We appreciate all reviewers for their helpful and constructive comments. We'll further improve the paper in the final version. Below we address their detailed comments.

R1: RGF outperforms NES: The major difference between RGF and NES [16] is that NES adopts the antithetic sampling, while RGF does not. Specifically, the gradient estimator is $\hat{g} = \frac{1}{q}\sum_{i=1}^{q}\frac{f(x+\sigma u_i,y)-f(x-\sigma u_i,y)}{2\sigma}u_i$ in NES and $\hat{g} = \frac{1}{q}\sum_{i=1}^{q}\frac{f(x+\sigma u_i,y)-f(x,y)}{\sigma}u_i$ in RGF (see Eq.(5)). The NES estimator can eliminate the second-order component of f through central differences, but it requires $2q$ queries while RGF only requires $q+1$ queries. When σ is small, the second-order component is often dominated by the first-order one. So RGF outperforms NES. We'll make it clearer.

R1: λ^* distribution and cosine similarity across the attack iterations: Thanks for the suggestion. As it's hard to plot the full distribution of λ^*, which changes during iteration, we show the average λ^* over all images w.r.t. iterations in Fig. A. It shows that λ^* decreases along with the iterations (i.e., the distribution concentrates on small λ^*). Fig. B shows the cosine similarity between the transfer and the true gradients, and that between the estimated and the true gradients, across iterations. The results show that the transfer gradient is useful at beginning, and becomes less useful along with the iterations. However, the estimated gradient can remain higher cosine similarity with the true gradient, which facilitates the adversarial attacks consequently. We'll add the results in the final version.

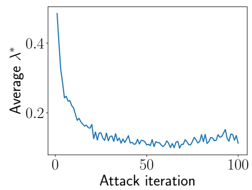

Figure A: The average λ^* across attack iterations.

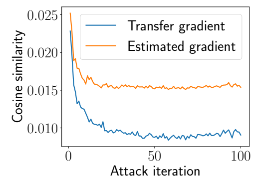

Figure B: The cosine similarity across attack iterations.

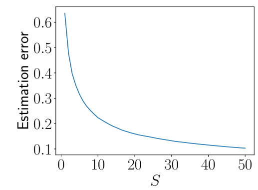

Figure C: The estimation error with different S.

R2: Novelty of the idea: As stated in L108-113, we consider the score-based setting while [4] focuses on the decision-based setting. [4] is built upon the Boundary method [3] and uses a fixed coefficient to incorporate the transfer gradient. Due to the different settings, we introduce a new objective (see Eq.(7)) for gradient estimation, and optimize it inside the proposed family of estimators, resulting in a generic P-RGF algorithm which incorporates the transfer gradient with an optimal coefficient. Technically, it's non-trivial to derive the optimal solution. Moreover, we found that it's necessary to use an adaptive coefficient rather than a fixed value since 1) the usefulness of the transfer gradient varies across iterations; 2) experiments show that our algorithm is beneficial from the adaptive coefficient. Overall, we propose a simple, yet novel and effective method, considering a different black-box setting from [4], as agreed by R1 and R3.

Table A: Additional experimental results.

Methods	Inception-v3		VGG-16		ResNet-50	
	ASR	AVG. Q	ASR	AVG. Q	ASR	AVG. Q
P-RGF ($\lambda = 0.05$)	97.8%	1021	99.7%	888	**99.6%**	790
P-RGF (λ^*, true norm)	**98.1%**	768	**99.8%**	**501**	99.5%	**427**
P-RGF (λ^*)	**98.1%**	**745**	**99.8%**	521	**99.6%**	452

R2: More analysis and experiments about the estimation of gradient norm: Thanks for the comment. The gradient norm (or cosine similarity) is easier to estimate than the true gradient since it's a scalar value. Fig. C shows the estimation error of the gradient norm, defined as the (normalized) RMSE—$\sqrt{\mathbb{E}(\frac{\widehat{\|\nabla f(x)\|_2}-\|\nabla f(x)\|_2}{\|\nabla f(x)\|_2})^2}$, w.r.t. the number of queries S. We chose $S = 10$ in all experiments to reduce the number of queries while the estimation error is acceptable. We also show the overall attack results of using the true gradient norm instead of the estimated norm in Table A (Row 2). The results are similar to those of using the estimated norm. We'll add the results in the final version.

R2: Experiments about P-RGF with a fixed $\lambda = 0.05$: Thanks for the suggestion. Table A (Row 1) shows the results of P-RGF with $\lambda = 0.05$ (optimal in Fig. 1(b)), which are better than P-RGF with $\lambda = 0.5$ (in Table 1). However, a significant performance gap still remains from using the adaptive λ^*. We'll add the results in the final version.

R3: The improvement over the RGF method is not significant: In Table A, P-RGF and RGF obtain similar attack success rates. The reason is that the maximum number of queries (i.e., 10,000) is sufficient for them to find adversarial perturbations, such that their attack success rates are similarly high. However, P-RGF requires fewer queries than RGF ($20\% \sim 45\%$ queries reduction). If the maximum number of queries is set to 1,000, the attack success rate against Inception-v3 becomes 56.4% using RGF and 78.6% using P-RGF (the average number of queries is 470 and 297 respectively). Moreover, in Table 2, P-RGF obtains much higher success rates than RGF, and also reduces the query complexity for attacking the defensive models. In summary, the improvement is significant in most of the cases.

R3: Attack results on adversarially trained defensive models: Thanks for the suggestion. We choose [*1] as our target model, which successfully performs PGD-based adversarial training on ImageNet. The gradient from ResNet-152 can hardly transfer to this model, and the results of RGF and P-RGF are similar. So we use another adversarially trained model (with a different architecture) to provide the transfer gradient. We perform ℓ_∞ attacks with $\epsilon = 16/255$, which is the same threat model used in adversarial training. Table B presents the results—P-RGF outperforms RGF significantly with the strong transfer-based prior. We'll add the results in the final version.

Table B: Attack results on adversarially trained model.

	ASR	AVG. Q
RGF	31.7%	1207
P-RGF (λ^*)	**64.7%**	**378**

[*1] C. Xie, Y. Wu, L. van der Maaten, et al. Feature denoising for improving adversarial robustness. CVPR 2019.