RETHINKING THE USE OF ENSEMBLE UPDATES IN ADVERSARIAL ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

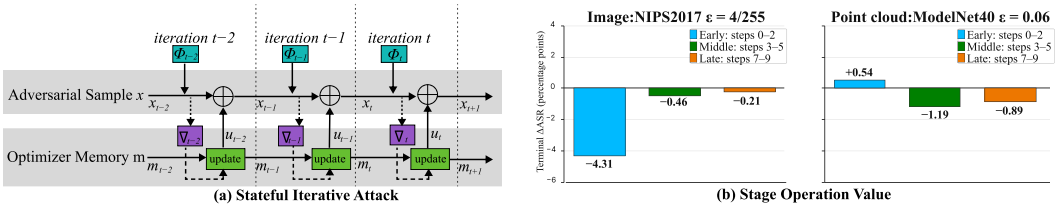
Adversarial examples can mislead deep neural networks with subtle perturbations and often transfer across models, a phenomenon known as adversarial transferability. Model ensemble attacks improve the transferability of adversarial examples to unknown models by jointly using information from multiple surrogate models. Existing methods primarily focus on constructing effective attack updates and typically apply the resulting update rule throughout the attack. However, iterative attacks are stateful sequential optimization processes, in which the same update may have different effects on final transferability depending on when it is applied along the attack trajectory. To investigate this question, we define Attack Operation Value (AOV), which measures the final benefit of method-specific computation by comparing the final transfer effects of method-specific and base ensemble updates from the same attack state. Our analysis reveals substantial variation in the returns of computationally expensive updates across attack stages: their effectiveness at one stage does not imply that they should be applied throughout the attack trajectory. Experiments under a fixed compute budget further show that a small number of early expensive updates, followed by sufficient base ensemble optimization, can yield a more effective allocation of computation. Motivated by these findings, we propose Temporal Ensemble Scheduling (TES), which concentrates a limited number of method-specific updates at the beginning of the attack and uses inexpensive base ensemble updates for the remaining iterations. Experiments on images and 3D point clouds show that TES substantially reduces computational cost and runtime while maintaining competitive black-box transferability.

1 INTRODUCTION

Deep neural networks are widely deployed in image recognition, autonomous driving, and biometric identification, yet remain vulnerable to adversarial examples: small input perturbations can induce incorrect predictions (Szegedy et al., 2014; Goodfellow et al., 2015). Adversarial examples also exhibit transferability, allowing perturbations generated on accessible surrogate models to fool unknown models with different architectures and parameters (Liu et al., 2017). Consequently, transfer-based attacks have become an important research paradigm for practical black-box adversarial attacks.

An effective strategy for improving transferability is to attack multiple surrogate models jointly. Combining information across models with different architectures and decision boundaries can reduce overfitting to an individual surrogate and improve generalization to unseen models. Early ensemble attacks typically average losses, prediction probabilities, or logits (Liu et al., 2017; Dong et al., 2018), while subsequent methods develop more elaborate mechanisms. These efforts broadly address two questions: which surrogate models to use, and how to combine their information. The former concerns ensemble size, diversity, and weight space coverage (Li et al., 2020; Gubri et al., 2022; Yao et al., 2025); the latter concerns output aggregation, gradient optimization, and model reweighting (Xiong et al., 2022; Chen et al., 2023; 2024; Tang et al., 2024). The refinement methods considered here normally execute their method-specific procedure at every outer iteration, even when its internal weights or gradient signals adapt.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Stateful Iterative Attack (b) Stage Operation Value

Figure 1: (a) Iterative attacks evolve over a joint adversarial and optimizer state. (b) The same ensemble update yields substantially different final transfer effects when applied at different attack stages.

Iterative attacks, however, are stateful sequential optimization processes. As illustrated in Figure 1(a), the current adversarial input x_t already incorporates the effects of preceding updates, while optimizer memory, such as momentum m_t, carries information across iterations. The ensemble operation Φ_t produces a gradient signal ∇_t, which is combined with the inherited optimizer state to form an update u_t and the next optimizer state. Applying u_t then produces x_{t+1}. Each attack operation thus changes both the current iterate and the conditions faced by subsequent operations. Consequently, the same operation may contribute differently to final transferability depending on when it is applied along the attack trajectory.

Figure 1(b) provides a simple test case. We consider a ten-iteration attack with five surrogate models and use logit ensembling as the baseline update rule. We then replace logit ensembling with loss-level updates in the early, middle, or late stage, while retaining logit ensemble updates in all remaining iterations. The results show that replacing logit ensembling with loss-level updates has an overall negative effect on terminal transferability, but the magnitude of this negative effect varies substantially across attack stages. Thus, merely changing the execution position of the same attack operation can lead to markedly different terminal transfer effects.

To further investigate this issue, we define *Attack Operation Value* (AOV). Starting from a given joint attack state, we compare method-specific and reference updates followed by the same continuation rule, and measure their difference in terminal transfer utility. On both images and point clouds, we observe that the same method-specific updates yield higher final returns early in the attack when operation counts and computational costs are matched. After these early updates, using expensive updates for all remaining iterations does not guarantee additional gains and reduces final attack performance in the analyzed settings. Allocation under a fixed budget also matters: spending more computation on expensive updates while shortening subsequent base ensemble optimization does not consistently improve transferability. A small number of early expensive updates followed by sufficient base ensemble optimization provides a more effective combination.

Based on these observations, we propose *Temporal Ensemble Scheduling* (TES). TES applies a limited number of method-specific updates at the beginning of the attack, then switches to inexpensive base ensemble updates to complete the original attack length. We integrate TES with ensemble attacks that have different computational structures and evaluate it on images and point clouds across datasets, target models, and perturbation budgets. With a fixed schedule across settings, TES reduces the number of method-specific updates and attack runtime while achieving black-box transferability comparable to, and often higher than, the full schedule that applies the method-specific rule at every iteration. Overall, the contributions of this paper are summarized as follows:

1. We formulate transfer-based ensemble attacks as stateful sequential optimization. We further define AOV to characterize the terminal effect of an attack operation applied at a particular joint state and propagated through the remaining trajectory.
2. Window and suffix interventions reveal stage-dependent returns of method-specific updates. Under a fixed compute budget, a small number of expensive early-stage updates followed by sustained base ensemble optimization enables a more effective allocation of computation.
3. We introduce TES, which schedules a limited number of method-specific updates early while preserving the full outer optimization process. Experiments across modalities and ensemble methods demonstrate its efficiency and competitive transferability.

2 Related Work

Gradient based attacks and conventional ensembles. Transfer attacks optimize perturbations on accessible surrogates and evaluate them on unknown targets. FGSM uses the input gradient to construct a perturbation (Goodfellow et al., 2015), while MI-FGSM accumulates gradients across iterations to stabilize the update direction (Dong et al., 2018). These optimization procedures also support model ensembles: Liu et al. (2017) generate transferable examples by jointly attacking multiple models, and Dong et al. (2018) compare fusion at the prediction, loss, and logit levels. In conventional ensembles, a weighted combination of model outputs or losses defines the objective whose input gradient drives the attack. This provides a basic way to exploit multiple surrogates without additional refinement of their interactions.

Constructing surrogate ensembles. One line of research improves which models contribute to the ensemble, including its size, diversity, and weight space coverage. Ghost Networks perturbs intermediate features to generate diverse model variants and uses longitudinal ensembling across attack iterations (Li et al., 2020). LGV collects weight snapshots during additional training at a high learning rate, forming surrogate ensembles from a broad geometric vicinity (Gubri et al., 2022). More recently, Yao et al. (2025) relate transferability to ensemble vulnerability, diversity, and complexity, motivating larger and more diverse surrogate sets while controlling overfitting. These studies improve the collection of models available to an attack, including approaches that vary the surrogate across iterations.

Aggregating surrogate information and refining updates. A second line improves how information from multiple models is used. SVRE reduces variation among surrogate gradients through stochastic variance reduction (Xiong et al., 2022). CWA seeks shared weaknesses through flatter input loss landscapes and proximity to the optima of individual surrogates (Chen et al., 2024). SMER combines stochastic minibatch attacks with ensemble reweighting through reinforcement learning (Tang et al., 2024). These methods can adapt their weights or gradient signals during optimization.

Beyond these approaches, Hu et al. (2022) view ensemble attacks as a multi-stage process. They accordingly design a model scheduling strategy that combines decision boundary similarity, model diversity, and information from preceding stages. However, their stage-wise analysis remains centered on model scheduling and does not examine the stages over which an attack method's updates effectively improve final transferability.

3 Mechanism Analysis and Method

In this section, we formulate ensemble attacks as stateful optimization and define AOV. Controlled CWA experiments then examine update placement, continued refinement, and compute allocation. These observations motivate TES; Section 4 tests its effectiveness and efficiency across methods.

3.1 Problem Formulation

Given a clean input x with its ground-truth label y, a transfer-based attack seeks an adversarial example x^{adv} that misleads unknown target models. Let x' denote a candidate perturbed input, $\mathcal{X}$ denote the valid input space, and $d(\cdot,\cdot)$ denote the distance metric. We denote by $\mathcal{C}(x,\epsilon)$ the feasible set of all valid perturbed inputs whose distance from x does not exceed the perturbation budget ϵ, defined as

$$\mathcal{C}(x,\epsilon) = \{x' \in \mathcal{X} : d(x',x) \le \epsilon\}. \tag{1}$$

Initialized at $x_0 = x$, the attack executes T outer updates to output $x_T = x^{\mathrm{adv}}$.

Under the same ensemble attack, each outer iteration applies either: (i) a base update B, which performs a standard momentum step over aggregated surrogate logits with fixed parameters; or (ii) a method-specific update R, which may introduce extra gradient evaluations, inner optimization, or model reweighting.

Let $z_t = (x_t, s_t)$ denote the joint attack state before iteration t, where s_t includes persistent optimizer information such as momentum and method-specific historical statistics. The attack evolves according to

$$z_{t+1} = \mathcal{T}_{a_t,t}(z_t;\xi_t), \qquad a_t \in \{B,R\}, \qquad t = 0,\dots,T-1, \tag{2}$$

where $\mathcal{T}_{a_t,t}$ is the state transition induced by the selected update, and ξ_t represents internal randomness.

Earlier updates jointly shape the current state, while the current update changes the states encountered by subsequent updates. The final effect of an attack operation can therefore depend on when it is applied along the trajectory.

Figure 1(b) illustrates this dependence by placing three loss ensemble updates at different stages of a logit ensemble attack. Table 1 reports the complete results, where G and L denote logit and loss ensemble updates, respectively. With the surrogate models, perturbation budget, attack length, and update counts held fixed, changing only the update positions leads to different final transfer performance. To further investigate this issue, we quantify the terminal value of an attack operation from a given trajectory state.

Table 1: Results for Figure 1(b). ASR (%) and its change from Logits-only (pp), computed before rounding.

Schedule	Updates	ASR	ΔASR
NIPS2017			
Logits-only	G^{10}	48.12	–
Loss-Early	L^3G^7	43.80	-4.31
Loss-Middle	$G^3L^3G^4$	47.66	-0.46
Loss-Late	G^7L^3	47.91	-0.21
ModelNet40			
Logits-only	G^{10}	66.16	–
Loss-Early	L^3G^7	66.71	$+0.54$
Loss-Middle	$G^3L^3G^4$	64.97	-1.19
Loss-Late	G^7L^3	65.27	-0.89

3.2 Attack Operation Value

The immediate surrogate loss, gradient magnitude, or single step displacement describes a local effect but does not directly measure its contribution to transferability after the remaining optimization. We therefore define operation value in terms of terminal transfer utility.

Consider a joint state z_t and an intervention window of length L, where $1 \le L \le T - t$. Starting from the same state, the intervention trajectory applies R^L within the window, while the reference trajectory applies B^L. Both trajectories then follow the same continuation rule, with the same remaining number of iterations and perturbation constraints. When switching from R to B after the intervention window, the current adversarial input and persistent optimizer state are carried forward without reinitialization; only the update rule changes, and the method-specific refinement is no longer executed. Let x_T^R and x_T^B denote their terminal inputs, with the superscripts indicating only the operations applied within the window. We define

$$\mathrm{AOV}_{t,L}(z_t) = \mathbb{E}\big[U(x_T^R) - U(x_T^B) \mid z_t\big]. \tag{3}$$

Here, larger values of U indicate stronger terminal attack performance, and the expectation is over attack randomness. Positive AOV indicates a terminal benefit from the method-specific updates within the window, whereas negative AOV indicates reduced utility.

Window AOV compares a bounded intervention followed by a common continuation rule; suffix AOV compares alternative updates over the entire remaining trajectory. We estimate terminal ASR differences on fixed analysis targets, including downstream effects.

3.3 Temporal Structure of Attack Operation Value

This section and Section 3.4 study CWA (Chen et al., 2024) on NIPS2017 ($\epsilon = 4/255$) and ModelNet40 ($\epsilon = 0.06$), using three analysis targets per modality. These panels differ from method evaluation; Appendix A.1 specifies the protocols.

We fix the attack length to $T = 10$ and compare the final returns of different placements and continued use of method-specific updates. Computational cost is measured by a reference count of surrogate input gradient computations, denoted by graphs. In the five surrogate CWA setting, each B and R requires 5 and 10 graphs, respectively.

Window AOV at different positions. To test whether the return on the same additional computation varies with its position, we fix $L = 2$ and place the two method-specific updates in the early, middle, or late stage: Early (R^2B^8), Middle ($B^4R^2B^4$), and Late (B^8R^2).

All three schedules contain two R updates and eight B updates, with identical outer iteration counts and total costs of 60 graphs. Only the positions of the two R updates differ.

The reference B^{10} costs 50 graphs. Each paired intervention replays the same base prefix, changes only the two window updates, and follows the same base continuation rule. The terminal ASR difference estimates window AOV, with equal additional computation across positions (Appendix B.1).

Suffix AOV after early updates. The positive return of early method-specific updates raises a further question: should these updates continue throughout the remaining attack? We compare Early-K2, R^2B^8, with the full schedule R^{10}. Both trajectories share the prefix R^2; from the resulting state z_2, they apply either B^8 or R^8.

This comparison corresponds to the suffix AOV in Equation (3) with $t = 2$ and $L = 8$. The intervention covers all remaining iterations, so no continuation remains beyond the window. Its value is the terminal utility of Full minus that of Early-K2, computed before rounding.

As shown in Table 2, Full incurs more method-specific computation but achieves lower final transferability than Early-K2, giving a negative suffix return in both modalities. After the early investment, replacing all remaining base ensemble updates with expensive updates reduces final performance in these settings.

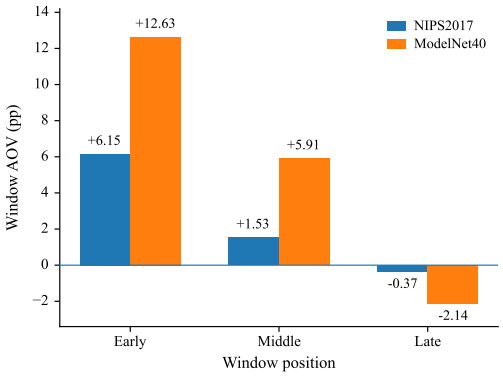


Figure 2: Window AOV relative to B^{10} across attack stages.

Table 2: Suffix comparison after the shared R^2 prefix. ASR (%) is also shown for the base-only schedule B^{10}; Suffix AOV is Full minus Early-K2 (pp).

Updates	Graphs	NIPS2017	ModelNet40
B^{10}	50	41.55	65.86
R^2B^8	60	**47.70**	**78.49**
R^{10}	100	43.45	68.70
Suffix AOV		-4.26	-9.79

In these CWA settings, the positive value of early expensive updates differs from the value of continuing them after the early investment. Expensive updates should therefore be used selectively, rather than throughout the attack simply because they are effective at a particular stage. We next examine how limited computation should be allocated between these updates and subsequent base ensemble optimization.

3.4 Allocation between Refinement and Continuation

The CWA case study motivates early placement but leaves a second question: how should a fixed budget be divided between refinement and base continuation?

Fixed budget allocation. For CWA, $c_B = 5$ and $c_R = 10$. With 60 reference graphs and k early method-specific updates, we use $12 - 2k$ base updates and $T_k = 12 - k$ outer iterations:

$$R^k B^{12-2k}, \qquad 10k + 5(12 - 2k) = 60, \quad k = 0, \ldots, 6. \tag{4}$$

Thus, $k = 0$ is the base-only schedule B^{12}, while $k = 6$ spends the entire budget on refinement. All configurations initially use the same outer step size.

Figure 3 gives the highest mean ASR at $K = 2$ for images and $K = 3$ for point clouds. Both retain substantial base continuation and outperform $K = 6$. $K = 1$ also falls below the best allocation: simply minimizing refinement is insufficient.

Step size rescaling. Because the schedules differ in length, we additionally match their cumulative nominal outer step size to the original ten iteration attack:

$$\alpha_k = \frac{10}{T_k}\alpha = \frac{10}{12 - k}\alpha. \tag{5}$$

The perturbation budget, CWA reverse neighborhood radius, and inner step size remain unchanged. K2 retains its original step size. After rescaling, the highest mean ASR still occurs at K2 for images and K3 for point clouds, and both outperform K6. In these CWA settings, the benefit of retaining base continuation therefore persists after this control. These observations motivate allocating a limited early investment to refinement while preserving subsequent inexpensive optimization.

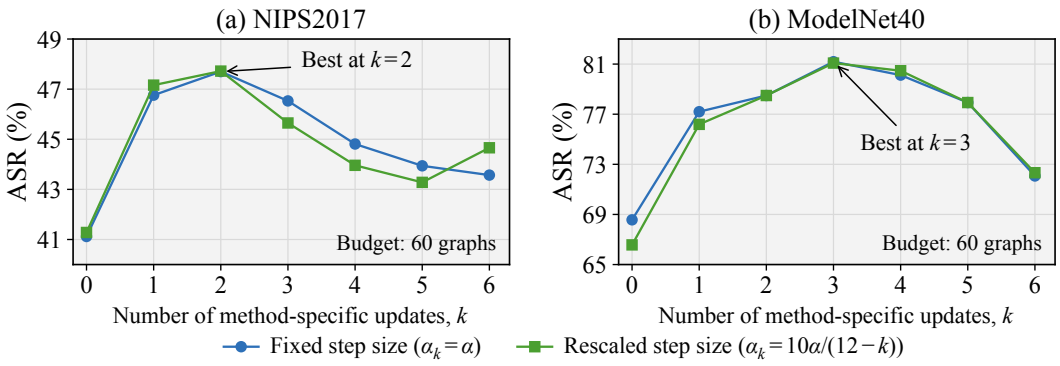


Figure 3: Allocation under 60 reference graphs: $R^k B^{12-2k}$. Rescaled uses $\alpha_k = 10\alpha/(12-k)$.

3.5 Temporal Ensemble Scheduling

Motivated by these findings, TES concentrates method-specific updates at the beginning of the attack while preserving the original number of outer iterations. Given an attack length T and a method-specific update count K, it uses

$$a_t = \begin{cases} R, & 0 \le t < K, \\ B, & K \le t < T. \end{cases} \tag{6}$$

Starting from $z_0 = (x, s_0)$, TES follows Equation (2). The switch from R to B preserves the current input and persistent optimizer state; internal updates and resets follow the original method without restarting the trajectory.

Table 3: CWA across attack lengths on ModelNet40.

	Prefix K / ASR (%)			Retained (%)	
T	20%	30%	Best tested	20%	30%
10	2 / 67.54	3 / 68.72	3 / 68.72	98.29	100.00
20	4 / 70.40	6 / 71.72	8 / 72.11	97.63	99.47
30	6 / 72.04	9 / 73.02	12 / 73.40	98.14	99.48
40	8 / 72.50	12 / 73.64	16 / 73.82	98.21	99.75

Tested K grids: $T10$: $0, 1, 2, 3, 4, 5, 10$; $T20$: $0, 2, 4, 6, 8, 20$; $T30$: $0, 2, 3, 6, 9, 12, 30$; $T40$: $0, 2, 4, 8, 12, 16, 40$.

TES decouples the number of method-specific updates from the outer attack length. If the reference costs of the two updates are c_B and c_R, its total cost is

$$C_{\mathrm{TES}}(K) = (T-K)c_B + Kc_R = Tc_B + K(c_R - c_B). \tag{7}$$

The first term is the base trajectory cost; the second is the refinement investment. Full costs Tc_R.

Choosing the prefix length. The CWA analysis in Table 3 suggests that setting K between $0.2T$ and $0.3T$ preserves most of the peak performance observed in the CWA experiments, whereas applying the method-specific update throughout the attack rarely yields the best performance. Based on this observation, we adopt $K = 0.2T$ as a common configuration for the following experiments, corresponding to $K = 2$ when $T = 10$. The experimental evaluations and ablation studies in Section 4 assess whether this observation generalizes across different methods and parameter settings.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate NIPS2017 (Kurakin et al., 2018), ImageNet-Val (Russakovsky et al., 2015), ImageNet-Sketch (Wang et al., 2019a), ModelNet40 (Wu et al., 2015), ScanObjectNN PB_T50_RS (Uy et al., 2019), and the evaluated ShapeNet HDF5 split (Chang et al., 2015). Dataset details and sample selection are provided in Appendix A.1.

Evaluation models. We evaluate transferability on 12 standard black-box models for each image dataset and 19, 14, and 19 for ModelNet40, ScanObjectNN, and ShapeNet, respectively. Architecture references are provided in Appendix A.1. Image defenses include FastAT-R50 (Wong et al., 2020), JPEG Q75 (Guo et al., 2018), HGD (Liao et al., 2018), and DiffPure SDE150 (Nie et al., 2022). For point clouds, we evaluate SOR, SRS, and DUP-Net (Zhou et al., 2019).

Table 4: NIPS2017 ASR (%) at $\epsilon = 4/255$. Bold marks the higher value in each Original/+Ours pair; ties are unbolded.

Method	Version	*Standard models*							*Defended models*			
		ResNet 152	DenseNet 201	VGG 19	AlexNet	MNASNet 1.0	ViT B/16	MaxViT T	FastAT R50	JPEG Q75	HGD	DiffPure SDE150
SVRE	Original	56.72	73.18	90.13	**16.64**	70.27	20.02	55.56	**2.76**	25.37	13.39	2.87
	+ Ours	**60.92**	**75.43**	**92.38**	15.85	**73.65**	**22.35**	**60.64**	2.43	**26.65**	**14.64**	**3.42**
SMER	Original	56.51	72.75	91.37	16.48	71.96	21.29	54.93	2.43	25.37	13.49	3.09
	+ Ours	**60.00**	**76.82**	**92.04**	**16.96**	**75.90**	**23.52**	**61.16**	**2.59**	**26.54**	**14.54**	**3.20**
CWA	Original	44.82	66.63	**91.03**	**16.32**	69.03	17.06	47.35	2.27	20.91	10.77	3.86
	+ Ours	**54.87**	**72.10**	88.90	16.01	**70.38**	**19.92**	**55.24**	2.27	**23.04**	**11.51**	**3.97**

Table 5: Image results at $\epsilon = 4/255$. Time: seconds/sample; speedup: Original/+Ours runtime.

Method	Version	NIPS2017			ImageNet-Val			ImageNet-Sketch		
		ASR ↑	Time ↓	Speedup ↑	ASR ↑	Time ↓	Speedup ↑	ASR ↑	Time ↓	Speedup ↑
SVRE	Original	53.88	1.5185	1.00×	56.18	0.882	1.00×	63.09	0.876	1.00×
	+ Ours	**56.80**	0.4484	3.39×	**58.36**	0.293	3.01×	**64.53**	0.292	3.00×
SMER	Original	54.77	2.6541	1.00×	54.86	1.867	1.00×	63.66	1.804	1.00×
	+ Ours	**57.54**	0.6803	3.90×	**57.74**	0.496	3.76×	**64.95**	0.476	3.79×
CWA	Original	50.01	0.3980	1.00×	50.66	0.293	1.00×	61.82	0.292	1.00×
	+ Ours	**53.28**	0.2258	1.76×	**54.17**	0.172	1.70×	**62.31**	0.171	1.71×

Methods and schedules. We apply TES to SVRE (Xiong et al., 2022), SMER (Tang et al., 2024), and CWA (Chen et al., 2024). In the main comparison, Original is Full (R^{10}) and Ours is TES (R^2B^8). Implementation details and attack parameters are provided in Appendix A.2.

Evaluation metrics and runtime. We use attack success rate (ASR) as the primary evaluation metric, computed on each target's clean-correct test samples and averaged equally over the standard target panel. Runtime is measured in seconds per sample on 48 fixed samples under the dataset-specific protocols in Appendix E.2.

4.2 Experimental Results

Image classification. Tables 4 and 5 report the black-box transfer results and attack time on image classification tasks. Across three datasets and three ensemble attacks, replacing full method-specific updates with TES substantially reduces attack time while maintaining, and in most cases improving, average transferability. In other words, reducing expensive ensemble updates in later iterations does not result in a simple performance–efficiency trade-off. Instead, on image tasks, a short prefix of method-specific updates simultaneously achieves higher transferability and lower runtime. Importantly, this trend is not limited to standard black-box models. TES also generally maintains or improves performance against input defenses such as JPEG, HGD, and DiffPure, while the results on FastAT exhibit some fluctuations. Per-target standard, robust, and defense results are given in Appendices C.1 and C.3.

Point-cloud classification. Tables 6 and 7 report the corresponding black-box transfer results and attack time on point-cloud classification tasks. CWA largely follows the same pattern as on images: shortening the interval of method-specific refinement yields more favorable overall transferability across point-cloud datasets while reducing runtime. In contrast, SMER largely preserves the transferability of Full despite a substantial reduction in computation, whereas SVRE exhibits a clearer effectiveness–efficiency trade-off, with a modest loss in transferability in exchange for significant computational savings. This method-dependent behavior is consistently observed across the three point-cloud datasets. Per-target point-cloud results are provided in Appendices C.2 and C.3.

Table 6: ModelNet40 ASR (%) at $\epsilon = 0.06$. Defenses average five targets. PN++: PointNet++; PMLP: PointMLP; PT: Point Transformer. Bold marks pairwise higher ASR.

Method	Version	*Standard models*							*Defended models*		
		PointNet	PN++ SSG	PMLP Elite	DGCNN	RepSurf	PT v2	SimpleView	SOR	SRS	DUP-Net
SVRE	Original	32.84	86.71	90.27	**84.33**	79.05	89.79	34.61	**80.05**	**77.13**	**78.38**
	+ Ours	**34.38**	**87.05**	**91.59**	84.16	**81.25**	**89.89**	**34.70**	79.09	76.49	76.36
SMER	Original	**36.13**	89.77	**94.11**	**90.59**	80.72	92.45	**44.87**	**83.75**	**80.20**	**83.19**
	+ Ours	35.19	**90.04**	94.06	88.53	**84.08**	**92.93**	39.26	82.27	79.89	80.32
CWA	Original	**32.38**	89.37	91.84	81.26	80.64	**92.01**	28.30	76.45	77.63	74.33
	+ Ours	31.84	**90.44**	**92.45**	**85.46**	**81.78**	91.58	**34.82**	**79.93**	**78.12**	**77.17**

Table 7: Point cloud results at $\epsilon = 0.06$. Time: seconds/sample; speedup: Original/+Ours runtime.

Method	Version	ModelNet40			ScanObjectNN			ShapeNet		
		ASR ↑	Time ↓	Speedup ↑	ASR ↑	Time ↓	Speedup ↑	ASR ↑	Time ↓	Speedup ↑
SVRE	Original	**66.85**	3.8231	1.00×	**86.40**	3.656	1.00×	**55.37**	3.680	1.00×
	+ Ours	**67.59**	1.2178	3.14×	85.63	1.067	3.43×	54.04	1.067	3.45×
SMER	Original	**71.48**	6.327	1.00×	86.42	6.275	1.00×	**59.12**	6.297	1.00×
	+ Ours	70.39	1.579	4.01×	**87.82**	1.548	4.05×	58.68	1.534	4.11×
CWA	Original	65.40	0.762	1.00×	75.49	0.731	1.00×	51.01	0.745	1.00×
	+ Ours	**67.54**	0.459	1.66×	**84.09**	0.421	1.74×	**53.64**	0.436	1.71×

Overall, the benefits of early refinement are more concentrated on image tasks, where later expensive updates enter a diminishing-return, and in some cases negative-return, regime more rapidly. On point clouds, early refinement remains highly valuable, but for some methods the high-value interval extends farther along the attack trajectory, making the switch from method-specific updates to base updates at $K = 2$ potentially premature. We also note that the performance difference between TES and the corresponding Full method typically remains within only a few percentage points. This is consistent with the role of TES: it is a temporal scheduling strategy. Its goal is to make an existing ensemble attack execute more efficiently, while its final attack strength remains largely determined by the underlying method itself.

4.3 Ablation and Sensitivity Studies

To further analyze TES, we study three factors: attack horizon, the number of method-specific updates, and the allocation of computation between refinement and base optimization. Additional ablations are provided in Appendix D.

Temporal allocation of method-specific updates. Figure 4 examines the effect of varying the number of method-specific updates while keeping the total attack length fixed. The overall trend is not that transferability continually improves as more refinement is introduced. Starting from the base attack, a small number of method-specific updates typically produces a rapid improvement, whereas the marginal benefit decreases as refinement is further extended, eventually saturating or even declining in some settings. The amount of refinement required to reach strong performance also differs across methods: the benefit of CWA is more concentrated in a short prefix, while SMER and SVRE can continue to benefit from a somewhat longer refinement interval.

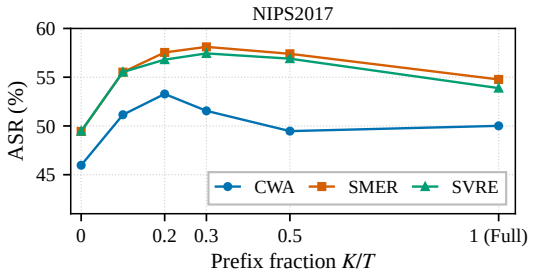


Figure 4: NIPS2017 prefix sensitivity, $T = 10$. Only tested prefixes are marked.

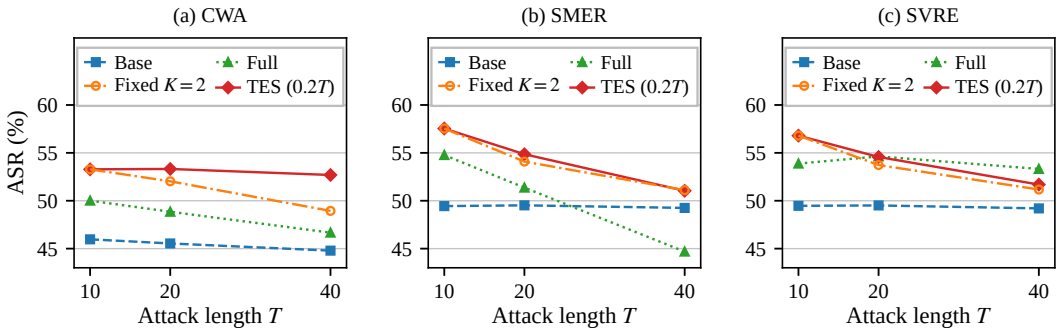


Figure 5: NIPS2017 attack-length sensitivity.

These results indicate that the benefit of method-specific computation does not accumulate uniformly along the attack trajectory. If a method-specific update retained the same advantage throughout optimization, increasing K should continually favor Full. Instead, the observed diminishing returns and eventual performance degradation suggest that the additional terminal contribution of the same refinement changes as the attack state evolves. This behavior is consistent with the trajectory-dependent value characterized by AOV.

Sensitivity to attack horizon. Figure 5 varies the attack horizon to examine whether the observed temporal structure is specific to ten-step attacks. As the trajectory becomes longer, keeping a fixed number of early refinement steps becomes progressively less sufficient, while expanding the refinement prefix generally recovers additional transferability. This suggests that the high-value region should not be interpreted as an absolute number of early iterations, such as the first two steps, but rather as an early stage of the attack trajectory. When the trajectory is extended, the effective refinement interval may expand accordingly.

Comparison with extended base optimization. The preceding prefix analysis shows that the marginal benefit of later refinement can decrease, but a simpler explanation remains possible: a similar computational budget might instead be spent on additional base updates. We therefore compare TES with longer base-only trajectories in Figure 6, considering both the original step size and a step size rescaled according to the longer attack length. On image tasks, allocating the additional budget entirely to extended base optimization does not reproduce the gains of TES. Thus, the advantage of TES cannot be explained simply by performing fewer expensive updates and replacing them with more inexpensive iterations. Instead, the results suggest that early method-specific refinement and subsequent base continuation play different roles. Early refinement modifies the attack state during a high-value region of the trajectory, while base updates continue the remaining optimization from the resulting state.

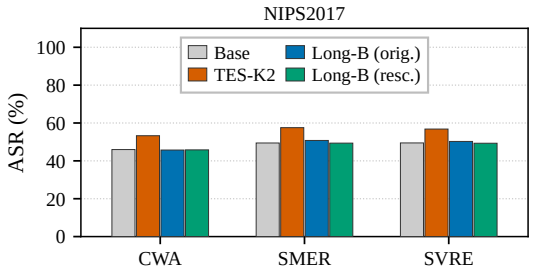


Figure 6: NIPS2017 base controls. Orig./resc.: original/rescaled step size.

5 Conclusion

In this paper, we rethink ensemble updates in transfer-based black-box adversarial attacks. Our systematic analysis shows that the value of method-specific updates varies along the attack trajectory, and that applying refinement throughout the entire optimization is often suboptimal, with early refinement yielding higher returns than continued refinement. Based on this insight, we propose Temporal Ensemble Scheduling (TES), which concentrates expensive ensemble updates in early iterations while retaining baseline optimization later. Across image and 3D point cloud datasets, TES substantially reduces runtime while maintaining competitive transferability.

AI USE STATEMENT

We used generative AI tools to assist with drafting and polishing manuscript text, translation, and literature retrieval and discovery, including identifying relevant related work. AI tools were also used to assist with editing figures and tables and formatting LaTeX and references. In addition, generative AI tools were used to assist with experimental implementation and execution, including supporting code development, debugging, and the organization and verification of experimental outputs. The authors reviewed and verified all AI-assisted experimental procedures and results and are responsible for all claims, citations, results, and the final content of the paper.

ETHICS STATEMENT

This work investigates transfer-based adversarial attacks to support controlled evaluation of model robustness. More efficient attacks can also facilitate misuse against deployed systems. The proposed method is intended for authorized robustness assessment; its results should not be interpreted as permission to attack systems without authorization.

REPRODUCIBILITY STATEMENT

Section 3 specifies the attack dynamics, AOV formulation and interventions, compute allocation, and the TES scheduling rule, while Section 4 summarizes the datasets, evaluation models, matched attack settings, metrics, and runtime measurements used in the main experiments. Appendix A provides the core reproduction protocol: Appendix A.1 documents the datasets, surrogate and target model panels, and model sources; Appendix A.2 specifies preprocessing, native attack transitions, state handling, step sizes, batch sizes, random seeds, and random-number streams, with Algorithm A1 giving the TES procedure and Table 10 summarizing the default batch and schedule settings; Appendix A.3 defines the clean-correct evaluation masks, macro-ASR aggregation, paired comparisons, and uncertainty estimation.

The controlled mechanism analyses are documented in Appendix B, including the window and suffix interventions, fixed-budget allocation, and optimizer-state controls. Appendix D reports additional sensitivity and stability experiments, including prefix-length and horizon studies, surrogate-composition controls, attack-seed repetitions, and the fixed random scheduling masks. Appendix C provides target-level standard, robust, and defense results. Appendix E documents reference computation, the hardware and software environment, runtime measurement scope, repeated timing procedures, shared-load conditions, and timing-quality criteria. The known native CUDA nondeterminism and the scope of the exact adapter checks are explicitly stated in Appendix A.2.

REFERENCES

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. arXiv preprint arXiv:1512.03012, 2015.

Bin Chen, Jia-Li Yin, Shukai Chen, Bo-Hao Chen, and Ximeng Liu. An Adaptive Model Ensemble Adversarial Attack for Boosting Adversarial Transferability. In *IEEE/CVF International Conference on Computer Vision*, 2023.

Huanran Chen, Yichi Zhang, Yinpeng Dong, Xiao Yang, Hang Su, and Jun Zhu. Rethinking Model Ensemble in Transfer-based Adversarial Attacks. In *International Conference on Learning Representations*, 2024.

Edoardo Debenedetti, Vikash Sehwag, and Prateek Mittal. A Light Recipe to Train Robust Vision Transformers. In *IEEE Conference on Secure and Trustworthy Machine Learning*, 2023.

Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. In *IEEE/CVF International Conference on Computer Vision*, 2021.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting Adversarial Attacks with Momentum. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, and Dimitris Tsipras. Robustness (Python Library). 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting Point Cloud Shape Classification with a Simple and Effective Baseline. In *International Conference on Machine Learning*, 2021.

Martin Gubri, Maxime Cordy, Mike Papadakis, Yves Le Traon, and Koushik Sen. LGV: Boosting Adversarial Example Transferability from Large Geometric Vicinity. In *European Conference on Computer Vision*, 2022.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. In *International Conference on Learning Representations*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

Zichao Hu, Heng Li, Liheng Yuan, Zhang Cheng, Wei Yuan, and Ming Zhu. Model scheduling and sample selection for ensemble adversarial example attacks. *Pattern Recognition*, 130:108824, 2022.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *IEEE/CVF International Conference on Computer Vision*, 2019.

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv preprint arXiv:1602.07360, 2016.

Karim Knaebel, Jonas Schult, Alexander Hermans, and Bastian Leibe. Point2Vec for Self-Supervised Representation Learning on Point Clouds. In *German Conference on Pattern Recognition*, 2023.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 2012.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, Alan Yuille, Sangxia Huang, Yao Zhao, Yuzhe Zhao, Zhonglin Han, Junjiajia Long, Yerkebulan Berdibekov, Takuya Akiba, Seiya Tokui, and Motoki Abe. Adversarial Attacks and Defences Competition. arXiv preprint arXiv:1804.00097, 2018.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution On $\mathcal{X}$-Transformed Points. In *Advances in Neural Information Processing Systems*, 2018.

Yingwei Li, Song Bai, Yuyin Zhou, Cihang Xie, Zhishuai Zhang, and Alan Yuille. Learning Transferable Adversarial Examples via Ghost Networks. In *AAAI Conference on Artificial Intelligence*, 2020.

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-box Attacks. In *International Conference on Learning Representations*, 2017.

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-Shape Convolutional Neural Network for Point Cloud Analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *IEEE/CVF International Conference on Computer Vision*, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *European Conference on Computer Vision*, 2018.

Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework. In *International Conference on Learning Representations*, 2022.

Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion Models for Adversarial Purification. In *International Conference on Machine Learning*, 2022.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017a.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*, 2017b.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing Network Design Spaces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Haoxi Ran, Jun Liu, and Chengjie Wang. Surface Representation for Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do Adversarially Robust ImageNet Models Transfer Better? In *Advances in Neural Information Processing Systems*, 2020.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Bowen Tang, Zheng Wang, Yi Bin, Qi Dou, Yang Yang, and Heng Tao Shen. Ensemble Diversity Facilitates Adversarial Transferability. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, 2021.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-Axis Vision Transformer. In *European Conference on Computer Vision*, 2022.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In *IEEE/CVF International Conference on Computer Vision*, 2019.

Haohan Wang, Songwei Ge, Zachary C. Lipton, and Eric P. Xing. Learning Robust Global Representations by Penalizing Local Predictive Power. In *Advances in Neural Information Processing Systems*, 2019a.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 2019b.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point Transformer V2: Grouped Vector Attention and Partition-based Pooling. In *Advances in Neural Information Processing Systems*, 2022.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015.

Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the Cloud: Learning Curves for Point Clouds Shape Analysis. In *IEEE/CVF International Conference on Computer Vision*, 2021.

Yifeng Xiong, Jiadong Lin, Min Zhang, John E. Hopcroft, and Kun He. Stochastic Variance Reduced Ensemble Adversarial Attack for Boosting the Adversarial Transferability. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Xincheng Yang, Mingze Jin, Weiji He, and Qian Chen. PointCAT: Cross-Attention Transformer for Point Cloud. arXiv preprint arXiv:2304.03012, 2023.

Wei Yao, Zeliang Zhang, Huayi Tang, and Yong Liu. Understanding Model Ensemble in Transferable Adversarial Attack. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Renrui Zhang, Liuhui Wang, Yali Wang, Peng Gao, Hongsheng Li, and Jianbo Shi. Starting From Non-Parametric Networks for 3D Point Cloud Analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point Transformer. In *IEEE/CVF International Conference on Computer Vision*, 2021.

Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. DUP-Net: Denoiser and Upsampler Network for 3D Adversarial Point Clouds Defense. In *IEEE/CVF International Conference on Computer Vision*, 2019.

Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A Self-supervised Approach for Adversarial Robustness. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating Adversarial Effects Through Randomization. In *International Conference on Learning Representations*, 2018.

Weilin Xu, David Evans, and Yanjun Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Network and Distributed System Security Symposium*, 2018.

A Experimental Protocols and Implementation Details

A.1 Datasets and Evaluation Models

Table 8 summarizes the evaluated datasets. The sample counts refer to the evaluated collections, rather than the denominator of every target's ASR. Each target is evaluated on its own clean-correct subset, as specified in Appendix A.3. ShapeNet denotes the evaluated HDF5 split. The supplementary NIPS2017 and ModelNet40 experiments use the same 1,000 and 2,468 registered samples, respectively.

For ImageNet-Sketch, the raw collection contains 50,889 images. We evaluate a fixed class-balanced subset of 50,000 images: for each of the 1,000 classes, the frozen sample manifest retains files `sketch_0.JPEG` through `sketch_49.JPEG`, excluding the remaining 889 images. The same subset is used for all three methods, Full and TES, both perturbation budgets, and the original and extension target panels. Thus, 50,000 denotes the evaluated subset size, not the raw dataset size; target-specific ASR denominators further restrict this subset to clean-correct samples.

Table 8: Datasets and standard black-box evaluation panels. Defended configurations are excluded from the standard-target counts.

Modality	Dataset	Evaluated samples	Standard targets
Image	NIPS2017	1,000	12
Image	ImageNet-Val	50,000	12
Image	ImageNet-Sketch (balanced subset)	50,000	12
Point cloud	ModelNet40	2,468	19
Point cloud	ScanObjectNN PB_T50_RS	2,882	14
Point cloud	ShapeNet HDF5 split	10,261	19

Mechanism versus method evaluation. The original mechanism analyses in Section 3.3 use three targets per modality. Their panels differ from the method-evaluation panels; their ASR values are therefore not pooled. Table 9 specifies the default surrogate ensembles and the original analysis targets. Expanded mechanism panels and reduced surrogate ensembles are specified alongside their corresponding experiments.

Table 9: Default surrogate ensembles and original mechanism targets. Model-family references also apply to the corresponding variants in the target lists below.

Setting	Models
Image surrogates	ResNet-50 (He et al., 2016), DenseNet-121 (Huang et al., 2017), ConvNeXt-Tiny (Liu et al., 2022), Swin-T (Liu et al., 2021), and VGG-16 (Simonyan & Zisserman, 2015).
Image mechanism targets	MobileNetV3-Large (Howard et al., 2019), DeiT-Small/16 (Touvron et al., 2021), and Inception-v3 (Szegedy et al., 2016).
ModelNet40 method surrogates	CurveNet (Xiang et al., 2021), PointCNN (Li et al., 2018), PointConv (Wu et al., 2019), RSCNN (Liu et al., 2019), and Point2Vec (Knaebel et al., 2023).
ModelNet40 mechanism surrogates	PointNet (Qi et al., 2017a), DGCNN (Wang et al., 2019b), PointMLP (Ma et al., 2022), and PointNet++ SSG/MSG (Qi et al., 2017b).
ModelNet40 mechanism targets	CurveNet, PointCNN, and Point2Vec.
ScanObjectNN / ShapeNet method surrogates	DGCNN, PointNet++ MSG/SSG, PointMLP, and CurveNet.

Standard image targets. The 12-target panel comprises ResNet-152, DenseNet-201, VGG-19, AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2015), SqueezeNet 1.1 (Iandola et al., 2016), ShuffleNetV2 $\times$1.0 (Ma et al., 2018), MnasNet 1.0 (Tan et al., 2019), EfficientNet-B0 (Tan & Le, 2019), RegNetY-400MF (Radosavovic et al., 2020), ViT-B/16 (Dosovitskiy et al., 2021), and MaxViT-T (Tu et al., 2022). The incremental image method ablations use this standard panel.

Standard point-cloud targets. ModelNet40 uses DGCNN, DGCNN-V, PT_Hengshuang, PT_Menghao, Point-PN (Zhang et al., 2023), PointCAT (Yang et al., 2023), PointMLP-Elite, PointNet, PointNet++ SSG, Point Transformer v1-Cls26/v1-Cls38/v1-Cls50/v2, RepSurf, RepSurf ($2\times$),

SimpleView, VN-DGCNN, VN-PointNet, and VN-Transformer. Relevant model families include Point Transformer (Zhao et al., 2021), Point Transformer V2 (Wu et al., 2022), RepSurf (Ran et al., 2022), SimpleView (Goyal et al., 2021), and Vector Neurons (Deng et al., 2021). ScanObjectNN uses PT_Hengshuang, PT_Menghao, Point-PN, PointCAT, PointConv, PointMLP-Elite, Point Transformer v1-Cls26/v1-Cls38/v1-Cls50/v2, RSCNN, RepSurf, RepSurf ($2\times$), and SimpleView. ShapeNet adds DGCNN-V, PointNet, VN-DGCNN, VN-PointNet, and VN-Transformer to this 14-target list.

Defended evaluations. Standard models, adversarially trained models, and individual defense configurations are reported separately. The supplementary defense comparisons retain TES and Full; incremental sensitivity and seed experiments use standard targets. Existing point-cloud SOR/SRS/DUP-Net evaluations cover five victim models.

A.2 Implementation and Attack Settings

TES changes the temporal choice between the base update B and the method-specific update R. For CWA, SMER, and SVRE, R retains the corresponding native procedure (Chen et al., 2024; Tang et al., 2024; Xiong et al., 2022); B is its registered base ensemble transition. Method-internal operations, stochastic rules, and resets retain their original definitions. At the scheduling boundary, TES preserves the current adversarial input and persistent optimizer state instead of initializing a new attack. Algorithm A1 states this shared scheduling rule without replacing the native transitions.

Perturbation coordinates and preprocessing. Both modalities use an untargeted, coordinatewise ℓ_∞ constraint. Images are converted to RGB, resized to 224×224 using bilinear interpolation with antialiasing, and represented in $[0, 1]$. Perturbations are applied before the checkpoint-specific input normalization. Thus $\Pi_{\mathcal{C}}(z) = \mathrm{clip}_{[0,1]}(x + \mathrm{clip}(z - x, -\epsilon, \epsilon))$ for images. The image budgets $4/255$ and $8/255$ refer to these pixel coordinates.

For the three point-cloud datasets, deterministic farthest-point sampling selects 1,024 points from the 2,048-point input, starting at index zero. The sampled cloud is centered and divided by its maximum Euclidean radius: $x_i = (p_i - \bar{p})/\max_j \|p_j - \bar{p}\|_2$. This preprocessing is performed once and cached before attacking. Budgets 0.06 and 0.12 apply to the resulting normalized coordinates. Projection is $\Pi_{\mathcal{C}}(z) = x + \mathrm{clip}(z - x, -\epsilon, \epsilon)$; no additional unit-sphere normalization is performed after each attack step. Model-specific forward adapters retain their checkpoint input conventions. Sampling, source order and preprocessing are shared by the compared schedules.

Common outer transition and base update. Let f_i denote source logits, L cross-entropy, and N_1, N_2, and N_{ma} normalization by the per-example sum of absolute values, Euclidean norm, and mean absolute value, respectively (denominators are floored at 10^{-12}). All attacks start from $x_0 = x$, with zero outer momentum m_0, and use

$$m_{t+1} = \mu m_t + d_t, \qquad x_{t+1} = \Pi_{\mathcal{C}}(x_t + \alpha\ \mathrm{sign}(m_{t+1})). \tag{8}$$

For CWA, $g_t = \nabla_{x_t} L(\sum_i f_i(x_t), y)$ and a B update uses $d_t = \mathsf{N}_1(g_t)$. For SMER and SVRE, the base gradient instead differentiates the mean source logits, $g_t = \nabla_{x_t} L(M^{-1} \sum_i f_i(x_t), y)$, and $d_t = \mathsf{N}_{\mathrm{ma}}(g_t)$. These distinct logit aggregations are preserved.

CWA refinement and state. An R update first sets $z_0 = \Pi_{\mathcal{C}}(x_t - (\epsilon/15)\,\mathrm{sign}(g_t))$. It then visits each source once in the order listed in Appendix A.1, updating the persistent inner momentum u and inner input by

$$u \leftarrow \mu u + \mathsf{N}_2(\nabla_z L(f_i(z), y)), \qquad z \leftarrow \Pi_{\mathcal{C}}(z + 250u). \tag{9}$$

After the final source, $d_t = \mathsf{N}_1(z - x_t)$ is used in Eq. 8. The coefficient 250 is the native inner step multiplier, not a number of inner iterations. Both u and m start at zero and persist across outer iterations. During B updates, u is retained but not updated; switching back to R resumes it. The outer step size is $\alpha = 2\epsilon/T$.

SMER refinement and state. We follow the authors' official MI-FGSM-based SMER implementation. Its ensemble-reweighting component, described as reinforcement learning in the original paper (Tang et al., 2024), updates persistent source weights by SGD. The weights are initialized as $w_i = 1$.

Each R update initializes $z = x_t$ and $v = 0$, then visits the M sources according to four independently shuffled NumPy permutations. For each selected source i, we first compute

$$g = \nabla_z L(w_i f_i(z), y). \tag{10}$$

Using detached source logits, the weight-update objective and SGD step are

$$\begin{aligned} \ell_w &= -\log\left[\max\left\{L\left(\frac{1}{M}\sum_{j=1}^{M} w_j \ \mathrm{stopgrad}(f_j(z)), y\right), 10^{-12}\right\}\right], \\ w &\leftarrow w - 0.02\left(\nabla_w \ell_w + 0.002w\right). \end{aligned} \tag{11}$$

The SGD optimizer uses no momentum. Logit detachment and the loss floor are implementation details of our evaluator. The previously computed input gradient g then updates the inner trajectory:

$$v \leftarrow v + \mathsf{N}_{\mathrm{ma}}(g), \qquad z \leftarrow \Pi_{\mathcal{C}}\left(z + \alpha \,\mathrm{sign}(v)\right). \tag{12}$$

After all $4M$ inner steps, the outer direction is $d_t = \mathsf{N}_{\mathrm{ma}}(v)$, with step size

$$\alpha = \begin{cases} \epsilon/T, & \text{images,} \\ 1.5\epsilon/T, & \text{point clouds.} \end{cases} \tag{13}$$

The temporary inner input z and accumulation v are reset at each R update. Source weights and outer momentum persist across outer iterations, including intervening B updates. A B update uses the unweighted mean-logit gradient and leaves the source weights unchanged.

SVRE refinement and state. At every outer update compute the mean-logit snapshot gradient g_t. For an R update initialize $z = x_t$, $v = 0$, and perform 16 inner updates. Each update samples a source uniformly with replacement, computes

$$h = \nabla_z L(f_i(z), y) - \nabla_{x_t} L(f_i(x_t), y) + g_t, \qquad v \leftarrow v + \mathsf{N}_{\mathrm{ma}}(h), \tag{14}$$

and sets $z \leftarrow \Pi_{\mathcal{C}}(z + \alpha\,\mathrm{sign}(v))$. Use $d_t = \mathsf{N}_{\mathrm{ma}}(v)$ in the outer update. Inner momentum resets at every R update; outer momentum persists, including at scheduling boundaries. SVRE uses $\alpha = \epsilon/T$ and momentum coefficient one in both loops.

Parameters and randomness. The outer momentum coefficient is one for all image methods; for point clouds it is 0.9 for CWA/SMER and one for SVRE. Unless explicitly rescaled, the horizon studies use the native α above; the original-step extended-base controls retain the $T = 10$ value. Attack seeds are 17 for images and 31 for point clouds unless a seed study is specified. In the supplementary harness, Python, NumPy and Torch are seeded per attack batch with $1000003s + 10007q + 97b + 11$ for images and $1000003s + 97b$ for point clouds, where q, b are zero-based shard and batch indices. SMER retains its native NumPy permutation stream. SVRE uses a separate Torch source-selection stream: its image seed is 1000 at the default attack seed, and $1000 + s$ for the additional image seeds s; the point-cloud seed is $s + b \cdot 1000003$. The outer-step source-selection seed adds $t \cdot 1000003$. No RNG or persistent state is reset merely because the schedule switches between B and R.

The point-cloud CWA RSCNN CUDA backward is not bitwise deterministic. Adapter checks replay the same recorded native gradient sequence and require exact step inputs and outputs; this establishes conditional transition parity, not bitwise reproducibility of independent native CUDA executions. Formal runs retain the native operator.

The default method comparison uses $T = 10$ and $K = 2$, so the expensive prefix occupies 20% of the outer iterations. $K = 0$ and $K = T$ recover the base-only and Full schedules, respectively. Longer-horizon studies specify K explicitly.

Within each matched comparison, the perturbation constraint, sample identities, source checkpoints, attack batch size, and evaluation panel are held fixed unless the experiment explicitly varies that quantity. The corresponding horizon, extended-base, and rescaled step-size rules are reported with their results. Equal reference computation does not imply equal runtime; actual timing protocols are reported separately.

Algorithm A1 Temporal Ensemble Scheduling

Input: clean input x, label y, surrogate ensemble $\mathcal{F}$; feasible set $\mathcal{C}(x, \epsilon)$; horizon T; prefix length $K \in \{0, \dots, T\}$; native transitions $\mathcal{T}_{B,t}$ and $\mathcal{T}_{R,t}$; attack seed ω.
Output: adversarial input x_T.

```
Initialize (x_0, s_0) and the random-number generator using the original attack protocol and ω.
for t = 0, ..., T − 1 do
    Set a_t ← R if t < K; otherwise set a_t ← B.
    Obtain the native random inputs ξ_t required by the selected transition.
    (x_{t+1}, s_{t+1}) ← T_{a_t,t}(x_t, s_t; y, F, ξ_t).
    Retain (x_{t+1}, s_{t+1}) for the next iteration, including at the R-to-B boundary.
end for
return x_T.
```

Each native transition includes its prescribed step size, optimizer-state update, and projection onto $\mathcal{C}(x, \epsilon)$. No target-model feedback is used to select a_t.

Table 10: Attack batch sizes in the NIPS2017 and ModelNet40 method experiments. The default budgets are $4/255$ and 0.06, respectively.

Method	NIPS2017 batch	ModelNet40 batch	Default T	Default K
CWA	8	8	10	2
SMER	8	8	10	2
SVRE	8	6	10	2

A.3 Evaluation and Statistical Protocol

Attack success rate. For target f_j, let $\mathcal{C}_j = \{i : f_j(x_i) = y_i\}$ denote its clean-correct subset. We compute

$$\mathrm{ASR}_j = \frac{100}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} \mathbf{1}\left[f_j(x_i^{\mathrm{adv}}) \neq y_i\right], \qquad \mathrm{MacroASR} = \frac{1}{M} \sum_{j=1}^{M} \mathrm{ASR}_j, \tag{15}$$

where M is the number of targets in the specified reporting panel. Target models are equally weighted; standard and defended panels are not pooled. For stochastic defenses, clean and adversarial evaluation follow the recorded paired seeds and draws.

Paired comparisons. Compared schedules use the same samples, target panel, and clean-correct masks. Mechanism interventions additionally match the prescribed incoming attack state and random conditions; their branch and continuation rules are given with the corresponding analysis. Only complete shards enter an aggregate. Results generated under different mechanism/method panels are not combined into one ASR comparison.

Seeds and uncertainty. The method-stability experiments use image attack seeds 17/29/43 and point-cloud attack seeds 31/37/43, keeping the samples and reporting panels fixed. We report individual seed results and their mean and sample standard deviation. Paired sample-bootstrap intervals instead measure uncertainty conditional on the recorded attack seed and target panel: the supplementary records use 2,000 resamples, while the retained historical base-extension records use 4,000.

B Additional Analysis of Temporal Update Allocation

This appendix supplements the temporal analysis in the main text with placement comparisons, an expanded analysis panel, and controlled interventions on optimizer state. These experiments use the mechanism-analysis setting. Their ASRs are not pooled with the method-evaluation results, even when some target architectures overlap. Unless otherwise specified, the horizon is $T = 10$, and the perturbation budgets are $4/255$ on NIPS2017 and 0.06 on ModelNet40. All branches obey the same feasible set $\mathcal{C}(x, \epsilon)$.

B.1 Window and Suffix Interventions

For a two-update window beginning after t base updates, we compare

$$B^t R^2 B^{T-t-2} \quad \text{with} \quad B^T, \qquad t \in \{0, 4, 8\}. \tag{16}$$

The paired branches start from the same state after the shared base prefix. They differ only in the update rule inside the intervention window and subsequently apply the same base-continuation rule. Each branch carries forward its own resulting input and persistent optimizer state; the states are not forced to coincide after the intervention. We estimate window AOV from the difference in terminal ASR on the same target-specific clean-correct samples. Thus, the estimate includes downstream effects rather than only the immediate change in surrogate loss.

The suffix comparison answers a different question. After the shared prefix R^2, it compares $R^2 R^8$ with $R^2 B^8$. For the original three-target CWA analysis in the main text, Full minus Early is -4.26 percentage points on NIPS2017 and -9.79 points on ModelNet40. This negative suffix return does not contradict a positive early-window return: the branches start from different intervention states and replace different portions of the trajectory. It also does not imply that every later refinement update is individually harmful.

B.2 Allocation under a Fixed Reference Budget

The fixed-budget analysis changes both the number of refinement updates and the remaining base continuation. With five CWA surrogates, $c_B = 5$ and $c_R = 10$ reference graphs. The schedules

$$R^k B^{12-2k}, \qquad T_k = 12 - k, \qquad k = 0, \ldots, 6 \tag{17}$$

all cost 60 reference graphs. This comparison differs from a fixed-T prefix-length scan, whose reference cost increases with K. The original-step-size condition keeps the outer step size α fixed. The rescaled condition uses $\alpha_k = 10\alpha/(12 - k)$ to match the cumulative nominal outer step size of the original ten-update attack. The perturbation budget, reverse-neighborhood radius, and inner step size remain unchanged. Equal reference-graph counts do not imply equal wall-clock time.

Table 11: NIPS2017 fixed-budget allocation, ASR (%). Every schedule costs 60 reference graphs. The two step-size conditions share the same $k = 2$ reference.

k	MobileNetV3-L	DeiT-S/16	Inception-v3	Mean
Original step size				
0	61.50	23.24	38.62	41.12
1	68.22	27.08	45.00	46.77
2	70.93	25.37	46.81	47.70
3	69.41	24.84	45.21	46.49
4	66.81	23.67	43.94	44.80
5	65.18	24.09	42.55	43.94
6	64.43	23.88	42.45	43.58
Rescaled step size				
0	61.93	23.67	38.19	41.26
1	68.55	27.51	45.32	47.12
2	70.93	25.37	46.81	47.70
3	68.87	23.03	45.00	45.63
4	66.59	22.49	42.77	43.95
5	65.40	22.39	42.02	43.27
6	66.81	23.88	43.30	44.66

The best measured allocation is $k = 2$ on NIPS2017 under both step-size rules (47.70% ASR), whereas ModelNet40 peaks at $k = 3$ (81.08% with the original step size and 81.05% with rescaling). Increasing refinement to $k = 6$ reduces the original-step-size ASR to 43.58% and 72.04%, respectively. Thus, equal computation does not remove the importance of base continuation, and the preferred allocation is not identical across modalities.

Table 12: ModelNet40 fixed-budget allocation, ASR (%). Every schedule costs 60 reference graphs. The two step-size conditions share the same $k = 2$ reference.

k	CurveNet	PointCNN	Point2Vec	Mean
Original step size				
0	61.37	75.18	69.00	68.52
1	70.90	79.61	80.95	77.15
2	73.35	78.88	83.25	78.49
3	76.34	80.70	86.21	81.08
4	73.79	81.07	85.55	80.14
5	70.63	79.24	84.02	77.96
6	63.96	73.54	78.63	72.04
Rescaled step size				
0	59.66	73.36	66.59	66.54
1	69.80	78.19	80.47	76.15
2	73.35	78.88	83.25	78.49
3	75.99	80.57	86.60	81.05
4	74.01	80.93	86.21	80.38
5	70.15	79.84	83.89	77.96
6	64.27	73.08	79.33	72.23

B.3 Placement Checks beyond CWA

We apply the same placement comparison to SMER and SVRE. Base, Early, Middle, Late, and Full denote B^{10}, R^2B^8, $B^4R^2B^4$, B^8R^2, and R^{10}, respectively. Within each method, Early, Middle, and Late use identical numbers of base and refinement updates. Full uses more refinement computation and is included as a separate reference, not as an equal-budget placement.

The image analysis uses MobileNetV3-Large, DeiT-Small, and Inception-v3; the point-cloud analysis uses CurveNet, PointCNN, and Point2Vec. Table 13 reports their individual results and the macro-average. The evaluations use 1,000 image samples with seed 17 and 2,468 point clouds with seed 31. These three-target mechanism panels are distinct from the standard method-evaluation panels.

Table 13: Placement comparisons for SMER and SVRE. Values are ASR (%) on the three-target mechanism panels; Full has a higher reference cost than the three placements.

Target	Base	Early	Middle	Late	Full
NIPS2017, SMER					
MobileNetV3-Large	63.56	74.19	67.14	63.88	70.82
DeiT-Small	28.04	30.92	28.68	28.57	26.76
Inception-v3	43.51	52.66	45.96	44.68	48.83
Macro-average	45.04	52.59	47.26	45.71	48.80
NIPS2017, SVRE					
MobileNetV3-Large	63.56	74.19	67.14	63.99	71.26
DeiT-Small	28.04	29.42	29.00	28.78	26.87
Inception-v3	43.51	51.60	45.85	44.89	48.62
Macro-average	45.04	51.74	47.33	45.89	48.91
ModelNet40, SMER					
CurveNet	68.00	78.45	71.73	68.09	82.88
PointCNN	75.99	80.61	79.56	75.86	80.70
Point2Vec	75.79	87.52	81.39	75.88	89.33
Macro-average	73.26	82.19	77.56	73.28	84.30
ModelNet40, SVRE					
CurveNet	59.83	70.59	61.11	60.32	75.68
PointCNN	70.14	71.60	71.47	69.41	74.49
Point2Vec	69.93	79.45	71.78	70.77	84.44
Macro-average	66.63	73.88	68.12	66.83	78.20

Early has the highest macro-ASR among the three equal-cost placements in all four settings. However, the comparison with Full depends on the modality: Early exceeds Full on NIPS2017, whereas Full exceeds Early on ModelNet40 by 2.11 points for SMER and 4.32 points for SVRE. These results are consistent with our analysis in the main text: early placement generally outperforms middle or late placement, indicating that refinement updates applied at early stages have the most pronounced impact on final transferability. However, for point clouds, refinement at other stages can still provide positive gains, and applying only two early refinement updates does not necessarily outperform Full.

B.4 Expanded-Panel Placement

We expand the CWA image analysis to 15 targets by adding the 12 standard method targets to the original three analysis targets, without changing the mechanism surrogates. The same adversarial examples are evaluated on both the original and expanded panels. Table 14 reports all 15 targets and keeps the original-three and expanded-panel averages separate.

Table 14: Expanded CWA image placement analysis on NIPS2017 ($N = 1{,}000$, $\epsilon = 4/255$).

Target	Base	Early	Middle	Late	Full
MobileNetV3-Large	61.71	70.93	62.36	60.63	65.73
DeiT-Small	24.31	25.37	25.80	24.31	21.96
Inception-v3	38.62	46.81	41.06	38.62	42.66
ResNet-152	48.82	55.49	50.05	48.31	44.92
EfficientNet-B0	54.04	63.83	53.94	52.87	59.79
RegNetY-400MF	56.63	67.96	56.85	55.88	62.03
ViT-B/16	18.54	19.81	20.23	19.17	16.84
MaxViT-T	51.30	55.76	52.23	50.05	47.46
AlexNet	15.06	15.21	13.63	14.10	16.96
GoogLeNet	34.83	41.61	35.63	34.25	38.97
SqueezeNet1.1	30.88	41.83	32.38	30.43	40.63
ShuffleNetV2	39.23	45.88	40.68	38.01	46.00
MNASNet1.0	60.47	70.72	61.04	59.57	68.24
VGG-19	79.15	88.45	78.81	77.35	90.36
DenseNet-201	63.20	71.78	64.16	63.41	66.85
Original 3 targets	41.55	47.70	43.08	41.18	43.45
All 15 targets	45.12	52.10	45.92	44.47	48.63

On the expanded panel, Early achieves 52.10% ASR, compared with 45.92% for Middle, 44.47% for Late, and 48.63% for Full. The early-placement advantage therefore persists in this image setting beyond the original three targets.

Expanded point-cloud analysis. The corresponding experiment uses 2,468 ModelNet40 samples, 19 mechanism-analysis targets.

The expanded-panel ASRs are 67.5204% for Early, 62.2530% for Middle, 57.1712% for Late, and 61.7069% for Full. Late is also below Base (57.9921%).

B.5 Fixed-Input Optimizer-State Controls

To test whether the return of a refinement window depends on the incoming optimizer state, we branch after $t \in \{2, 4, 8\}$ base updates. At each fork, we hold the incoming adversarial input and inner state fixed and intervene only on the outer momentum: *keep* retains it, *reset* sets it to zero, and *shuffle* permutes its coordinates while preserving its norm. Each intervention is applied to both members of a paired comparison: one executes an R^2 window and the other a B^2 window, followed by the same remaining base-continuation rule until $T = 10$.

The inputs are fixed at the fork, not throughout the subsequent attack. The two trajectories evolve normally after the intervention. Incoming input and state identities are checked before the intervention, so differences between keep, reset, and shuffle do not arise from different incoming adversarial examples. For each intervention, AOV is the paired terminal ASR of the R^2 branch minus that of

Table 15: Expanded CWA ModelNet40 placement analysis ($N = 2{,}468$, $\epsilon = 0.06$).

Target	Base	Early	Middle	Late	Full
CurveNet	58.56	73.35	64.49	55.84	60.45
PointCNN	72.93	78.88	74.99	69.50	65.84
Point2Vec	66.09	83.25	75.84	65.83	79.81
DGCNN-V	20.20	22.76	19.73	19.96	22.76
PT_Hengshuang	67.62	81.06	74.83	66.14	72.59
PT_Menghao	61.86	78.34	66.68	60.65	66.64
Point-PN	60.83	73.31	68.69	62.88	71.18
PointCAT	77.54	90.77	85.76	77.32	86.11
PointMLPElite	83.20	92.75	89.52	84.07	89.39
PointTransformer-v1-Cls26	30.55	35.91	30.46	29.65	31.98
PointTransformer-v1-Cls38	44.09	51.23	45.83	43.65	46.14
PointTransformer-v1-Cls50	47.16	55.57	49.32	45.40	52.57
PointTransformer-v2	84.81	92.14	89.31	85.73	90.62
RepSurf	74.99	85.97	81.34	71.90	76.80
RepSurf2	69.83	80.30	75.59	68.51	70.05
SimpleView	23.32	24.58	21.71	21.55	20.14
VN-DGCNN	64.27	78.06	70.87	64.36	66.82
VN-PointNet	68.02	75.05	70.45	66.55	70.85
VN-Transformer	25.98	29.61	27.41	26.75	31.69
Original 3 targets	65.86	78.49	71.77	63.72	68.70
All 19 targets	57.99	67.52	62.25	57.17	61.71

the B^2 branch. Changes in AOV relative to keep are differences of these paired effects; they are not changes in the ASR of the refinement branch alone.

The image control uses the first 32 samples from each of four fixed shards, totaling 128 samples, and the 15-target mechanism panel. Figure 7 reports the result.

The effect of resetting momentum is position-dependent. At fork 2, reset increases AOV from 2.5140 to 5.9575 points. The change relative to keep is 3.4435 points, with a paired sample-bootstrap 95% interval of $[1.800, 5.231]$ for seed 17. At fork 4, reset instead changes AOV from 0.7057 to -0.8866 points; the difference is -1.5923 points with interval $[-3.267, 0.191]$. This interval includes zero. Shuffle also gives lower AOV than keep at forks 4 and 8 in all three recorded seeds. Thus, the results do not identify momentum removal as a uniformly beneficial intervention.

Point-cloud state controls. The corresponding ModelNet40 experiment uses 128 samples, 19 mechanism-analysis targets, the same three fork positions.

C Detailed Transfer and Defense Results

This appendix reports target-level comparisons between the native full attack and TES. Throughout the tables, Full denotes R^{10} and TES denotes R^2B^8, using the corresponding method's native updates and the same five-surrogate ensemble within each comparison. All values are ASR percentages on each target's clean-correct subset; higher values indicate stronger attacks. Macro-averages give equal weight to the reported targets. A sample misclassified before the attack is excluded from that target's denominator.

The comparisons below use the method-evaluation setting, not the mechanism-analysis panels in Appendix B. Standard targets, robust models, and input defenses are reported separately.

C.1 Standard-Model Transfer on Image Data

NIPS2017. Table 16 provides the complete 12-standard-target comparison at $\epsilon = 4/255$, using 1,000 samples and attack seed 17. Full and TES are evaluated on the same samples and target-specific clean-correct masks.

TES increases standard-panel macro-ASR from 50.01% to 53.28% for CWA, from 54.77% to 57.54% for SMER, and from 53.88% to 56.80% for SVRE, matching Table 5.

At $8/255$, TES improves CWA from 73.95% to 76.66%, but is slightly below Full for SMER (79.43% versus 79.77%) and SVRE (79.08% versus 79.13%).

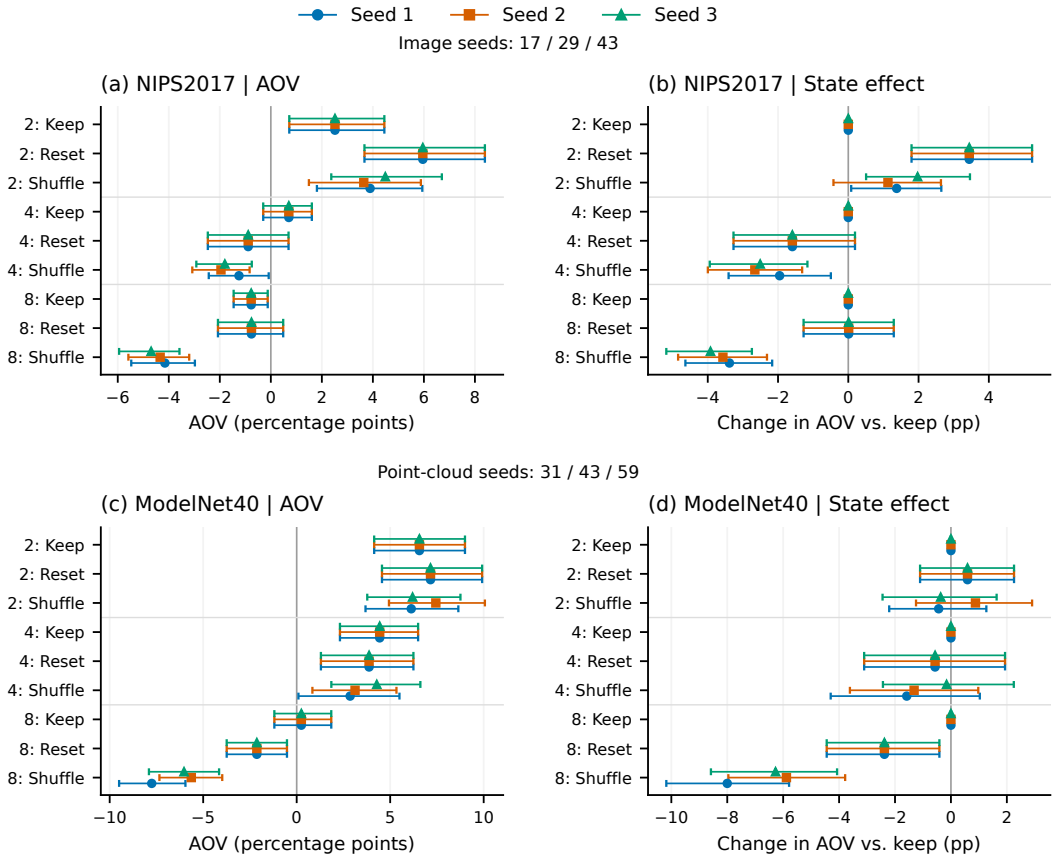


Figure 7: Fixed-input CWA optimizer-state controls on NIPS2017 (15 mechanism targets) and ModelNet40 (19 mechanism targets), $N = 128$ per seed. Left: paired window AOV. Right: its paired change relative to keep. Error bars are 95% sample-bootstrap intervals conditional on each seed, not intervals across seeds. Rows identify fork position and momentum intervention. Seeds 1–3 correspond to 17/29/43 for images and 31/43/59 for point clouds. Horizontal offsets are measured effects; vertical offsets separate overlapping seeds.

Table 16: NIPS2017 standard-model transfer ($N = 1{,}000$, $\epsilon = 4/255$, $T = 10$, seed 17). Full and TES are paired within each method.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
ResNet-152	44.82	54.87	56.51	60.00	56.72	60.92
EfficientNet-B0	60.11	63.83	65.11	68.62	64.36	68.30
RegNetY-400MF	62.03	66.67	65.91	70.55	66.56	71.41
ViT-B/16	17.06	19.92	21.29	23.52	20.02	22.35
MaxViT-T	47.35	55.24	54.93	61.16	55.56	60.64
AlexNet	16.32	16.01	16.48	16.96	16.64	15.85
GoogLeNet	40.00	42.07	45.40	47.47	44.83	46.32
SqueezeNet1.1	40.63	42.28	45.13	46.63	40.33	43.18
ShuffleNetV2	45.16	47.09	50.36	50.85	47.94	51.21
MNASNet1.0	69.03	70.38	71.96	75.90	70.27	73.65
VGG-19	91.03	88.90	91.37	92.04	90.13	92.38
DenseNet-201	66.63	72.10	72.75	76.82	73.18	75.43
Macro-average	50.01	53.28	54.77	57.54	53.88	56.80

Table 17: NIPS2017 standard-model transfer at $\epsilon = 8/255$ ($N = 1{,}000$, $T = 10$, seed 17).

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
ResNet-152	71.28	80.51	86.87	86.67	87.79	88.10
EfficientNet-B0	86.38	89.04	89.79	91.17	89.79	90.43
RegNetY-400MF	83.60	86.08	88.67	90.08	89.86	89.75
ViT-B/16	38.35	41.63	49.47	48.52	45.23	45.55
MaxViT-T	72.38	79.54	82.87	84.22	84.84	85.36
AlexNet	39.78	38.67	40.10	37.24	36.93	35.82
GoogLeNet	71.95	75.17	80.00	78.39	79.54	78.97
SqueezeNet1.1	72.26	71.96	76.01	73.31	74.96	71.96
ShuffleNetV2	71.31	73.85	78.33	77.24	76.03	77.24
MNASNet1.0	89.64	91.33	91.44	92.79	91.89	92.23
VGG-19	98.65	98.21	98.54	98.43	98.21	98.54
DenseNet-201	91.85	93.88	95.17	95.06	94.53	94.96
Macro-average	73.95	76.66	79.77	79.43	79.13	79.08

ImageNet-Val and ImageNet-Sketch. The evaluations use 50,000 ImageNet-Val samples and the fixed 50,000-image class-balanced ImageNet-Sketch subset specified in Appendix A.1, at $\epsilon \in \{4/255, 8/255\}$. The tables combine five original targets and seven extension targets, including separately generated extension attacks.

Table 18: ImageNet-Val standard-panel results ($N = 50{,}000$, $\epsilon = 4/255$).

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
ResNet-152	46.44	55.74	56.87	60.22	59.04	62.19
EfficientNet-B0	59.70	64.37	63.36	67.78	66.02	68.72
RegNetY-400MF	59.40	64.59	63.99	67.84	67.02	69.27
ViT-B/16	21.73	23.69	24.81	27.05	24.40	26.60
MaxViT-T	50.21	56.78	55.44	60.63	58.79	62.81
AlexNet	17.74	17.79	18.52	18.88	18.48	18.84
GoogLeNet	42.57	45.99	48.28	50.97	48.93	50.69
SqueezeNet1.1	43.33	43.82	46.28	48.56	46.60	48.64
ShuffleNetV2	46.52	49.00	51.13	53.55	51.42	53.35
MNASNet1.0	67.37	70.40	69.99	73.43	72.16	74.20
VGG-19	87.57	86.69	88.17	90.09	88.67	91.50
DenseNet-201	65.28	71.18	71.44	73.94	72.59	73.54
Macro-average	50.66	54.17	54.86	57.74	56.18	58.36

For the ImageNet-Val comparisons, CWA improves from 50.66% to 54.17% at $4/255$ and from 73.40% to 76.40% at $8/255$. SMER improves from 54.86% to 57.74% at $4/255$.

C.2 Standard-Model Transfer on Point Clouds

ModelNet40. Tables 22 and 23 report the 19-target method panel at the two perturbation budgets, using 2,468 samples and attack seed 31. The original result vectors are retained, while the macro-ASR is computed only over the 19 reported targets. The two budgets are evaluated and summarized separately.

At $\epsilon = 0.06$, applying TES to SMER yields more than a threefold speedup while its macro-ASR is 1.09 percentage points lower than Full. For SVRE, TES also achieves more than a threefold speedup and increases macro-ASR by 0.74 percentage points over Full. For CWA, TES increases macro-ASR by 2.14 percentage points while also reducing runtime.

Table 19: ImageNet-Val standard-panel results ($N = 50{,}000$, $\epsilon = 8/255$).

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
ResNet-152	70.33	80.80	84.66	84.20	86.81	86.12
EfficientNet-B0	84.02	87.00	88.49	88.94	89.98	89.61
RegNetY-400MF	81.40	85.38	87.19	87.66	89.13	88.45
ViT-B/16	41.92	45.10	51.49	50.24	49.50	49.17
MaxViT-T	73.02	78.85	82.11	82.84	84.68	85.17
AlexNet	39.55	39.31	42.01	39.88	40.95	39.38
GoogLeNet	72.44	76.17	80.40	79.72	81.14	79.67
SqueezeNet1.1	71.99	71.75	75.50	74.74	75.67	74.58
ShuffleNetV2	70.66	72.90	77.35	76.74	77.18	76.26
MNASNet1.0	88.14	89.67	90.89	91.28	91.95	91.52
VGG-19	98.51	97.76	98.21	98.46	98.45	98.86
DenseNet-201	88.77	92.12	93.34	93.25	94.14	93.29
Macro-average	73.40	76.40	79.30	79.00	79.97	79.34

Table 20: ImageNet-Sketch standard-panel results on the fixed class-balanced subset ($N = 50{,}000$, $\epsilon = 4/255$).

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
ResNet-152	62.97	66.41	68.63	70.42	68.00	70.11
EfficientNet-B0	74.56	74.68	76.42	77.64	75.92	77.41
RegNetY-400MF	74.31	75.24	76.08	77.73	76.26	77.43
ViT-B/16	37.66	39.42	40.24	43.03	39.43	42.21
MaxViT-T	59.31	61.24	62.21	65.06	61.63	64.77
AlexNet	28.78	28.66	28.39	28.58	28.01	27.89
GoogLeNet	50.94	51.70	53.07	54.49	52.73	54.08
SqueezeNet1.1	51.77	50.64	53.42	53.58	52.49	53.38
ShuffleNetV2	60.52	60.51	62.91	63.77	61.65	63.00
MNASNet1.0	76.24	75.92	77.10	78.25	77.00	77.69
VGG-19	88.44	85.55	87.33	88.08	86.34	88.80
DenseNet-201	76.34	77.71	78.11	78.81	77.61	77.56
Macro-average	61.82	62.31	63.66	64.95	63.09	64.53

Table 21: ImageNet-Sketch standard-panel results on the fixed class-balanced subset ($N = 50{,}000$, $\epsilon = 8/255$).

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
ResNet-152	77.86	81.48	85.02	85.23	84.99	85.05
EfficientNet-B0	88.34	88.34	90.36	90.85	90.24	90.48
RegNetY-400MF	85.44	86.12	88.34	88.65	88.44	88.48
ViT-B/16	55.45	56.48	61.34	61.41	59.01	59.96
MaxViT-T	75.70	76.40	79.21	79.59	78.29	78.97
AlexNet	46.98	47.37	47.67	46.51	46.88	46.37
GoogLeNet	75.09	75.54	79.32	78.20	78.66	77.23
SqueezeNet1.1	73.94	72.37	76.29	74.27	74.68	74.59
ShuffleNetV2	74.61	74.71	78.74	78.08	77.43	77.13
MNASNet1.0	87.63	87.57	89.45	89.76	89.08	89.26
VGG-19	96.43	95.11	96.37	96.44	95.74	96.85
DenseNet-201	90.38	91.14	92.40	92.38	92.23	91.65
Macro-average	77.32	77.72	80.38	80.11	79.64	79.67

Table 22: ModelNet40 standard-model transfer ($N = 2{,}468$, $\epsilon = 0.06$, $T = 10$, seed 31). Averages use the 19 targets listed here.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
DGCNN	81.26	85.46	90.59	88.53	84.33	84.16
DGCNN-V	21.52	23.20	26.33	25.35	24.56	26.20
PT (Hengshuang)	75.50	79.80	79.67	81.55	76.13	77.81
PT (Menghao)	79.54	82.67	91.92	88.39	83.70	82.83
Point-PN	74.07	73.31	74.64	77.31	71.67	72.46
PointCAT	89.36	90.55	93.85	92.75	90.24	90.07
PointMLPElite	91.84	92.45	94.11	94.06	90.27	91.59
PointNet	32.38	31.84	36.13	35.19	32.84	34.38
PointNet++ SSG	89.37	90.44	89.77	90.04	86.71	87.05
PT-v1 Cls26	35.73	42.04	52.06	47.54	43.07	44.30
PT-v1 Cls38	51.09	53.28	58.09	55.42	54.84	55.80
PT-v1 Cls50	55.21	51.65	53.67	54.99	52.13	54.69
PT-v2	92.01	91.58	92.45	92.93	89.79	89.89
RepSurf	80.64	81.78	80.72	84.08	79.05	81.25
RepSurf2	76.52	81.53	78.32	82.32	77.13	79.47
SimpleView	28.30	34.82	44.87	39.26	34.61	34.70
VN-DGCNN	77.70	82.79	89.39	86.44	81.29	80.84
VN-PointNet	78.04	80.72	88.26	83.96	81.98	81.21
VN-Transformer	32.52	33.35	43.34	37.40	35.85	35.51
Macro-average	65.40	67.54	71.48	70.39	66.85	67.59

Table 23: ModelNet40 standard-model transfer ($N = 2{,}468$, $\epsilon = 0.12$, $T = 10$, seed 31). Averages use the 19 targets listed here.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
DGCNN	96.10	95.93	96.80	97.15	96.23	96.80
DGCNN-V	54.67	53.64	64.47	59.20	57.42	58.87
PT (Hengshuang)	94.04	93.37	91.85	94.04	93.06	93.60
PT (Menghao)	96.20	97.36	97.86	97.95	97.63	97.32
Point-PN	94.18	93.61	93.96	94.67	94.67	94.45
PointCAT	96.18	95.65	96.44	96.40	96.18	96.22
PointMLPElite	96.46	97.08	97.16	97.34	96.90	96.94
PointNet	60.43	59.44	64.14	62.65	61.20	61.56
PointNet++ SSG	96.40	96.71	95.91	96.71	96.31	96.53
PT-v1 Cls26	83.72	87.61	92.49	91.01	88.51	88.37
PT-v1 Cls38	86.58	86.36	88.68	88.68	87.12	87.34
PT-v1 Cls50	86.36	81.70	80.86	84.16	80.77	81.96
PT-v2	96.03	95.94	95.68	96.16	95.77	95.94
RepSurf	95.54	95.68	94.84	96.21	95.28	96.07
RepSurf2	95.16	94.94	92.66	95.47	94.99	95.12
SimpleView	70.44	83.32	87.32	86.10	81.40	81.56
VN-DGCNN	95.22	95.27	95.99	95.13	94.86	95.04
VN-PointNet	95.55	95.85	96.41	96.41	95.90	95.29
VN-Transformer	68.31	74.08	80.80	78.60	76.04	74.02
Macro-average	87.24	88.08	89.70	89.69	88.43	88.58

ScanObjectNN and ShapeNet. The respective evaluations contain 2,882 and 10,261 samples and use 14 and 19 reported targets. The tables below report both budgets using the same target panels as the main text.

Table 24: ScanObjectNN standard-model transfer at $\epsilon = 0.06$; 14 reporting targets, 2882 samples.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
PT (Hengshuang)	76.40	85.61	85.00	88.50	86.96	86.59
PT (Menghao)	81.79	87.71	93.07	93.11	91.96	91.50
Point-PN	81.86	89.08	89.36	92.92	91.55	91.41
PointCAT	86.13	95.09	96.71	98.11	97.06	97.76
PointConv	72.83	86.35	87.48	91.00	87.97	87.25
PointMLP-Elite	79.72	93.10	93.85	96.65	95.61	95.57
PT-v1 Cls26	65.86	77.05	81.05	80.95	80.27	77.37
PT-v1 Cls38	76.78	83.23	87.60	87.06	86.61	85.21
PT-v1 Cls50	77.17	87.82	90.62	91.81	89.88	88.69
PT-v2	74.92	78.11	78.35	79.37	79.03	78.40
RSCNN	81.99	89.28	94.24	93.52	91.95	90.38
RepSurf	78.36	86.37	86.87	88.75	88.01	87.77
RepSurf2	83.02	87.93	87.60	89.87	88.61	88.13
SimpleView	39.97	50.58	58.16	57.86	54.12	52.80
Macro-average	75.49	84.09	86.42	87.82	86.40	85.63

Table 25: ScanObjectNN standard-model transfer at $\epsilon = 0.12$; 14 reporting targets, 2882 samples.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
PT (Hengshuang)	86.96	91.21	89.86	93.55	91.17	92.20
PT (Menghao)	92.88	93.39	96.72	97.04	97.23	97.74
Point-PN	90.22	92.55	92.87	95.52	94.38	95.89
PointCAT	92.63	94.60	96.09	97.63	97.24	98.24
PointConv	81.92	89.96	91.73	93.31	91.68	92.95
PointMLP-Elite	89.88	92.60	92.64	96.07	93.94	95.73
PT-v1 Cls26	85.68	87.24	89.26	89.95	89.26	91.92
PT-v1 Cls38	89.50	89.40	92.20	93.55	93.60	93.24
PT-v1 Cls50	89.57	91.19	93.38	95.14	95.14	95.53
PT-v2	82.81	84.75	84.02	84.60	84.99	84.70
RSCNN	91.47	93.33	94.00	96.14	94.76	96.05
RepSurf	87.97	89.53	88.05	89.57	89.85	90.18
RepSurf2	90.11	91.09	90.92	91.53	91.53	91.17
SimpleView	62.71	67.81	72.41	71.70	70.24	71.80
Macro-average	86.74	89.19	90.30	91.81	91.07	91.95

At $\epsilon = 0.06$, TES improves CWA on ScanObjectNN from 75.49% to 84.09% and on ShapeNet from 51.01% to 53.64%. SMER improves on ScanObjectNN and is only slightly lower than Full on ShapeNet, while SVRE is lower on both datasets. Even in the largest of these decreases, however, the ASR drop is only about 1 percentage point, while TES achieves several-fold speedups, as summarized in Table 7. This indicates that the benefit of TES is not merely a conventional speed–performance trade-off: substantially reducing later method-specific computation can preserve nearly the full attack strength and, in some settings, even improve transferability.

C.3 Robust Models and Input Defenses

Robust image models. Table 28 reports the seven robust-model configurations on NIPS2017 at $\epsilon = 4/255$. The seven-target robust mean is separate from the standard-model average. The Salman et al. ResNet-18/50 and WideResNet-50-2 models follow Salman et al. (2020); the XCiT-M12/L12

Table 26: ShapeNet standard-model transfer at $\epsilon = 0.06$; 19 reporting targets, 10261 samples.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
DGCNN-V	15.00	15.01	17.97	16.30	16.02	15.66
PT (Hengshuang)	44.47	44.61	50.37	49.95	48.88	46.51
PT (Menghao)	39.01	43.03	51.30	48.55	47.32	46.24
Point-PN	57.06	55.61	58.88	62.27	57.11	56.26
PointCAT	81.78	82.42	88.04	87.88	83.85	81.50
PointConv	76.22	75.54	82.82	81.15	77.28	73.76
PointMLP-Elite	70.44	73.23	77.65	78.90	71.32	71.87
PointNet	36.20	36.82	43.49	42.34	43.36	42.51
PT-v1 Cls26	31.19	30.98	36.18	35.20	35.58	34.69
PT-v1 Cls38	59.84	64.10	67.42	68.89	64.80	65.02
PT-v1 Cls50	54.92	59.30	64.31	65.50	61.74	61.93
PT-v2	63.18	63.64	66.46	68.55	62.35	61.13
RSCNN	65.68	71.34	75.21	77.21	72.58	72.80
RepSurf	68.11	73.97	77.67	81.50	73.92	74.38
RepSurf2	53.54	59.80	63.95	67.27	60.84	59.69
SimpleView	41.54	48.76	55.70	53.70	50.10	46.93
VN-DGCNN	47.91	53.89	65.29	58.66	55.60	50.90
VN-PointNet	45.21	49.80	57.13	51.77	49.56	47.07
VN-Transformer	17.84	17.31	23.49	19.39	19.86	17.90
Macro-average	51.01	53.64	59.12	58.68	55.37	54.04

Table 27: ShapeNet standard-model transfer at $\epsilon = 0.12$; 19 reporting targets, 10261 samples.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
DGCNN-V	28.61	25.47	31.11	27.60	28.26	29.10
PT (Hengshuang)	70.18	62.44	63.99	65.63	66.66	69.15
PT (Menghao)	79.65	76.07	80.98	80.33	79.56	80.55
Point-PN	81.80	75.09	75.38	79.72	78.79	80.03
PointCAT	97.80	95.35	95.91	97.28	96.70	97.52
PointConv	96.09	93.03	95.36	95.03	95.12	93.76
PointMLP-Elite	96.34	93.46	94.67	95.60	94.59	94.56
PointNet	59.86	54.28	61.33	59.83	62.07	63.84
PT-v1 Cls26	56.85	47.59	52.17	51.52	52.87	55.36
PT-v1 Cls38	89.58	86.19	87.18	88.77	88.03	89.02
PT-v1 Cls50	84.67	81.89	84.41	84.59	84.02	84.26
PT-v2	93.31	90.97	91.43	93.61	92.16	91.74
RSCNN	91.52	92.27	93.11	95.17	92.87	93.12
RepSurf	93.46	93.46	93.19	96.01	94.20	95.03
RepSurf2	89.12	89.53	88.57	93.07	90.92	91.41
SimpleView	86.29	85.35	86.93	87.87	85.72	85.68
VN-DGCNN	93.49	93.20	95.52	94.18	94.09	92.48
VN-PointNet	90.73	93.22	95.11	93.24	92.92	91.07
VN-Transformer	46.99	46.30	56.89	49.20	49.50	46.22
Macro-average	80.34	77.64	80.17	80.43	79.95	80.21

models follow Debenedetti et al. (2023); and the Engstrom et al. ResNet-50 model follows Engstrom et al. (2019).

Table 28: Robust-model ASR on NIPS2017 ($N = 1{,}000$, $\epsilon = 4/255$, attack seed 17). Clean-correct denominators are specific to each robust target.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
Salman et al., R18	1.13	1.29	1.29	1.13	1.29	1.13
Salman et al., WR50-2	1.09	0.85	0.97	1.21	1.09	1.09
Debenedetti et al., XCiT-M12	1.25	1.59	1.59	1.59	1.48	1.59
Debenedetti et al., XCiT-L12	1.47	1.59	1.36	1.59	1.59	1.59
FastAT-R50 (4px)	2.27	2.27	2.43	2.59	2.76	2.43
Salman et al., R50	1.44	1.71	1.44	1.44	1.57	1.44
Engstrom et al., R50	1.06	1.06	1.32	1.19	0.93	0.93
Seven-target mean	1.39	1.48	1.49	1.53	1.53	1.46

The robust-panel ASRs are much lower than the standard-panel ASRs. TES has a slightly higher robust-panel mean for CWA and SMER, but a lower mean for SVRE (1.46% versus 1.53%). This may be because the surrogate set does not include robust models, resulting in relatively low attack success rates on robust targets. Consequently, the improvements observed on standard models do not consistently extend to every robust target. Due to limited computational resources, we did not conduct experiments using robust models as surrogates; therefore, the conclusions drawn from standard models cannot yet be generalized to robust models.

Image input defenses. Tables 29 and 30 report all 47 recorded input-defense configurations for the same NIPS2017 comparison. The first table includes 5-bit input quantization (Xu et al., 2018), JPEG quality 75, the registered resize-and-pad configuration (Xie et al., 2018), and NRP (Naseer et al., 2020) on five victim models, followed by HGD and DiffPure on their respective recorded victims. The second table retains all five random resize-and-pad draws for each of the five victims. The draw labels index the fixed evaluation repetitions; they are not five additional victim models or five independent attack seeds. The registered defense parameters and evaluation randomness are held fixed between Full and TES.

Additional image datasets and point-cloud defenses. Additional image-defense results are reported by dataset and perturbation budget. The point-cloud defense comparison covers SOR, SRS, and DUP-Net on the five evaluated victims: PointNet, DGCNN, PointMLP, PointNet++ SSG, and PointNet++ MSG. This is a five-victim defense panel, not a defense evaluation on all 19 standard ModelNet40 targets.

At $\epsilon = 0.06$, the five-victim means increase for CWA under all three defenses, while SMER and SVRE decrease under all three. In particular, the pooled-count SRS means are 80.20% versus 79.89% for SMER Full and TES.

Table 29: Input-defense ASR on NIPS2017 ($\epsilon = 4/255$). Each row denotes a defense–victim configuration; no average is taken across heterogeneous defenses.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
5-bit quantization						
ResNet-152	39.53	47.74	50.21	53.18	49.08	53.90
EfficientNet-B0	53.24	58.02	58.66	62.17	58.98	62.59
RegNetY-400MF	55.08	58.91	59.67	63.17	60.22	63.61
ViT-B/16	16.10	18.22	19.70	22.46	18.75	20.55
MaxViT-T	43.70	50.41	49.07	56.20	49.79	56.82
JPEG quality 75						
ResNet-152	20.91	23.04	25.37	26.54	25.37	26.65
EfficientNet-B0	34.12	34.68	39.08	40.20	37.50	38.96
RegNetY-400MF	37.08	37.77	39.72	42.48	40.30	40.53
ViT-B/16	14.51	13.41	15.71	15.82	15.16	14.73
MaxViT-T	21.95	24.86	25.95	28.65	24.76	27.03
Resize-and-pad (208)						
ResNet-152	29.82	35.29	39.53	42.83	38.91	43.76
EfficientNet-B0	40.18	44.04	45.18	49.94	45.63	49.04
RegNetY-400MF	48.67	54.10	53.55	60.20	53.99	58.09
ViT-B/16	13.69	14.66	15.52	16.81	16.27	16.49
MaxViT-T	36.68	43.01	42.90	48.70	43.11	48.19
NRP						
ResNet-152	23.63	26.79	27.53	31.86	26.05	30.59
EfficientNet-B0	30.28	30.16	32.00	36.12	32.68	36.47
RegNetY-400MF	33.88	35.99	37.05	40.80	35.64	40.33
ViT-B/16	9.04	8.27	9.26	11.80	10.36	11.36
MaxViT-T	26.84	29.39	29.50	35.78	27.37	35.78
HGD (IncRes-v2)	10.77	11.51	13.49	14.54	13.39	14.64
DiffPure (ResNet-152)	3.86	3.97	3.09	3.20	2.87	3.42

Table 30: All five recorded random resize-and-pad draws on NIPS2017 ($\epsilon = 4/255$). Full and TES use matched defense randomness within each draw.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
Draw 1						
ResNet-152	26.79	33.85	38.01	40.29	35.41	39.46
EfficientNet-B0	38.05	40.09	44.51	47.23	43.83	46.55
RegNetY-400MF	44.79	50.11	51.55	55.88	50.78	54.10
ViT-B/16	11.60	12.67	13.86	14.72	12.14	13.21
MaxViT-T	34.38	40.10	40.73	44.79	40.21	46.25
Draw 2						
ResNet-152	30.30	34.75	37.85	42.09	37.23	41.37
EfficientNet-B0	40.23	43.68	45.98	50.11	46.32	49.54
RegNetY-400MF	48.56	51.88	54.43	57.76	54.55	56.54
ViT-B/16	12.33	13.18	15.97	16.40	14.68	14.90
MaxViT-T	36.28	43.35	43.24	48.23	42.83	48.44
Draw 3						
ResNet-152	32.00	37.65	40.43	43.93	39.51	44.34
EfficientNet-B0	39.87	44.54	46.44	48.66	46.99	49.11
RegNetY-400MF	48.33	53.00	53.89	60.00	54.11	58.89
ViT-B/16	12.77	14.38	16.85	17.27	15.34	16.20
MaxViT-T	37.01	43.97	44.07	48.86	43.35	49.27
Draw 4						
ResNet-152	33.02	39.73	43.65	47.47	41.90	47.06
EfficientNet-B0	44.01	48.35	50.40	55.19	51.65	54.96
RegNetY-400MF	52.48	57.66	57.88	63.07	57.22	60.53
ViT-B/16	11.92	13.33	15.60	16.03	14.95	15.38
MaxViT-T	37.55	44.61	43.36	48.86	43.67	49.48
Draw 5						
ResNet-152	36.01	43.00	47.94	50.10	46.19	50.72
EfficientNet-B0	49.84	52.99	54.52	59.63	54.41	58.43
RegNetY-400MF	52.20	56.70	58.35	62.64	58.46	61.98
ViT-B/16	14.79	16.18	19.08	19.40	17.79	18.76
MaxViT-T	39.00	45.12	43.98	50.41	44.19	51.04

Table 31: NIPS2017 robust-model ASR at $\epsilon = 8/255$. The last two targets use extension runs; no combined mean is reported.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
Salman et al., R18	6.29	6.61	5.97	5.81	5.97	5.48
Salman et al., WR50-2	3.02	3.14	2.54	2.42	2.66	3.02
Debenedetti et al., XCiT-M12	2.73	2.73	2.61	2.50	2.39	2.50
Debenedetti et al., XCiT-L12	2.38	2.27	2.27	2.27	2.04	2.04
FastAT-R50 (4px)	5.83	5.51	5.67	5.67	6.00	5.83
Salman et al., R50	3.81	3.81	3.41	3.28	3.67	3.41
Engstrom et al., R50	4.90	4.37	4.24	4.11	4.11	3.84

Table 32: NIPS2017 input-defense ASR at $\epsilon = 8/255$; each row identifies its victim.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
5-bit quantization						
ResNet-152	69.20	79.57	85.83	85.42	86.34	87.06
EfficientNet-B0	85.12	87.14	88.95	89.37	88.31	89.05
RegNetY-400MF	80.66	84.70	87.98	88.42	87.65	88.63
ViT-B/16	39.62	41.63	48.41	46.29	44.07	44.70
MaxViT-T	72.00	77.27	82.54	82.33	83.47	83.26
JPEG quality 75						
ResNet-152	49.04	56.37	64.23	61.15	63.59	60.51
EfficientNet-B0	68.47	70.72	78.60	76.58	76.13	73.99
RegNetY-400MF	69.23	71.64	77.73	75.77	78.07	76.00
ViT-B/16	34.95	34.84	42.86	41.43	40.22	37.58
MaxViT-T	52.32	57.51	64.43	62.81	63.46	60.11
Resize-and-pad (208)						
ResNet-152	58.31	66.87	73.68	73.27	74.72	73.89
EfficientNet-B0	69.81	73.10	78.09	77.53	77.98	76.96
RegNetY-400MF	73.84	77.38	83.26	83.48	82.48	82.71
ViT-B/16	29.53	31.90	38.58	37.07	36.75	35.67
MaxViT-T	62.90	71.09	76.37	76.89	74.92	76.68
HGD (IncRes-v2)	23.33	26.67	37.87	35.56	36.82	34.21
DiffPure (ResNet-152)	6.84	6.73	7.39	7.17	6.06	6.06

Table 33: NIPS2017 random resize-and-pad ASR at $\epsilon = 8/255$, retaining all five registered draws.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
Draw 1						
ResNet-152	55.35	62.31	71.24	70.61	70.40	69.47
EfficientNet-B0	68.52	70.78	77.01	76.10	75.08	75.76
RegNetY-400MF	70.73	74.17	79.16	78.38	80.04	78.82
ViT-B/16	29.11	31.47	35.34	34.91	32.65	32.33
MaxViT-T	60.83	68.44	74.58	74.69	74.06	73.96
Draw 2						
ResNet-152	57.60	65.56	73.01	71.98	72.39	72.29
EfficientNet-B0	71.15	73.79	79.31	77.82	77.13	77.24
RegNetY-400MF	73.39	77.05	81.93	81.82	81.26	80.93
ViT-B/16	28.08	30.33	37.83	35.58	35.69	34.19
MaxViT-T	63.20	70.58	74.64	75.05	76.09	76.30
Draw 3						
ResNet-152	59.05	67.70	75.62	74.90	75.82	75.10
EfficientNet-B0	72.72	75.39	79.18	77.95	78.73	78.62
RegNetY-400MF	73.67	78.22	82.00	82.67	82.22	82.89
ViT-B/16	31.87	33.69	40.56	37.98	36.80	36.91
MaxViT-T	63.10	71.10	77.86	77.75	76.20	76.82
Draw 4						
ResNet-152	60.68	69.56	77.30	76.37	76.57	76.26
EfficientNet-B0	73.89	76.62	81.19	80.50	79.93	80.27
RegNetY-400MF	77.18	80.49	84.12	84.12	84.90	85.67
ViT-B/16	32.61	34.56	40.52	39.33	38.79	38.35
MaxViT-T	62.24	70.64	75.93	75.62	76.35	76.87
Draw 5						
ResNet-152	61.83	72.02	80.56	80.56	81.48	80.97
EfficientNet-B0	76.82	80.41	84.33	84.44	85.42	84.44
RegNetY-400MF	77.25	80.55	86.26	85.60	85.49	86.59
ViT-B/16	34.30	36.66	43.94	42.34	41.48	41.59
MaxViT-T	64.52	71.58	77.49	76.56	76.35	78.42

Table 34: ModelNet40 defense ASR at $\epsilon = 0.06$ ($N = 2{,}468$). Each defense mean equally weights the five victims.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
SOR						
PointNet	36.70	39.76	46.80	43.24	40.54	39.81
DGCNN	79.87	86.22	93.88	90.66	88.40	86.76
PointMLP	93.81	94.25	95.44	94.96	94.52	93.72
PointNet++ SSG	86.59	90.70	92.43	91.56	89.28	88.19
PointNet++ MSG	85.27	88.74	90.21	90.92	87.49	86.96
Five-victim mean	76.45	79.93	83.75	82.27	80.05	79.09
SRS						
PointNet	33.50	32.46	37.34	35.99	32.87	33.36
DGCNN	84.13	87.02	92.67	89.82	87.16	85.20
PointMLP	93.63	93.33	94.29	94.11	92.98	92.32
PointNet++ SSG	89.47	89.83	89.47	90.32	87.09	85.88
PointNet++ MSG	87.43	87.96	87.26	89.24	85.54	85.68
Five-victim mean	77.63	78.12	80.20	79.89	77.13	76.49
DUP-Net						
PointNet	35.83	39.25	48.36	43.19	41.24	39.63
DGCNN	79.13	83.53	93.61	90.01	86.28	83.77
PointMLP	92.86	92.96	94.45	93.63	93.01	92.30
PointNet++ SSG	81.23	84.36	89.46	86.38	84.91	81.74
PointNet++ MSG	82.59	85.74	90.06	88.41	86.47	84.38
Five-victim mean	74.33	77.17	83.19	80.32	78.38	76.36

Table 35: ModelNet40 defense ASR at $\epsilon = 0.12$ ($N = 2{,}468$). Each defense mean equally weights the five victims.

	CWA		SMER		SVRE	
Target	Full	TES	Full	TES	Full	TES
SOR						
PointNet	70.48	73.95	81.17	77.01	75.46	75.05
DGCNN	96.10	96.19	96.74	97.42	96.65	96.60
PointMLP	96.06	95.62	96.06	95.93	96.11	96.02
PointNet++ SSG	96.40	96.40	96.44	96.67	96.53	96.44
PointNet++ MSG	95.95	96.66	96.31	96.80	96.35	96.53
Five-victim mean	91.00	91.76	93.34	92.77	92.22	92.13
SRS						
PointNet	61.80	60.04	65.37	64.01	62.07	62.66
DGCNN	96.27	96.27	96.53	96.71	96.49	96.49
PointMLP	96.12	95.86	96.08	96.12	96.12	96.12
PointNet++ SSG	96.28	96.50	96.10	96.41	96.28	96.28
PointNet++ MSG	96.18	96.40	95.69	96.62	96.00	96.75
Five-victim mean	89.33	89.01	89.95	89.98	89.39	89.66
DUP-Net						
PointNet	71.67	76.65	85.10	80.97	78.26	77.22
DGCNN	95.88	96.21	96.78	96.64	96.26	96.36
PointMLP	95.12	95.12	95.12	95.12	95.12	95.12
PointNet++ SSG	95.36	96.27	96.47	96.62	96.06	96.12
PointNet++ MSG	94.91	95.78	96.27	96.07	95.83	95.83
Five-victim mean	90.59	92.01	93.95	93.08	92.31	92.13

D Additional Ablations, Controls, and Stability

This appendix examines prefix length, attack horizon, base continuation, surrogate composition, and attack-seed variation. All experiments use the method-evaluation setting. Unless a different budget is stated, NIPS2017 uses $\epsilon = 4/255$, 1,000 samples, and the standard 12-target panel; ModelNet40 uses $\epsilon = 0.06$, 2,468 samples, and the standard 19-target reporting panel. Single-seed comparisons use seed 17 for images and seed 31 for point clouds. Robust and defended targets are excluded from the averages in this appendix. Base denotes B^{10}, TES denotes R^2B^8, and Full denotes R^{10} unless the horizon is explicitly varied.

D.1 Prefix Length at a Fixed Horizon

Figure 8 compares $R^K B^{10-K}$ at fixed $T = 10$. Increasing K replaces base updates with refinement updates without changing the outer horizon or perturbation budget. The two modalities are summarized separately using their full standard reporting panels.

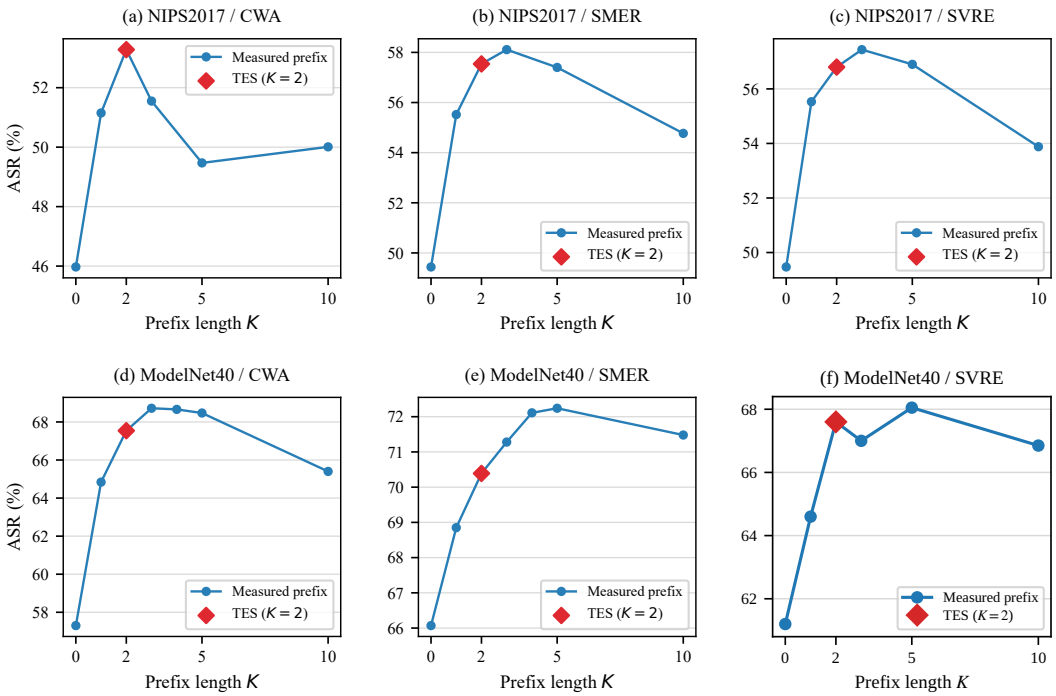


Figure 8: Prefix-length sensitivity at $T = 10$. Panels (a–c) use NIPS2017 ($4/255$, standard 12 targets); panels (d–f) use ModelNet40 (0.06, standard 19 targets). The plotted configurations are retained; the subsequently completed point-cloud SVRE $K = 4$ result is reported in Table 38. The red diamond marks TES.

Figure 8 shows a rapid improvement from Base followed by method-dependent saturation. The image CWA curve peaks at $K = 2$, while image SMER and SVRE peak at $K = 3$. On ModelNet40, the largest completed ASR occurs at $K = 3$ for CWA and $K = 5$ for SMER and SVRE. These experimental results align closely with our main claim: early-stage refinement alone is sufficient to achieve near-optimal performance on images, whereas prolonged refinement may even lead to performance degradation. While point clouds are similarly most sensitive to early refinements, subsequent refinements continue to yield positive gains, typically leading to peak performance in the intermediate stages. Furthermore, both modalities exhibit a consistent trend—applying the refinement method throughout the entire process rarely yields optimal results.

Larger point-cloud budget. Table 36 retains the corresponding $\epsilon = 0.12$ comparison. It is reported separately from the default-budget grid.

Table 36: ModelNet40 prefix sensitivity at $\epsilon = 0.12$ ($T = 10$). ASR (%) uses the standard 19-target panel.

K	CWA	SMER	SVRE
0	84.28	87.46	86.10
1	87.20	88.96	88.02
2	88.08	89.69	88.58
3	88.29	89.97	88.99
5	88.15	90.27	89.05
10	87.24	89.70	88.43

D.2 Attack Horizon and Proportional Prefixes

When the horizon changes, a 20% refinement prefix means $K = 0.2T$: $K = 2, 4, 6, 8$ for $T = 10, 20, 30, 40$, respectively. A fixed $K = 2$ must not be described as a 20% prefix for every horizon. Figure 9 compares Base, Full, fixed $K = 2$, and a proportional prefix at $T = 10, 20, 40$.

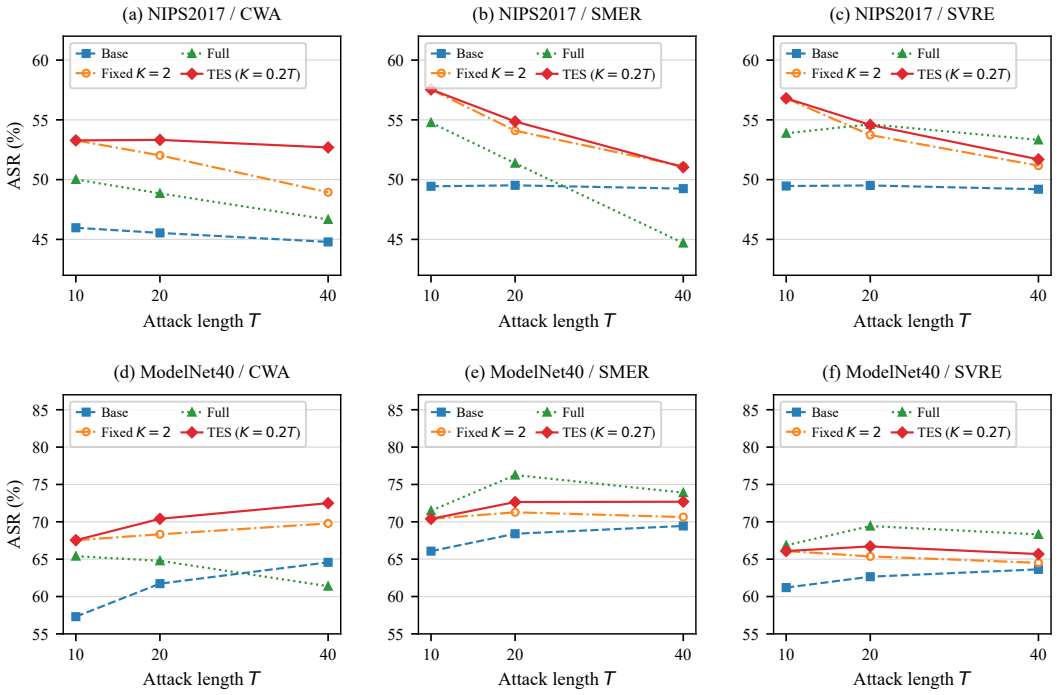


Figure 9: Attack length and prefix allocation. Panels (a–c) show NIPS2017 and (d–f) ModelNet40 under the same standard panels as Figure 8. Fixed $K = 2$ and $K = 0.2T$ coincide at $T = 10$. Each horizon retains its native step-size rule; increasing T is not a fixed-step continuation experiment.

In Figure 9, proportional prefixes improve over fixed $K = 2$ in most longer-horizon comparisons, especially point-cloud CWA. The effect is not universal: image SMER at $T = 40$ has 51.0410% ASR at $K = 8$, slightly below 51.1139% at $K = 2$. Full remains stronger than the proportional prefix for long-horizon point-cloud SMER and SVRE; image SVRE at $T = 40$ likewise gives 53.3204% for Full versus 51.6928% for $K = 8$. Thus, the curves support horizon-dependent prefix allocation.

For a completed tested prefix grid $\mathcal{K}_T$, we define the descriptive retention ratio

$$\rho_T(K) = \frac{\mathrm{ASR}(K, T)}{\max_{k \in \mathcal{K}_T} \mathrm{ASR}(k, T)}. \tag{18}$$

The denominator is the best ASR in the specified tested grid, not a global optimum. It is not the improvement over Base. We distinguish $\rho \geq 0.95$, $0.90 \leq \rho < 0.95$, and $\rho < 0.90$ as descriptive

categories; none establishes statistical equivalence. A missing 30% configuration is not interpolated from adjacent values.

Table 37: Completed ModelNet40 proportional-prefix grids. ASRs and retention ratios are percentages. The 20% and 30% configurations are assessed separately against the listed complete grid.

Method	T	Tested K	ASR_{20}	ASR_{30}	Best	ρ_{20}	ρ_{30}
CWA	10	0,1,2,3,4,5,10	67.54	68.72	68.72	98.29	100.00
CWA	20	0,2,4,6,8,20	70.40	71.72	72.11	97.63	99.47
CWA	30	0,2,3,6,9,12,30	72.04	73.02	73.40	98.14	99.48
CWA	40	0,2,4,8,12,16,40	72.50	73.64	73.82	98.21	99.75
SMER	20	0,2,4,6,8,20	72.64	73.57	76.26	95.27	96.48
SMER	30	0,2,3,6,9,12,30	72.90	73.96	76.48	95.33	96.71
SMER	10	0,1,2,3,4,5,10	70.39	71.28	72.24	97.44	98.66
SMER	40	0,2,4,8,12,16,40	72.69	73.09	73.91	98.35	98.89
SVRE	10	0,1,2,3,4,5,10	67.59	67.03	68.07	99.29	98.48
SVRE	20	0,2,4,6,8,20	66.71	68.24	69.43	96.08	98.29
SVRE	30	0,2,3,6,9,12,30	66.27	67.33	68.90	96.18	97.72
SVRE	40	0,2,4,8,12,16,40	65.68	66.65	68.31	96.15	97.57

The completed rows in Table 37 exceed 95% retention at both proportions. This statement is restricted to these listed grids. The completed SMER $T = 30$ grid gives 72.9034% at $K = 6$ and 73.9621% at $K = 9$, compared with the measured best of 76.4778% at $K = 30$. Their retention ratios are 95.3263% and 96.7106%, respectively; both exceed 95%, although Full has the higher endpoint ASR. The additional complete SMER and SVRE grids are included in Table 37.

Table 38: Completed additional $T = 10, K = 4$ point-cloud configurations ($N = 2{,}468$, 19 standard targets), ASR (%).

Method	ASR
SMER	72.1129
SVRE	67.6709

D.3 Per-Iteration Attack Trajectories

Endpoint sensitivity to K and evolution along one attack are different measurements. For the trajectory experiment, one native attack is run per batch and read-only snapshots are collected after its updates. All snapshots belong to the same attack trajectory; they are not independently regenerated attacks or independent stochastic repetitions.

Figure 10 reports every observed iterate for all three methods at $T = 10, 20, 30, 40$, with $K = 0.2T$. The marked refinement boundaries and subsequent curves illustrate the base continuation.

For SMER, ASR is 11.7835% at the end of the two refinement updates and rises to 70.4024% after the full base continuation. This supports the contribution of subsequent optimization within this trajectory. It does not replace the endpoint K-grid retention test, and the independently generated trajectory endpoint need not equal the cached reference endpoint exactly.

D.4 Surrogate Number and Composition

We compare the default five-surrogate ensemble with M2, M3A, and M3B. For images, M2 contains ResNet-50 and Swin-T; M3A adds DenseNet-121; M3B contains DenseNet-121, ConvNeXt-Tiny, and VGG-16. For point clouds, M2 contains CurveNet and RSCNN; M3A adds PointCNN; M3B contains PointCNN, PointConv, and Point2Vec. M3A versus M3B changes composition at fixed ensemble size; comparisons with M2 or M5 also change the number of surrogates.

Table 39 reports Base, TES, and Full on the unchanged standard target panel. The M5 reference is the matching default result and is not counted as a new independent experiment.

Across different surrogate combinations, early prefix observations exhibit varying effects. For instance, in CWA+M2, TES achieves 91.46% of the best ASR among Base, TES, and Full; in point

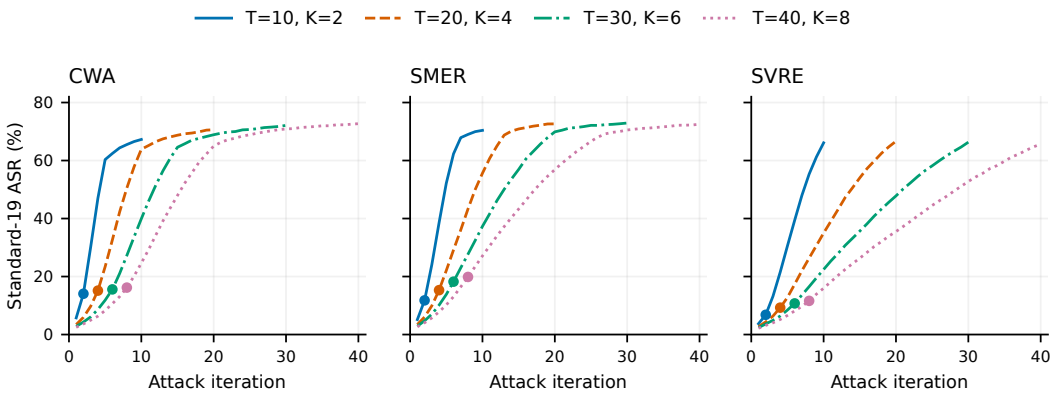


Figure 10: All observed ModelNet40 trajectories ($N = 2{,}468$, $\epsilon = 0.06$, 19 standard reporting targets). Each line is one native trajectory per batch with $K = 0.2T$; the dot marks step K, after which base updates continue. Snapshots are read from that same trajectory, not independently generated prefix-scan attacks.

Table 39: Surrogate-composition comparisons, ASR (%). Rows vary the surrogate ensemble, while the reporting panel remains fixed within each modality.

Method	Ensemble	Base	TES	Full
NIPS2017				
CWA	M2	21.20	19.39	13.52
CWA	M3A	34.20	36.39	30.39
CWA	M3B	42.55	46.78	43.94
CWA	M5	45.97	53.28	50.01
SMER	M2	20.36	22.31	22.64
SMER	M3A	34.81	40.37	38.83
SMER	M3B	46.43	50.08	45.63
SMER	M5	49.44	57.54	54.77
SVRE	M2	20.36	23.52	23.25
SVRE	M3A	34.78	40.94	37.52
SVRE	M3B	46.39	51.74	46.12
SVRE	M5	49.47	56.80	53.88
ModelNet40				
CWA	M2	58.18	61.49	62.63
CWA	M3A	56.50	59.61	56.83
CWA	M3B	57.01	62.92	64.04
CWA	M5	57.30	67.54	65.40
SMER	M2	60.06	64.43	68.15
SMER	M3A	61.00	64.47	65.80
SMER	M3B	59.93	65.20	68.96
SMER	M5	66.07	70.39	71.48
SVRE	M2	56.12	62.82	66.17
SVRE	M3A	56.97	61.07	62.24
SVRE	M3B	55.24	61.76	62.43
SVRE	M5	61.19	67.59	66.85

cloud SMER, the corresponding ratios for M2 and M3B reach 94.54% and 94.55%, respectively. Nevertheless, even in the worst-case scenario, the utility of the early stage consistently exceeds 90% of that achieved by the Full configuration.

D.5 Extended-Base and Short-Refinement Controls

To test whether the refinement budget is better spent on additional base updates, we compare TES with a longer base-only attack. CWA uses B^{12} (60 versus 60 reference graphs for TES), SMER uses B^{16} (80 versus 80), and SVRE uses B^{22} (110 versus 114).

The original-step-size condition keeps the $T = 10$ base step size, while the rescaled condition multiplies it by $10/T_{\text{long}}$. Other native attack settings and the feasible set $\mathcal{C}(x, \epsilon)$ are retained. Figure 11 evaluates all controls on the same standard panel within each dataset and method. The red TES bar exceeds both extended-base bars for all three methods on both image and point-cloud tasks.

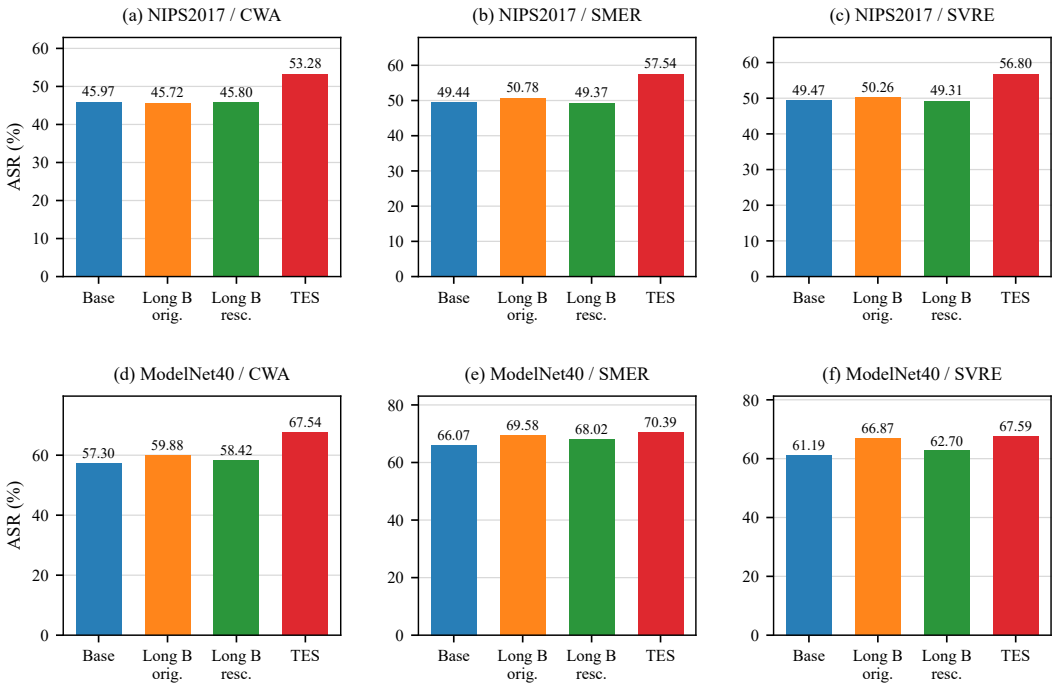


Figure 11: TES versus extended Base on NIPS2017 (a–c) and ModelNet40 (d–f). The extended horizons are 12, 16, and 22 for CWA, SMER, and SVRE. Original-step and rescaled-step controls are separate bars. Graph budgets match TES exactly for CWA/SMER (60/80); SVRE uses 110 versus 114 graphs. No equal-runtime claim is made.

The short-refinement controls instead ask whether base continuation can be replaced by stopping after refinement alone. We evaluate R^2 with the original $T = 10$ step size, and R^q with both original and recalibrated step sizes, where $q = 4$ for SMER and $q = 3$ for SVRE. SMER R^4 and TES each cost 80 reference graphs; SVRE R^3 costs 111 versus 114 for TES. Recalibration uses the native short-horizon step-size rule; the original-step-size condition preserves the ten-update step size. The point-cloud SMER native step-size factor is retained.

TES exceeds the recalibrated short-refinement control in these four completed default-budget settings, by 1.66 and 5.53 points on images for SMER and SVRE, and 5.19 and 6.61 points on point clouds. This supports retaining base continuation in these comparisons.

Table 40: Short-refinement controls at the default budgets, ASR (%). Short R means R^4 for SMER and R^3 for SVRE. No equal-time claim is made.

Data	Method	R^2 orig.	Short R orig.	Short R recal.	TES
Image	SMER	5.85	17.31	55.88	57.54
Image	SVRE	4.61	9.19	51.27	56.80
Point	SMER	11.80	37.13	65.20	70.39
Point	SVRE	6.80	11.87	60.98	67.59

D.6 Stability across Attack Seeds

The three-seed method evaluation uses seeds 17, 29, and 43 for images and 31, 37, and 43 for point clouds. All seeds use the same evaluated sample collection, surrogate ensemble, standard target panel, and attack configuration. Table 41 reports each seed separately; Table 42 gives the arithmetic mean and sample standard deviation across the three seeds.

Table 41: Per-seed standard-panel ASR (%) at $T = 10$ and the default perturbation budgets.

Data	Method	Seed	Base	TES	Full
Image	CWA	17	45.97	53.28	50.01
Image	CWA	29	46.00	53.45	50.06
Image	CWA	43	45.96	53.45	50.06
Image	SMER	17	49.44	57.54	54.77
Image	SMER	29	49.40	57.64	55.06
Image	SMER	43	49.47	57.27	54.87
Image	SVRE	17	49.47	56.80	53.88
Image	SVRE	29	49.41	56.63	54.14
Image	SVRE	43	49.48	57.38	53.34
Point	CWA	31	57.30	67.54	65.40
Point	CWA	37	57.26	66.90	65.48
Point	CWA	43	57.08	67.21	65.47
Point	SMER	31	66.07	70.39	71.48
Point	SMER	37	66.00	70.60	71.85
Point	SMER	43	65.72	70.33	71.64
Point	SVRE	31	61.19	67.59	66.85
Point	SVRE	37	61.18	66.27	66.71
Point	SVRE	43	61.18	66.18	66.91

Table 42: Mean $\pm$ sample standard deviation of ASR (%) across the three attack seeds in Table 41.

Data	Method	Base	TES	Full
Image	CWA	45.98 ± 0.02	53.39 ± 0.10	50.04 ± 0.03
Image	SMER	49.44 ± 0.03	57.48 ± 0.19	54.90 ± 0.15
Image	SVRE	49.45 ± 0.03	56.94 ± 0.39	53.79 ± 0.41
Point	CWA	57.22 ± 0.12	67.22 ± 0.32	65.45 ± 0.05
Point	SMER	65.93 ± 0.19	70.44 ± 0.14	71.66 ± 0.19
Point	SVRE	61.18 ± 0.01	66.68 ± 0.79	66.82 ± 0.10

The seed results preserve the modality-dependent comparison with Full: TES is higher on the image standard panel, whereas point-cloud SMER and SVRE remain slightly lower than Full on average.

D.7 Matched-Budget Early, Uniform, and Random Schedules

This control holds $T = 10$, $K = 2$, the perturbation budget, sources, samples, target panel, attack/evaluation seeds, batch size, and native step-size rule fixed within each method and modality. It varies only the positions of the two R updates. Early selects positions $(1, 2)$ and Uniform selects $(1, 6)$, with positions numbered from one. Five random schedules are fixed before their evaluation by `random.Random(s).sample(range(10),2)`, with schedule seeds $s = 19001, \ldots, 19005$.

In this order, the selected one-based pairs are $(1, 10)$, $(1, 3)$, $(3, 8)$, $(5, 10)$, and $(5, 7)$. Each mask is shared across samples and used for all three methods and both modalities. Schedule seeds are distinct from attack seeds; masks are neither optimized nor selected using target results.

All schedules use the method-evaluation setting: NIPS2017 has 1,000 samples, $\epsilon = 4/255$, attack seed 17 and 12 standard targets; ModelNet40 has 2,468 samples, $\epsilon = 0.06$, seed 31 and 19 standard reporting targets. Attack batches follow Table 10. Their reference graph counts are 60 for CWA, 80 for SMER, and 114 for SVRE. These counts match within each method and do not imply equal elapsed time. Temporary and persistent states follow Appendix A.2, including when a random schedule returns from B to R.

Figure 12 shows Early, Uniform, and all five random-mask ASRs for each setting; the accompanying figure data provide exact values, including the random-mask mean and sample standard deviation. This standard deviation measures variation across these fixed position masks, conditional on the attack seed and evaluated samples.

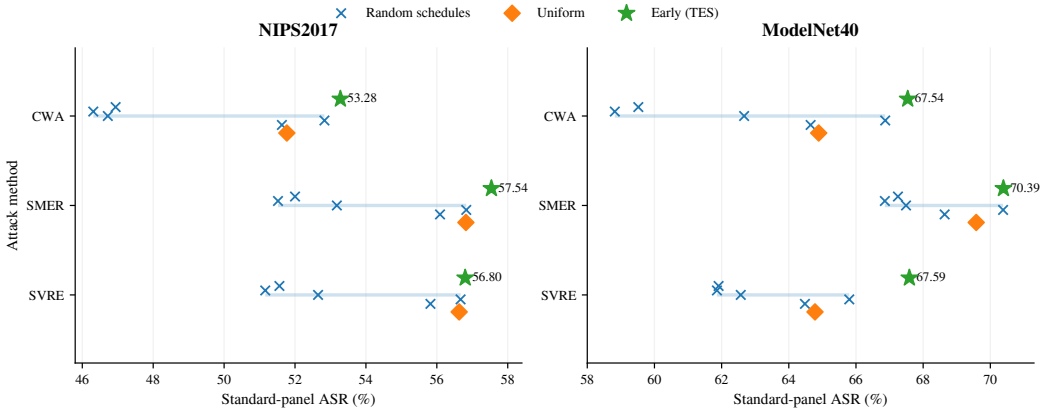


Figure 12: Matched-budget scheduling on NIPS2017 (standard 12 targets) and ModelNet40 (standard 19 targets). Stars denote Early and diamonds Uniform; R1–R5 denote all five fixed random masks in seed order 19001–19005. Vertical offsets separate masks and schedules; gray segments show the random-mask range. These are descriptive ASRs conditional on the fixed attack seed, not independent attack-seed repetitions.

Early exceeds Uniform and every tested random mask in all six settings; even the best-performing random mask does not surpass Early.

E Computational Cost and Runtime Measurements

This appendix distinguishes reference computation from measured attack-generation time. Reference graphs describe the prescribed surrogate-gradient work. Wall-clock time additionally depends on implementation, batch size, hardware, and concurrent load.

E.1 Reference Computation

Let c_B and c_R denote the reference numbers of differentiated surrogate input-gradient computations for one base and one refinement update. For a horizon T and prefix length K, the scheduled attack has reference cost

$$G(T, K) = (T - K)c_B + Kc_R. \tag{19}$$

The five-surrogate configurations use $c_B = 5$, with $c_R = 10$, 20, and 37 for CWA, SMER, and SVRE, respectively. Table 43 gives the resulting $T = 10$ costs.

Reference-budget matching is exact for the CWA B^{12} versus TES comparison (60/60), SMER B^{16} versus TES (80/80), and SMER R^4 versus TES (80/80). For SVRE, B^{22} costs 110 and R^3 costs 111, compared with 114 for TES, so these are approximate-budget controls.

Table 43: Reference gradient-computation counts for the five-surrogate $T = 10$ configurations. The reduction is relative to Full and is not a measured runtime reduction.

Method	c_B	c_R	Base	TES ($K = 2$)	Full	Reduction
CWA	5	10	50	60	100	40.00%
SMER	5	20	50	80	200	60.00%
SVRE	5	37	50	114	370	69.19%

Hardware and software. The experimental server uses NVIDIA GeForce RTX 4090 GPUs (24 GB each), running Ubuntu 22.04.3 LTS with NVIDIA driver 535.247.01. The experiment environment uses Python 3.10.18, PyTorch 2.1.0+cu118, torchvision 0.16.0+cu118, CUDA 11.8 (the PyTorch build), and cuDNN 8.7.0. Other key dependencies are NumPy 1.26.4, SciPy 1.14.0, timm 1.0.24, and Pillow 11.0.0.

E.2 Timing Protocol and Measurement Scope

Attack-call timing. The measured interval covers the native attack call, with CUDA synchronization at the interval boundaries. Model loading, warm-up, target-model evaluation, and cached-result retrieval are excluded. Seconds per sample are computed from the timed attack calls and their processed sample counts. Timing repetitions reuse a fixed sample collection.

The main-text NIPS2017 times use the batch-8 measurements in Table 44; ModelNet40 SVRE uses the six-round shared-load $T = 10$ measurements in Table 45. The remaining timing benchmarks use 48 samples and three rounds on an exclusive GPU, with image batch 12, point-cloud CWA/SMER batch 8, and SVRE batch 6. When reporting an average over datasets, the dataset times receive equal weight, and the aggregate speedup is the ratio of the unrounded mean Full time to the unrounded mean TES time.

Exclusive-GPU sweep. The image prefix sweep uses batch size 8, 48 fixed samples, and three rounds, with warm-up excluded. Table 44 supplies the main-text NIPS2017 Full and TES times at $K = 10$ and $K = 2$, respectively.

Table 44: Image $T = 10$ prefix timing on an exclusive GPU, in seconds per sample (batch 8, 48 samples, three rounds). These are attack-call times, not full evaluation times.

K	CWA	SMER	SVRE
0	0.1773	0.1801	0.1733
1	0.2040	0.4280	0.2940
2	0.2258	0.6803	0.4484
3	0.2430	0.9315	0.5580
5	0.2906	1.4173	0.8257
10	0.3980	2.6541	1.5185

Shared-GPU measurements. The additional point-cloud measurements use six or seven fixed rounds with balanced, rotated configuration orders. CUDA is synchronized, warm-up is excluded, and concurrent GPU processes are recorded around each timed batch.

For each round r, the relative time of configuration K is computed against its same-round $K = 0$ measurement:

$$q_{r,K} = \frac{\tau_{r,K}}{\tau_{r,0}}. \tag{20}$$

The average and uncertainty of these paired ratios are evaluated from the fixed rounds.

E.3 Quality Criteria and Qualified Results

A shared-load timing group is eligible for the paper's time–ASR plots only if every configuration has all prescribed rounds and satisfies three checks: the coefficient of variation of absolute seconds per sample is at most 5%; the relative half-width of its descriptive 95% interval is at most 5%; and

the relative half-width of the 95% interval for its paired time ratio to $K = 0$ is at most 5%. All fixed rounds are retained.

Table 45 reports the qualified ModelNet40 groups: CWA at $T = 20, 30, 40$ and SVRE at $T = 10$, all at $\epsilon = 0.06$. The timing runs use 48 samples on the same recorded physical GPU under shared load, with batch 8 for CWA and batch 6 for SVRE. CWA $T = 20$ and SVRE $T = 10$ use six rounds; CWA $T = 30/40$ use seven. The ASR column comes from the matching complete 2,468-sample evaluation on the standard 19-target panel.

Table 45: Qualified shared-load ModelNet40 measurements. Intervals describe seconds per sample over the fixed timing rounds; ASR is evaluated separately on the full 2,468 samples.

K	ASR (%)	Seconds/sample	Descriptive 95% interval
CWA, $T = 20$			
0	61.7231	0.7528	[0.7443, 0.7613]
2	68.3288	0.8286	[0.8107, 0.8465]
4	70.4013	0.9055	[0.8881, 0.9228]
6	71.7225	0.9794	[0.9564, 1.0025]
8	72.1078	1.0657	[1.0529, 1.0785]
20	64.7837	1.5273	[1.5043, 1.5503]
CWA, $T = 30$			
0	63.6091	1.1441	[1.1242, 1.1640]
2	69.4766	1.2295	[1.1986, 1.2605]
3	69.7560	1.2986	[1.2644, 1.3329]
6	72.0383	1.3901	[1.3741, 1.4062]
9	73.0191	1.5179	[1.4980, 1.5378]
12	73.4010	1.6368	[1.6101, 1.6636]
30	62.6991	2.3794	[2.3261, 2.4326]
CWA, $T = 40$			
0	64.5745	1.4999	[1.4751, 1.5247]
2	69.7834	1.6014	[1.5808, 1.6221]
4	70.8128	1.6801	[1.6369, 1.7234]
8	72.4970	1.8367	[1.8126, 1.8607]
12	73.6377	1.9582	[1.9316, 1.9848]
16	73.8215	2.1267	[2.1038, 2.1496]
40	61.3824	3.1202	[3.0846, 3.1559]
SVRE, $T = 10$			
0	61.1947	0.4916	[0.4792, 0.5041]
1	64.6001	0.8561	[0.8291, 0.8831]
2	67.5900	1.2178	[1.1917, 1.2439]
3	67.0294	1.5575	[1.5247, 1.5904]
5	68.0673	2.2109	[2.1552, 2.2666]
10	66.8513	3.8231	[3.7482, 3.8979]

For SVRE at $T = 10$, $K = 2$ reduces the measured attack time from 3.8231 to 1.2178 seconds per sample, achieving more than a threefold speedup while increasing ASR from 66.8513% for Full to 67.5900%. Thus, TES simultaneously improves transferability and substantially reduces runtime in this setting.

Table 46: SMER ModelNet40 timings for $T = 20, 30, 40$. All prescribed rounds are retained; intervals describe repeated passes under the recorded shared load.

Method	T	K	Seconds/sample	95% interval
SMER	20	0	0.7569	[0.6922, 0.8215]
SMER	20	2	2.0001	[1.8320, 2.1683]
SMER	20	4	3.0211	[2.9198, 3.1224]
SMER	20	6	4.2648	[3.8853, 4.6442]
SMER	20	8	5.5074	[4.9847, 6.0302]
SMER	20	20	12.3354	[11.7118, 12.9589]
SMER	30	0	1.1069	[1.0352, 1.1786]
SMER	30	2	2.2228	[2.1610, 2.2846]
SMER	30	3	2.7959	[2.7171, 2.8748]
SMER	30	6	4.4468	[4.2994, 4.5943]
SMER	30	9	6.7114	[5.8484, 7.5743]
SMER	30	12	9.3564	[7.4591, 11.2536]
SMER	30	30	18.8042	[17.2441, 20.3642]
SMER	40	0	1.5755	[1.3234, 1.8276]
SMER	40	2	2.9988	[2.3402, 3.6574]
SMER	40	4	4.4440	[3.5120, 5.3760]
SMER	40	8	6.6460	[5.3630, 7.9290]
SMER	40	12	9.7020	[7.4380, 11.9659]
SMER	40	16	11.4137	[9.5542, 13.2731]
SMER	40	40	26.0994	[23.4682, 28.7306]

Table 47: SVRE ModelNet40 timings for $T = 20, 30, 40$. All prescribed rounds are retained; intervals describe repeated passes under the recorded shared load.

Method	T	K	Seconds/sample	95% interval
SVRE	20	0	0.9393	[0.9013, 0.9773]
SVRE	20	2	1.6009	[1.5373, 1.6644]
SVRE	20	4	2.2580	[2.1914, 2.3246]
SVRE	20	6	2.8875	[2.7851, 2.9899]
SVRE	20	8	3.4338	[3.3464, 3.5212]
SVRE	20	20	7.4250	[7.1767, 7.6734]
SVRE	30	0	1.4644	[1.4563, 1.4725]
SVRE	30	2	2.1743	[2.0902, 2.2583]
SVRE	30	3	2.5640	[2.3491, 2.7790]
SVRE	30	6	3.4914	[3.2725, 3.7104]
SVRE	30	9	4.4766	[4.2083, 4.7449]
SVRE	30	12	5.4520	[5.0865, 5.8174]
SVRE	30	30	11.8347	[11.5289, 12.1405]
SVRE	40	0	1.9186	[1.8554, 1.9818]
SVRE	40	2	2.5762	[2.4960, 2.6565]
SVRE	40	4	3.2127	[3.1281, 3.2973]
SVRE	40	8	4.4540	[4.3127, 4.5952]
SVRE	40	12	5.7583	[5.4611, 6.0556]
SVRE	40	16	6.9983	[6.5324, 7.4642]
SVRE	40	40	15.2769	[14.7949, 15.7589]

Table 48: Stepwise radius matching on images. ASR is computed on samples correctly classified by each target before the attack. ASR differences are reported in percentage points (pp). Intervals use 30,000 paired sample bootstrap resamples with seed 58. Raw margin scales are not directly comparable across models.

Metric	MobileNetV3 Large	DeiT Small
Immediate margin difference after branching, natural	-0.2808	-0.1800
Terminal margin difference, natural	$+0.2407$	-0.1559
Terminal margin difference, controlled	$+0.2104$	-0.1925
Intervention effect on terminal margin, I	-0.0303	-0.0365
97.5% interval for the margin intervention effect	$[-0.0578, -0.0012]$	$[-0.0709, -0.0007]$
Terminal ASR difference, natural (pp)	$+2.093$	-1.712
Terminal ASR difference, controlled (pp)	$+2.093$	-2.854
Intervention effect on terminal ASR, I (pp)	0.000	-1.142
97.5% interval for the ASR intervention effect (pp)	$[-1.744, +1.744]$	$[-2.626, +0.342]$

F Analysis and Interpretation of Stage-Dependent Gains

The main text and Appendices B.3 and D.1 establish that refinement placement and prefix length affect attack performance. This section discusses a conditional explanation and examines some of its contributing factors through additional interventions on images. For point clouds, we present only a conjecture that remains to be tested.

F.1 Images: Early Refinement Shapes Trajectories That Subsequent Updates Can Exploit

We hypothesize that refinement does more than improve the immediate attack outcome: by changing the perturbation and the update state, it may also influence how subsequent base updates evolve. Early refinement may therefore produce terminal gains after base continuation even without an immediate advantage. In some image configurations, a short refinement prefix may already establish much of the useful change that subsequent updates can exploit, reducing the additional value of continued refinement.

To examine whether perturbation magnitude and subsequent feedback contribute to this process, we conduct a CWA intervention experiment on 1,000 NIPS2017 images with $\epsilon = 4/255$ and $T = 10$, retaining the original five surrogate models and step size rule. After a shared refinement update R, the two branches apply either another R update or a base update B, followed by eight B updates in both branches.

We define the attack margin as the largest logit among incorrect classes minus the true-class logit. All branch differences in the table are computed as $R - B$. The intervention effect is defined as

$$I = E_{\text{controlled}} - E_{\text{natural}}, \tag{21}$$

where E denotes the corresponding branch difference under the controlled or natural condition.

For MobileNetV3-Large, the natural margin difference changes from negative immediately after branching to positive at the terminal step, indicating that the immediate outcome does not fully characterize the final value of this refinement update. Dynamic radius matching reduces its positive margin advantage, consistent with perturbation magnitude and subsequent feedback contributing to that advantage, although most of the mean advantage remains.

This evidence has clear boundaries. For DeiT-Small, the natural terminal difference is already negative, so its intervention result cannot be described as removing a positive gain. Moreover, the ASR intervention intervals include zero for both targets. The experiment therefore provides **partial evidence about the mechanism at the margin level, with effects that depend on the target model**.

F.2 Point Clouds: A Conjecture About Persistent Opportunities for Effective Correction

For the point cloud configurations in which longer refinement prefixes remain beneficial, we propose the following conjecture:

Early refinement may not yet have sufficiently shaped a useful trajectory for subsequent base updates to exploit. Later refinement may still alter the final target response, allowing its beneficial window to extend to later stages.

Such opportunities need not exist for every object or target model. Further trajectory adjustments may improve the attack margin on some targets without causing samples to cross a decision boundary. They may also produce additional successes on some targets while causing losses on others. Thus, placing refinement early can be more efficient while extending the refinement prefix still provides a net benefit.

Due to time and computational constraints, we have not directly tested this conjecture. It should therefore be regarded as a plausible explanation rather than a verified mechanism.