DENSITY FEEDBACK CONTROL FOR TRANSFERABILITY OF GENERATIVE SPARSE ADVERSARIAL ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generator-based sparse adversarial attacks train encoder–decoder networks to produce transferable sparse perturbations for black-box robustness evaluation. Their perturbation density, however, is set only indirectly by a fixed sparsity coefficient whose mapping to density is unknown and shifts during training, so reaching a target density requires re-tuning; this fixed pressure also yields little transferability gain once density reduction begins. We propose Density Feedback Control (DFC), which tightens or relaxes this coefficient at the end of each epoch according to the gap between the observed hard-mask density and the target, without changing the generator architecture or loss. At a 10% target density, DFC brings the mean density of three base attacks within one percentage point of the target in a single run and improves the transfer attack success rate (ASR) of two of them on CNN and Transformer targets, including under per-image ℓ_0 budgets, while perturbing fewer pixels. Tracing GPG's training, we observe that as DFC relaxes the coefficient, the transfer ASR recovers with the density and overtakes the fixed-coefficient baseline with fewer pixels. For EGS-TSSA, which restricts perturbations to source-model feature-importance regions, DFC yields no consistent gain; we analyze how this restriction interacts with density feedback.

1 INTRODUCTION

Sparse adversarial attacks expose a failure mode in which a vision classifier can be deceived by modifying only a small subset of input pixels (Modas et al., 2019; Dong et al., 2020; He et al., 2022; Song et al., 2024; Ming et al., 2024; Song et al., 2026; Jiang et al., 2026). Sparse attacks are important for assessing robustness, as they serve as stress tests under threat models that restrict the number of modifiable input elements (Croce et al., 2022). Sensitivity to perturbations affecting only a few pixels also offers a complementary perspective on the analysis of deep neural networks (DNNs) (Dong et al., 2020). In particular, generator-based sparse attacks have been proposed to learn a source model's vulnerabilities using training data and transfer the resulting sparse attacks to a target model, thereby revealing its vulnerabilities in a black-box setting where its internal information is unavailable.

Such attacks, including TSAA (He et al., 2022), EGS-TSSA (Ming et al., 2024), and GPG (Jiang et al., 2026), train the generator on an adversarial loss plus a sparsity loss weighted by a coefficient λ, which sets the perturbation density only indirectly. Reaching a desired density therefore requires repeated training runs for TSAA, multi-stage optimization for EGS-TSSA, and fine-tuning with a re-tuned coefficient for GPG. In each case, λ is specified before a training or fine-tuning stage, whereas the resulting density is observed only afterwards. The mapping from λ to the density is unknown, differs across attacks, and evolves during generator training, preventing this open-loop procedure from directly targeting a prescribed density.

Beyond making the density difficult to prescribe, fixed-coefficient training also limits transferability. Fig. 1 traces GPG, TSAA, and EGS-TSSA under the protocol of Jiang et al. (2026), which proceeds through warm-up, density reduction, stabilization, and density fine-tuning with a re-tuned coefficient. Transferability improves primarily during warm-up and, as expected, drops during density reduction; however, the stabilization and fine-tuning stages recover little of this loss despite occupying more than half of training, while the density remains nearly constant (Fig. 1b). Because the coefficient is fixed within each stage and changed only by manual re-tuning between stages,

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.

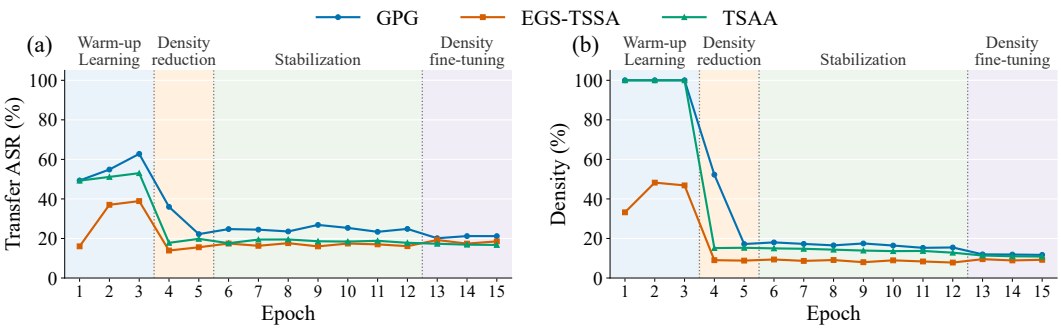

Figure 1: Training dynamics of generator-based sparse adversarial attacks. Evolution of (a) transferability and (b) mean pixel density across four training stages: warm-up, density reduction, stabilization, and density fine-tuning. Transferability is measured by the mean attack success rate (ASR) across five CNN-based and five Transformer-based target models.

the sparsity pressure does not respond to where the density lies relative to the target. The generator nonetheless keeps changing under this pressure: it replaces over 30% of its support per epoch and discards locations more effective than random ones, a loss its ASR never recovers (Sec. 4.6). This motivates making the sparsity pressure respond to the observed density, tightening it when the density is above the target and relaxing it when it falls below.

We propose Density Feedback Control (DFC), which links the weight of the differentiable sparsity loss to the hard-mask density that is actually evaluated. At the end of each epoch, DFC measures the mean hard-mask density of the generated perturbations and updates the sparsity coefficient from its normalized deviation from the target. DFC changes only how the coefficient is set; the architecture and loss of the base attack are unchanged, so it replaces the multi-stage schedules of existing generator-based attacks without further modification.

Our contributions are summarized as follows:

- DFC treats the sparsity coefficient as a control input driven by the hard-mask density, bringing the mean density of three base attacks to within one percentage point of a prescribed target in a single training run, without stage-wise re-tuning.
- DFC improves the mean transfer ASR of GPG and TSAA on CNN and Transformer targets while perturbing fewer pixels on average. Under per-image ℓ_0 budgets, its gain comes not only from placing more images within the budget but also from attacking the same images more successfully.
- Tracing training, we observe that as DFC relaxes the coefficient, the density recovers from below and the transfer ASR recovers with it, overtaking the fixed-coefficient baseline with fewer pixels.

2 Related Works

Black-box adversarial attacks, which do not require access to the target model's internal parameters or gradients, can be broadly categorized into query-based and transfer-based methods.

2.1 Query-based Sparse Attacks

Query-based attacks craft adversarial examples by iteratively modifying the input, guided by the target model's prediction scores or labels. Early sparse variants used evolutionary search over individual pixels (Su et al., 2019) or over localized perturbations parameterized as lines and curves (Jere et al., 2019). Later work searched for vulnerable patches with reinforcement learning (Yang et al., 2020), rearranged a small number of pixels (Pomponi et al., 2023), and selected pixel replacements from a synthetic image via Bayesian search under an ℓ_0 constraint (Vo et al., 2024). Song et al. (2024) introduced a REINFORCE-based framework with remember-and-forget processes for classification and object detection, which Song et al. (2026) extended to decision-based attacks on seman-

tic segmentation. Because these methods require repeated queries to the target model, transfer-based approaches instead craft perturbations on a source model without target-model feedback.

2.2 TRANSFER-BASED SPARSE ATTACKS

Transfer-based sparse attacks generate adversarial perturbations against a source model under white-box access and transfer them to a target model whose internal information is unavailable. PGD_0 adapted projected gradient descent (PGD) to sparse attacks by projecting perturbations onto an ℓ_0 ball (Madry et al., 2018; Croce & Hein, 2019). SparseFool extended DeepFool, which seeks perturbations that cross a nearby decision boundary, to the sparse setting (Modas et al., 2019; Moosavi-Dezfooli et al., 2016). GreedyFool uses input gradients to greedily select influential pixels and subsequently prunes redundant perturbations to reduce the number of modified pixels (Dong et al., 2020). More recently, sPGD jointly optimized perturbation magnitudes and a sparse support mask through PGD-like gradient updates under an ℓ_0 constraint (Zhong & Liu, 2026). Despite their strong attack performance on the source model, these gradient-based methods exhibit limited transferability to unseen target models.

Generator-based adversarial attacks train a generator to produce adversarial perturbations with improved transferability (Naseer et al., 2019; Xiao et al., 2018; Baluja & Fischer, 2018; Poursaeed et al., 2018; Song et al., 2018). Inspired by this paradigm, He et al. (2022) proposed TSAA, which encodes the input image into a compact representation and learns a pixel-selection mask M and perturbation values P to construct a sparse perturbation δ. TSAA enables substantially faster attack generation than gradient-based methods while improving transferability. Building on this approach, EGS-TSSA (Ming et al., 2024) improves transferability by restricting the selection of M to regions containing features important for classification, using a Group Feature Importance (GFI) mask derived from the feature maps of the source model. Similarly, GPG (Jiang et al., 2026) enhances transferability by encouraging P to resemble class-specific perturbations generated by PGD.

The training objective for a sparse perturbation generator is:

$$\mathcal{L}(\theta; \lambda) = \mathcal{L}_{\text{adv}}(\theta) + \lambda \mathcal{L}_{\text{spa}}(\theta), \tag{1}$$

where $\mathcal{L}_{\text{adv}}$ encourages misclassification, $\mathcal{L}_{\text{spa}}$ promotes sparsity, and λ controls their relative importance. Prior methods have primarily improved transferability through the design of $\mathcal{L}_{\text{adv}}$.

2.3 ADAPTIVE CONTROL OF REGULARIZATION WEIGHTS

The weight of a regularization term can be adjusted during training by a prescribed schedule or by feedback. Prescribed schedules, such as cyclical KL annealing (Fu et al., 2019), repeatedly vary the regularization weight during training. Feedback methods treat the weight as a Lagrange multiplier updated from constraint violation (Platt & Barr, 1987; Rezende & Viola, 2018), with proportional–integral (Shao et al., 2020) or PID (Stooke et al., 2020) variants that suppress oscillations and converge smoothly to the target.

Target sparsity has also been pursued through adaptive sparsity losses (Retsinas et al., 2021) and constrained optimization (Gallego-Posada et al., 2022). In sparse adversarial attacks, per-image methods such as PGD_0 enforce the ℓ_0 budget directly by projection, and Zhu et al. (2021) replace a fixed sparsity weight with a decreasing sequence of weights. For generator-based attacks, whose density is a property of the learned mapping, λ is still set manually, as reviewed in Sec. 2.2; to our knowledge, none controls the density by feedback during training.

DFC brings such feedback to generator-based sparse attacks. As in proxy-Lagrangian methods (Cotter et al., 2019) and unlike loss-level controllers (Shao et al., 2020), the generator is trained on a differentiable surrogate while the coefficient responds to the observed hard-mask density, the quantity that is actually evaluated.

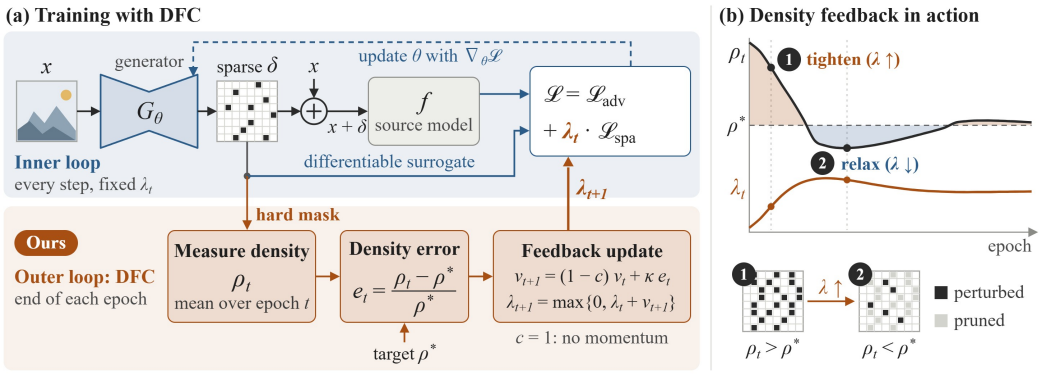

Figure 2: Overview of DFC. (a) The generator G_θ of the base attack is trained at every step on $\mathcal{L}_{\text{adv}} + \lambda_t \mathcal{L}_{\text{spa}}$, with λ_t fixed within epoch t. At the end of each epoch, DFC measures the mean hard-mask density ρ_t, computes its normalized error from the target ρ^*, and updates the sparsity coefficient. (b) Schematic training dynamics: the sparsity pressure tightens while $\rho_t > \rho^*$ and relaxes once ρ_t falls below ρ^*, so the mean density approaches the target. Masks illustrate one image; DFC regulates the mean density.

3 Method

3.1 Problem Formulation

Given a target mean pixel density ρ^*, we seek a generator that maximizes attack effectiveness while producing perturbations at that density:

$$\min_\theta \ \mathcal{L}_{\text{adv}}(\theta) \quad \text{s.t.} \quad \rho(\theta) = \rho^*, \qquad \rho(\theta) = \mathbb{E}_x \left[\frac{1}{HW} \sum_{i,j} \mathbb{1}\left[\|\delta_{:,i,j}(x)\|_0 > 0 \right] \right]. \tag{2}$$

Because $\rho(\theta)$ is computed from the hard mask and is non-differentiable, existing methods instead minimize the surrogate $\mathcal{L}_{\text{adv}} + \lambda \mathcal{L}_{\text{spa}}$ with a fixed λ. The map $\lambda \mapsto \rho$ is unknown, attack-specific, and non-stationary during training, so a fixed λ leaves ρ uncontrolled. We instead treat λ as a control input and adjust it from the observed density error, closing the loop between the differentiable surrogate and the density that is actually evaluated.

3.2 Density Feedback Control

Density measurement. DFC updates the sparsity coefficient once per epoch from the observed density. Let ρ_t denote the mean density during epoch t, i.e., the average fraction of active spatial locations in the hard masks of all training samples processed in that epoch, where a location counts as active if any color channel is perturbed. Since the masks are already computed in the forward pass, ρ_t is obtained at no additional cost. Given a target density $\rho^* > 0$, the normalized density error is

$$e_t = \frac{\rho_t - \rho^*}{\rho^*}. \tag{3}$$

Controller. The generator is trained with a fixed λ_t throughout epoch t. At the end of the epoch, DFC updates a velocity state v_t and then the coefficient:

$$v_{t+1} = (1-c)\, v_t + \kappa e_t, \qquad \lambda_{t+1} = \max\{0, \lambda_t + v_{t+1}\}, \tag{4}$$

where $\kappa > 0$ is the feedback gain and $c \in (0, 1]$ is the damping coefficient. A density above (below) the target pushes the coefficient up (down). While the clamp is inactive, eliminating v_t gives $\lambda_{t+1} = \lambda_t + \kappa e_t + (1-c)(\lambda_t - \lambda_{t-1})$: the coefficient follows the density error as in the basic multiplier update of Platt & Barr (1987), plus a momentum term that carries the previous adjustment forward (Polyak, 1964). Setting $c = 1$ removes the momentum and recovers the basic update. Unlike dual ascent on a single Lagrangian, the generator minimizes the differentiable surrogate L_{spa} while the coefficient responds to the hard-mask density, as in proxy-Lagrangian methods (Cotter et al., 2019). Because $\rho_t \in [0, 1]$, the error is bounded below by -1 but can be large above the target (up to 9 for $\rho^* = 10\%$); the error term therefore tightens the pressure quickly when the density is too high and relaxes it gradually when it is too low.

Gain and initialization. Since e_t is dimensionless, κ sets the scale of each adjustment. We match it to the order of magnitude of the initial coefficient,

$$\kappa = 10^{\lfloor \log_{10} \lambda_0 \rfloor}, \tag{5}$$

so that the adjustments are commensurate with the coefficient range of each base attack. This allows the same c to be shared across base attacks whose coefficients differ in scale, leaving c and λ_0 as the only hyperparameters of DFC. We initialize $v_0 = 0$. The controller runs outside the generator's computational graph (see Fig. 2 and Algorithm 1).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTING

Setups. We conducted our experiments on the widely used ImageNet dataset (Deng et al., 2009). Following Jiang et al. (2026), we used ResNet-50 as the source model and DenseNet-161, VGG-16, Inception-v3, Wide ResNet-50, EfficientNet-B6, ViT-Small, DeiT-Small, TNT-Small, Twins-Small, and Swin-Tiny as the target models (Huang et al., 2017; Simonyan & Zisserman, 2015; Szegedy et al., 2016; Zagoruyko & Komodakis, 2016; Tan & Le, 2019; Dosovitskiy et al., 2021; Touvron et al., 2021; Han et al., 2021; Chu et al., 2021; Liu et al., 2021). We set $\epsilon = 10/255$.

Benchmark Dataset. Generator-based sparse attacks, including DFC, were trained on the ImageNet training set and evaluated on the entire ImageNet validation set. All other methods were evaluated on a subset of 5,000 images randomly sampled from the ImageNet validation set using a fixed random seed of 0.

Evaluation Metrics and Comparative Methods. Sparsity is defined as the average pixel density of adversarial perturbations. To evaluate transferability, we use the attack success rate (ASR), measured on images correctly classified by each victim model. To assess the effectiveness of DFC, we primarily compare TSAA, EGS-TSSA, and GPG with their respective DFC-enhanced variants. For a more comprehensive evaluation, we additionally include PGD_0, sPGD, SparseFool, and GreedyFool as baselines. For these methods, we use the official implementations with default hyperparameter settings, except for those controlling perturbation density.

Generator details. We follow the architectures and loss functions of the official implementations and the training procedure of Jiang et al. (2026). All generator-based methods, including their DFC variants, are trained for 15 epochs, starting with a warm-up with $\lambda = 0$ of three epochs for the baselines and two for the DFC variants. The baselines then use fixed coefficients for reduction and stabilization (nine epochs) and fine-tuning (three epochs) toward a target density of 10%, the density at which Jiang et al. (2026) compare sparse attacks. For the DFC variants, DFC replaces these phases with feedback toward the same target, $\rho^* = 10\%$; for TSAA, we lower λ_0 because the default drove the density to zero after warm-up. Coefficients and DFC hyperparameters are given in App. E.

4.2 TRANSFERABILITY ON CNN MODELS

Table 1 reports the transfer ASR of sparse adversarial examples crafted on ResNet-50 and evaluated on five unseen CNN models. Conventional sparse attacks (PGD_0, sPGD, SparseFool, and GreedyFool) achieve near-perfect white-box ASR but at most 20.27% mean transfer ASR, indicating that their perturbations overfit to the source model. Transfer-oriented methods (GPG, TSAA, and EGS-TSSA) alleviate this issue (25.23–32.99%) at comparable sparsity. DFC substantially improves both GPG (32.99% $\rightarrow$ 42.09%) and TSAA (25.23% $\rightarrow$ 40.83%) while lowering their mean density by 0.87 pp, so the gains do not come from perturbing more pixels. The gains are largest on Dense161 and WideRes50, whereas DFC lowers the ASR on IncV3 and VGG16 (by up to 4.92 pp). In contrast, DFC barely helps EGS-TSSA (29.24% $\rightarrow$ 29.54%), which extends TSAA with a GFI mask for transferability; as a result, the plain TSAA-DFC outperforms EGS-TSSA with or without DFC. This suggests that the GFI mask and DFC are not complementary, which we analyze in Sec. 4.5.

Table 1: Transfer ASR (%) on CNN models and mean density (%). * denotes source-model results. Mean ASR is averaged over the five CNN target models, excluding the source model. Bold indicates the higher ASR within each baseline–DFC pair.

Method	Density	ResNet-50	IncV3	VGG16	Dense161	WideRes50	EffNetB6	Mean ASR
PGD_0 (100×5)	10.00	100.00*	10.09	20.94	23.54	29.27	10.09	18.78
$sPGD_{proj}$ ($T = 500$)	9.99	99.95*	6.93	16.25	18.35	22.59	8.16	14.46
$sPGD_{unproj}$ ($T = 500$)	9.99	99.95*	9.93	21.98	26.08	32.99	10.39	20.27
SparseFool ($\lambda = 20$)	12.90	100.00*	9.04	15.83	15.53	18.24	9.74	13.67
GreedyFool ($\kappa = 30$)	12.72	100.00*	9.01	14.95	19.83	24.40	9.80	15.60
GPG	11.69	80.67*	**18.44**	**42.49**	35.80	54.73	13.50	32.99
GPG-DFC ($c = 0.5$)	10.82	**83.12***	13.52	37.89	**62.90**	**73.75**	**22.41**	**42.09** (+27.58%)
TSAA	10.86	75.41*	**13.47**	**39.45**	28.71	32.30	12.22	25.23
TSAA-DFC ($c = 0.5$)	9.99	**84.63***	11.92	38.60	**59.13**	**75.33**	**19.15**	**40.83** (+61.81%)
EGS-TSSA	10.22	76.81*	**17.75**	44.63	32.16	35.36	**16.32**	29.24
EGS-TSSA-DFC ($c = 0.5$)	10.89	**87.54***	14.99	**45.13**	**34.53**	**39.41**	13.60	**29.54** (+1.00%)

4.3 Transferability on Transformer Models

Table 2 reports the transfer ASR from ResNet-50 to five Transformer models. Transferring across architecture families is considerably more challenging: conventional sparse attacks reach at most 6.93% mean ASR, and transfer-oriented methods at most 9.08%. In this setting, DFC yields even larger relative gains than on CNNs, raising the mean ASR of GPG from 9.08% to 13.45% (+48.01%) and that of TSAA from 7.48% to 12.65% (+69.25%). The gains are largest on ViT-S, DeiT-S, and TNT-S; on TNT-S, DFC more than doubles the ASR of both attacks. In contrast, DFC lowers the ASR on Swin-T for both attacks and on Twins-S for GPG (by up to 1.94 pp). For EGS-TSSA, DFC slightly reduces the mean ASR (Sec. 4.5). **Overall, DFC alone narrows the gap between TSAA and GPG, proposed 4 years later, from 7.76 to 1.26 pp on CNN models and from 1.60 to 0.80 pp on Transformer models.**

Table 2: Transfer ASR (%) on Transformer models and mean density (%). * denotes source-model results. Mean ASR is averaged over the five Transformer target models, excluding the source model. Bold indicates the higher ASR within each baseline–DFC pair.

Method	Density	ResNet-50	ViT-S	DeiT-S	TNT-S	Swin-T	Twins-S	Mean ASR
PGD_0 (100×5)	10.00	100.00*	6.73	4.37	7.97	7.34	7.83	6.85
$sPGD_{proj}$ ($T = 500$)	9.99	99.95*	5.69	3.67	6.45	6.45	5.99	5.65
$sPGD_{unproj}$ ($T = 500$)	9.99	99.95*	6.49	4.69	7.76	7.57	8.13	6.93
SparseFool ($\lambda = 20$)	12.90	100.00*	5.77	2.72	5.54	4.07	4.82	4.58
GreedyFool ($\kappa = 30$)	12.72	100.00*	6.73	3.74	6.19	6.20	6.69	5.91
GPG	11.69	80.67*	8.78	5.60	10.49	**9.43**	**11.12**	9.08
GPG-DFC ($c = 0.5$)	10.82	**83.12***	**16.81**	**9.64**	**23.58**	7.49	9.70	**13.45** (+48.01%)
TSAA	10.86	75.41*	9.77	4.68	7.21	**8.06**	7.66	7.48
TSAA-DFC ($c = 0.5$)	9.99	**84.63***	**16.22**	**9.95**	**19.20**	7.48	**10.41**	**12.65** (+69.25%)
EGS-TSSA	10.22	76.81*	**10.07**	**4.52**	**10.64**	**9.54**	**9.58**	**8.87**
EGS-TSSA-DFC ($c = 0.5$)	10.89	**87.54***	9.52	4.52	10.63	8.45	9.23	8.47 (−4.54%)

4.4 Transferability under Per-Image ℓ_0 Budgets

Prior generator-based sparse attacks measure sparsity as the average density over the dataset, which allows more pixels on difficult images as long as the average stays within the budget. In contrast, query-based sparse attacks impose a strict ℓ_0 budget on each image, which better reflects practical threat models (Song et al., 2024). Following this protocol, Table 3 reports the transfer ASR averaged over the ten victim models, counting an attack as successful only if its perturbation density is within the budget; gradient-based attacks are run separately for each budget. Under this protocol, the fixed-coefficient generator baselines, which lead the gradient-based attacks in Tables 1 and 2, fall behind them: at $\ell_0 \leq 10\%$, GPG and TSAA achieve 7.95% and 6.41%, whereas PGD_0 and $sPGD_{unproj}$ reach 12.82% and 13.60%.

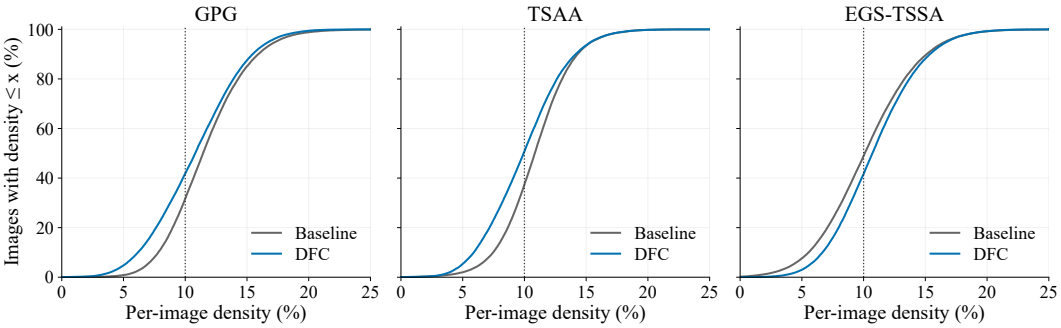

Figure 3: Per-image perturbation density over all 50,000 validation images, without success filtering. The dotted line marks the 10% budget.

Generator-based attacks, including their DFC variants, control density only on average (Fig. 3); for the baselines, over 60% of GPG and TSAA perturbations exceed the 10% budget and thus count as failures. DFC puts GPG and TSAA back in the lead: their DFC variants outperform all baselines under every budget, improving their in-budget ASR by 5.79–6.12 and 9.31–10.78 pp, respectively. Notably, TSAA-DFC under the 10% budget already surpasses every baseline under the 12% budget. In contrast, DFC does not improve EGS-TSSA.

Table 3: In-budget transfer ASR (%) under per-image ℓ_0 budgets.

Method	$\ell_0 \leq 10\%$	$\ell_0 \leq 11\%$	$\ell_0 \leq 12\%$
PGD_0	12.82	13.12	13.65
sPGD_{proj}	10.06	10.36	10.75
sPGD_{unproj}	13.60	14.25	14.81
GPG	7.95	10.81	13.51
GPG-DFC	13.74 (+72.68%)	16.82 (+55.61%)	19.63 (+45.30%)
TSAA	6.41	8.94	11.41
TSAA-DFC	15.72 (+145.31%)	19.25 (+115.34%)	22.19 (+94.55%)
EGS-TSSA	9.48	11.57	13.43
EGS-TSSA-DFC	8.96 (-5.54%)	11.16 (-3.49%)	13.30 (-0.97%)

Where does the per-image advantage of DFC come from? DFC can gain under a per-image budget in two ways: by placing more images within the budget, or by attacking the same images more successfully. We separate the two by decomposing the difference in in-budget ASR between DFC and each baseline into a density term, from images whose budget status changes, and a success term, from changes in attack success on the same images; images whose budget status changes also enter the success term with half weight (Table 4; App. C). For GPG and TSAA, both terms are positive, and the success term is the larger (3.19 vs. 2.60 pp for GPG and 5.52 vs. 3.79 pp for TSAA).

Table 4: Decomposition of the in-budget ASR gain of DFC at $\ell_0 \leq 10\%$ (pp), averaged over the ten victims. C_{in} (C_{out}) denotes the images whose perturbation is within the budget for DFC but not for the baseline (for the baseline but not for DFC). Success rates (%) of the baseline (B) and DFC (D) on these images are measured regardless of the budget.

Method	Total	Density	Success	C_{in}: B→D	C_{out}: B→D
GPG	+5.78	+2.60	+3.19	20.22→29.84	21.53→27.28
TSAA	+9.31	+3.79	+5.52	18.24→33.59	12.99→19.30
EGS-TSSA	−0.53	−0.51	−0.01	24.31→17.79	13.13→17.82

The density term follows the shift in density: DFC lowers the density of both attacks (Table 1, Fig. 3), and the term is positive on all ten victims. This term thus reflects where each run ends relative to the budget, as EGS-TSSA illustrates below, whereas the success term measures how effective the perturbations are on the same images. The success term favors DFC whether it perturbs fewer or more pixels than the baseline. On C_{in}, where only DFC is within the budget and thus perturbs fewer pixels, DFC succeeds more often (29.84% vs. 20.22% for GPG and 33.59% vs. 18.24% for TSAA); on C_{out}, where it perturbs more, it also succeeds more often (27.28% vs. 21.53% and 19.30% vs. 12.99%). On the images within the budget for both, which carry full weight in this term, DFC succeeds 35.52% vs. 26.64% for GPG and 31.99% vs. 18.76% for TSAA. Unlike the density term, the success term varies with the victim, consistent with the transfer ASR without the budget (Tables 1 and 2): for GPG, it is negative on exactly the four victims on which GPG-DFC has a lower transfer ASR than GPG (IncV3, VGG16, Swin-T, and Twins-S).

EGS-TSSA shows the opposite pattern. Its DFC variant is denser (10.89% vs. 10.22%), and the density term is negative on all ten victims. Its success also follows density: on all ten victims, DFC

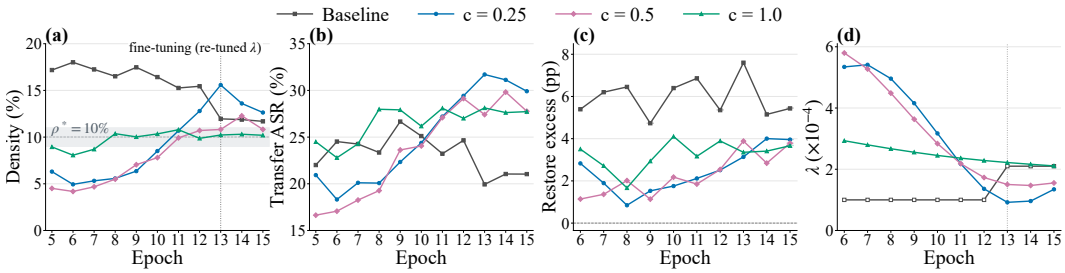

Figure 4: Training dynamics of GPG with a fixed coefficient (Baseline) and GPG-DFC with damping c; the dotted line in (a) and (d) marks where the baseline's coefficient is re-tuned for fine-tuning. (a) Mean density, with $\rho^* = 10\% \pm 1$ pp shaded. (b) Transfer ASR averaged over the ten victims. (c) Restore excess: the ASR gain from restoring the locations discarded since the previous epoch, minus that from adding as many random inactive locations (App. A). (d) Sparsity coefficient λ_t.

succeeds less often than the baseline on C_{in}, where it is sparser, and more often on C_{out}, where it is denser. The success term is nearly zero (-0.01 pp), so the density term accounts for almost all of the change in in-budget ASR (-0.51 of -0.53 pp).

4.5 DFC AND THE GFI RESTRICTION

EGS-TSSA restricts the perturbations of TSAA to a GFI support derived from the feature maps of the source model. Under a fixed coefficient, this restriction improves TSAA: the mean transfer ASR rises from 25.23% to 29.24% on CNN targets (Table 1), and the in-budget ASR at $\ell_0 \leq 10\%$ from 6.41% to 9.48% (Table 3). With DFC, it does not: EGS-TSSA-DFC remains close to EGS-TSSA (29.54% vs. 29.24% on CNNs and 8.47% vs. 8.87% on Transformers) and falls well below TSAA-DFC (40.83% and 12.65%).

What the restriction contributes. We decompose the change in in-budget ASR at $\ell_0 \leq 10\%$ as in Table 5. Under a fixed coefficient, the restriction raises the in-budget ASR of TSAA by 3.07 pp, of which 2.14 pp comes from the density term: EGS-TSSA places 48.93% of its perturbations within the budget, compared with 37.40% for TSAA. DFC alone yields a larger density term (3.79 pp), together with a success term of 5.52 pp. With DFC, the restriction instead lowers the in-budget ASR by 6.76 pp, of which 5.20 pp comes from the success term: it mainly lowers attack success on the same images.

What density feedback cannot change. The GFI support $S(x) \in \{0,1\}^{H \times W}$ is computed from the clean image and the fixed source model, and the restriction requires $(1-S) \odot \delta = 0$ regardless of the sparsity coefficient. DFC can therefore change how many pixels the generator perturbs, but not which locations it may use. Under a density target, the restriction thus acts in two ways: it shifts the density, which DFC achieves on its own, and it restricts the support, which DFC cannot relax. With DFC, the first effect is redundant, while the second remains.

Table 5: Decomposition of the in-budget ASR change at $\ell_0 \leq 10\%$, averaged over the ten victims. Each row adds the component to the base; ρ is the mean density before $\rightarrow$ after (Table 1).

		ASR change (pp)		
Base	ρ (%)	Total	Dens.	Succ.
+ GFI mask				
TSAA	10.86→10.22	+3.07	+2.14	+0.94
TSAA-DFC	9.99→10.89	−6.76	−1.56	−5.20
+ DFC				
TSAA	10.86→ 9.99	+9.31	+3.79	+5.52

What the restriction removes. Evaluating the generator's restore perturbation (Sec. 4.6) before the GFI mask (App. B), we find that DFC makes the proposal of EGS-TSSA more effective only outside the GFI support; within the support, it is no more effective than under a fixed coefficient. The restriction thus removes the gain that DFC brings. This is consistent with Tab. 5, where the restriction lowers the in-budget ASR of TSAA-DFC mainly through the success term.

4.6 Training Dynamics

Fig. 4 traces GPG with a fixed coefficient and GPG-DFC with damping $c \in \{0.25, 0.5, 1\}$ after warm-up. Transfer ASR is averaged over the ten victims, and the final-epoch values of the baseline and of $c = 0.5$ correspond to Tab. 1 and 2.

Density. The baseline remains at 15–18% during stabilization and approaches the target only after its coefficient is re-tuned for fine-tuning at epoch 13 (Fig. 4a). With $c = 1$, DFC stays within one percentage point of the target from epoch 8 onward. Smaller damping carries more momentum: the density first undershoots to 4–5% and then returns. With $c = 0.5$, it reaches the target band at epoch 11, briefly exceeds it at epoch 14, and ends at 10.8%; with $c = 0.25$, it overshoots to 15.6% and ends at 12.6%, outside the band. The damping coefficient thus shapes the transient, and both $c = 0.5$ and $c = 1$ end within one percentage point of the target with similar final ASR (27.76% and 27.72%).

Transferability recovers as the coefficient relaxes. The first updates of DFC tighten the coefficient enough to drive the density below the target. Once λ_t relaxes (Fig. 4d), the density recovers toward the target, and the ASR recovers with it (Fig. 4a,b): from epoch 6 until the density enters the target band, it rises from 17.07% to 27.08% with $c = 0.5$ and from 22.79% to 27.98% with $c = 1$. The recovered ASR overtakes the baseline while DFC still perturbs fewer pixels: $c = 1$ matches it at epoch 7 and exceeds it from epoch 8, and $c \leq 0.5$ from epoch 11, when $c = 0.5$ reaches 27.08% at a density of 9.91% against 23.22% at 15.26% for the baseline. When they overtake it, all three runs already exceed the highest ASR the baseline reaches in Fig. 4b (26.65%), and they stay above it at every later epoch except epoch 10 of $c = 1$. As the pressure relaxes, the ASR of DFC with $c = 0.5$ reaches the level of restoring what it discarded in the previous epoch, which the baseline never reaches (Fig. 5), suggesting that DFC ends above the baseline by offsetting what it discards. The baseline instead approaches the target by raising its coefficient, which lowers its ASR from 24.6% to 19.9% at fine-tuning.

What each epoch discards. The generators replace much of their support every epoch: 31–44% of the active locations under a fixed coefficient and 35–66% with DFC (App. A). To measure what they discard, we evaluate the perturbation values P_t with the mask $M_t \vee M_{t-1}$, which restores the locations discarded since epoch $t - 1$, and with $M_t \vee Q_t$, which instead activates as many random locations inactive in both epochs. The difference in ASR, the restore excess, thus reflects where the discarded locations lie rather than how many there are (Fig. 4c). The baseline has a larger excess than all three DFC runs at every epoch, even in epochs 6–9, when the density of DFC recovers and it replaces up to 66% of its support per epoch. The baseline's excess averages 5.9 pp during stabilization, whereas that of DFC ($c \geq 0.5$) near the target (epochs 13–15) averages 3.5 pp despite replacing more. Under a fixed coefficient, the generator thus discards more effective pixels. Since random additions lower the ASR of every DFC run (App. A), its rise during density recovery likely comes from where the generator places the added locations rather than from their number.

5 Conclusion

We proposed Density Feedback Control (DFC), which sets the sparsity coefficient of generator-based sparse attacks by feedback from the hard-mask density instead of fixing it in advance. In a single training run, DFC brings the mean density of three base attacks to within one percentage point of the target, and for GPG and TSAA it raises the mean transfer ASR on both CNN and Transformer targets at a lower density, including under per-image ℓ_0 budgets; with DFC, the earlier TSAA surpasses the fixed-coefficient GPG. The training dynamics suggest why: under a fixed coefficient, the density stays nearly constant after density reduction and the ASR recovers little, whereas DFC relaxes the coefficient once the density falls below the target, and the ASR recovers with the density, overtaking the baseline with fewer pixels. For EGS-TSSA, the feature-importance restriction helps under a fixed coefficient mainly by placing more images within the budget, which DFC does on its own, while the restricted support removes the gain of DFC. Our evaluation covers a single target density, source model, and dataset, the training dynamics come from one seed of GPG, and λ_0 was lowered for TSAA. Since DFC controls only the mean density (Fig. 3), extending the feedback to per-image densities is left for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with and polish the writing, support literature retrieval and discovery, and help draft and revise parts of the manuscript. We also used generative AI tools to assist with code generation and the presentation of experimental results in figures and tables to improve readability and visual clarity. We did not use generative AI tools for any other applicable tasks requiring disclosure. We reviewed all AI-assisted work. In particular, we manually verified AI-assisted text and references, tested and checked generated code for correctness, and confirmed that all figures and tables accurately reflect the underlying experimental results. We take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work investigates adversarial attack methods for evaluating model robustness. Although such techniques may have dual-use implications, the purpose of this research is to better understand model vulnerabilities and to facilitate the development and evaluation of more robust and secure learning systems.

REPRODUCIBILITY STATEMENT

We provide the algorithmic details of our method in the main paper and report the datasets, model configurations, training settings, and evaluation protocols used in our experiments. Additional implementation details and hyperparameters required to reproduce our results are provided. We also provide the source code as part of the supplementary materials.

REFERENCES

Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *The Association for the Advancement of Artificial Intelligence*, 2018.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Advances in Neural Information Processing Systems*, 2021.

Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*, 2019.

Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *International Conference on Computer Vision*, 2019.

Francesco Croce, Maksym Andriushchenko, Naman D. Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: A versatile framework for query-efficient sparse black-box adversarial attacks. In *AAAI Conference on Artificial Intelligence*, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

Xiaoyi Dong, Dongdong Chen, Jianmin Bao, Chuan Qin, Lu Yuan, Weiming Zhang, Nenghai Yu, and Dong Chen. Greedyfool: Distortion-aware sparse adversarial attack. In *Advances in Neural Information Processing Systems*, 2020.

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.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Jose Gallego-Posada, Juan Ramirez, Akram Erraqabi, Yoshua Bengio, and Simon Lacoste-Julien. Controlled sparsity via constrained optimization or: How i learned to stop tuning penalties and love constraints. In *Advances in Neural Information Processing Systems*, 2022.

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *Advances in Neural Information Processing Systems*, 2021.

Ziwen He, Wei Wang, Jing Dong, and Tieniu Tan. Transferable sparse adversarial attack. *Conference on Computer Vision and Pattern Recognition*, 2022.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, 2017.

Malhar Jere, Briland Hitaj, Gabriela F. Ciocarlie, and Farinaz Koushanfar. Scratch that! an evolution-based adversarial attack against neural networks. *CoRR*, abs/1912.02316, 2019.

Chengze Jiang, Zhuangzhuang Wang, Minjing Dong, Jie Gui, Lu Dong, Yuan Yan Tang, and James Tin-Yau Kwok. Gradient perturbation guidance for boosting sparse adversarial attack transferability. *IEEE Trans. Circuits Syst. Video Technol.*, 36(5):6361–6373, 2026.

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 *International Conference on Computer Vision*, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Di Ming, Peng Ren, Yunlong Wang, and Xin Feng. Transferable structural sparse adversarial attack via exact group sparsity training. In *Conference on Computer Vision and Pattern Recognition*, 2024.

Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: A few pixels make a big difference. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Conference on Computer Vision and Pattern Recognition*, 2016.

Muzammal Naseer, Salman H. Khan, Muhammad Haris Khan, Fahad Shahbaz Khan, and Fatih Porikli. Cross-domain transferability of adversarial perturbations. In *Advances in Neural Information Processing Systems*, 2019.

John C. Platt and Alan H. Barr. Constrained differential optimization. In *Neural Information Processing Systems*, 1987.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Jary Pomponi, Daniele Dántoni, Alessandro Nicolosi, and Simone Scardapane. Rearranging pixels is a powerful black-box attack for RGB and infrared deep learning models. *IEEE Access*, 11: 11298–11306, 2023.

Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge J. Belongie. Generative adversarial perturbations. In *Conference on Computer Vision and Pattern Recognition*, 2018.

George Retsinas, Athena Elafrou, Georgios Goumas, and Petros Maragos. Online weight pruning via adaptive sparsity loss. In *International Conference on Image Processing*, 2021.

Danilo Jimenez Rezende and Fabio Viola. Taming VAEs. *CoRR*, abs/1810.00597, 2018.

Huajie Shao, Shuochao Yao, Dachun Sun, Aston Zhang, Shengzhong Liu, Dongxin Liu, Jun Wang, and Tarek F. Abdelzaher. ControlVAE: Controllable variational autoencoder. In *International Conference on Machine Learning*, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Dongsu Song, Daehwa Ko, and Jay Hoon Jung. Amnesia as a catalyst for enhancing black box pixel attacks in image classification and object detection. In *Advances in Neural Information Processing Systems*, 2024.

Dongsu Song, DaeYun Go, Boseung Seo, and Jay Hoon Jung. Segpar: Class-centric decision-based sparse attack for semantic segmentation. In *European Conference on Computer Vision*, 2026.

Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems*, 2018.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID lagrangian methods. In *International Conference on Machine Learning*, 2020.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.*, 23(5):828–841, 2019.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *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.

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.

Viet Quoc Vo, Ehsan Abbasnejad, and Damith Ranasinghe. Brusleattack: a query-efficient score-based black-box sparse adversarial attack. In *International Conference on Learning Representations*, 2024.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *International Joint Conference on Artificial Intelligence*, 2018.

Chenglin Yang, Adam Kortylewski, Cihang Xie, Yinzhi Cao, and Alan L. Yuille. Patchattack: A black-box texture-based attack with reinforcement learning. In *European Conference on Computer Vision*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

Xuyang Zhong and Chen Liu. Sparse-pgd: A unified framework for sparse adversarial perturbations generation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 48(3):3056–3069, 2026.

Mingkang Zhu, Tianlong Chen, and Zhangyang Wang. Sparse and imperceptible adversarial attack via a homotopy algorithm. In *International Conference on Machine Learning*, 2021.

A RESTORE EXPERIMENT

This appendix defines the restore experiment of Sec. 4.6 and reports its per-epoch results (Table 6).

Masks and replacement. For a run and epoch t, let $M_t(x) \in \{0,1\}^{H\times W}$ denote the hard spatial mask produced by the generator at the end of epoch t, where a location is active if any color channel is perturbed, and let $P_t(x)$ denote its perturbation values before masking, so that the current perturbation is $\delta_t = M_t \odot P_t$. The locations discarded and added between epochs $t-1$ and t are

$$R_t = M_{t-1} \wedge \neg M_t, \qquad A_t = M_t \wedge \neg M_{t-1}. \tag{6}$$

We report ρ_t, r_t, and a_t as the fractions of the HW spatial locations that are in M_t, R_t, and A_t, averaged over images. The replacement rate r_t/ρ_{t-1} is the fraction of the locations active at epoch $t-1$ that are inactive at epoch t, computed from these averages with $\rho_{t-1} = \rho_t + r_t - a_t$.

Restore and random conditions. Both conditions evaluate the current perturbation values on an enlarged mask:

$$x_{\text{restore}} = \text{clip}\big(x + P_t \odot (M_t \vee M_{t-1}),\, 0,\, 1\big), \tag{7}$$

$$x_{\text{random}} = \text{clip}\big(x + P_t \odot (M_t \vee Q_t),\, 0,\, 1\big), \tag{8}$$

where, for each image, Q_t is drawn uniformly without replacement from the locations inactive in both epochs, $\neg(M_t \vee M_{t-1})$, with $|Q_t| = |R_t|$. Both conditions therefore perturb $|M_t| + |R_t|$ locations on each image, and their mean density is $\rho_t + r_t$. We define

$$G_{\text{restore}} = \text{ASR}_{\text{restore}} - \text{ASR}_{\text{cur}}, \qquad G_{\text{random}} = \text{ASR}_{\text{random}} - \text{ASR}_{\text{cur}}, \tag{9}$$

and the restore excess as $G_{\text{restore}} - G_{\text{random}} = \text{ASR}_{\text{restore}} - \text{ASR}_{\text{random}}$.

Protocol. We use the checkpoints of adjacent epochs $t-1$ and t for $t = 6, \ldots, 15$. Epoch 5 is omitted because, for the baseline, its previous epoch belongs to density reduction; we use the same epochs for all runs. All conditions are evaluated on the 50,000 ImageNet validation images with $\epsilon = 10/255$; for each victim, ASR is measured on the images it classifies correctly, and the ten victims are weighted equally. Q_t is drawn with a single random seed shared by all runs, so we do not report variation across seeds. At epoch 11 of the baseline, one image has fewer candidate locations than $|R_t|$; we exclude it from all three conditions of that row (49,999 images). Densities are computed over all 50,000 images. The epoch-15 values of ASR_{cur} for the baseline and for $c = 0.5$ match Tables 1 and 2.

Results. The replacement rate ranges from 31 to 44% for the baseline and from 35 to 66% with DFC; the highest rates occur while the density of $c \le 0.5$ recovers from the undershoot (epochs 6–9). Random additions raise the ASR of the baseline by 0.90–2.18 pp and lower that of every DFC run by 0.25–0.89 pp. The excess of the baseline exceeds that of all three DFC runs at every epoch, by at least 1.15 pp (epoch 14, $c = 0.25$), and its smallest value (4.73 pp, epoch 9) is larger than the largest value of any DFC run (4.11 pp, $c = 1$ at epoch 10). Fig. 5 plots the current and restore ASR of each run.

B GFI RESTRICTION ON THE RESTORE PERTURBATION

This appendix details the evaluation in Sec. 4.5 and reports its results (Table 7).

Proposal and restriction. The generator of EGS-TSSA outputs a mask M_t and perturbation values P_t over the whole image, and the GFI support $S(x) \in \{0,1\}^{H\times W}$, computed from the clean image and the fixed source model, is applied afterwards, so the perturbation it uses is $S \odot M_t \odot P_t$. We call $M_t \odot P_t$, before S is applied, the proposal of the generator.

Conditions. Since the current ASR of GPG-DFC follows the restore ASR (Sec. 4.6), we ask how much of the restore ASR depends on locations outside the support. We evaluate the restore perturbation of Eq. 7, with M_t and M_{t-1} taken before S, without and with S:

$$x^S_{\text{restore}} = \text{clip}\big(x + S \odot P_t \odot (M_t \vee M_{t-1}),\, 0,\, 1\big). \tag{10}$$

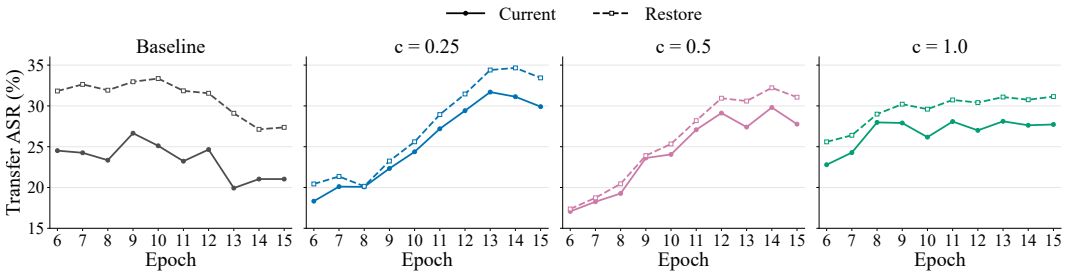

Figure 5: Current ASR (solid) and restore ASR (dashed) of GPG with a fixed coefficient (Baseline) and GPG-DFC with damping c, averaged over the ten victims. The restore ASR evaluates the current perturbation after restoring the locations discarded in the previous epoch (Eq. 7). With $c = 0.5$, the current ASR at epochs 7–12 reaches the restore ASR of the previous epoch, whereas the baseline never reaches its restore level.

The drop $\mathrm{ASR}_{\mathrm{restore}} - \mathrm{ASR}^{S}_{\mathrm{restore}}$ measures how much of the restore ASR depends on locations outside the support.

Protocol. We use the checkpoints of epochs 7–14, over which DFC relaxes the sparsity pressure and the transfer ASR of EGS-TSSA-DFC recovers, for both EGS-TSSA and EGS-TSSA-DFC. All other settings follow App. A: the 50,000 ImageNet validation images, $\epsilon = 10/255$, ASR measured on the images each victim classifies correctly, and the ten victims weighted equally. We average the ASR over these epochs of a single training run.

Results. Without S, the proposal of EGS-TSSA-DFC is more effective than that of EGS-TSSA (25.36% vs. 22.44%); with S, it is not (16.74% vs. 17.19%). The mask thus lowers the restore ASR by 8.62 pp for EGS-TSSA-DFC, against 5.25 pp for EGS-TSSA. The gain that DFC brings to the proposal lies outside the support, and the restriction removes it.

C Decomposition of the Per-Image Gain

For a victim model and a correctly classified image i, let $b_i^M = \mathbb{1}[\rho_i^M \le \tau]$ indicate whether the perturbation of method M is within the ℓ_0 budget τ, and let s_i^M indicate whether it fools the victim regardless of the budget. The in-budget ASR is $\mathrm{ASR}^M = \frac{1}{N}\sum_i b_i^M s_i^M$. For DFC ($D$) and a baseline ($B$),

$$\mathrm{ASR}^D - \mathrm{ASR}^B = \underbrace{\frac{1}{N}\sum_i \left(b_i^D - b_i^B\right)\frac{s_i^D + s_i^B}{2}}_{\text{density term}} + \underbrace{\frac{1}{N}\sum_i \left(s_i^D - s_i^B\right)\frac{b_i^D + b_i^B}{2}}_{\text{success term}}. \tag{11}$$

The density term is nonzero only for images whose budget status differs between the two methods, and the success term only for images on which attack success differs. The decomposition is exact and symmetric in the two methods. We compute it for each victim model and report the average over the ten victims.

Table 6: Per-epoch restore results for GPG with a fixed coefficient (Baseline) and GPG-DFC with damping c. Densities, replacement rates, and ASR are in %; gains and excess are in pp. Both the restore and random conditions have a mean density of $\rho_t + r_t$. ASR is averaged over the ten victims.

Run	t	ρ_t	r_t	a_t	r_t/ρ_{t-1}	$\mathrm{ASR}_{\mathrm{cur}}$	G_{restore}	G_{random}	Excess
Baseline	6	18.01	5.89	6.72	34.29	24.52	7.31	1.91	5.39
	7	17.25	6.67	5.91	37.06	24.25	8.38	2.18	6.20
	8	16.51	6.74	6.00	39.09	23.35	8.58	2.12	6.45
	9	17.47	5.11	6.07	30.96	26.65	6.31	1.57	4.73
	10	16.43	6.41	5.36	36.67	25.11	8.24	1.84	6.40
	11	15.26	6.10	4.94	37.14	23.22	8.63	1.77	6.86
	12	15.44	5.10	5.28	33.43	24.65	6.90	1.55	5.35
	13	11.95	6.74	3.25	43.63	19.93	9.17	1.57	7.60
	14	11.87	4.32	4.24	36.15	21.04	6.09	0.94	5.15
	15	11.69	4.37	4.18	36.77	21.03	6.33	0.90	5.43
$c = 0.25$	6	4.94	4.07	2.70	64.48	18.33	2.11	−0.72	2.83
	7	5.32	2.68	3.06	54.27	20.11	1.24	−0.65	1.89
	8	5.58	3.52	3.78	66.14	20.08	0.07	−0.77	0.84
	9	6.36	3.13	3.91	56.21	22.33	0.91	−0.62	1.53
	10	8.51	2.81	4.96	44.25	24.38	1.23	−0.53	1.76
	11	10.69	3.31	5.50	38.95	27.19	1.74	−0.38	2.12
	12	12.79	4.08	6.18	38.13	29.41	2.05	−0.46	2.51
	13	15.58	4.44	7.23	34.73	31.70	2.70	−0.44	3.14
	14	13.60	6.62	4.65	42.53	31.12	3.53	−0.47	4.00
	15	12.63	5.70	4.73	41.92	29.91	3.53	−0.43	3.96
$c = 0.5$	6	4.18	2.66	2.33	58.92	17.07	0.30	−0.84	1.14
	7	4.68	2.32	2.81	55.39	18.25	0.48	−0.88	1.36
	8	5.50	2.39	3.21	51.14	19.27	1.18	−0.83	2.02
	9	7.05	2.90	4.45	52.71	23.61	0.30	−0.84	1.14
	10	7.80	3.46	4.21	49.11	24.05	1.28	−0.89	2.18
	11	9.91	3.52	5.64	45.19	27.08	1.11	−0.74	1.85
	12	10.70	4.21	5.00	42.45	29.12	1.82	−0.72	2.54
	13	10.82	4.90	5.01	45.75	27.40	3.19	−0.70	3.88
	14	12.27	3.77	5.22	34.83	29.81	2.41	−0.43	2.84
	15	10.82	6.09	4.64	49.59	27.76	3.30	−0.49	3.79
$c = 1$	6	8.06	4.69	3.80	52.41	22.79	2.80	−0.70	3.50
	7	8.70	3.48	4.11	43.12	24.28	2.12	−0.59	2.72
	8	10.37	3.81	5.47	43.76	27.98	1.01	−0.65	1.67
	9	10.03	4.53	4.19	43.66	27.91	2.31	−0.63	2.94
	10	10.34	4.53	4.84	45.17	26.17	3.43	−0.67	4.11
	11	10.79	4.68	5.13	45.24	28.08	2.65	−0.52	3.16
	12	9.87	5.11	4.19	47.37	27.00	3.40	−0.49	3.89
	13	10.22	3.94	4.30	39.92	28.11	2.98	−0.38	3.36
	14	10.31	4.05	4.13	39.60	27.63	3.13	−0.28	3.41
	15	10.19	4.32	4.21	41.95	27.72	3.42	−0.25	3.67

Table 7: Restore ASR (%) of EGS-TSSA and EGS-TSSA-DFC without and with the GFI support S, averaged over the ten victims and epochs 7–14. Without S, the restore perturbation is evaluated on the proposal before the GFI mask.

Method	Without S	With S	Drop (pp)
EGS-TSSA	22.44	17.19	5.25
EGS-TSSA-DFC	25.36	16.74	8.62

D ALGORITHM

Algorithm 1 summarizes DFC. The controller runs outside the generator's computational graph and adds no trainable parameters.

Algorithm 1 Density Feedback Control (DFC)

Require: target density ρ^*, initial coefficient λ_0, damping c, warm-up epochs T_w, total epochs T

$\kappa \leftarrow 10^{\lfloor \log_{10} \lambda_0 \rfloor}$; $v \leftarrow 0$; $\lambda \leftarrow \lambda_0$

Train G_θ for T_w epochs with $\lambda = 0$ ▷ warm-up

for $t = T_w + 1, \ldots, T$ **do**

  Train G_θ for one epoch on $\mathcal{L}_{\mathrm{adv}} + \lambda \mathcal{L}_{\mathrm{spa}}$, recording hard-mask densities

  $\rho_t \leftarrow$ mean density over the epoch; $e_t \leftarrow (\rho_t - \rho^*)/\rho^*$

  $v \leftarrow (1 - c)\, v + \kappa e_t$; $\lambda \leftarrow \max(0,\ \lambda + v)$ ▷ keep $\lambda \geq 0$

end for

E Training details of generator-based attacks

All generator-based methods are trained for 15 epochs and start with a warm-up with $\lambda = 0$. For the baselines, a three-epoch warm-up is followed by reduction and stabilization phases spanning nine epochs and three epochs of fine-tuning toward a target density of 10%, each with a fixed λ (Table 8). For the DFC variants, we shorten the warm-up to two epochs so that the density can settle at the target within the same 15-epoch budget, and DFC replaces the remaining phases. We initialize $v_0 = 0$ and update the sparsity coefficient at the end of each epoch toward $\rho^* = 10\%$. For GPG and EGS-TSSA, we set λ_0 to 10^{-4}, the default sparsity coefficient of GPG. For TSAA, this setting drove the density to zero after warm-up, discarding what the generator had learned during warm-up; we therefore lower λ_0 to 5×10^{-5}.

Table 8: Sparsity coefficients of the generator-based attacks.

	Baseline λ		DFC	
Method	Red./Stab.	Fine-tune	λ_0	κ
GPG	1×10^{-4}	2.1×10^{-4}	10^{-4}	10^{-4}
TSAA	1.5×10^{-4}	2×10^{-4}	5×10^{-5}	10^{-5}
EGS-TSSA	8×10^{-5}	2×10^{-5}	10^{-4}	10^{-4}

F Limitations

Strict per-image budgets. Because DFC controls only the mean density, the density still varies across images, and perturbations whose density exceeds a per-image ℓ_0 budget do not satisfy the constraint. DFC improves transferability under per-image budgets over the fixed-coefficient baselines (Table 3), but under strict budgets such as $\ell_0 \leq 10\%$, its attack success rate remains below that obtained without the per-image constraint at the same mean density. Closing this gap, for example by feeding back an upper quantile of the per-image density distribution instead of the mean, is left for future work.

Restricted perturbation regions. DFC improves transferability by letting the generator re-select pixels when the sparsity pressure is relaxed. When the base attack restricts perturbations to a predefined region, as EGS-TSSA does with feature-importance regions of the source model, this reselection is confined to the region, and DFC yields no consistent gain (Tables 1 and 2). As analyzed in Section 4.5, a region that benefits fixed-coefficient training becomes a constraint once pixel selection is driven by density feedback. Combining density feedback with such region restrictions is left for future work.