We thank all reviewers for their insightful feedback. We are encouraged they find ODS to be simple (R2), well-motivated (R1), applicable to many existing attacks (R2) including both white- and black-box attacks (R1,3), and evaluated with extensive experiments (R2) which show significant improvements in black-box attacks (R1,3) and justifications of surrogate models (R3). We address some specific comments below and will incorporate all feedback received.

@R2,3 – "Comparison with black-box attacks using surrogate models would strengthen the paper." Great suggestion! We focus on [25] which R2 and R3 cited. [25] proposed P-RGF which uses prior knowledge to estimate the gradient of the target model more efficiently than RGF. RGF uses random sampling to estimate the gradient, so we can combine ODS with RGF and compare it with P-RGF under ℓ_2 and ℓ_∞ norms (results in Table i below) . **The average number of queries required by ODS-RGF is smaller than P-RGF ([25]) in all settings**. It suggests ODS-RGF can estimate the gradient more precisely than P-RGF by exploiting diversity obtained from surrogate models. R2 also cited [23]. While we did not have enough time for an additional experimental comparison, we note that [23] is specific to the ℓ_∞ norm and needs to train a generator per target class, which is quite restricted compared to ODS.

Table i: Comparison of ODS-RGF and P-RGF for 300 images on ImageNet. The target and surrogate models are pre-trained ResNet50 and ResNet34 models, respectively. As for hyperparameters, the number of max queries is 10000, sample size is 10, step size is 0.5 (ℓ_2) and 0.005 (ℓ_∞), and epsilon is $\sqrt{0.001 \cdot 224^2 \cdot 3}$ (ℓ_2) and 0.05 (ℓ_∞).

		untargeted			targeted		
norm	attack	success	queries	ℓ_2 perturbation	success	queries	ℓ_2 perturbation
ℓ_2	RGF	**100.0%**	633	3.07	**99.3%**	3141	8.23
	P-RGF [25]	**100.0%**	211	2.08	97.0%	2296	7.03
	ODS-RGF	**100.0%**	**133**	**1.50**	**99.3%**	**1043**	**4.47**
ℓ_∞	RGF	97.0%	520	-	25.0%	2971	-
	P-RGF [25]	99.7%	88	-	65.3%	2123	-
	ODS-RGF	**100.0%**	**74**	-	**92.0%**	**985**	-

@R2 – "Does ODS suffer from differences in the training scheme (e.g. adversarially and naturally)?" Yes, partly. That being said, we can mitigate the problem by simultaneously using surrogates obtained with various training schemes (which are mostly publicly available). We run a new experiment to attack a robust target model using SimBA-ODS with both natural and robust surrogate models (a natural model and a robust model). SimBA-ODS still outperforms SimBA-DCT without surrogate models (e.g. average query is 1304 vs 2824). This suggests that if the set of surrogates includes one that is similar to the target, ODS still works (even when some other surrogates are "wrong").

@R3 – "What happens if a search direction of ODS is a vector aligned with the true class?" It might accelerate attacks but make the perturbation large due to less diversity (A related phenomenon is shown in Figure G for MultiTargeted).

@R3 – "Why do you combine SimBA and Boundary attack with ODS?" A reason is these attacks use random sampling. Another one is popularity. These attacks are common benchmarks.

@R1 – "Comparison under the same step for Table 1 and 2 would be fairer." We agree and performed new experiments with ODI-PGD-(k-2), which outperforms PGD-k in Table 1 on all datasets (90.21% vs 90.31% on MNIST, 44.45% vs 46.06% on CIFAR-10 and 42.3% vs 43.5% on ImageNet). For Table 2, we also ran tuned ODI-PGD with 1000 total steps on MNIST and can confirm the result in a single run is within the confidence intervals in Figure C.

@R2 – "Comparison of diversity between 2 steps of ODI and PGD would be helpful." We compare diversity like in Figure A. In the left panel of Figure i, losses for points generated by ODI-2 (2 steps of ODI) are more diverse than PGD-2. This diversity also brings diversity in attack results after 20 steps (see the right panel).

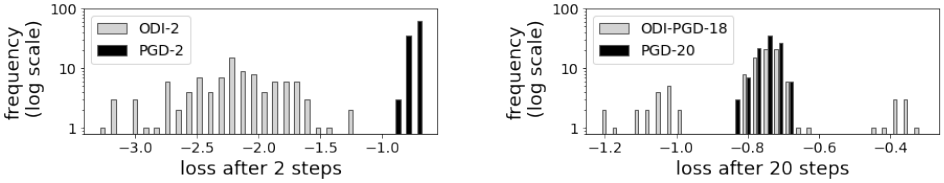

Figure i: Histogram of loss values after some update steps. Each attack runs 100 times for one sample image. PGD-2 and PGD-20 are initialized by naïve uniform initialization. The loss function is the margin loss.

@R1 – "What is computational complexity in Table 2?" It is the number of gradient computations, e.g. 42 steps × 20 restarts = 840. We confirmed the wall-clock time for calculation of a ODI step is the same as PGD.