A Appendix

A.1 More details of experimental settings

A.1.1 Implementation details of our method

Training Algorithm. The DRIFT filters are trained jointly with a frozen base model M using a composite loss. Each batch update includes: (i) preserving clean accuracy via cross-entropy; (ii) enforcing gradient diversity through Jacobian separation; (iii) enforcing divergence in logit-space VJPs, including against the identity path; and (iv) adversarial training with base-PGD adversaries. Warmup schedules ensure stability by gradually activating each component. The procedure is summarized in Algorithm 1.

Algorithm 1: Training DRIFT Filters

Input: Frozen base model M; trainable filters $\mathcal{F} = \{f_i\}_{i=1}^{K}$; dataloader $\mathcal{D}$; weights $\alpha, \beta_{\text{JS}}, \beta_{\text{LVJP}}, \gamma$; warmups $w_{\text{js}}, w_{\text{lvjp}}, w_{\text{adv}}$; PGD budget ϵ, steps T, step size η

Output: Trained filters $\mathcal{F}$

Define identity path $f_{\text{id}}(x) = x$;
for *epoch* $= 1, \ldots, E$ **do**
  for *batch* $(x, y) \sim \mathcal{D}$ **do**
    `// (i) Clean CE across filters`
    $\mathcal{L}_{\text{CE}} \leftarrow \frac{1}{K} \sum_i \text{CE}(M(f_i(x)), y)$;
    $\mathcal{L} \leftarrow \alpha \cdot \mathcal{L}_{\text{CE}}$;
    `// (ii) Jacobian-space separation (after warmup)`
    if *epoch* $> w_{\text{js}}$ *and* $K \geq 2$ **then**
      $\mathcal{L}_{\text{JS}} \leftarrow \text{avg}_{i<j}\, \mathbb{E}_v[\cos^2(J_{f_i}(x)^\top v,\ J_{f_j}(x)^\top v)]$;
      $\mathcal{L} \leftarrow \mathcal{L} + \beta_{\text{JS}} \cdot \mathcal{L}_{\text{JS}}$;
    `// (iii) Logit-VJP separation (after warmup)`
    if *epoch* $> w_{\text{lvjp}}$ **then**
      $\mathcal{L}_{\text{LVJP}} \leftarrow$ pairwise terms + vs-identity terms with random logits w;
      $\mathcal{L} \leftarrow \mathcal{L} + \beta_{\text{LVJP}} \cdot \mathcal{L}_{\text{LVJP}}$;
    `// (iv) Base-PGD adversary (after warmup)`
    if *epoch* $> w_{\text{adv}}$ **then**
      $\delta_M \leftarrow \text{PGD}(x, y; M, \epsilon, \eta, T)$;
      $\mathcal{L}_{\text{adv}} \leftarrow \max_i \text{CE}(M(f_i(x + \delta_M)), y)$;
      $\mathcal{L} \leftarrow \mathcal{L} + \gamma \cdot \mathcal{L}_{\text{adv}}$;
    `// (v) Update`
    Backpropagate $\mathcal{L}$; sanitize NaNs/Infs; clip grads; optimizer step;

A.1.2 Implementation Details of Adversarial Attacks

PGD: We implement ℓ_∞ and ℓ_2 variants of Projected Gradient Descent (PGD) Madry et al. (2018). For ℓ_∞ PGD we set $\epsilon = 4/255$, 40 steps with step size $\epsilon/10$. For ℓ_2 PGD we set $\epsilon = 1$, 40 steps with step size $\epsilon/10$.
MIM: The Momentum Iterative Method (MI-FGSM) Dong et al. (2018) introduces a momentum term to stabilize the gradient direction. We use $\epsilon = 4/255$, 40 steps, and momentum decay $= 1.0$.
SGM: The Skip Gradient Method (SGM) Wu et al. (2020) applies gradient smoothing to mitigate gradient shattering in residual connections. We set $\epsilon = 4/255$, 40 steps, momentum decay $= 1.0$, and $\lambda = 0.01$.
VMI: The Variance-Minimized Iterative Method (VMI-FGSM) Wang & He (2021) reduces gradient variance across steps. We set $\epsilon = 4/255$, 40 steps, momentum decay $= 1.0$, and neighborhood samples $N = 5$.
AA: AutoAttack (AA) Croce & Hein (2020b) is an ensemble of parameter-free attacks designed for reliable robustness evaluation. We run the standard configuration with $\epsilon = 4/255$, 40 steps.
FAB: The Fast Adaptive Boundary (FAB) Attack Croce & Hein (2020a) minimizes perturbation

norm while crossing the decision boundary. We set $\epsilon = 4/255$ and allow up to 40 steps.
Square: The Square Attack Andriushchenko et al. (2020) is a gradient-free black-box attack using random square-shaped updates. We set $\epsilon = 4/255$ and a query budget of 5000.
BPDA+EOT: Following Athalye et al. (2018), we evaluate adaptive white-box attacks using *Backward Pass Differentiable Approximation* (BPDA) combined with *Expectation over Transformation* (EOT). BPDA approximates the backward pass of non-differentiable components with the identity mapping, while EOT averages gradients across multiple stochastic forward passes. We use 40 PGD steps, $\epsilon = 4/255$, and 5 EOT samples per step. This setting provides the attacker full access to the defense while accounting for stochasticity.

A.1.3 Implementation Details of Adversarial Defenses

JPEG Compression. Following Dziugaite et al. (2016), we evaluate JPEG-based defenses with compression quality factors of 75% and 50%.
BaRT. For BaRT Raff et al. (2019), we apply $k = 5$ and $k = 10$ randomized transformations sequentially at inference time.
DiffPure. We use DiffPure Nie et al. (2022) with the authors' recommended hyperparameters for ImageNet-scale evaluation.
Adversarial Training (AT). Standard adversarial training (AT) Madry et al. (2018) is included as a baseline for robustness.
ANF. For ANF Suresh et al. (2025), we configure the first layer with a convolutional kernel size of 15×15, 256 filters, and a max-pooling kernel size of 5×5.
FFR. We evaluate Filter Frequency Regularization (FFR) Lukasik et al. (2023) as a frequency-based regularization defense. Since ANF and FFR alone do not yield sufficient robustness on ImageNet, we follow prior work and combine them with adversarial training using a 1-step PGD attack at $\epsilon = 4/255$.

A.2 More experimental results

Cross-Filter Transferability. To assess the transferability of adversarial examples across different trained filters, we employ Projected Gradient Descent (PGD) with $\epsilon = 4/255$ and 40 attack steps. Table 7 shows the robust accuracy (RA) when adversarial samples generated using one filter (source/attacker) are evaluated on another filter (target/victim). The results demonstrate that robust accuracy remains high across all off-diagonal entries, indicating that adversarial perturbations crafted on one filter do not easily transfer to others. This strongly suggests that our Jacobian- and logit-space separation objectives effectively induce gradient divergence among filters, thereby reducing the consensus subspace exploited by transferable attacks. In other words, although each filter alone is capable of mitigating adversarial perturbations, the ensemble diversity ensures that perturbations optimized against one filter fail to generalize to others. This property is critical for our defense strategy, as it directly targets the root cause of adversarial transferability: gradient alignment. The consistently high RA values across ViT-16, ResNet-v2, DeiT-S, and Inception-v3 models confirm the effectiveness of our design in breaking gradient consensus across diverse architectures. Additional analysis and ablation studies are provided in Appendix A.3.

Table 7: Cross-filter transferability reported as robust accuracy (RA %) under PGD ($\epsilon = 4/255$, 40 steps). Rows: source/attacker filter; Columns: target/victim filter.

	ViT-16				**ResNet-v2**				**DeiT-S**				**Inception-v3**			
Src↓/Tgt→	f1	f2	f3	f4	f1	f2	f3	f4	f1	f2	f3	f4	f1	f2	f3	f4
f1	–	76.24	77.30	73.60	–	60.32	58.73	61.38	–	61.90	61.38	56.03	–	58.83	47.17	43.67
f2	73.60	–	70.95	71.00	57.67	–	59.26	59.79	58.20	–	66.67	44.92	64.17	–	62.67	42.50
f3	75.10	76.24	–	72.01	58.20	59.79	–	57.67	56.56	55.03	–	60.85	48.83	62.83	–	61.17
f4	73.07	76.24	74.66	–	61.38	62.43	65.61	–	53.92	43.92	67.20	–	45.67	46.50	60.17	–

A.3 More Analysis

A.3.1 Impact of Noise Budget

We assess non-adaptive attacks (the attacker is *not* aware of the defense) using PGD under both ℓ_∞ and ℓ_2 norms. As expected, increasing the perturbation budget degrades accuracy monotonically,

but the drop remains gradual across typical evaluation ranges (See Table 8). These results should be interpreted as an *upper bound* on robustness; see our adaptive evaluations for a stricter assessment.

Table 8: Non-adaptive PGD on ViT-B/16 under ℓ_∞ and ℓ_2 norms.

ℓ_∞		ℓ_2	
ϵ	**RA (%)**	ϵ	**RA (%)**
$2/255$	76.13	0.5	76.71
$4/255$	74.66	1.0	74.07
$8/255$	68.25	1.5	73.01
$16/255$	62.96	2.0	70.89
$20/255$	58.62	3.0	69.30

Non-adaptive results provide useful signal about baseline robustness but can overestimate true security; adaptive, defense-aware attacks (e.g., BPDA+EOT) are reported elsewhere in this paper for completeness.

A.3.2 Impact of EOT

To quantify the effect of expectation over transformation (EOT) on a stochastic defense, we run PGD-ℓ_∞ with $K \in \{5, 10, 20\}$ EOT samples per gradient step (common randomness across paired evaluations; all other hyperparameters fixed). As K increases, the attack better estimates the gradient of the *expected* loss and therefore becomes stronger. Table 9 reports Top-1 robust accuracy (%) across ϵ.

(i) Robust accuracy decreases monotonically with K for every ϵ, confirming that EOT makes the attack more faithful to the defense's stochasticity.
(ii) The *marginal* gain from $K{=}10$ to $K{=}20$ is smaller than from $K{=}5$ to $K{=}10$ at low budgets (e.g., -1.06 points at $\epsilon{=}1/255$), indicating partial saturation; however, at larger budgets the $10 \to 20$ gain remains non-negligible (e.g., -6.34 at $\epsilon{=}8/255$). (iii) Overall drops from EOT-5 to EOT-20 grow with ϵ (from -5.82 at $1/255$ to about -19 points at $4/255$–$8/255$), showing that higher budgets benefit more from accurate gradient estimation.

A.4 Impact of Filter Architecture

The architectural design of the filters f_i crucially influences both their ability to diversify gradients and their efficiency in deployment.

Expressivity vs. Simplicity. Filters must be lightweight to avoid prohibitive overhead during inference. At the same time, they need sufficient expressivity to induce distinct gradient directions. For example, a simple residual block with two convolutions and a ReLU nonlinearity

$$f(x) = x + \mathrm{Conv}_2(\mathrm{ReLU}(\mathrm{Conv}_1(x)))$$

already introduces nontrivial nonlinear transformations, ensuring gradients are not trivially aligned. Deeper or wider filters could increase diversity further, but at the risk of overfitting or collapsing to similar functions without additional regularization.

Table 9: **PGD-EOT on ViT-B/16 (ImageNet).** Increasing EOT samples strengthens the attack, lowering robust accuracy. Δ columns show absolute point drops.

ϵ	**EOT-5**	**EOT-10**	**EOT-20**	**Δ 5→10**	**Δ 10→20**	**Δ 5→20**
$1/255$	81.96	77.20	76.14	-4.76	-1.06	-5.82
$2/255$	74.55	64.50	60.79	-10.05	-3.71	-13.76
$4/255$	57.62	44.39	38.57	-13.23	-5.82	-19.05
$8/255$	37.51	24.81	18.47	-12.70	-6.34	-19.04

Gradient Geometry. Architectures with strong local sensitivity (e.g., small convolutional kernels) tend to decorrelate gradients across filters more effectively. Conversely, architectures with large receptive fields or strong averaging effects may inadvertently align filters, weakening separation. Thus, the choice of kernel size, depth, and activation plays a direct role in the effectiveness of $\mathcal{L}_{JS}$ and $\mathcal{L}_{LVJP}$ in promoting divergence.

Training Stability. Overly complex filters can destabilize joint optimization: with many parameters, filters may collapse toward learning identity-like transformations, reducing their utility. Lightweight architectures constrain the search space, making separation losses more effective in driving gradient divergence.

Computation and Deployment. Inference cost scales linearly with the complexity of each filter. Since DRIFT samples only one filter per forward pass, lightweight architectures preserve the real-time feasibility of the defense. In contrast, heavier filters would diminish the main advantage of DRIFT—efficient deployment without modifying the base model.

Practical Guidance. We find that residual-style shallow convolutional filters strike the best balance: they preserve input dimensionality (ensuring model compatibility), introduce sufficient non-linearity for gradient diversification, and remain efficient at both training and inference. More elaborate architectures (e.g., multi-layer CNNs or attention blocks) may be explored, but lightweight residual filters already achieve strong robustness without sacrificing clean accuracy.

To evaluate how the choice of filter architecture influences DRIFT, we compare several lightweight designs while keeping the base model (ViT-B/16) and training setup fixed. Each filter preserves input dimensionality to maintain compatibility with the backbone.

Architectures tested.

- **SingleConv:** A single 3×3 convolution with ReLU.
- **ResBlock (ours):** Two 3×3 convolutions with ReLU and a residual skip connection.
- **DeepConv:** A stack of four 3×3 convolutions with ReLU activations.
- **MLPFilter:** A shallow two-layer MLP applied patchwise to image embeddings.

Table 10: Effect of filter architecture on clean and robust accuracy (%) with ViT-B/16 under adaptive PGD ($\epsilon = 4/255$, 40 steps).

Filter Architecture	Clean Accuracy	Robust Accuracy
SingleConv	77.3	42.5
DeepConv	75.8	47.9
MLPFilter	74.9	44.2
ResBlock (ours)	**80.5**	**56.7**

Results show that the residual block (our chosen design) achieves the best balance: it maintains clean accuracy close to the undefended baseline while significantly improving robust accuracy. SingleConv lacks sufficient expressivity and yields poor robustness. DeepConv provides moderate robustness gains but at the cost of lower clean accuracy and higher training instability. Patchwise MLPs decorrelate gradients somewhat but underperform compared to convolutional filters. Overall, lightweight residual filters offer the most effective and efficient choice for DRIFT.

A.5 Impact of the Number of Filters

The number of filters K plays a critical role in shaping the defense's effectiveness.

Gradient Diversity. Each filter induces a distinct gradient geometry. Increasing K expands the ensemble of gradient subspaces, which makes it harder for an adversary to find perturbations that transfer across all filters. In theory, if the pairwise consensus $\Gamma(f_i, f_j)$ is kept small, the expected transfer success of perturbations decreases roughly as $\mathcal{O}(1/K)$, since adversaries must overfit to one

filter at a time. Thus, larger K strengthens robustness by increasing gradient divergence opportunities.

Robustness–Accuracy Trade-off. While more filters can improve adversarial robustness, they may also introduce redundancy or training instability if K is too large. Empirically, small ensembles ($K = 2$–4) already capture strong divergence effects without hurting clean accuracy. Beyond a certain point, gains diminish: additional filters may converge to similar behaviors unless carefully regularized by LJS and $LLVJP$.

Training and Inference Cost. The cost of training grows quadratically with K for pairwise losses:

$$\text{Cost} \sim \binom{K}{2}(P_v + P_w).$$

Thus, increasing K both increases backprop passes (for each probe) and the number of pairwise comparisons. Inference, however, remains efficient: only one filter is sampled and applied per forward pass, so runtime cost grows only linearly with K when switching filters randomly. This makes DRIFT scalable in deployment even with moderate K.

Practical Guidance. We find $K = 3$–5 filters offers the best balance: sufficient gradient diversity to suppress transferability, while keeping training cost manageable. Larger ensembles can be explored if resources allow, but diminishing returns beyond $K = 6$ suggest focusing instead on improving separation losses.

A.6 Probe Count: Variance–Compute Trade-off

Estimators. DRIFT uses Hutchinson-style probing to estimate pairwise alignment. For two filters (f_i, f_j), let

$$\phi_v(x; f_i, f_j) = \cos^2\big(J_{f_i}(x)^\top v,\ J_{f_j}(x)^\top v\big),$$
$$\psi_w(x; f_i, f_j) = \cos^2\big(\nabla_x\langle M(f_i(x)), w\rangle,\ \nabla_x\langle M(f_j(x)), w\rangle\big),$$

with v a unit probe in the filter output space, and w a unit probe in the logit space. With P_v i.i.d. probes $\{v_p\}_{p=1}^{P_v}$ and P_w i.i.d. probes $\{w_q\}_{q=1}^{P_w}$, the Monte Carlo estimators used in $\mathcal{L}_{JS}$ and $\mathcal{L}_{LVJP}$ are

$$\widehat{\Gamma}_{\text{JS}}(x; f_i, f_j) = \frac{1}{P_v}\sum_{p=1}^{P_v}\phi_{v_p}(x; f_i, f_j), \qquad \widehat{\Gamma}_{\text{LVJP}}(x; f_i, f_j) = \frac{1}{P_w}\sum_{q=1}^{P_w}\psi_{w_q}(x; f_i, f_j).$$

Concentration. Each summand is bounded: $\phi_v \in [0, 1]$ and $\psi_w \in [0, 1]$. Hence, by Hoeffding's inequality, for any $\epsilon > 0$,

$$\Pr\Big(|\widehat{\Gamma}_{\text{JS}} - \mathbb{E}[\phi_v]| \geq \epsilon\Big) \leq 2\exp\big(-2P_v\epsilon^2\big), \qquad \Pr\Big(|\widehat{\Gamma}_{\text{LVJP}} - \mathbb{E}[\psi_w]| \geq \epsilon\Big) \leq 2\exp\big(-2P_w\epsilon^2\big).$$

Consequently, the mean-squared error (MSE) scales as $\mathcal{O}(1/P_v)$ and $\mathcal{O}(1/P_w)$, respectively. In practice, this means doubling the number of probes reduces the estimator's standard deviation by roughly $1/\sqrt{2}$.

Bias. The estimators are unbiased for the *probe-averaged* consensus, i.e., $\mathbb{E}[\widehat{\Gamma}_{\text{JS}}] = \mathbb{E}[\phi_v]$ and $\mathbb{E}[\widehat{\Gamma}_{\text{LVJP}}] = \mathbb{E}[\psi_w]$, where the expectation is over the probe distributions (Rademacher or Gaussian, normalized). Any residual bias relative to *full* Jacobian/gradient alignment comes from using random projections instead of exhaustively scanning all directions; this bias decreases as probe count grows.

Compute cost. Each probe requires a VJP/gradient evaluation. Thus the per-batch cost scales linearly in probes:

$$\text{Cost per batch} \approx \mathcal{O}\Big(P_v \cdot \tbinom{K}{2} + P_w \cdot \tbinom{K}{2} + P_w \cdot K\Big),$$

corresponding to JS pairs, LVJP pairs, and LVJP-vs-identity terms. Larger K or probe counts improve statistical stability but increase backprop passes.

Practical guidance. We find a simple schedule balances stability and throughput:

1. **Warmup (epochs** 1**–**w**):** P_v=2 and P_w=2 (fast, lets filters stabilize).
2. **Main training:** P_v=5 and P_w=5 (good variance–compute trade-off).
3. **High-fidelity refinement (last** 10**–**20% **epochs, optional):** P_v=8–10, P_w=8–10 if compute permits.

Empirically, $P_v, P_w \in [5, 10]$ yield stable separation signals and stronger cross-filter non-transferability, while larger values show diminishing returns relative to their linear compute cost.