTACK is robust to the biases introduced by differences on training images and class labels, making the proposed method more practical in real world scenarios.

7.4. Ablation Studies on DCT Downsampling Operation

In this section, we conduct experiments to study the effect of DCT downsampling method. Intuitively, the impact of downsampling is two-folds: 1) it improves query efficiency for black-box attacks since the optimization will be performed on a more compact subspace; 2) during training phase, the learning of c-Glow models may be more difficult since the dimension-reduction operation will also drop information about the input. Specifically, we present results for untargeted attacks on CIFAR-10 and ImageNet for four sets of downsampling ratio rs, *i.e.* $\{1.0, 0.5, 0.25, 0.125\}$. The performance are demonstrated in Table 8. Note that when $r = 0.125$ on CIFAR-10, the input dimension is reduced to 4×4, and most of the

[5]https://github.com/fastai/imagenette

Table 7. Attack success rate (ASR %), mean and median number of queries of open-set untargeted attack on ImageNet (**Case 1** and **Case 2**). The first 5 methods (from 'NES' to 'Signhunter') are pure query-based attacks, while the other methods are query-and-transfer-based attacks. The best values among methods are highlighted in bold.

	Case 1												Case 2											
Target Model →	ResNet			GoogleNet			VGG			SqueezeNet			ResNet			GoogleNet			VGG			SqueezeNet		
Attack Method ↓	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
NES [13]	92.5	972.3	641.0	90.4	895.1	643.0	90.9	1025.4	844.0	91.3	865.1	682.0	91.5	1041.5	862.0	89.4	685.2	476.0	92.9	849.3	655.0	87.1	744.3	487.0
$\mathcal{N}$ATTACK [17]	96.1	1285.4	794.0	97.2	864.7	642.0	96.0	936.6	652.0	95.3	855.3	592.0	94.1	882.5	593.0	95.6	799.8	569.0	93.7	695.3	408.0	95.4	684.3	422.0
Bandits [14]	92.4	892.5	524.0	90.4	763.4	586.0	91.3	692.1	427.0	91.2	724.9	506.0	93.2	743.5	503.0	94.8	862.5	679.0	93.9	908.0	723.0	91.0	755.6	582.0
SimBA [8]	97.1	401.5	316.0	96.4	394.2	301.0	98.3	582.3	369.0	98.4	519.4	341.0	95.8	415.8	247.0	97.4	395.4	186.0	95.7	319.4	204.0	96.8	429.4	297.0
Signhunter [1]	**100**	225.7	37.0	**100**	217.5	64.0	**100**	193.4	54.0	**100.0**	269.7	62.0	**100**	196.5	81.0	**99.8**	164.3	101.0	**100**	294.2	189.0	99.2	194.1	**87.0**
Subspace [9]	91.9	637.5	412.0	93.1	576.4	392.0	94.1	588.1	284.0	95.1	563.5	471.0	93.3	733.5	582.0	94.5	724.1	644.0	92.7	842.5	584.0	92.0	683.0	393.0
P-RGF [3]	96.7	422.7	216.0	96.8	322.7	126.0	96.6	379.5	281.0	91.8	588.3	220.0	94.5	525.4	394.0	93.0	492.3	315.0	93.1	475.2	258.0	92.1	363.4	194.0
TREMBA [12]	95.2	310.4	141.0	97.2	288.1	121.0	94.3	253.4	141.0	94.9	347.5	161.0	91.3	189.4	91.0	93.2	**174.5**	**71.0**	94.5	226.2	161.0	92.1	168.3	101.0
MetaAttack [6]	93.1	413.7	228.0	92.7	473.6	362.0	95.0	393.5	216.0	94.1	462.1	287.0	94.7	386.4	201.0	93.2	425.9	361.0	93.8	362.4	161.0	95.1	374.3	191.0
AdvFlow [20]	92.8	627.4	326.0	94.2	756.4	542.0	96.2	1766.2	1362.0	91.1	612.5	362.0	91.6	743.7	642.0	93.5	1285.4	843.0	94.1	973.5	632.0	95.1	1044.1	832.0
$\mathcal{CG}$-ATTACK	94.2	**148.6**	**31.0**	98.0	**163.9**	**21.0**	98.4	**156.2**	**21.0**	97.8	**172.1**	**21.0**	98.8	**165.3**	**61.0**	97.3	246.1	101.0	98.9	**173.4**	**81.0**	**99.7**	**167.2**	91.0

Table 8. Attack success rate (ASR, %), mean and median number of queries of untargeted attack on CIFAR-10 and ImageNet for different downsampling ratios. The best value among methods are highlighted in bold.

	Downsampling ratio →	$r = 1.0$			$r = 0.5$			$r = 0.25$			$r = 0.125$		
	Target model ↓	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
CIFAR-10	ResNet	**100.0**	**81.6**	**1.0**	84.2	288.4	181.0	61.3	897.4	821.0	-	-	-
	DenseNet	**100.0**	**43.4**	**1.0**	82.3	343.6	141.0	59.6	758.5	521.0	-	-	-
	VGG	**99.9**	**56.4**	**1.0**	81.7	363.8	161.0	63.9	636.2	521.0	-	-	-
	PyramidNet	**100.0**	**30.1**	**1.0**	87.2	334.2	161.0	70.6	756.4	421.0	-	-	-
ImageNet	ResNet	73.2	972.6	861.0	80.5	596.7	461.0	95.4	372.9	201.0	**97.3**	**210.4**	**21.0**
	GoogleNet	65.3	1139.2	961.0	77.4	813.5	641.0	96.2	442.3	281.0	**100.0**	**138.8**	**21.0**
	VGG	69.5	773.2	541.0	78.1	554.3	421.0	94.7	252.3	121.0	**99.4**	**77.3**	**1.0**
	SqueezeNet	61.2	877.3	641.0	73.2	729.3	561.0	93.8	320.8	141.0	**99.3**	**132.9**	**21.0**

Table 9. Attack success rate (ASR %), mean and median number of queries of untargeted attack on CIFAR-10. The best and values among methods are highlighted in boldface.

Target Model →	ResNet			DenseNet			VGG			PyramidNet		
Attack Method ↓	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
$\mathcal{CG}$-ATTACK-NES	98.5	**67.3**	**1.0**	98.3	57.9	**1.0**	**100**	78.2	**1.0**	97.9	85.2	**1.0**
$\mathcal{CG}$-ATTACK-SA-ES	98.3	141.2	**1.0**	**100**	97.3	**1.0**	96.7	103.6	**1.0**	99.2	67.1	**1.0**
$\mathcal{CG}$-ATTACK-CMA-ES	**100**	81.6	**1.0**	**100**	**43.3**	**1.0**	99.9	**56.4**	**1.0**	**100**	**30.1**	**1.0**

useful information within the input is dropped, increasing difficulty for the learning of c-Glow models. Therefore, we omit this part when reporting the results.

From Table. 8, we can see that for CIFAR-10, the attacks are less effective in terms of ASR, mean query number, and median query numbers as the downsampling ratio r decreases. This can be attributed to the fact that sizes of images from CIFAR-10 dataset is already small enough for optimization, thus continuing reducing the input size will worsening the second impact of downsampling that harms the training of c-Glow models. A different trend, however, can be observed for ImageNet dataset from Table. 8. The attack results improve as the input sizes become smaller. This shows that the original size of ImageNet is too large for black-box optimization and the images contain redundant information that can be reduced without harming the training results.

7.5. Ablation Studies on ES Search Algorithm

In this section, we analyze the effect of basic search algorithm on our proposed $\mathcal{CG}$-ATTACK. $\mathcal{CG}$-ATTACK adopts CMA-ES as the basic search algoritm, here we provide results on adopting NES [27, 30] and SA-ES [16] as basic search algorithm in our method for the untargeted attack on CIFAR-10 dataset. As shown in Tab. 9, adopting CMA-ES will have the best overall performance while the algorithm choice will not significantly influence our method.

7.6. Discussion of the complexity of $\mathcal{CG}$-ATTACK

The main computational complexity arises from the use of c-Glow model in both pre-training and attack stage. As specified in Sec. 1 of Supplementary, c-Glow consists of 24 Glow steps, and each step is further divided into there layers: conditional actnorm layer, conditional 1×1 convolutional layer and conditional affine coupling layer. We list the floating point operations (FLOPs) of each layer in the table below.

Table 10. The FLOPs for each of the c-Glow step layer.

Layer $\rightarrow$	Actnorm	1×1 Convolution	Affine Coupling
FLOPs	6.8M	6.7M	0.2M

Table 11. Untargeted attack on CIFAR-10 with different ratios of adjusted parameters in c-Glow.

Target Model $\rightarrow$	ResNet			DenseNet			VGG			PyramidNet		
Adjusted Parameters $\downarrow$	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
Pure Transfer	74.9	-	-	84.1	-	-	84.5	-	-	92.1	-	-
Gaussian	100.0	81.6	1.0	100.0	43.3	1.0	99.9	56.4	1.0	100.0	30.1	1.0
Gaussian + First	97.2	133.9	61.0	98.6	127.3	61.0	99.5	108.1	41.0	98.8	129.3	81.0
Gaussian + All	85.2	527.9	421.0	82.4	386.2	221.0	87.5	421.7	261.0	85.9	479.3	261.0

Table 12. Untargeted attack on CIFAR-10 with c-Glow networks of different blocks.

Target Model $\rightarrow$	ResNet			DenseNet			VGG			PyramidNet		
Number of Blocks $\downarrow$	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median	ASR	Mean	Median
2	97.3	136.9	21.0	98.4	158.2	21.0	99.7	179.6	41.0	97.9	121.5	1.0
3	100.0	81.6	1.0	100.0	43.3	1.0	99.9	56.4	1.0	100.0	30.1	1.0
4	98.6	108.3	1.0	99.7	36.2	1.0	99.2	60.8	1.0	99.3	27.4	1.0

Therefore, the total FLOPs of c-Glow model is $T = (6.8M + 6.7M + 0.2M) \times 24 = 328.8M$. In training stage, each iteration consists of 1 forward operation (T FLOPs) and 1 backward operation ($2T$ FLOPS), making the total of $3T = 986.4M$ FLOPs In attack stage, each query consists of 1 forward operation thus taking $T = 328.8M$ FLOPs.

7.7. Ablation Studies on Ratio of parameters transferred for $\mathcal{CG}$-ATTACK

The key of $\mathcal{CG}$-ATTACK is to partially transfer the parameters of CAD, which can be seen as a **trade-off between attack generalization and query efficiency**. Here we experimentally verify the effects of different ratios of adjusted parameters to $\mathcal{CG}$-ATTACK: **1)** transferring all parameters without adjusting; **2)** adjusting Gaussian parameters; **3)** adjusting Gaussian parameters and the first layer of c-Glow; **4)** adjusting Gaussian and all parameters of c-Glow. As shown in Tab. 11, under the query limit (10,000 times), only adjusting Gaussian parameters achieves the best performance in both ASR and efficiency.

7.8. Ablation Studies on the depth (capacity) of c-Glow networks

As specified in Sec. 1 of Supplementary, we adopted c-Glow of 3 blocks in main experiments. To verify the depth's effect, here we conduct some experiments with c-Glow of 2,3,4 blocks, respectively. As shown in Tab. 12, 2-block network performs worst in both ASR and query efficiency, probably due to its limited capacity for modeling CAD. There is no significant performance difference between 3-block and 4-block networks, verifying their suitability to this task. Generally, we think there should be a tradeoff between modeling capacity and generalization $w.r.t.$ the depth of c-Glow.

8. Further Discussions

Limitations and possible defenses of $\mathcal{CG}$-ATTACK. The superior performance of $\mathcal{CG}$-ATTACK is mainly due to the good modeling of the conditional adversarial distribution (CAD) and the partial transfer mechanism. The assumptions behind are that CAD of one model is derived by mapping a Gaussian distribution through the c-Glow network (Section 3.2.2), and CADs of different models correspond to different Gaussian distributions while with the same c-Glow network (Assumption 1). Although these assumptions have been thoroughly verified and analyzed in our work, they could be utilized to design defenses. **2) Possible defenses**: There are three possible approaches. **2.a) Ensemble models**. A few works attempted to resist the adversarial transferability by enhancing the diversity of sub-models in ensemble models, *e.g.*, DVERGE (NeurIPS 2020) and ADP (NeurIPS 2019). It may be somewhat effective against $\mathcal{CG}$-ATTACK, because the CAD of the diverse ensemble model may correspond to a Gaussian-mixture distribution. However, $\mathcal{CG}$-ATTACK can be further adjusted to model the CAD of ensemble model to evade this defense. **2.b) Adversarial training (AT)**, where the perturbation is generated by the pre-trained c-Glow network. **2.c) Random noise defense (RND)** (NeurIPS 2021) shows that adding random noise to each query sample could effectively defend against query-based attacks. It may be effective against $\mathcal{CG}$-ATTACK, causing more queries for the target model to learn Gaussian parameters.

Potential negative social effects. Deep learning methods have been widely adopted in numerous application domains. Our work focuses on the black-box adversarial attack, which is currently one of the most effective attacks against deep learning algorithms. Therefore, our work has a profound impact on the security of real-world deep learning applications.

our work may introduce some negative impacts. Firstly, the attackers may adopt our method to attack real-world deep learning based applications, which is also why we hope to boost the development of defense methods. Secondly, we feel obligated to emphasize the diversity of real-world attack methods and have no intention to claim that our work covers all the attack techniques, which may introduce a false sense of security.

The potential benefit/usage of the *CAD* approximated by c-Glow. As verified above, the *CAD* approximated by c-Glow is very close to the real distribution, and is very useful to improve the performance of black-box attack. We believe that this distribution can be also used to improve the robustness of DNNs. For example, it can be utilized to efficiently generate diverse adversarial examples during the adversarial training framework [19]. It will be explored in our future work.

References

[1] Abdullah Al-Dujaili and Una-May O'Reilly. Sign bits are all you need for black-box attacks. In *ICLR*, 2020.

[2] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE S&P*, 2017.

[3] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. In *NeurIPS*, 2019.

[4] Philip J. Davis. Leonhard euler's integral: A historical profile of the gamma function. *American Mathematical Monthly*, 66(10):849–869, 1959.

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[6] Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, and Jiashi Feng. Query-efficient meta attack to deep neural networks. In *ICLR*, 2020.

[7] Chuan Guo, Jared S. Frank, and Kilian Q. Weinberger. Low frequency adversarial perturbation. In *UAI*, 2019.

[8] Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q. Weinberger. Simple black-box adversarial attacks. In *ICML*, 2019.

[9] Yiwen Guo, Ziang Yan, and Changshui Zhang. Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks. In *NeurIPS*, 2019.

[10] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[11] Nikolaus Hansen, Dirk V. Arnold, and Anne Auger. Evolution strategies. In *Springer Handbook of Computational Intelligence*. 2015.

[12] Zhichao Huang and Tong Zhang. Black-box adversarial attack with transferable model-based embedding. In *ICLR*, 2020.

[13] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *ICML*, 2018.

[14] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *ICLR*, 2019.

[15] Rie Johnson and Tong Zhang. A framework of composite functional gradient methods for generative adversarial models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[16] Dipl. Ing. Karl Heinz Kellermayer. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Journal of Cybernetics*, pages 319–320, 1977.

[17] Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. NATTACK: learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *ICML*, 2019.

[18] You Lu and Bert Huang. Structured output learning with conditional generative flows. In *AAAI*, 2020.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[20] Hadi Mohaghegh Dolatabadi, Sarah Erfani, and Christopher Leckie. Advflow: Inconspicuous black-box adversarial attacks using normalizing flows. In *NeurIPS*, 2020.

[21] Ryan Murray, Brian Swenson, and Soummya Kar. Revisiting normalized gradient descent: Fast evasion of saddle points. *IEEE Transactions on Automatic Control*, 64(11):4818–4824, 2019.

[22] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[23] I Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*. 1978.

[24] Ludger Rüschendorf. The wasserstein distance and approximation theorems. *Probability Theory and Related Fields*, 70(1):117–129, 1985.

[25] H.-P Schwefel. Numerische optimierung von computer-modellen mittels der evolutionsstrategie: mit einer vergleichenden einführung in die hill-climbing-und zufallsstrategie. In *Birkhäuser*, 1977.

[26] Xinwei Shen, Tong Zhang, and Kani Chen. Bidirectional generative modeling using adversarial gradient estimation. *arXiv preprint arXiv:2002.09161*, 2020.

[27] Yi Sun, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber. Efficient natural evolution strategies. In *GECCO*, 2009.

[28] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

[29] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *JMLR*, 15(1):949–980, 2014.

[30] Daan Wierstra, Tom Schaul, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. In *IEEE CEC*, 2008.