ROBUSTNESS BEYOND PIXELS FOR VISION–LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt learning adapts frozen vision–language models using a small number of trainable tokens, but adversarial prompt-learning methods typically defend only against image-space perturbations. We study a complementary white-box setting where an adversary jointly perturbs the input image and selected residual-stream activations, as may arise in shared-encoder or untrusted-adapter pipelines. This compositional threat model is challenging because pixel and feature perturbations interact through a common forward pass, while activation scales vary across transformer depth. We propose **JPFAT** (Joint Pixel–Feature Adversarial Prompt Tuning). JPFAT uses alternating projected gradient ascent over a fixed product threat set, Relative Activation Budgets to calibrate feature radii to layerwise activation scale, and entropy-regularized effort allocation to prioritize attack paths without changing their feasible radii. A vulnerability-adaptive consistency loss emphasizes samples with larger output and internal representation drift. Using fixed-budget, non-adaptive pixel, feature, sequential, and joint attacks for evaluation, JPFAT improves accuracy under the joint pixel–feature adversary and its individual components while maintaining competitive clean base-to-new generalization.

1 INTRODUCTION

Vision–language models (VLMs) such as CLIP (Radford et al., 2021) transfer effectively across recognition tasks through natural-language supervision, yet their visual encoders remain vulnerable to adversarial perturbations (Szegedy et al., 2014; Goodfellow et al., 2015). Full adversarial fine-tuning improves image-space robustness but is costly and can weaken the zero-shot transfer that makes VLMs useful (Mao et al., 2023; Schlarmann et al., 2024). Prompt learning offers a parameter-efficient alternative: it freezes the encoders and adapts only a small set of continuous tokens, often inserted into internal transformer blocks (Zhou et al., 2022; Khattak et al., 2023). Adversarial prompt learning extends this by training prompts against perturbed images (Zhang et al., 2024; Zhou et al., 2024), but every such method defines the adversary in pixel space alone, even though the prompts themselves act *inside* the encoder.

Prompts act inside the encoder — what if perturbations do too?

This gap is practical, not just conceptual: in a shared-encoder, adapter-based, or plugin-based pipeline, an untrusted component may perturb intermediate activations directly, a capability pixel-only threat models cannot express. We study an adversary that jointly perturbs input pixels *and* residual-stream activations at selected transformer blocks. Feature perturbations are complementary, not a substitute for pixel attacks, since they need not be reachable by any valid image perturbation, so we always report pixel-only robustness separately from the joint one.

Crucially, this is not two independent attacks combined: a pixel perturbation changes representations throughout the encoder, so a later feature perturbation acts on an already-altered state rather than the clean one, and the two paths compose through a single forward pass and interact during optimization. We formalize this as a Cartesian product of one pixel ball and several feature balls,

Code will be publicly released upon acceptance.

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.

jointly constrained and jointly optimized, rather than the independent-threat-model union used in prior multi-perturbation work (Tramèr & Boneh, 2019; Maini et al., 2020) (Table 1).

Two obstacles make this threat model hard to use in practice. First, residual-stream magnitudes vary sharply across depth (Heimersheim & Turner, 2023; Csordás et al., 2025), so a single fixed feature radius ends up negligible at some layers and excessive at others. Second, a joint adversary has several coupled paths but only a finite per-step budget, and shrinking some paths' radii to favor others would silently redefine the threat model, making robustness numbers incomparable across methods.

Table 1: **Threat-model positioning.** JPFAT alone jointly optimizes pixel and feature perturbations through a common forward pass.

Setting	Pixel	Feature	Jointly optimized
Pixel-only prompt robustness	✓	✗	✗
Feature-only robustness	✗	✓	✗
Union-based multi-attack (Tramèr & Boneh, 2019; Maini et al., 2020)	✓	✓	✗
JPFAT (ours)	✓	✓	✓

JPFAT. We propose **JPFAT** (Joint Pixel–Feature Adversarial Prompt Tuning), a TRADES-style framework (Zhang et al., 2019) built from four components. **JA-PGD** alternates projected pixel and feature updates at each iteration, so the attack always responds to the other path's latest state instead of a stale gradient. **Relative Activation Budgets (RAB)** set each feature radius and step relative to a stop-gradient estimate of that layer's own activation scale, replacing hand-tuned absolute budgets with two transferable hyperparameters. **Gradient-effort allocation** then redirects attack effort toward whichever paths are currently most influential, while provably leaving every radius, and hence the threat model itself, unchanged. **VAAL** weights training examples by their output and internal drift under the finished attack, concentrating training pressure where the attack succeeds.

Novelty relative to prior work

Prior adversarial prompt learning computes one pixel gradient per step; **JA-PGD** alternates so each path attacks the other's latest state. Multi-perturbation defenses use a fixed radius per independent threat type (Tramèr & Boneh, 2019; Maini et al., 2020); **RAB** instead derives every radius from the layer's own activation scale. Prior budget-splitting shrinks some radii to favor others, changing the threat model; **allocation** reweights only step size, provably preserving every radius. TRADES-style training (Zhang et al., 2019) weights all examples equally; **VAAL** instead weights by the attack's own output and internal drift.

A method that generates its own training attacks should not also grade its own homework, so we strictly separate the adaptive training attack from the reporting adversary: RAB and allocation shape training perturbations only, and every method is evaluated at test time with the same fixed-budget, non-adaptive attack. We report clean, pixel-only, feature-only, sequential, and joint accuracy separately, and use AutoAttack (Croce & Hein, 2020) as an external check on pixel-space robustness.

In summary,

- We formulate a **compositional pixel–residual-stream threat model**: a white-box activation-access setting, absent from prior work, in which pixel and feature perturbations act jointly through a frozen encoder.
- We propose **JPFAT**: alternating pixel–feature PGD (JA-PGD, Algorithm 1), RAB-calibrated feature budgets, and entropy-regularized effort allocation that provably preserves the fixed feasible set (Appendix E).
- We introduce **VAAL**, a detached vulnerability-weighted consistency loss that concentrates training pressure where the joint attack succeeds.
- Under strict train/test attack separation and a common fixed-budget protocol (pixel-only, feature-only, sequential, joint, and AutoAttack), JPFAT improves joint robustness substantially over pixel-only baselines while remaining competitive on clean accuracy (Section 4).

A full discussion of **related work** is deferred to Appendix B.

2 ATTACKING INSIDE THE ENCODER: A JOINT THREAT MODEL

Most adversarial attacks only touch the input image. We consider a stronger adversary that can also reach inside the network and perturb its internal computations. Notation is summarized in Appendix A.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a C-class few-shot dataset, $x_i \in [0,1]^d$, $y_i \in \{1, \dots, C\}$. A pretrained VLM, such as CLIP (Radford et al., 2021), provides frozen visual and text encoders V and T. Following Khattak et al. (2023), prompt learning inserts continuous tokens into the first P blocks of both encoders; their concatenation θ is the only trainable parameter. Let $z_\theta(x)$ and $t_c(\theta)$ be prompted image and class-text representations. With $\bar{v} = v/\|v\|_2$, logit scale $s > 0$, and B the minibatch size,

$$f_{\theta,c}(x) = s\, \bar{z}_\theta(x)^\top \bar{t}_c(\theta), \qquad p_\theta(c \mid x) = \mathrm{softmax}_c\big(f_\theta(x)\big). \tag{1}$$

$$\mathcal{L}_{\mathrm{nat}} = -\frac{1}{B}\sum_{i=1}^{B} \log p_\theta(y_i \mid x_i) \tag{2}$$

is the ordinary cross-entropy loss on clean images — the starting point every term below perturbs or augments.

An image passes through encoder blocks, each producing a hidden state. Let $V = \mathcal{E}_{M-1} \circ \cdots \circ \mathcal{E}_0$, and let $h_i^{(l)} \in \mathbb{R}^{N_l \times D_l}$ be the clean hidden state after block $\mathcal{E}_l$ (N_l tokens, D_l channels). To let the adversary reach both pixels *and* internal computation, we inject perturbations at every stage of the forward pass, not only at the input:

$$\tilde{h}_i^{(l)} = \mathcal{E}_l\big(\tilde{h}_i^{(l-1)}\big) + \mathbf{1}\{l < L\}\delta_i^{(l)}, \qquad \tilde{h}_i^{(-1)} = \text{patch-embed}(x_i + \delta_{x,i}), \tag{3}$$

so $\delta_{x,i}$ perturbs the pixels and each $\delta_i^{(l)}$ perturbs the hidden state entering block $l+1$, up to attack depth $L \le M$. Write $\boldsymbol{\delta}_i = (\delta_{x,i}, \delta_i^{(0)}, \dots, \delta_i^{(L-1)})$ and $\mathcal{K} = \{x, 0, \dots, L-1\}$ indexing the $K = L+1$ attack paths — one for pixels, one per attacked layer. $p_\theta(\cdot \mid x_i, \boldsymbol{\delta}_i)$ denotes the prediction under equation 3. Feature perturbations are injected through forward hooks that edit a layer's output in transit without altering weights, so the encoder itself is never modified.

A perturbation this large is not adversarial — it is a different image or a broken activation — so each path is capped to stay "small." For image radius ε_x and feature radii $\{\varepsilon_l\}_{l=0}^{L-1}$,

$$\Delta_i = \Big\{\boldsymbol{\delta}_i : \|\delta_{x,i}\|_\infty \le \varepsilon_x,\ x_i + \delta_{x,i} \in [0,1]^d,\ \|\delta_i^{(l)}\|_\infty \le \varepsilon_l\ \forall l\Big\} = \mathcal{B}_x(\varepsilon_x) \times \prod_{l=0}^{L-1} \mathcal{B}_l(\varepsilon_l), \tag{4}$$

i.e. a pixel ball times one ball per attacked layer — a product, not a sum, since every path is available at once.

Unlike a union of independent threat models (Tramèr & Boneh, 2019; Maini et al., 2020), all paths here act simultaneously through a single forward pass: the image perturbation changes every later hidden state, and each feature perturbation acts on top of that already-changed state. Each path keeps its full radius throughout; Section 3.3 only reallocates *how hard* we push on each path, never how far it may go.

Feature perturbations are a complementary white-box capability and need not be induced by valid image perturbations, so we report pixel-only robustness separately from the joint one. Training uses RAB-calibrated radii $\varepsilon_l = \rho m_l$ (Section 3.2); evaluation uses fixed radii $\varepsilon_l = \varepsilon_f$ shared across methods (Section 3.5).

Training is a minimax problem: find θ that stays accurate under the worst perturbation in Δ_i. Following TRADES (Zhang et al., 2019), we optimize

$$\min_\theta \frac{1}{N}\sum_{i=1}^{N}\left[-\log p_\theta(y_i \mid x_i) + \lambda \max_{\boldsymbol{\delta}_i \in \Delta_i} D_{\mathrm{KL}}(p_\theta(\cdot \mid x_i) \,\|\, p_\theta(\cdot \mid x_i, \boldsymbol{\delta}_i))\right], \tag{5}$$

where $\lambda > 0$ trades accuracy against robustness: the first term is ordinary accuracy, the second penalizes large KL drift under a worst-case perturbation. For $L = 0$, equation 5 reduces to image-space TRADES. JPFAT approximates the inner $\max$ with an attack called JA-PGD, and reweights the outer loss using a vulnerability signal from that same attack.

3 JPFAT: COORDINATED PIXEL–FEATURE ADVERSARIAL PROMPT TUNING

The threat model above lets an adversary act on pixels and on selected internal hidden states at once (Figure 1). These levels are not independent: a changed image changes downstream hidden states, and a hidden-state perturbation then changes how the encoder processes that already-changed image. JPFAT handles this with four pieces: **JA-PGD** (Section 3.1) builds one coordinated adversarial example so pixel and feature perturbations respond to each other; **Relative Activation Budgets** (RAB, Section 3.2) sets each layer's feature budget relative to that layer's typical activation scale; **gradient-effort allocation** (Section 3.3) spends limited attack updates on whichever paths currently look most effective, without changing what any path is allowed to do; and **VAAL** (Section 3.4) up-weights training examples the finished attack disrupted most. Only the prompt parameters θ are ever updated; both VLM encoders stay frozen.

Training and evaluation share the threat model Δ_i but use it differently: training uses RAB and adaptive allocation to build the strongest adversarial examples we can afford, while evaluation uses a fixed, non-adaptive attack with identical budgets across methods, so reported robustness is not an artifact of method-specific statistics or schedules (Section 3.5).

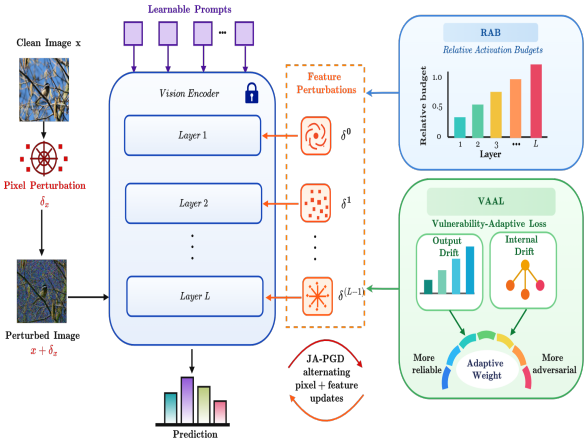


Figure 1: **Overview of JPFAT.** The adversary jointly perturbs input pixels δ_x and selected residual-stream states $\{\delta^{(l)}\}_{l=0}^{L-1}$.

3.1 JA-PGD: JOINT ALTERNATING PIXEL–FEATURE ATTACK

For each image, JA-PGD searches for both pixel and feature perturbations that together maximally change the prediction while staying in Δ_i. Since the attack needs a fixed target to push away from, the clean prediction is frozen (stop-gradient, $\mathrm{sg}[\cdot]$) rather than left free to drift, which would let the attack "win" by moving the target, not the prediction:

$$p_i^{\mathrm{sg}} = \mathrm{sg}[p_\theta(\cdot \mid x_i)]\,. \tag{6}$$

The candidate perturbed prediction under $\boldsymbol{\delta}_i$ is

$$\tilde{p}_i(\boldsymbol{\delta}_i) = p_\theta\big(\cdot \mid x_i + \delta_{x,i}; \delta_i^{(0)}, \ldots, \delta_i^{(L-1)}\big), \tag{7}$$

and JA-PGD searches for the $\boldsymbol{\delta}_i$ that pulls it as far as possible from the frozen target:

$$\boldsymbol{\delta}_i^\star \approx \arg\max_{\boldsymbol{\delta}_i \in \Delta_i} D_{\mathrm{KL}}\left(p_i^{\mathrm{sg}} \,\|\, \tilde{p}_i(\boldsymbol{\delta}_i)\right). \tag{8}$$

Updating pixels and features simultaneously would use gradients that go stale as soon as the pixels move, since every downstream hidden state then changes; so instead of one simultaneous step, JA-PGD alternates. At iteration t it updates the pixel perturbation with the current feature perturbations active, then updates all feature perturbations using the newly changed image:

$$\begin{aligned}
\delta_{x,i}^{t+1} &= \Pi_{\mathcal{B}_x(\varepsilon_x)}\left[\delta_{x,i}^t + a_x^{(t)}\alpha_x \,\mathrm{sign}\left(\nabla_{\delta_{x,i}} J_i(\delta_{x,i}^t; \delta_{f,i}^t)\right)\right], \\
\delta_i^{(l),t+1} &= \Pi_{\mathcal{B}_l(\varepsilon_l)}\left[\delta_i^{(l),t} + a_l^{(t)}\alpha_l \,\mathrm{sign}\left(\nabla_{\delta_i^{(l)}} J_i(\delta_{x,i}^{t+1}; \delta_{f,i}^t)\right)\right], \qquad l = 0, \ldots, L-1,
\end{aligned} \tag{9}$$

where $J_i(\boldsymbol{\delta}_i) = D_{\mathrm{KL}}(p_i^{\mathrm{sg}} \| \tilde{p}_i(\boldsymbol{\delta}_i))$; α_x, α_l are base step sizes (set in Section 3.2 for feature paths) and $a_x^{(t)}, a_l^{(t)}$ are the per-path effort weights from Section 3.3 (uniform, $1/K$, until then); $\delta_{f,i}^t$ collects all feature perturbations at step t. Π clips each perturbation back into its ball after every step, enforcing equation 4; pixel projection also keeps the image valid, $x_i + \delta_{x,i}^{t+1} \in [0,1]^d$. Paths start from independently sampled points inside their own ℓ_∞ balls, not zero.

The first backward pass gives the pixel gradient and the gradients used to score every path (Section 3.3); a second pass after the pixel update gives fresh feature gradients. JA-PGD thus costs exactly two backward passes per iteration, independent of L, though compute and memory per pass still scale with the attacked layers.

3.2 Relative Activation Budgets: Calibrating "Small" to Every Layer

A fixed raw radius (e.g. 0.01) is huge at a layer with small activations and negligible at a layer with large ones, so a single global radius does not mean the same thing at every depth. RAB instead sets each layer's budget relative to its own activation scale, which we must first measure.

For a clean token (i, n) at layer l, RAB computes its root-mean-square magnitude, takes the median over the minibatch, and smooths the estimate across minibatches with decay rate $\beta \in [0, 1)$, so a single noisy batch cannot swing the budget:

$$r_{i,n}^{(l)} = \left(\frac{1}{D_l} \sum_{d=1}^{D_l} \left(h_{i,n,d}^{(l)} \right)^2 \right)^{1/2}, \qquad s_l^{(t)} = \mathrm{median}_{i,n}\{ r_{i,n}^{(l)} \},$$
$$m_l^{(t)} = \begin{cases} s_l^{(t)}, & t = 0, \\ \beta m_l^{(t-1)} + (1-\beta) s_l^{(t)}, & t > 0. \end{cases} \tag{10}$$

This running scale $m_l^{(t)}$ then directly sets the layer's feature radius and base step, via two relative hyperparameters $\rho, \rho_{\mathrm{step}} > 0$ shared across all layers:

$$\varepsilon_l = \rho \, \mathrm{sg}\left[m_l^{(t)} \right], \qquad \alpha_l = \rho_{\mathrm{step}} \, \mathrm{sg}\left[m_l^{(t)} \right]. \tag{11}$$

RMS removes dependence on layer width, the median limits sensitivity to rare large tokens, and stop-gradient prevents the optimizer from shrinking m_l to weaken the attack. RAB is also scale-consistent: rescaling a layer's clean and perturbed activations by the same positive factor rescales its radius and step by that factor, leaving the normalized displacement unchanged (formal statement in Appendix E).

3.3 Gradient-Effort Allocation: Spending a Fixed Budget Wisely

With $K = L + 1$ paths but few PGD iterations, uniform updates waste effort on paths the model already resists while under-attacking the currently most vulnerable one. Enlarging a path's radius is not an option, since that would change the threat model; we instead keep every radius fixed and reallocate only the step size, which first requires scoring each path's current usefulness.

For path $k \in \mathcal{K}$, with gradient $g_{i,k}^{(t)}$ at the current pre-update state, we score each path by radius-weighted gradient magnitude, so a path that both allows a large step and currently has a large gradient scores highest:

$$q_k^{(t)} = \varepsilon_k \left(\frac{1}{B} \sum_{i=1}^{B} \mathrm{mean}_j \left| g_{i,k,j}^{(t)} \right| \right), \qquad \bar{q}^{(t)} = \frac{1}{K} \sum_{r \in \mathcal{K}} q_r^{(t)}, \qquad \hat{q}_k^{(t)} = \frac{q_k^{(t)}}{\max(\bar{q}^{(t)}, \eta_q)}, \tag{12}$$

with $\eta_q > 0$ a small floor avoiding division by near-zero. A raw score is noisy from batch to batch, so we turn scores into a stable allocation by softening them with temperature $\tau_a > 0$ and smoothing across iterations with momentum $\mu \in [0, 1)$:

$$\pi^{(t)} = \mathrm{softmax}\left(\hat{\boldsymbol{q}}^{(t)} / \tau_a \right), \qquad a^{(t)} = \mu a^{(t-1)} + (1-\mu)\pi^{(t)}, \qquad a^{(0)} = \frac{1}{K}\mathbf{1}. \tag{13}$$

Small τ_a concentrates effort on the top-scoring path; large τ_a spreads it near-evenly; μ damps swings between iterations. These scores reuse the first backward pass from Section 3.1, so allocation adds no extra pass. Crucially, allocation only ever touches the step, never the radius:

$$\alpha_k^{\mathrm{eff}} = a_k^{(t)} \alpha_k, \qquad \varepsilon_k^{\mathrm{eff}} = \varepsilon_k, \tag{14}$$

so the route the optimizer takes through Δ_i changes, not Δ_i itself (entropy-regularized interpretation, simplex preservation, and fixed-set feasibility proved in Appendix E).

3.4 VAAL: Weighting Training by Where the Attack Actually Lands

Some images are barely moved by the joint attack, others substantially so; uniform training pressure ignores this. VAAL scores each example's vulnerability *after* the attack runs, by first measuring how much the attack actually moved the model's output and internal state, and then uses that score purely as a detached loss weight, with no effect on the attack itself.

Let $z_i, \tilde{z}_i$ be clean and adversarial visual embeddings; d_i^{out} below measures how much the final prediction moved. Internal drift instead measures how much the hidden states moved, using a RAB-normalized Huber penalty ϕ (quadratic near zero, linear for outliers, so a few extreme tokens cannot dominate the score):

$$d_i^{\text{out}} = \frac{1 - \cos(z_i, \tilde{z}_i)}{2}, \qquad d_{i,l} = \frac{1}{N_l D_l} \sum_{n,d} \phi\left(\frac{\tilde{h}_{i,n,d}^{(l)} - h_{i,n,d}^{(l)}}{\text{sg}[m_l]} \right), \tag{15}$$

Not every layer drifts equally, so layerwise drift is combined with detached, temperature-τ_d attention, letting layers with larger drift contribute more to the summary:

$$b_{i,l} = \text{sg}[\text{softmax}_l\left(d_{i,\cdot}/\tau_d\right)], \qquad \bar{d}_i^{\text{int}} = \frac{\sum_{l=0}^{L-1} b_{i,l} d_{i,l}}{1 + \sum_{l=0}^{L-1} b_{i,l} d_{i,l}}. \tag{16}$$

The denominator keeps $\bar{d}_i^{\text{int}}$ bounded; detaching $b_{i,l}$ stops the model from "solving" this by distorting attention rather than reducing drift.

Let $a = a^{(T-1)}$ be the final allocation (T the total JA-PGD iterations) and $c(a) = 1 - H(a)/\log K$ for $K > 1$ ($c(a) = 0$ for $K = 1$) its concentration. A concentrated allocation signals a path worth attending to, so we weight such examples more, combining output and internal drift into one normalized, bounded per-example weight ($\eta_u > 0$ again a small floor):

$$u_i = \frac{d_i^{\text{out}} + \bar{d}_i^{\text{int}}}{2}, \qquad \bar{u} = \frac{1}{B}\sum_i u_i, \qquad \hat{u}_i = \frac{u_i}{\text{sg}[\max(\bar{u}, \eta_u)]},$$

$$w_i = \text{sg}[1 + \hat{u}_i(1 + c(a))], \qquad \mathcal{L}_{\text{adv}} = \frac{1}{B}\sum_i w_i D_{\text{KL}}\left(p_\theta(\cdot \mid x_i) \,\|\, p_\theta(\cdot \mid x_i, \boldsymbol{\delta}_i^\star)\right). \tag{17}$$

$p_\theta(\cdot \mid x_i)$ inside $\mathcal{L}_{\text{adv}}$ is *not* detached, so θ is updated through both clean and adversarial predictions; only the ingredients of w_i are detached, so it acts purely as a per-example weight (weight bounds and the detached-gradient property in Appendix E). We build u_i from embedding and internal-layer drift rather than reusing the attack's own objective $J_i(\boldsymbol{\delta}_i^\star)$ directly as the weighting signal; Appendix E.4 discusses why.

***RAB** sets the budget, **JA-PGD** and **allocation** spend it, **VAAL** decides how much it counts.*

3.5 Training and Evaluation Regimes

Combining the clean loss with this vulnerability-weighted adversarial term gives the full objective, with training epoch e:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{nat}}(\theta) + \lambda(e)\mathcal{L}_{\text{adv}}(\theta). \tag{18}$$

with $\lambda(e)$ warmed up from zero during early training, so the model first learns the task before being pushed toward robustness.

JA-PGD-A (training). Uses the KL objective in equation 8, RAB-derived layerwise budgets, and adaptive step allocation (Algorithm 1).

JA-PGD-F (evaluation). A simpler, fixed version used for reporting (Algorithm 2): it drops RAB and step allocation, uses fixed, pre-specified pixel and feature radii with uniform step sizes, and optimizes a label-based objective instead of the KL one. Budgets, attacked layers, iterations, and restarts are identical across all compared methods, so no method benefits from an easier evaluation attack induced by its own training statistics; ε_f is one global constant applied identically everywhere, not RAB-calibrated.

4 EXPERIMENTS: DOES ATTACKING INSIDE THE ENCODER PAY OFF?

We evaluate whether JPFAT improves robustness to the proposed joint pixel–feature threat model while preserving few-shot transfer. Specifically, we test: clean and joint robust base-to-new generalization; robustness of the individual pixel and feature paths; transfer across datasets; and the effects of VAAL, RAB, allocation, attack schedule, robust-loss weight, and attack depth.

4.1 EXPERIMENTAL PROTOCOL

We use the 11-dataset few-shot adversarial prompt-learning benchmark (Zhou et al., 2024), covering object, fine-grained, scene, texture, satellite, and action recognition. We use MaPLe-style prompts (Khattak et al., 2023) on CLIP ViT-B/32 (Radford et al., 2021), prompt depth $P = 9$, and attack depth $L = 3$. Unless stated otherwise, all experiments follow the 16-shot protocol with training batch size 4 and test batch size 200. Results on the CLIP ViT-B/16 backbone are reported separately in Appendix F.13.

Prompts are trained for 20 epochs using SGD with learning rate 0.0035, cosine decay, and a one-epoch warm-up at 10^{-5}. JA-PGD-A uses $T = 5$ iterations, $\varepsilon_x = \alpha_x = 1/255$, $\rho = 0.005$, and $\rho_{\text{step}} = 0.001$. We linearly warm up λ from zero to 1.5 over the first three epochs.

For robustness reporting, every method is evaluated with the same fixed-budget, uniform-step, labelled-cross-entropy JA-PGD-F attack (Algorithm 2) for 100 iterations, using the same attack depth $L = 3$ and fixed feature budgets $\varepsilon_f = 0.002$, $\alpha_f = 0.0004$ for the joint attack across all compared methods. We report clean, pixel-only, feature-only, and joint JA-PGD-F accuracy separately, since these attacks represent distinct capabilities.

4.2 MAIN RESULTS

Table 2 evaluates **base-to-new generalization** under clean and fixed-budget adversarial evaluation. JPFAT obtains the strongest joint JA-PGD-F harmonic mean, 5.4%, against 0.0% to 0.4% for every baseline, while its clean HM of 55.6% trails FAP by only 1.1 points. It also achieves the best pixel-PGD and feature-PGD HM, 27.8% and 18.4% respectively, so the joint-robustness gain is not won on one path at the expense of the other; both constituent paths improve together.

Table 2: **Base-to-new generalization and fixed-budget robustness,** averaged over all 11 datasets. Joint denotes the fixed-budget JA-PGD-F reporting attack; Pixel and Feature PGD isolate the corresponding paths. **B**: base; **N**: novel; **HM**: harmonic mean.

	Clean			Joint JA-PGD-F			Pixel PGD			Feature PGD		
Method	B	N	HM	B	N	HM	B	N	HM	B	N	HM
ZsRobust ICLR'23	41.6±3.5	35.0±4.1	37.3	0.7±0.3	0.3±0.1	0.4	15.0±1.9	10.0±1.4	11.5	8.4±2.4	4.4±1.2	5.5
AdvPT ECCV'24	64.1±0.6	**53.2±1.7**	**58.1**	0.0±0.0	0.1±0.0	0.0	4.3±0.2	3.8±0.9	3.9	1.5±0.2	1.5±0.9	1.4
FAP NeurIPS'24	**69.1±2.9**	49.0±3.7	56.7	6.2±0.4	0.4±0.3	0.4	33.8±1.3	21.0±1.4	25.3	22.4±2.5	1.9±1.3	2.5
JPFAT (Ours)	68.6±1.0	47.9±3.4	55.6	**9.8±0.7**	**4.2±0.6**	**5.4**	**40.0±1.7**	**22.7±2.7**	**27.8**	**29.1±1.0**	**14.6±2.6**	**18.4**

JPFAT's clean HM trails FAP and AdvPT by small margins (55.6% vs. 56.7% and 58.1%). This is expected, not component-specific: AdvPT and FAP train only against Δ_i at $L = 0$ (pixels alone), so their prompts fit clean features freely, while JPFAT's outer objective (equation 18) also penalizes KL drift from perturbations injected inside the encoder at all $L = 3$ blocks, trading some clean sharpness for stability under a strictly larger threat set. Figure 2 shows that JPFAT tracks FAP's clean accuracy closely as the shot count grows, while its joint robustness improves more sharply with additional adaptation data; the right panel confirms that allocation redistributes update effort across pixel and feature paths while their projection radii stay fixed throughout.

For **cross-dataset transfer**, we train prompts on ImageNet-1k and evaluate them zero-shot on 10 target datasets. Table 3 reports clean and JA-PGD accuracy separately, averaged over all 11 datasets. JPFAT improves average JA-PGD robustness over FAP by roughly 1.4× (4.4%→6.2%), while giving up only 2.1 points of average clean transfer relative to the strongest baseline (45.5% vs. AdvPT's 47.6%), the same modest, consistent trade-off already seen under base-to-new generalization.

Figure 3 shows a qualitative Oxford Pets example. Under the joint attack, FAP's prediction flips and its Grad-CAM attention shifts off the target animal, whereas JPFAT keeps both the correct prediction and object-focused attention. This illustration complements, rather than replaces, the quantitative fixed-budget evaluation above.

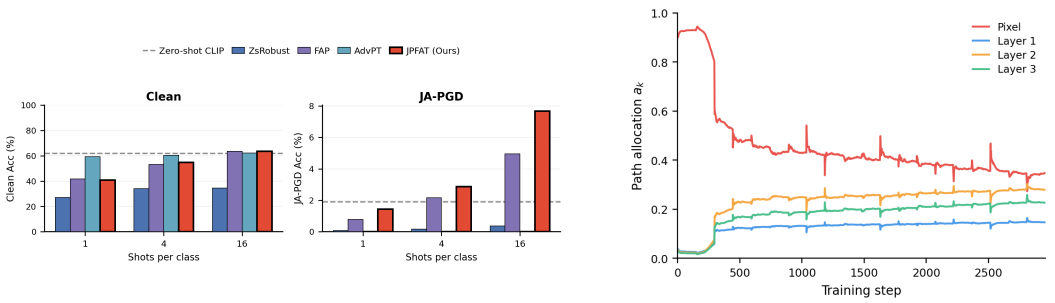


Figure 2: **Few-shot scaling and allocation dynamics. Left:** clean and fixed-budget joint JA-PGD-F accuracy versus shots per class, averaged over 11 datasets. **Right:** mean allocation across pixel and feature paths during training. Allocation redistributes update effort using radius-weighted scores while all projection radii remain fixed.

Table 3: **Cross-dataset transfer from ImageNet-1k.** Prompts are trained on ImageNet-1k and evaluated zero-shot on 10 target datasets. Clean and JA-PGD denote standard and fixed-budget JA-PGD-F accuracy (%). Best value per dataset and metric in **bold**.

Method	Metric	ImageNet	DTD	Caltech101	Flowers102	Food101	Oxford Pets	UCF101	EuroSAT	FGVC Aircraft	Stanford Cars	SUN397	Avg.
ZsRobust ICLR'23	Clean	42.4	**27.4**	82.7	40.6	48.6	65.9	39.8	24.5	7.7	24.7	39.8	40.4
	JA-PGD	0.1	0.1	3.3	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.4
AdvPT ECCV'24	Clean	**56.4**	**27.4**	**84.6**	**49.2**	51.7	78.1	47.0	**39.1**	**8.9**	32.5	**49.1**	**47.6**
	JA-PGD	0.0	0.4	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
FAP NeurIPS'24	Clean	55.0	27.2	83.6	47.7	**52.3**	**78.8**	**47.4**	30.7	8.5	30.8	47.2	46.3
	JA-PGD	3.1	5.3	18.7	3.6	0.4	4.8	1.9	8.5	0.1	0.0	1.8	4.4
JPFAT (Ours)	Clean	54.5	24.7	81.5	41.7	50.4	77.5	46.2	36.6	6.4	**34.8**	45.8	45.5
	JA-PGD	**5.0**	**6.5**	**23.7**	**5.7**	**1.9**	**7.8**	**3.7**	**10.8**	**0.2**	**0.2**	**2.3**	**6.2**

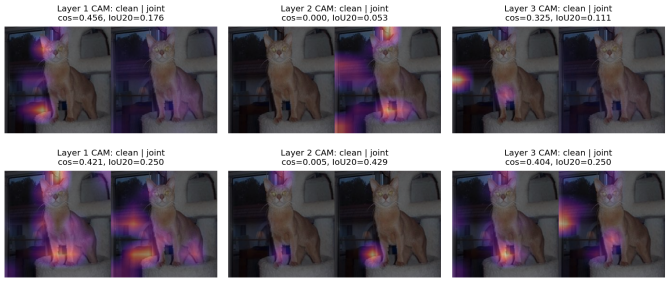


Figure 3: **Qualitative joint-attack example on Oxford Pets.** Grad-CAM maps (clean | joint) at Layers 1–3 for target class *abyssinian*, with FAP (top) and JPFAT, ours (bottom). Under the joint JA-PGD-F attack, FAP's prediction flips to *persian* and its attention shifts off the cat, while JPFAT stays correct with attention concentrated on the object.

5 Ablations: Which Piece Is Doing the Work?

Does VAAL improve vulnerability-aware training? Figure 4(a–b) compares VAAL against uniform weighting, $w_i \equiv 1$. Enabling VAAL costs a small amount of clean accuracy (80.9%→79.8%) but improves pixel robustness from 40.7% to 45.1% and joint robustness from 3.3% to 6.2%. The right panel also shows lower, more stable allocation concentration once VAAL is active, consistent with the intended interaction between vulnerability weighting and multi-path adversarial training: VAAL discourages the optimizer from over-relying on any single dominant path.

Do RAB and allocation both contribute? Table 4(c) isolates RAB and gradient-effort allocation on Oxford Pets. Combining both yields the highest pixel-PGD and joint JA-PGD-F accuracy of any configuration tested. Figure 4(c) varies attack depth on Caltech101: increasing L makes feature-only and joint attacks noticeably harder, while pixel-only robustness barely moves. Together, these results support scale-calibrated, multi-layer perturbation budgets rather than a single feature radius.

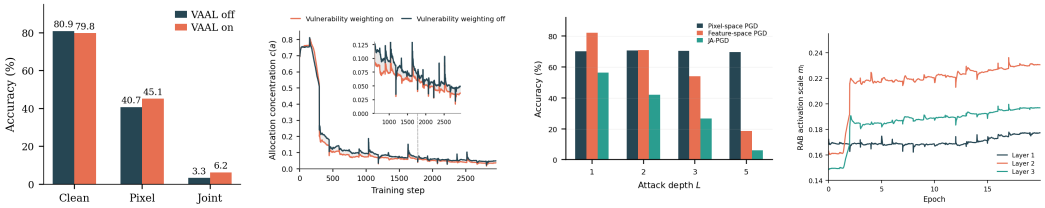


Figure 4: **VAAL ablation and attack-depth/RAB diagnostics. (a–b)** VAAL vs. uniform $w_i \equiv 1$ (Oxford Pets) and allocation concentration $c(a)$: VAAL trades a small clean-accuracy drop for larger pixel/joint robustness with lower, more stable concentration. **(c)** Attack depth, Caltech101: larger L makes feature-only and joint attacks harder, motivating layer-calibrated budgets. **(d)** RAB scale m_l (equation 10), Oxford Pets: differs substantially across layers ($\sim$1.3–1.5$\times$), non-monotonic in depth, each converging to a stable value (Appendix E)

.

Method	B	N	HM
ZsRobust	1.3±0.8	0.7±0.2	0.8
AdvPT	0.1±0.0	0.2±0.0	0.1
FAP	7.6±0.5	0.5±0.4	0.6
JPFAT	**11.2±0.8**	**5.0±0.9**	**6.5**

λ	Clean	Joint	HM
1.0	90.0	24.7	38.8
1.5	**90.2**	30.1	45.1
2.0	89.7	26.1	40.4
3.0	88.4	**30.4**	**45.2**

RAB	Alloc.	Pixel PGD	Joint
✗	✗	43.2	5.1
✗	✓	42.3	5.4
✓	✗	43.7	5.0
✓	✓	**45.1**	**6.2**

Table 4: **Schedule, robust-loss, and mechanism ablations. (a)** Sequential pixel-then-feature PGD accuracy (%), avg. over 11 datasets; B/N/HM: base/novel/harmonic mean. **(b)** Clean–robustness trade-off from λ, Caltech101; Joint is fixed-budget JA-PGD-F. **(c)** RAB and allocation ablated individually/jointly, Oxford Pets; Pixel PGD and Joint are fixed-budget accuracy.

Are gains specific to the joint evaluation schedule? Table 4(a) evaluates sequential pixel-then-feature PGD using matched perturbation paths and iteration budget. JPFAT remains the strongest method here too, indicating its gains are not an artifact of the alternating reporting schedule specifically. Table 4(b) sweeps λ on Caltech101 to characterize the clean-robustness trade-off; we use $\lambda = 1.5$ throughout the main experiments.

Generalization to deeper attacks. Since a defender may not know the attacker's true access depth, we test L_{train}=3 against a deeper test-time attack, L_{eval}=5. Across three datasets it tracks a model trained and evaluated at the matched depth within 1–3.3 points, with the smallest gap on JA-PGD (0.2 points), showing JPFAT generalizes beyond its training depth rather than overfitting to L_{train}=3 (Appendix F.9).

Computational efficiency. JPFAT matches FAP's parameter count (3.56M) and clean-forward GFLOPs (224.37), adding no inference-time cost. At training time it is 3.05$\times$ slower than AdvPT per step but needs 36% fewer GFLOPs than FAP (3450 vs. 5383; Appendix F.8), since JA-PGD computes its clean target once per step and evaluates gradients per path rather than a full backward pass every PGD iteration. Its only overhead relative to FAP is 1.37$\times$ peak memory, from the per-block activation buffers and the clean/adversarial hidden-state pairs VAAL's drift scoring requires.

Additional ablations. We report further validation, most notably AutoAttack (external, non-adaptive) and a mechanism-aware adaptive attacker, alongside additional robustness and internal-mechanism diagnostics, in the appendix F.

6 Conclusions

Adversarial prompt learning defends only against pixel-space attacks, yet the prompts it trains sit inside the encoder. JPFAT trains against joint pixel–residual-stream perturbations using JA-PGD, which alternates pixel and feature updates under RAB-calibrated radii and reallocates step effort without shrinking any radius. VAAL weights each example by the drift the attack induces. Across 11 datasets, joint robustness rises well above pixel-only baselines while clean accuracy stays competitive, and the gains survive a stronger, RAB-aware adaptive attacker. We hope this encourages defences that operate inside the encoder, not only at its input.

Ethics Statement

As with other adversarial-training methods, the adaptivity of JPFAT's attack (its entropy-regularized reallocation of effort across the pixel and feature paths, acting inside a frozen encoder) obscures its behavior, making analyses like those in Section 5 and Appendix F.1 both challenging and essential for understanding such defences. Additionally, as with other work on adversarial robustness, these advances hold promise for safer and more reliable vision–language models but also introduce risks, including misuse (a stronger attack in the wrong hands) and bias (inherited from the pretrained backbone and benchmark data).

Reproducibility Statement

All hyperparameters and experimental details are reported in Section 4.1 and Appendix C, with complete pseudocode for JA-PGD, RAB, and the JPFAT training step in Appendix D and full assumptions and proofs of the feasible-set guarantees in Appendix E. The supplementary material provides code (including plotting scripts) and evaluation data to fully reproduce all the main results in this work.

Use of LLMs

Large language models were used as a writing and analysis aid throughout the preparation of this paper, including polishing grammar and clarity and drafting and revising prose from author-provided technical content. All research ideation, experimental design, execution, and interpretation of results were carried out by the authors; language-model output was reviewed and verified by the authors, who take full responsibility for the content of the paper.

References

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, 2014.

Xiaoyun Chen and Ning Zhang. Layer-wise adversarial training approach to improve adversarial robustness. In *International Joint Conference on Neural Networks (IJCNN)*, 2020.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, 2020.

Róbert Csordás, Christopher D. Manning, and Christopher Potts. Do language models use their depth efficiently? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative-Model Based Vision*, 2004.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

Stefan Heimersheim and Alex Turner. Residual stream norms grow exponentially over the forward pass, 2023. LessWrong.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 125–136, 2019.

Xin Jia et al. Evolution-based region adversarial prompt learning for robustness enhancement in vision-language models. *arXiv preprint arXiv:2503.12874*, 2025.

Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop on 3D Representation and Recognition*, 2013.

Chao Li, Yonghao Liao, Caichang Ding, and Zhiwei Ye. Mdapt: Multi-modal depth adversarial prompt tuning to enhance the adversarial robustness of visual language models. *Sensors*, 25(1): 258, 2025. doi: 10.3390/s25010258.

Lin Luo, Xin Wang, Bojia Zi, Shihao Zhao, and Xingjun Ma. Adversarial prompt distillation for vision-language models. *arXiv preprint arXiv:2411.15244*, 2025.

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 (ICLR)*, 2018.

Pratyush Maini, Eric Wong, and J. Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning (ICML)*, 2020.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Chengzhi Mao, Scott Geng, Junfeng Yang, Xin Wang, and Carl Vondrick. Understanding zero-shot adversarial robustness for large-scale models. In *International Conference on Learning Representations (ICLR)*, 2023.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Christian Schlarmann, Naman Deep Singh, Francesco Croce, and Matthias Hein. Robust clip: Unsupervised adversarial fine-tuning of vision embeddings for robust large vision-language models. In *International Conference on Machine Learning (ICML)*, 2024.

Lijun Sheng, Jian Liang, Zilei Wang, and Ran He. R-tpt: Improving adversarial robustness of vision-language models through test-time prompt tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29958–29967, 2025.

Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, and Stephen Casper. Targeted latent adversarial training improves robustness to persistent harmful behaviors in llms. *arXiv preprint arXiv:2407.15549*, 2024.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

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 (ICLR)*, 2014.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019.

Xin Wang, Kai Chen, Jiaming Zhang, Jingjing Chen, and Xingjun Ma. Tapt: Test-time adversarial prompt tuning for robust inference in vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Songlong Xing, Zhengyu Zhao, and Nicu Sebe. Clip is strong enough to fight back: Test-time counterattacks towards zero-shot adversarial robustness of clip. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15172–15182, 2025.

Chiu Wai Yan, Tsz-Him Cheung, and Dit-Yan Yeung. ILA-DA: Improving transferability of intermediate level attack with data augmentation. In *International Conference on Learning Representations (ICLR)*, 2023.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019.

Jiaming Zhang, Xingjun Ma, Xin Wang, Lingyu Qiu, Jiaqi Wang, Yu-Gang Jiang, and Jitao Sang. Adversarial prompt tuning for vision-language models. In *European Conference on Computer Vision (ECCV)*, pp. 56–72, 2024.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

Yiwei Zhou, Xiaobo Xia, Zhiwei Lin, Bo Han, and Tongliang Liu. Few-shot adversarial prompt learning on vision-language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

APPENDIX CONTENTS

A **Notation and Symbols** 14

B **Related Work** 15

C **Implementation Details** 16

C.1 Datasets 16

C.2 Baselines 16

D **Algorithms** 16

E **Theoretical Properties** 17

E.1 Conventions and Assumptions 17

E.2 Properties of Relative Activation Budgets 18

E.3 Properties of Gradient-Effort Allocation 18

E.4 Properties of VAAL 19

E.5 Scope of the Guarantees 20

F **Additional Ablation** 20

F.1 Qualitative Case Study: Drift and Attribution Analysis 20

F.2 Few-Shot Scaling Across All Datasets 21

F.3 Full Per-Dataset Robustness Across All Attack Settings 22

F.4 AutoAttack Evaluation 22

F.5 RAB Scale Sensitivity (ρ) 23

F.6 Allocation Dynamics Across Datasets 24

F.7 Per-Dataset RAB, Allocation, and VAAL Ablation 25

F.8 Computational Overhead 25

F.9 Generalization to a Deeper Evaluation Attack 26

F.10 Adaptive Attack Against JPFAT's Internal Mechanism 27

F.11 Sensitivity to JA-PGD's Pixel–Feature Leader Order 27

F.12 Correlation Between VAAL's Vulnerability Score and the Attack Objective 27

F.13 Generalization to the ViT-B/16 Backbone 28

G **Limitations and Future Work** 28

A NOTATION EXPLANATIONS

Table 5 summarizes the notation used throughout the paper.

Table 5: Notation table.

Notation	Description
Data and model	
x_i, y_i	input image and label of sample i
θ	learnable multi-modal prompt parameters
V, T	frozen visual and text encoders
$z_\theta(x)$, $t_c(\theta)$	prompted visual and text representations
$p_\theta(\cdot \mid x)$	predictive distribution
s	logit scale
M	number of residual blocks in the visual transformer
P	prompt depth (blocks carrying learnable prompt tokens)
L	attack depth (blocks carrying feature perturbations)
$h_i^{(l)}$, $\tilde{h}_i^{(l)}$	clean and adversarial hidden state at block l
N_l, D_l	number of tokens and channel width at block l
Threat model	
$\delta_{x,i}$	pixel perturbation
$\delta_i^{(l)}$	feature perturbation at block l
$\boldsymbol{\delta}_i$	joint perturbation tuple
ε_x, α_x	pixel radius and step
ε_l, α_l	feature radius and step at block l
ε_f, α_f	fixed feature radius and step (evaluation)
Δ_i	joint threat set
$K = L + 1$	number of attack paths
$\mathcal{K}$	index set of attack paths
RAB	
$r_{i,n}^{(l)}$	token RMS at block l
s_l	observed (median) activation scale at block l
m_l	stop-gradient EMA scale at block l
β	EMA smoothing coefficient
ρ, ρ_{step}	relative feature radius and step
Attack and allocation	
$J_i(\boldsymbol{\delta}_i)$	inner (KL-to-clean) attack objective
$g_{i,k}$	gradient of path k
q_k, $\hat{q}_k$	radius-weighted block score and its normalization
$\bar{q}$, η_q	path-mean score and allocation-score floor
$\pi^{(t)}$	proposed allocation at iteration t
$a^{(t)}$	realized allocation (after momentum)
τ_a, μ	allocation temperature and momentum
t, T	JA-PGD iteration index and total iteration count
VAAL	
d_i^{out}	output (embedding) drift
$d_{i,l}$	per-depth Huber drift
$\phi(\cdot)$	RAB-normalized Huber penalty
$b_{i,l}$	depth attention weight
τ_d	depth-attention temperature
d_i^{int}, $\bar{d}_i^{\text{int}}$	aggregated and squashed internal drift
$c(a)$	allocation concentration
u_i, $\hat{u}_i$	raw and batch-normalized vulnerability
$\bar{u}$, η_u	batch-mean vulnerability and weight floor
w_i	detached per-sample weight
$\mathcal{L}_{\text{nat}}$, $\mathcal{L}_{\text{adv}}$	natural and adversarial loss
$\lambda(e)$	robust coefficient

B Related Work

Adversarial examples and robustness. Adversarial examples were identified by Szegedy et al. (2014) and Goodfellow et al. (2015), who showed that imperceptible input perturbations reliably fool deep networks. Ilyas et al. (2019) reframe this: adversarial vulnerability arises from non-robust features that are genuinely predictive yet brittle, so adversarial examples are a natural consequence of learning from data rather than an artefact of model architecture. Tsipras et al. (2019) show that natural and robust accuracy generally trade off, since their optimal classifiers differ. Projected gradient descent (Madry et al., 2018) and its consistency-based surrogate TRADES (Zhang et al., 2019) remain the standard inner maximizers; reliable evaluation of the resulting defences requires diverse parameter-free attacks (Croce & Hein, 2020).

Adversarial robustness of vision–language models. CLIP-style models (Radford et al., 2021) inherit the adversarial vulnerability of their visual encoders, and standard adversarial training is expensive to apply to a frozen foundation model and tends to erode zero-shot transfer. Mao et al. (2023) identify text-encoder-free fine-tuning as the cause of this erosion and propose TeCoA, a text-guided contrastive adversarial fine-tuning objective that preserves vision–language alignment; FARE (Schlarmann et al., 2024) fine-tunes the vision encoder unsupervised so that adversarial embeddings remain close to the originals, leaving downstream heads intact. A parallel line defends at inference: test-time prompt tuning (Sheng et al., 2025) and test-time counterattacks (Xing et al., 2025) adapt or purify at deployment rather than during training. All of these operate entirely in pixel space.

Adversarial prompt learning. Prompt learning adapts a frozen VLM through a small set of continuous parameters (Zhou et al., 2022), optionally shared across modalities and inserted at depth (Khattak et al., 2023). AdvPT (Zhang et al., 2024) applies adversarial training within this budget, aligning learnable text prompts with adversarial image embeddings. FAP (Zhou et al., 2024) observes that naive adversarial prompt learning collapses natural–adversarial feature consistency in the few-shot regime and introduces a consistency objective that preserves it; APD (Luo et al., 2025) instead distills knowledge from a clean teacher into bimodal visual and textual prompts, avoiding reliance on a robustly pre-trained backbone; TAPT (Wang et al., 2025) moves the adaptation to test time; MDAPT (Li et al., 2025) jointly trains visual and textual prompts across multiple insertion depths for stronger semantic alignment; and ER-APT (Jia et al., 2025) combines gradient-based PGD with an evolutionary search over adversarial examples to increase attack diversity during training. Every method in this line trains the prompts against a *pixel-only* adversary, even though the prompts act inside the encoder: MDAPT varies prompt insertion depth but still attacks only the input image. JPFAT closes that asymmetry by admitting perturbations at the same depths at which the prompts are inserted.

Feature- and latent-space adversarial training. Perturbing hidden representations rather than inputs reaches failure modes that pixel-space attacks miss. ILA (Huang et al., 2019) maximizes the projection of a perturbation onto a pre-specified intermediate layer of the source model to improve black-box transfer; ILA-DA (Yan et al., 2023) extends it with three data augmentation techniques that further boost transferability. Layer-wise adversarial training (Chen & Zhang, 2020) perturbs intermediate activations of convolutional networks for defence, and latent adversarial training (Sheshadri et al., 2024) perturbs a hidden layer of a language model to elicit persistent harmful behaviours that input-space attacks do not surface. These methods perturb a single hand-picked layer with an absolute budget and do not compose feature perturbations with an input-space attack. JPFAT attacks L depths jointly, conditions them on an already-perturbed image, and derives every radius from a measured per-layer activation scale via RAB.

Multiple perturbation models and adaptive budgets. Robustness to more than one threat model is usually formulated as a maximum over a finite *union* of independent perturbation sets (Tramèr & Boneh, 2019; Maini et al., 2020), where the adversary picks one set per example and the sets do not interact. Our setting is compositional: the feature perturbation is applied to representations of an already-perturbed image, so the paths are coupled through a single forward pass and a joint optimizer can exploit their interaction. Adaptive schemes that redistribute a budget across components typically shrink some radii, making the effective threat model a function of the attack's own state;

our allocation instead scales step sizes while leaving every path's projection radius intact, so the declared threat set is unchanged and reported numbers remain comparable across methods.

C Implementation Details

All experiments are conducted in an environment running PyTorch 2.1.2 (CUDA 11.8) on Python 3.8. Experiments on all 11 datasets are carried out on a single NVIDIA L40S GPU, except for the ImageNet-1k few-shot scaling analysis, which uses multiple A40 GPUs.

C.1 Datasets

We evaluate on the 11-dataset few-shot benchmark established for adversarial prompt learning (Zhou et al., 2024): ImageNet-1k (Deng et al., 2009) and Caltech101 (Fei-Fei et al., 2004) for generic object recognition; Oxford Pets (Parkhi et al., 2012), Stanford Cars (Krause et al., 2013), Flowers102 (Nilsback & Zisserman, 2008), Food101 (Bossard et al., 2014), and FGVC Aircraft (Maji et al., 2013) for fine-grained recognition; SUN397 (Xiao et al., 2010) for scene recognition; DTD (Cimpoi et al., 2014) for texture recognition; EuroSAT (Helber et al., 2019) for satellite imagery recognition; and UCF101 (Soomro et al., 2012) for action recognition. For base-to-new generalization, each dataset's classes are split evenly into a base half, on which all methods are trained, and a new half, seen only at evaluation time, following Zhou et al. (2022); training uses 16 shots per base class, and evaluation for both splits uses the dataset's standard test partition, subsampled to a fixed test batch size.

C.2 Baselines

We compare against ZsRobust (TeCoA) (Mao et al., 2023), which adversarially fine-tunes the CLIP vision encoder with a text-guided contrastive objective aligning adversarial image embeddings with their ground-truth text embeddings; AdvPT (Zhang et al., 2024), which tunes learnable text prompts against a pixel-space PGD adversary with a cross-entropy objective; and FAP (Zhou et al., 2024), which regularizes natural–adversarial feature consistency to address its collapse under few-shot adversarial prompt learning. All three are retrained under the same backbone, shot count, dataset splits, and evaluation pipeline as JPFAT, rather than taken from published numbers, so that clean and robust accuracy are directly comparable.

Table 6: Overall methodological comparison of baselines and JPFAT. All methods are evaluated with the same labelled cross-entropy JA-PGD-F attack (Section 3.5).

Method	Visual Prompts	Text Prompts	Deep Prompts	Threat Model	Training Loss
ZsRobust (TeCoA)	5	✗	✗	Pixel	TeCoA
AdvPT	✗	16	✗	Pixel	CE
FAP	2	2	✓	Pixel	KL-Cos
JPFAT (Ours)	2	2	✓	**Pixel + Feature**	VAAL

D Algorithms

This appendix gives the full pseudocode referenced in Section 3: the RAB update, the two JA-PGD regimes, and the complete JPFAT training step.

Algorithm 1 JA-PGD-A — adaptive alternating pixel–feature attack (training)

Require: batch $\{x_i\}$, prompts θ, pixel budget $(\varepsilon_x, \alpha_x)$, RAB budgets $\{\varepsilon_l, \alpha_l\}$, iterations T, temperature τ_a, momentum μ

Register hooks on blocks $0, \dots, L-1$; initialize all blocks uniformly on their ℓ_∞ balls; project $x + \delta_x$ into $[0,1]^d$

$p_i \leftarrow p_\theta(\cdot \mid x_i)$ (fixed target); $a^{(0)} \leftarrow \frac{1}{K}\mathbf{1}$, $K = L{+}1$

for $t = 0, \dots, T-1$ **do**

  Pixel block (feature perturbations active): obtain $\{g_k^{(t)}\}_{k \in \mathcal{K}}$ for all K paths from one backward pass of $J = \frac{1}{B}\sum_i D_{\mathrm{KL}}(p_i \| \tilde{p}_i(x_i{+}\delta_x^t; \delta_f^t))$

  $q_k \leftarrow \varepsilon_k \,\mathrm{mean}|g_k^{(t)}|$; $\hat{q}_k \leftarrow q_k / \max(\bar{q}, \eta_q)$; $a^{(t)} \leftarrow \mu\, a^{(t-1)} + (1-\mu)\,\mathrm{softmax}(\hat{q}/\tau_a)$, renormalize

  $\delta_x^{t+1} \leftarrow \Pi_{\mathcal{B}_x(\varepsilon_x)}\big[\delta_x^t + a_x^{(t)} \alpha_x \,\mathrm{sign}(g_x^{(t)})\big]$; clamp $x + \delta_x^{t+1}$ to $[0,1]^d$

  Feature block (pixel perturbation fixed): obtain $\{g_l\}$ for all L layers from one backward pass, at the updated δ_x^{t+1}

  $\delta^{(l,t+1)} \leftarrow \Pi_{\mathcal{B}_l(\varepsilon_l)}\big[\delta^{(l,t)} + a_l^{(t)} \alpha_l \,\mathrm{sign}(g_l)\big]$ for all l ▷ each block's full radius, never allocation-scaled

end for

return $\delta_x^\star$, $\{\delta^{(l)\star}\}$, final allocation $a = a^{(T-1)}$; remove all hooks

Algorithm 2 JA-PGD-F — fixed-budget alternating attack (evaluation and inference)

Require: batch $\{(x_i, y_i)\}$, prompts θ, fixed budgets $(\varepsilon_x, \alpha_x)$, $(\varepsilon_f, \alpha_f)$, iterations T

Register hooks on blocks $0, \dots, L-1$; initialize all blocks uniformly on their ℓ_∞ balls; project $x + \delta_x$ into $[0,1]^d$

for $t = 0, \dots, T-1$ **do**

  Pixel block (feature perturbations active): $g_x \leftarrow \nabla_{\delta_x} \ell_{\mathrm{CE}}(f_\theta(x{+}\delta_x^t; \delta_f^t), y)$; $\delta_x^{t+1} \leftarrow \Pi_{\mathcal{B}_x(\varepsilon_x)}\big[\delta_x^t + \alpha_x \,\mathrm{sign}(g_x)\big]$, clamp $x + \delta_x^{t+1}$ to $[0,1]^d$

  Feature block (pixel perturbation fixed): obtain $\{g_l\}$ for all L layers from one backward pass, at the updated δ_x^{t+1}

  $\delta^{(l,t+1)} \leftarrow \Pi_{\mathcal{B}_l(\varepsilon_f)}\big[\delta^{(l,t)} + \alpha_f \,\mathrm{sign}(g_l)\big]$ for all l ▷ uniform steps, $a \equiv 1$

end for

return δ_x^T, $\{\delta^{(l,T)}\}$; remove all hooks

E Theoretical Properties

What this appendix establishes

This appendix collects the structural properties underlying JPFAT's three mechanisms: RAB (Section 3.2), gradient-effort allocation (Section 3.3), and VAAL (Section 3.4). Together they formalize three claims made in the main text: RAB's radii transform consistently under activation rescaling (Prop. 1); allocation and JA-PGD's projections leave the declared threat set Δ_i unchanged (Prop. 4); and VAAL's weights are bounded and detached from the prompt gradient (Prop. 5). These results say nothing about whether finite-step JA-PGD solves the non-convex inner maximization in equation 8; that is assessed empirically via the matched fixed-budget attacks of Section 3.5.

E.1 Conventions and assumptions

Every attacked layer has $D_l \geq 1$ with a nonempty token collection for the minibatch median in equation 10; $0 \leq \beta < 1$; $0 \leq \mu < 1$; $\eta_q, \eta_u > 0$; and $x_i \in [0,1]^d$, so $B_x(\varepsilon_x)$ is nonempty for any

$\varepsilon_x > 0$. Let

$$\mathcal{S}_K = \Big\{ a \in \mathbb{R}^K_{\geq 0} : \textstyle\sum_k a_k = 1 \Big\}, \qquad H(a) = -\textstyle\sum_k a_k \log a_k \ (0 \log 0 := 0), \tag{19}$$

and, with $c(a) := 0$ for $K = 1$,

$$c(a) = 1 - \frac{H(a)}{\log K}, \qquad K > 1, \tag{20}$$

so $c(a) = 0$ at the uniform allocation and $c(a) \to 1$ as effort concentrates on one path.

E.2 Relative Activation Budgets

Proposition 1 (Equivariance of the RAB statistic)**.** *Under a positive rescaling $h^{(l)} \mapsto c_l h^{(l)}$, $\tilde{h}^{(l)} \mapsto c_l \tilde{h}^{(l)}$ ($c_l > 0$) of layer l's clean and perturbed activations,*

$$s_l \mapsto c_l s_l, \quad m_l^{(t)} \mapsto c_l m_l^{(t)}, \quad \varepsilon_l \mapsto c_l \varepsilon_l, \quad \alpha_l \mapsto c_l \alpha_l,$$

and the normalized displacement $(\tilde{h}^{(l)} - h^{(l)})/m_l$ is invariant.

Proof. RMS is positively homogeneous, so $r_{i,n}^{(l)} \mapsto c_l r_{i,n}^{(l)}$ and hence $s_l^{(t)} \mapsto c_l s_l^{(t)}$ by homogeneity of the median. Induct on the EMA equation 10: the base case gives $m_l^{(0)} \mapsto c_l m_l^{(0)}$ under rescaling, and $m_l^{(t)} = \beta m_l^{(t-1)} + (1-\beta) s_l^{(t)}$ preserves the factor c_l by linearity. Since $\varepsilon_l = \rho m_l$ and $\alpha_l = \rho_{\text{step}} m_l$, both scale by c_l; the ratio $(\tilde{h}^{(l)} - h^{(l)})/m_l$ is therefore unchanged. □

This is why a single ρ transfers across layers of differing scale (Figs. 9–10): its effect on each radius is relative, not absolute.

Lemma 1 (RAB scale envelope)**.** *If $m_l^{(0)} = s_l^{(0)}$ and $0 \leq \beta < 1$, then for all $t \geq 0$, $\min_{r \leq t} s_l^{(r)} \leq m_l^{(t)} \leq \max_{r \leq t} s_l^{(r)}$.*

Proof. Induction: $m_l^{(t)}$ is a convex combination of $m_l^{(t-1)}$ (which lies in the envelope of $\{s_l^{(r)}\}_{r<t}$ by hypothesis) and $s_l^{(t)}$, hence lies in the envelope of $\{s_l^{(r)}\}_{r \leq t}$. □

Consistent with the stable per-layer trajectories in Fig. 4(d).

E.3 Entropy-regularized allocation

Allocation (Sec. 3.3) and VAAL's depth attention (Sec. 3.4) are the same mechanism instantiated twice: a score vector reweighted into a softmax over a simplex, so we state it once.

Lemma 2 (Entropy-regularized simplex maximizer)**.** *For $\tau > 0$ and scores $\hat{q} \in \mathbb{R}^K$, the program $\arg\max_{a \in \mathcal{S}_K} \left[\sum_k a_k \hat{q}_k + \tau H(a) \right]$ has the unique solution $a_k = \exp(\hat{q}_k/\tau) / \sum_j \exp(\hat{q}_j/\tau)$.*

Proof. $G(a) = \sum_k a_k \hat{q}_k - \tau \sum_k a_k \log a_k$ is strictly concave on the compact convex set $\mathcal{S}_K$ (as $\tau > 0$), so it has at most one maximizer. Stationarity with multiplier ν on $\sum_k a_k = 1$ gives $\hat{q}_k - \tau(\log a_k + 1) - \nu = 0$, i.e. $a_k = C \exp(\hat{q}_k/\tau)$; normalizing fixes C and yields the softmax, which is strictly positive and hence the global maximizer. □

Corollary 2 (Allocation)**.** *For the allocation weights defined by*

$$\pi^{(t)} = \arg\max_{a \in \mathcal{S}_K} \left[\sum_{k=1}^K a_k \hat{q}_k^{(t)} + \tau_a H(a) \right], \tag{21}$$

$\pi_k^{(t)} = \exp(\hat{q}_k^{(t)}/\tau_a) / \sum_j \exp(\hat{q}_j^{(t)}/\tau_a)$ is the unique maximizer of equation 21 (Lemma 2 with $\hat{q} = \hat{q}^{(t)}$, $\tau = \tau_a$).

Corollary 3 (Depth attention). *For the depth-attention weights defined by*

$$b_i = \arg\max_{b \in \mathcal{S}_L} \left[\sum_{l=0}^{L-1} b_l d_{i,l} + \tau_d H(b) \right], \tag{22}$$

$b_{i,l} = \mathrm{softmax}_l(d_{i,\cdot}/\tau_d)$ *is the unique maximizer of equation 22 (Lemma 2 with* $K = L$, $\hat{q} = d_{i,\cdot}$, $\tau = \tau_d$*), and since* $b_i \in \mathcal{S}_L$, $\min_l d_{i,l} \le \sum_l b_{i,l} d_{i,l} \le \max_l d_{i,l}$.

The temperature (τ_a or τ_d) is the single knob interpolating between uniform effort and concentration on one path/layer, matching Fig. 2 (right) for allocation and keeping d_i^{int} a genuine summary of $\{d_{i,l}\}$ rather than a value exceeding all of them (Fig. 5) for depth attention.

Lemma 3 (Simplex preservation). *With* $a^{(0)} = K^{-1}\mathbf{1}$, $\pi^{(t)} \in \mathcal{S}_K$, $0 \le \mu < 1$, *the momentum update* $a^{(t)} = \mu a^{(t-1)} + (1-\mu)\pi^{(t)}$ *satisfies* $a^{(t)} \in \mathcal{S}_K$ *for all* t.

Proof. Immediate induction: a convex combination of two points of the convex set $\mathcal{S}_K$ stays in $\mathcal{S}_K$. □

Proposition 4 (Fixed-set feasibility). *For any finite allocation sequence and* $\boldsymbol{\delta}_i^{(0)} \in \Delta_i$, *the JA-PGD iterates equation 9 satisfy* $\boldsymbol{\delta}_i^{(t)} \in \Delta_i$ *for every* $t \ge 0$*; allocation therefore changes only step sizes, never the feasible set.*

Proof. $\mathcal{B}_x(\varepsilon_x)$ and each $\mathcal{B}_l(\varepsilon_l)$ are nonempty, closed, convex, so Euclidean projection onto them is range-preserving regardless of the pre-projection update. Hence $\delta_{x,i}^{(t+1)} = \Pi_{\mathcal{B}_x(\varepsilon_x)}[\cdot] \in \mathcal{B}_x(\varepsilon_x)$ and $\delta_i^{(l),t+1} = \Pi_{\mathcal{B}_l(\varepsilon_l)}[\cdot] \in \mathcal{B}_l(\varepsilon_l)$ for every l, which are exactly the constraints defining Δ_i. □

Together, Lemma 3 and Prop. 4 say allocation redistributes a fixed unit of effort across paths whose radii never move; the gap between allocation on/off in Table 4(c) reflects which paths are attacked harder, not what they are permitted to reach.

E.4 VAAL

Proposition 5 (VAAL weight bounds and detached gradient). *For* $\eta_u > 0$: $1 \le w_i \le 1 + 2/\eta_u$; *if* $\bar{u} > \eta_u$, *then* $\frac{1}{B}\sum_i w_i = 2 + c(a)$; *and since* w_i *is detached,* $\nabla_\theta \mathcal{L}_{\mathrm{adv}} = \frac{1}{B}\sum_i w_i \nabla_\theta D_{\mathrm{KL}}\big(p_\theta(\cdot \mid x_i) \,\|\, p_\theta(\cdot \mid x_i, \boldsymbol{\delta}_i^\star)\big)$.

Proof. $0 \le d_i^{\mathrm{out}} \le 1$ and $d_{i,l} \ge 0$ (Huber penalty), so by Corollary 3 $d_i^{\mathrm{int}} \ge 0$, giving $0 \le \bar{d}_i^{\mathrm{int}} = d_i^{\mathrm{int}}/(1 + d_i^{\mathrm{int}}) < 1$ and $0 \le u_i \le 1$. By Lemma 3, $a \in \mathcal{S}_K$, so $0 \le c(a) \le 1$ (with $c(a) := 0$ at $K = 1$). Since $\max(\bar{u}, \eta_u) \ge \eta_u$, $0 \le \hat{u}_i \le 1/\eta_u$, and hence $1 \le w_i = 1 + \hat{u}_i(1 + c(a)) \le 1 + 2/\eta_u$. If $\bar{u} > \eta_u$ the floor is inactive and $\frac{1}{B}\sum_i \hat{u}_i = 1$, so, since $c(a)$ is batch-common,

$$\frac{1}{B}\sum_i w_i = 1 + (1 + c(a)) = 2 + c(a). \tag{23}$$

Finally w_i is built under stop-gradient, so $\nabla_\theta w_i = 0$ and the product rule applied to $\mathcal{L}_{\mathrm{adv}} = \frac{1}{B}\sum_i w_i D_{\mathrm{KL}}(\cdot)$ gives the detached-gradient identity directly. □

Why not reuse $J_i(\boldsymbol{\delta}_i^\star)$ directly?

$J_i(\boldsymbol{\delta}_i^\star) = D_{\mathrm{KL}}(p_i^{\mathrm{sg}} \| \tilde{p}_i(\boldsymbol{\delta}_i^\star))$ is the attack's own optimization target and a purely output-space quantity; u_i instead draws on embedding and internal-layer drift, so it registers representational disruption at the feature paths even when the output-level KL has not yet moved, and vice versa. Reweighting by $J_i(\boldsymbol{\delta}_i^\star)$ directly would also reweight the outer consistency loss by a quantity derived from itself, since $J_i(\boldsymbol{\delta}_i^\star)$ is exactly the D_{KL} term in equation equation 17. Empirically the two remain closely tied: u_i correlates positively and significantly with $J_i(\boldsymbol{\delta}_i^\star)$ on every evaluation dataset (Pearson $r = 0.45$–0.64, all 95% bootstrap intervals excluding zero; Appendix F.12), confirming u_i tracks the attack's effect while remaining a distinct, bounded quantity.

Eq. equation 23 ties the batch-average weight directly to $c(a)$: the lower, more stable concentration under VAAL (Fig. 4(b); Fig. 5) implies a lower average weight. And because w_i is detached, enabling VAAL changes only how much consistency pressure each sample receives, not the gradient path carrying it; so the clean/robust trade-off in Fig. 4(a) is attributable to reweighting alone, not an added loss term.

E.5 Scope of the guarantees

The propositions and lemmas above establish four structural properties: RAB budgets transform consistently under positive rescaling of activations (Proposition 1); the EMA scale statistic remains bounded within its observed range (Lemma 1); allocation vectors remain valid simplex elements under momentum updates and all JA-PGD iterates remain within the declared product threat set (Lemma 3, Proposition 4); and VAAL weights are bounded and detached from the prompt gradient (Proposition 5). Global optimality of finite-step JA-PGD and the relative merits of alternating versus sequential update schedules under equal iteration budgets are not covered here; both are evaluated empirically via matched fixed-budget attacks, multiple restarts, and the schedule ablation in Table 4(a).

F Additional ablation

F.1 Qualitative Case Study: Drift and Attribution Analysis

What does JPFAT's internal robustness look like beyond the numbers in Table 2? To illustrate, we revisit the Oxford Pets example introduced in Figure 3 (target class *abyssinian*). Figure 5 compares RAB-normalized internal drift against allocation-weighted joint vulnerability for FAP (top) and JPFAT (bottom); *patch* is the Huber-penalized displacement on spatial patch tokens, and *internal* is the full-token quantity $d_{i,l}$ (equation 16) that VAAL aggregates into d_i^{int}.

JPFAT cuts per-layer drift by 46–72% relative to FAP, and the reduction grows with depth (patch: 46.4/61.8/71.9%; internal: 46.8/62.5/71.7% at Layers 1–3), consistent with RAB calibrating radii to layerwise activation scale rather than a fixed budget throughout. At convergence, the rightmost panel shows JPFAT lowering the inner KL objective $J_i(\delta^\star)$ (equation 8) by 66.4% ($0.7694 \rightarrow 0.2585$) and allocation concentration $c(a)$ (equation 20) by 42.6% ($0.061 \rightarrow 0.035$): the adversary disturbs JPFAT's representations less to begin with, and when it does succeed, it spreads its effort more evenly across paths rather than exploiting one dominant vulnerability, exactly the behavior VAAL is designed to induce (Section 3.4).

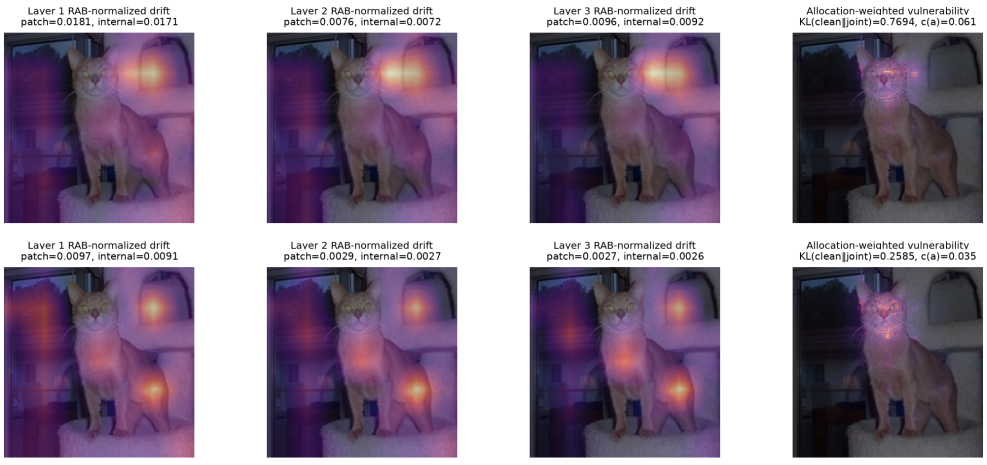


Figure 5: RAB-normalized internal drift and allocation-weighted joint vulnerability, FAP (top) vs. JPFAT (bottom).

Which attack paths actually drive the joint adversary here? Figure 6 shows gradient×perturbation contribution maps for the pixel path and each feature layer, alongside the

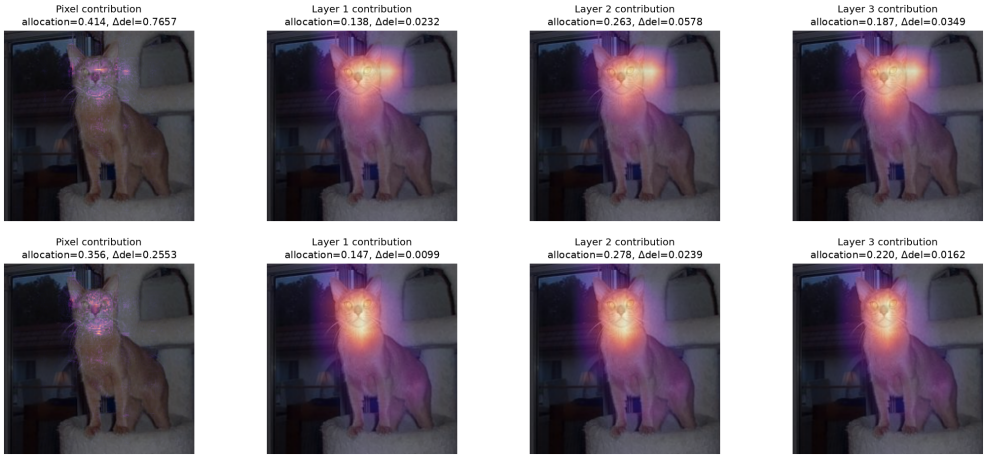


Figure 6: Path-attribution analysis for the same sample, FAP (top) vs. JPFAT (bottom).

realized allocation a_k and the leave-one-path-out deletion necessity Δ_{del}: the drop in attack KL when only that path is zeroed at the JA-PGD solution, holding the others fixed.

Under JPFAT, Δ_{del} is 53.6–66.7% lower at every path than under FAP (pixel 66.7%; Layers 1–3: 57.3/58.7/53.6%), so no path remains as decisive as it is for FAP. The allocation shifts accordingly: JPFAT assigns 14.0% less effort to the pixel path but 5.7–17.6% more to each feature layer, redistributing pressure toward a more balanced joint exploit. This is the same shift already visible in the lower $c(a)$ of Figure 5, and it matches the design intent of the allocation mechanism (Section 3.3).

F.2 Few-Shot Scaling Across All Datasets

Figure 7 breaks the averaged few-shot curves of Figure 2 down across all 11 evaluation datasets. Clean accuracy tracks FAP closely throughout, consistent with the clean–robust trade-off in the main results (Section 4.2), while for joint JA-PGD-F accuracy JPFAT is typically the only method that rises above the near-zero floor as shots increase, a pattern that holds across most datasets rather than a few easy cases.

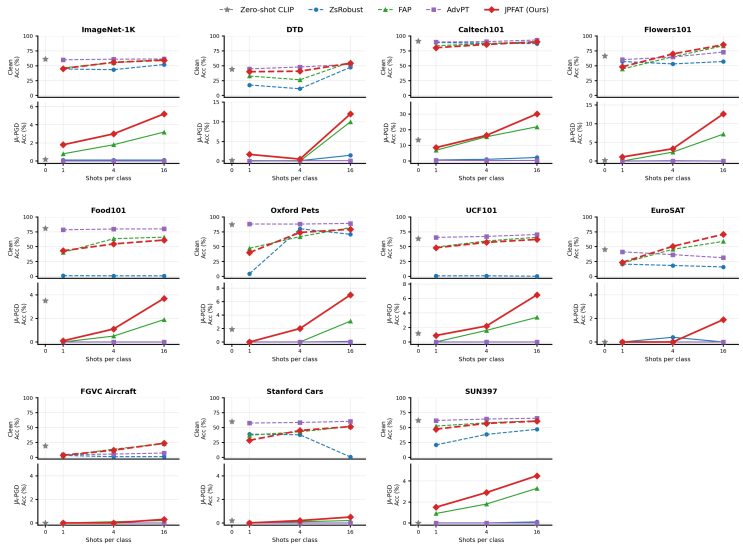


Figure 7: Clean and joint (JA-PGD) accuracy vs. shots per class, per dataset.

F.3 Full Per-Dataset Robustness Across All Attack Settings

Tables 2 and 3 summarize clean, joint, single-path, and sequential robustness averaged across datasets. Table 7 breaks the clean, joint, and single-path (pixel and feature) numbers out per dataset, with base (B), novel (N), and harmonic mean (HM) accuracy for each setting, one dataset block stacked after another; Table 8 gives the corresponding per-dataset breakdown for sequential attacks. In both, JPFAT's row is shaded to make it easy to track across columns, and the best value in each column is in **bold**.

Table 7: Robustness under Clean, JA-PGD, Pixel-space PGD, and Feature-space PGD evaluation, per dataset. B = base, N = novel, HM = harmonic mean. Best value per column in **bold**.

	Clean			JA-PGD			Pixel			Feature		
Method	B	N	HM	B	N	HM	B	N	HM	B	N	HM
ImageNet-1k												
ZsRobust	52.0±2.3	43.4±4.5	47.3	0.2±0.1	0.2±0.1	0.2	11.2±1.6	11.1±1.4	11.1	5.1±0.1	4.4±0.6	4.7
AdvPT	**60.5±0.3**	**49.2±5.4**	**54.2**	0.0±0.0	0.0±0.0	0.0	2.1±0.1	2.0±0.1	2.0	1.1±0.1	0.9±0.1	1.0
FAP	56.1±0.6	46.0±1.6	50.6	3.1±0.1	1.9±0.6	2.3	24.6±0.7	20.1±1.1	22.1	15.8±0.3	11.5±0.5	13.3
JPFAT (ours)	56.0±1.4	44.7±3.2	49.7	**4.5±0.0**	**3.0±0.0**	**3.6**	**26.8±0.8**	**20.6±1.0**	**23.3**	**19.4±1.2**	**13.1±0.5**	**15.6**
DTD												
ZsRobust	21.1±6.4	11.5±5.1	14.9	1.4±0.5	0.5±0.4	0.7	9.1±1.6	5.3±2.4	6.7	4.0±2.5	1.3±0.4	2.0
AdvPT	55.3±0.8	**44.5±0.5**	**49.3**	0.1±0.1	0.1±0.1	0.1	9.7±0.1	6.4±1.4	7.7	2.6±0.1	1.4±0.6	1.8
FAP	**68.3±2.0**	32.5±4.2	44.1	15.4±0.0	0.0±0.0	0.0	42.6±1.7	19.1±2.3	26.4	33.1±2.8	0.0±0.0	0.0
JPFAT (ours)	65.5±0.8	30.6±1.9	41.8	**18.9±0.2**	**5.6±0.8**	**8.6**	**44.9±0.9**	**21.0±2.8**	**28.6**	**36.2±0.8**	**10.3±2.0**	**16.0**
Caltech101												
ZsRobust	93.5±1.3	**86.2±2.0**	**89.7**	3.2±0.2	2.3±0.2	2.7	43.4±0.4	32.5±0.2	37.2	26.0±0.7	17.0±0.2	20.5
AdvPT	**95.2±0.3**	80.6±0.6	87.3	0.1±0.1	1.2±0.1	0.3	24.0±0.0	21.9±6.8	22.9	8.1±0.8	6.7±7.2	7.3
FAP	94.3±0.1	81.4±3.0	87.4	25.6±0.9	0.0±0.0	0.0	76.1±0.8	**56.9±2.6**	**65.1**	59.2±3.0	0.1±0.1	0.2
JPFAT (ours)	93.0±0.5	76.0±1.6	83.6	**36.2±1.4**	**17.8±0.1**	**23.8**	**77.8±0.6**	53.7±0.1	63.5	**64.7±0.7**	**40.1±1.3**	**49.5**
Flowers102												
ZsRobust	55.0±3.4	49.6±8.9	52.1	0.7±0.4	0.0±0.0	0.0	21.5±5.7	15.2±1.0	17.8	9.8±5.2	6.1±0.9	7.5
AdvPT	72.6±1.1	**58.5±2.1**	**64.8**	0.0±0.0	0.0±0.0	0.0	4.2±0.6	1.6±0.3	2.3	0.6±0.1	0.3±0.1	0.4
FAP	88.2±2.1	44.7±0.4	59.3	11.7±1.1	0.0±0.0	0.0	65.8±0.7	20.7±0.7	31.5	38.6±0.8	0.0±0.0	0.0
JPFAT (ours)	**89.8±1.7**	40.8±0.4	56.1	**21.2±2.1**	**2.0±0.1**	**3.7**	**73.5±2.5**	**21.4±1.8**	**33.1**	**50.5±0.1**	**8.2±0.1**	**14.0**
Food101												
ZsRobust	2.4±0.1	2.0±0.0	2.2	0.7±1.0	0.0±0.0	0.0	1.2±0.8	1.1±1.2	1.1	0.0±0.0	0.1±0.1	0.0
AdvPT	**77.3±0.4**	62.6±2.1	**69.2**	0.0±0.0	0.0±0.0	0.0	0.7±0.1	1.1±0.1	0.9	0.2±0.0	0.3±0.1	0.3
FAP	69.8±0.9	**66.0±1.6**	67.9	2.5±0.4	0.0±0.0	0.0	30.1±0.2	**26.2±1.5**	**28.1**	18.0±0.1	0.0±0.0	0.0
JPFAT (ours)	63.8±2.7	58.5±1.8	61.1	**4.7±0.0**	**2.8±1.2**	**3.5**	**31.4±2.6**	25.2±3.9	28.0	**20.2±1.0**	**14.2±3.5**	**16.7**
Oxford Pets												
ZsRobust	87.4±1.7	**82.0±2.0**	**84.6**	0.0±0.0	0.0±0.0	0.0	22.5±2.6	13.7±2.4	17.0	22.1±5.3	12.5±3.1	16.0
AdvPT	**89.2±0.4**	76.6±0.4	82.4	0.0±0.0	0.0±0.0	0.0	1.6±0.4	3.1±0.4	2.2	2.4±0.4	4.0±0.6	3.0
FAP	87.0±2.8	58.9±15.9	70.2	0.4±0.6	0.0±0.0	0.0	29.1±7.4	17.6±2.1	22.0	20.6±4.7	0.0±0.0	0.0
JPFAT (ours)	83.1±0.6	72.1±6.8	77.2	**6.6±1.7**	**6.4±3.5**	**6.5**	**49.8±3.3**	**39.6±10.8**	**44.1**	**38.5±1.4**	**32.0±9.1**	**35.0**
UCF101												
ZsRobust	2.3±0.3	3.0±1.8	2.6	0.1±0.1	0.0±0.0	0.0	1.8±0.5	0.1±0.1	0.1	0.4±0.6	0.0±0.0	0.0
AdvPT	69.3±1.5	**58.9±2.7**	**63.7**	0.0±0.0	0.0±0.0	0.0	2.5±0.4	1.8±0.2	2.0	0.3±0.1	0.6±0.7	0.4
FAP	**72.0±0.4**	48.5±4.3	57.9	4.4±0.8	0.0±0.0	0.0	37.9±0.3	**20.9±2.2**	**27.0**	22.9±1.8	0.0±0.0	0.0
JPFAT (ours)	70.5±1.2	42.5±1.3	53.0	**8.1±1.1**	**2.9±0.3**	**4.3**	**41.6±0.9**	18.1±1.1	25.2	**28.1±0.4**	**11.4±0.3**	**16.2**
EuroSAT												
ZsRobust	55.8±4.1	20.5±1.8	30.0	**0.7±0.9**	**0.5±0.6**	**0.5**	**33.6±3.2**	**8.7±0.2**	**13.8**	22.6±10.5	4.6±5.8	7.7
AdvPT	61.4±1.0	43.5±0.8	51.0	0.0±0.0	0.0±0.0	0.0	0.1±0.1	0.2±0.1	0.1	0.0±0.0	0.1±0.1	0.0
FAP	79.5±18.0	43.2±6.3	56.0	0.0±0.0	0.0±0.0	0.0	0.6±0.5	0.1±0.1	0.2	9.0±12.7	0.0±0.0	0.0
JPFAT (ours)	**86.2±0.2**	**50.6±10.5**	**63.8**	0.1±0.0	0.0±0.0	0.0	26.2±1.7	3.0±4.3	5.5	**29.7±1.7**	**7.4±10.2**	**11.8**
FGVC Aircraft												
ZsRobust	2.0±0.0	2.0±0.0	2.0	0.1±0.1	0.0±0.0	0.0	1.5±0.4	2.0±0.0	1.7	1.0±1.4	1.0±1.4	1.0
AdvPT	3.1±0.4	3.5±1.1	3.3	0.0±0.0	0.0±0.0	0.0	0.2±0.1	0.9±0.5	0.3	0.1±0.1	0.9±0.3	0.1
FAP	22.9±0.9	14.7±1.4	17.9	0.4±0.3	0.0±0.0	0.0	9.2±0.8	4.2±0.3	5.8	3.0±0.3	0.1±0.1	0.2
JPFAT (ours)	**24.4±0.5**	**15.2±2.8**	**18.8**	**0.7±0.1**	**0.1±0.1**	**0.2**	**9.6±0.3**	**4.8±0.6**	**6.4**	**3.2±0.2**	**1.1±0.1**	**1.6**
Stanford Cars												
ZsRobust	32.3±15.9	31.7±15.2	32.0	0.5±0.6	0.0±0.0	0.0	2.1±2.5	2.6±3.3	2.3	0.9±0.1	0.5±0.6	0.6
AdvPT	53.8±0.4	**50.1±1.1**	**51.9**	0.0±0.0	0.0±0.0	0.0	0.6±0.1	0.3±0.1	0.4	0.1±0.0	0.1±0.1	0.1
FAP	54.0±2.5	42.2±0.4	47.4	0.3±0.1	0.0±0.0	0.0	20.2±1.0	**14.2±1.1**	**16.7**	4.0±0.4	0.0±0.0	0.0
JPFAT (ours)	**55.5±1.2**	38.4±3.0	45.4	**1.0±0.6**	**0.2±0.1**	**0.4**	**23.1±1.4**	12.9±2.8	16.6	**6.5±0.5**	**3.1±0.6**	**4.2**
SUN397												
ZsRobust	53.7±3.4	52.9±3.3	53.3	0.0±0.0	0.0±0.0	0.0	16.7±1.8	17.9±3.3	17.3	0.1±0.1	0.3±0.1	0.2
AdvPT	67.0±0.2	56.8±2.1	61.5	0.0±0.0	0.0±0.0	0.0	1.9±0.1	2.1±0.1	2.0	0.8±0.3	1.1±0.4	0.9
FAP	**68.2±1.2**	**61.0±1.2**	**64.4**	4.1±0.1	1.9±2.6	2.5	**35.5±0.5**	**30.9±0.8**	**33.1**	21.7±0.0	9.6±13.5	13.3
JPFAT (ours)	66.5±0.5	56.9±3.5	61.3	**5.8±0.4**	**4.8±0.2**	**5.2**	35.0±4.0	29.1±0.8	31.8	**23.4±3.3**	**19.6±0.6**	**21.3**

F.4 AutoAttack Evaluation

We additionally evaluate with the stronger, parameter-free AutoAttack suite (Croce & Hein, 2020), used only at test time as an external, non-adaptive check on pixel-space robustness; training itself never sees AutoAttack and uses only JA-PGD-A (Section 3.5). This isolates whether the pixel path specifically remains robust after training against the joint, dual pixel–feature attack, rather than only against a pixel-only adversary, and whether that pixel-space robustness is an artifact of obfuscated gradients. Since adversarial prompt tuning does not rely on gradient masking, we use the two APGD variants, APGD-CE and APGD-DLR, at the same pixel radius $\varepsilon_x = 1/255$ used throughout training

Table 8: Robustness under Sequential Dual PGD evaluation, per dataset. B = base, N = novel, HM = harmonic mean. Best value per column in **bold**

(a)

Method	Sequential		
	B	N	HM
ImageNet-1k			
ZsRobust	0.2±0.1	0.2±0.1	0.2
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	4.1±0.0	2.7±0.7	3.3
JPFAT (ours)	**5.9±0.5**	**3.9±0.1**	**4.7**
DTD			
ZsRobust	1.6±0.8	0.5±0.1	0.7
AdvPT	0.0±0.0	0.2±0.3	0.0
FAP	16.8±0.1	0.0±0.0	0.0
JPFAT (ours)	**20.1±0.4**	**6.0±0.9**	**9.2**
Caltech101			
ZsRobust	7.7±1.6	5.5±0.2	6.4
AdvPT	0.8±0.4	1.9±0.1	1.1
FAP	32.1±1.1	0.0±0.0	0.0
JPFAT (ours)	**40.6±1.1**	**20.6±1.6**	**27.4**
Flowers102			
ZsRobust	1.5±1.2	0.3±0.0	0.5
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	13.2±1.2	0.0±0.0	0.0
JPFAT (ours)	**23.2±2.3**	**2.5±0.2**	**4.4**
Food101			
ZsRobust	0.7±0.9	0.0±0.0	0.0
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	3.3±0.3	0.0±0.0	0.0
JPFAT (ours)	**5.3±0.1**	**3.3±1.4**	**4.1**

(b)

Method	Sequential		
	B	N	HM
Oxford Pets			
ZsRobust	0.1±0.2	0.0±0.0	0.0
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	0.8±0.9	0.0±0.0	0.0
JPFAT (ours)	**8.5±1.7**	**8.4±5.0**	**8.5**
UCF101			
ZsRobust	0.1±0.1	0.0±0.0	0.0
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	6.5±1.1	0.0±0.0	0.0
JPFAT (ours)	**10.4±0.6**	**3.6±0.3**	**5.4**
EuroSAT			
ZsRobust	**1.6±2.2**	**0.1±0.1**	**0.2**
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	0.0±0.0	0.0±0.0	0.0
JPFAT (ours)	0.5±0.2	0.0±0.0	0.0
FGVC Aircraft			
ZsRobust	0.5±0.7	**1.0±1.4**	**0.6**
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	0.3±0.4	0.0±0.0	0.0
JPFAT (ours)	**0.6±0.0**	0.2±0.1	0.3
Stanford Cars			
ZsRobust	0.5±0.6	0.0±0.0	0.0
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	0.5±0.1	0.0±0.0	0.0
JPFAT (ours)	**1.2±0.6**	**0.4±0.1**	**0.6**
SUN397			
ZsRobust	0.1±0.1	0.1±0.0	0.1
AdvPT	0.0±0.0	0.0±0.0	0.0
FAP	5.8±0.1	2.5±3.5	3.4
JPFAT (ours)	**6.8±0.8**	**5.9±0.1**	**6.3**

and evaluation, applied to the same 16-shot prompts trained under the standard few-shot protocol (Section 4.1). Unlike JA-PGD-F's fixed uniform step, AutoAttack uses a fractional attack generator that explores the perturbation budget by automatically adjusting its step size α over the course of the attack, giving a step-size-adaptive check alongside our fixed-budget reporting adversary.

JPFAT achieves the strongest robust accuracy among all four methods on nearly every dataset, improving average AutoAttack robust accuracy over FAP from 27.0% to 29.8%, confirming that pixel-space robustness is preserved, not traded away, when the pixel path is trained jointly with internal feature-layer perturbations. ZsRobust and AdvPT collapse to near-zero accuracy across almost all datasets, with AdvPT nonzero only on DTD (0.06%) and Caltech101 (0.53%).

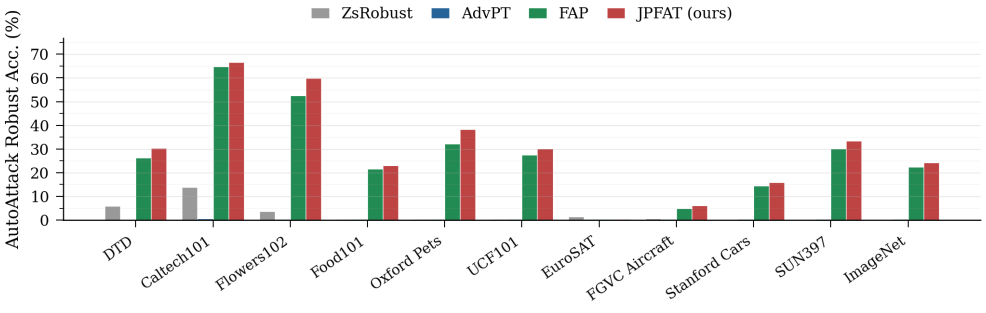


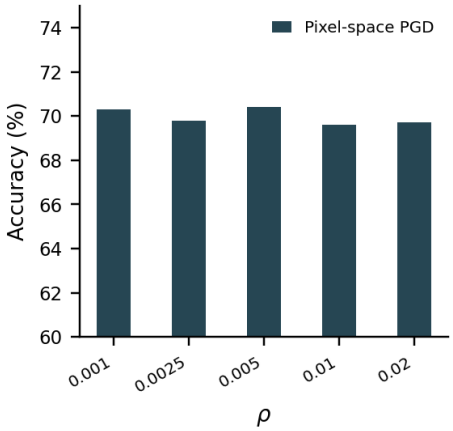
Figure 8: AutoAttack (APGD-CE + APGD-DLR, $\varepsilon_x = 1/255$) robust accuracy per dataset.

F.5 RAB Scale Sensitivity (ρ)

Figure 9 sweeps the RAB scale $\rho \in \{0.001, 0.0025, 0.005, 0.01, 0.02\}$ on Caltech-101 (16-shot), with $\rho_{\text{step}} = \rho/5$ throughout. Pixel-space PGD accuracy stays flat (69.6–70.4%), confirming ρ controls only the feature-path budgets $\varepsilon_l = \rho m_l$, while feature-space PGD and JA-PGD accuracy both increase monotonically with ρ (feature: 52.2%→62.5%; JA-PGD: 23.0%→31.5%). Figure 10

shows why: at $\rho = 0.001$ the pixel path retains nearly all effort ($a_x \approx 0.97$), effectively reducing JPFAT to a pixel-only attack, whereas larger ρ progressively redistributes effort to the feature layers. At the default $\rho = 0.005$, allocation reaches its most balanced distribution across all four paths ($a_x \approx 0.31$, Layers 1–3 absorbing the remainder) with the lowest allocation concentration $c(a)$, consistent with VAAL's goal of closing off single-path exploits; larger ρ shifts allocation further toward the feature layers at the cost of under-utilizing the pixel path, and yields further robustness gains at a modest clean-accuracy cost, consistent with the standard robustness–accuracy trade-off.

(a) ρ sensitivity effect on Pixel-space PGD.



(b) ρ sensitivity effect on Feature-space PGD and JA-PGD.

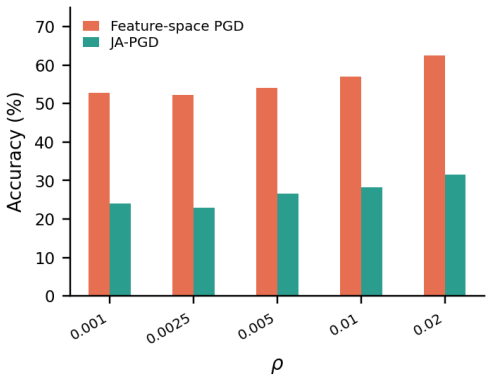


Figure 9: RAB scale sensitivity, Caltech-101 (16-shot), $\rho_{\text{step}} = \rho/5$. (a) Pixel-space PGD is insensitive to ρ. (b) Feature-space PGD and JA-PGD improve monotonically, confirming larger budgets yield stronger feature-path robustness.

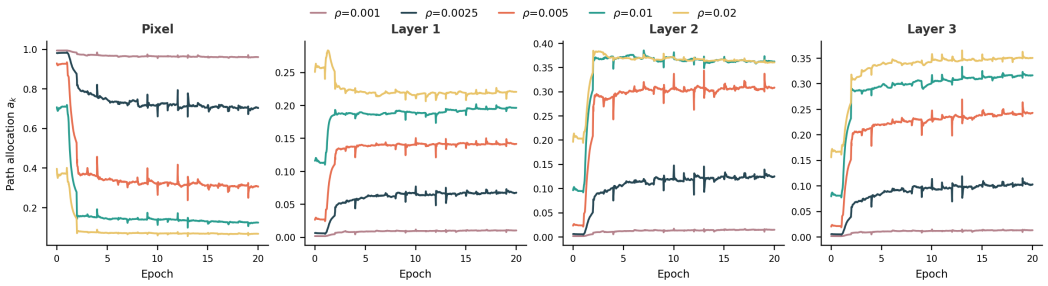


Figure 10: Realized path allocation a_k over training across all five ρ values (Caltech-101, 16-shot), $\rho_{\text{step}} = \rho/5$. Each panel shows one attack path: pixel (left) and Layers 1–3 (right). Larger ρ progressively redistributes effort from the pixel path to the feature layers; $\rho = 0.005$ yields the most balanced allocation across all paths.

F.6 Allocation Dynamics Across Datasets

Figure 11 extends the single-dataset allocation trajectory of Figure 2 (right) to four additional datasets, each showing the realized allocation a_k (Eq. equation 13) for the pixel path and the three attacked feature layers over the full training run. On all four datasets, training begins with effort concentrated almost entirely on the pixel path, consistent with its larger initial radius-weighted score, and is progressively redistributed toward the feature layers as training proceeds. By the end of training, allocation settles into a broadly comparable split across all four paths rather than remaining pixel-dominant, though the exact converged values and the point at which redistribution begins differ by dataset. This is consistent with allocation adapting to per-dataset gradient signal rather than following a fixed, dataset-independent schedule, and complements the RAB scale diagnostic of Figure 4 and the ρ-sensitivity sweep of Figure 9, both reported on a single dataset.

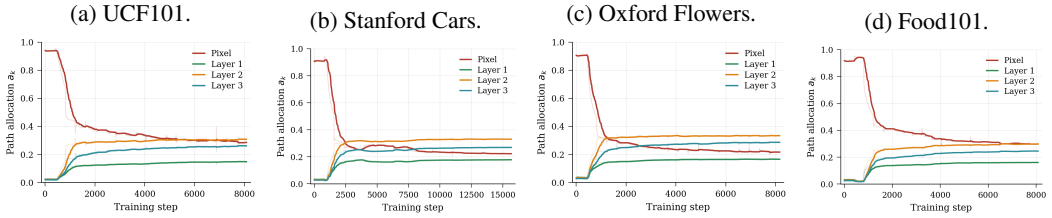


Figure 11: Realized path allocation a_k over training. On each of the four datasets shown, effort starts concentrated on the pixel path and is progressively redistributed toward the three feature layers as training proceeds, converging to a broadly balanced allocation across all four paths by the end of training.

F.7 Per-Dataset RAB, Allocation, and VAAL Ablation

On Oxford Pets, Table 4(c) and Figure 4(a–b) isolate the individual contributions of RAB, gradient-effort allocation, and VAAL. Table 9 extends this ablation to all 11 datasets, replacing each mechanism in turn with its uniform counterpart: RAB with a shared, non-layer-calibrated budget; allocation with equal effort across paths; and VAAL with uniform sample weighting, $w_i \equiv 1$.

JPFAT achieves the best average pixel-PGD and joint JA-PGD-F accuracy across datasets (36.7% and 7.9%, respectively), and the best joint accuracy on 8 of 11 individual datasets. Removing VAAL causes the largest average drop in joint accuracy (7.9%→5.2%), consistent with the ablation in Figure 4(a).

RAB and allocation contribute a smaller, more dataset-dependent share: on a few datasets (Caltech101, UCF101, Stanford Cars), uniform RAB or uniform allocation actually beats JPFAT's pixel-PGD accuracy, and uniform RAB matches or exceeds JPFAT's joint accuracy on EuroSAT and Stanford Cars. Removing either still lowers the 11-dataset average, so their benefit holds overall even though it isn't uniform dataset-by-dataset; VAAL remains the most consistently beneficial of the three mechanisms.

Table 9: **Per-dataset RAB, allocation, and VAAL ablation.** Fixed-budget pixel-PGD and joint JA-PGD-F accuracy (%) with RAB, allocation, and VAAL individually replaced by a uniform rule, versus the full JPFAT model (**ours**). Best value per metric per dataset in **bold**.

Method	Metric	DTD	Caltech101	Flowers102	Food101	Oxford Pets	UCF101	EuroSAT	Aircraft	Stanford Cars	SUN397	ImageNet	Avg.
Uniform RAB	Pixel	33.9	**72.6**	62.6	28.0	43.7	**36.9**	31.5	8.1	**22.2**	27.5	24.2	35.6
	Joint	10.6	28.4	11.9	3.0	5.0	5.2	**5.8**	**0.4**	**0.6**	4.3	4.5	7.2
Uniform allocation	Pixel	34.2	69.5	61.3	28.5	42.3	34.5	23.4	8.5	21.3	25.2	20.2	33.5
	Joint	11.1	24.5	**13.8**	2.8	5.4	5.3	1.4	0.2	0.5	4.2	3.4	6.6
VAAL off	Pixel	32.9	68.6	59.0	26.6	40.7	32.9	37.9	7.3	18.0	24.6	20.0	33.5
	Joint	9.4	23.1	7.2	1.5	3.3	3.3	3.5	0.1	0.1	2.4	2.8	5.2
JPFAT (ours)	Pixel	**35.2**	72.2	**63.3**	**29.3**	**45.1**	34.4	**39.9**	**9.8**	22.0	**27.8**	**24.6**	**36.7**
	Joint	**12.0**	**30.1**	12.6	**3.7**	**6.2**	**6.5**	4.9	0.3	0.5	**5.0**	**5.1**	**7.9**

F.8 Computational Overhead

Table 10 reports per-step cost at matched attack iterations (T=5). All methods share nearly identical clean-forward GFLOPs (≈224), so none add inference cost over standard CLIP. At training time, JPFAT uses 36% fewer GFLOPs than FAP (3450 vs. 5383) despite attacking more paths jointly, since JA-PGD computes its detached clean target once per step rather than recomputing clean logits and backpropagating within every PGD iteration as FAP does. AdvPT's low cost (17.23 GFLOPs/step) reflects its pixel-only attack bypassing the encoder's backward pass entirely, while ZsRobust falls between the two, requiring encoder backpropagation but without JPFAT's joint allocation overhead.

Table 10: **Computational cost.** Clean GFLOPs measure one clean forward pass including text encoding for all classes. Training-step GFLOPs include full adversarial generation at $T=5$. GFLOPs/attack step measures the marginal cost of adding one additional attack iteration.

Method	Clean GFLOPs	Training-step GFLOPs	GFLOPs/attack step
ZsRobust ICLR'23	224.87	1405.31	234.22
AdvPT ECCV'24	224.02	534.25	17.23
FAP NeurIPS'24	224.37	5383.01	897.17
JPFAT (Ours)	224.37	**3449.99**	465.69

F.9 Generalization to a Deeper Evaluation Attack

JPFAT trains and is evaluated at the same attack depth, $L=3$ (Section 4.1). Figure 4(c) already varies attack depth with $L_{\text{eval}}=L_{\text{train}}$, showing that feature-only and joint robustness fall as more layers are attacked; however, that sweep retrains a separate model at each depth, so it does not test whether a single deployed model generalizes to a stronger attacker at test time. Since the shared-encoder/adapter threat model motivating the feature paths (Section 2) implies the defender may not know the attacker's true access depth in advance, a stronger test is whether robustness holds when the test-time attacker perturbs *more* layers than the defender trained against, i.e. $L_{\text{eval}}=5$ while $L_{\text{train}}=3$. Table 11 compares this mismatched setting against a model trained directly at $L_{\text{train}}=5$ (matched to L_{eval}), across three datasets. The $L_{\text{train}}=3$ model tracks the matched-depth model within 3.3 points on Pixel-space and Feature-space PGD, and within 0.2 points on JA-PGD, indicating that JPFAT's robustness generalizes to attacked layers beyond those seen during training rather than overfitting to $L_{\text{train}}=3$.

Table 11: **Robustness under a deeper test-time attack.** $L_{\text{eval}}=5$ vs. training at $L_{\text{train}}=3$, compared against a model trained and evaluated at the matched depth ($L_{\text{train}}=5$). Better value per row in **bold**.

Dataset	Attack	$L_{\text{train}}=3,\ L_{\text{eval}}=5$	$L_{\text{train}}=5,\ L_{\text{eval}}=5$
Caltech-101	Pixel-space PGD	**70.5**	69.5
	Feature-space PGD	**19.1**	18.7
	JA-PGD	5.8	**6.0**
DTD	Pixel-space PGD	33.3	**34.7**
	Feature-space PGD	8.4	**8.9**
	JA-PGD	3.4	3.4
Oxford Pets	Pixel-space PGD	**45.1**	41.8
	Feature-space PGD	**4.6**	4.0
	JA-PGD	0.2	0.2

Table 12: **Robustness under an adaptive, mechanism-aware attack.** JPFAT on three datasets (clean accuracy): DTD (54.7%), Caltech101 (88.84%), and Oxford Pets (80.49%), under increasing attacker strength and mechanism-awareness. "RAB-aware" means the attacker adopts JPFAT's own per-layer budget normalization when allocating perturbation. Worst-case union reports the lowest robust accuracy across configurations for each dataset.

Dataset	Configuration	Steps	Restarts	RAB-aware	Robust Acc. (%)
DTD	Reported (JA-PGD, $L=3$)	100	1	✗	11.29
	Strong (budget only)	200	5	✗	11.29
	Strong + RAB-aware	200	5	✓	10.64
	Worst-case union		–		**10.64**
Caltech101	Reported (JA-PGD, $L=3$)	100	1	✗	26.61
	Strong (budget only)	200	5	✗	26.53
	Strong + RAB-aware	200	5	✓	25.03
	Worst-case union		–		**24.83**
Oxford Pets	Reported (JA-PGD, $L=3$)	100	1	✗	5.31
	Strong (budget only)	200	5	✗	5.21
	Strong + RAB-aware	200	5	✓	4.66
	Worst-case union		–		**4.63**

F.10 Adaptive Attack Against JPFAT's Internal Mechanism

The AutoAttack evaluation in Appendix F.4 is pixel-only and attacker-agnostic: it has no knowledge of JPFAT's Relative Activation Budget (RAB) mechanism or its per-layer allocation rule. To check whether our reported joint robustness reflects genuine robustness rather than a comparatively weak fixed evaluation attack, we instead evaluate JPFAT against an attacker that is both stronger and mechanism-aware: it gets substantially more optimization budget, and it adopts JPFAT's own RAB-style per-layer normalization when allocating perturbation across the pixel and feature paths, rather than splitting budget by a fixed or attack-side heuristic. We compare three configurations, the standard reporting protocol, a strong variant that increases optimization effort alone, and a strong variant that is additionally RAB-aware, and report the worst case across all three. Table 12 summarizes the results on DTD, Caltech101, and Oxford Pets; the pattern is consistent across all three: the RAB-aware attacker gains modestly over the budget-only attacker, and the worst-case union is at most about half a point below the strongest individual configuration.

F.11 Sensitivity to JA-PGD's Pixel–Feature Leader Order

JA-PGD is already alternating in the sense of equation 9: each iteration steps the pixel and feature paths sequentially rather than simultaneously, since a simultaneous step would use feature gradients that go stale as soon as the pixels move. All reported results fix one such order, pixel leads, then features update on the newly perturbed image. To isolate whether this fixed leader order biases the pixel-vs-feature-vs-joint split, we test three alternatives on the same initialization, iteration budget, and attack depth: *feature-first* (reversed), *leader-swap* (the leader alternates every iteration), and *random-leader* (drawn uniformly per iteration). Table 13 shows robust accuracy varies by at most 0.1 point across schedules, and only 2–4 examples per dataset ever change outcome relative to the reported order. The reported pixel-leads order is therefore not a meaningful source of bias.

Table 13: **JA-PGD is insensitive to which path leads.** Robust accuracy under four leader schedules for the pixel–feature update in equation 9, all sharing the same initialization, iteration budget, and attack depth. *Leader-swap* alternates which path updates first each iteration; *random-leader* draws it uniformly at random. Disagreement is the share of test examples whose robust outcome differs from the reported pixel-first schedule.

Dataset	Leader schedule	Robust Acc. (%)	Disagr. vs. Pixel-first (%)
DTD	Pixel-first (reported)	11.29	–
	Feature-first	11.29	0.00
	Leader-swap	11.29	0.00
	Random-leader	11.29	0.00
Caltech101	Pixel-first (reported)	26.57	–
	Feature-first	26.57	0.16
	Leader-swap	26.65	0.16
	Random-leader	26.57	0.08
Oxford Pets	Pixel-first (reported)	5.29	–
	Feature-first	5.31	0.08
	Leader-swap	5.21	0.08
	Random-leader	5.26	0.08

F.12 Correlation Between VAAL's Vulnerability Score and the Attack Objective

Appendix E.4 argues that VAAL's vulnerability score u_i is a useful, non-redundant proxy for the attack's own objective $J_i(\boldsymbol{\delta}_i^\star)$ rather than a direct restatement of it. We test this empirically: for each dataset, we compute u_i and $J_i(\boldsymbol{\delta}_i^\star)$ per test example under the reported attack and report their correlation, with 95% bootstrap confidence intervals ($2{,}000$ resamples). Table 14 shows u_i is positively and significantly correlated with $J_i(\boldsymbol{\delta}_i^\star)$ on every dataset (all intervals exclude zero), confirming that higher-weighted examples are indeed the ones the attack moves more.

Table 14: Correlation between VAAL's vulnerability score u_i and the attack's own objective $J_i(\boldsymbol{\delta}_i^\star)$, with 95% bootstrap confidence intervals.

Dataset	Pearson r (95% CI)	Spearman ρ (95% CI)
DTD	0.45 (0.40, 0.51)	0.47 (0.43, 0.51)
Caltech101	0.64 (0.61, 0.67)	0.62 (0.59, 0.64)
Oxford Pets	0.64 (0.61, 0.67)	0.61 (0.59, 0.63)

F.13 Generalization to the ViT-B/16 Backbone

All main results use ViT-B/32. Table 15 checks whether JPFAT's robustness gains transfer to a different backbone, ViT-B/16, on three datasets against FAP. The joint (JA-PGD) and single-path advantages seen with ViT-B/32 hold here too, at some cost to novel-class clean accuracy on Caltech101 and DTD.

Table 15: Robustness on the ViT-B/16 backbone. B = base, N = novel, HM = harmonic mean. Best value per column in **bold**; JPFAT's row is shaded.

	Clean			JA-PGD			Pixel			Feature		
Method	B	N	HM	B	N	HM	B	N	HM	B	N	HM
Caltech101												
FAP	**95.8**	**82.8**	**88.8**	9.4	**5.6**	7.0	73.2	**53.6**	**61.9**	41.4	**22.1**	28.8
JPFAT (ours)	93.7	78.7	85.5	**16.3**	5.5	**8.2**	**75.4**	50.3	60.3	**49.8**	21.2	**29.7**
DTD												
FAP	**74.3**	**45.8**	**56.7**	6.2	2.5	3.6	37.2	**19.4**	**25.5**	21.4	**7.6**	**11.2**
JPFAT (ours)	72.5	43.9	54.7	**9.6**	**2.7**	**4.2**	**38.5**	17.7	24.3	**22.2**	6.2	9.7
Flowers102												
FAP	87.9	**33.0**	**48.0**	2.8	0.0	0.0	60.6	9.4	16.3	22.2	1.8	3.3
JPFAT (ours)	**90.7**	31.2	46.4	**7.3**	**0.2**	**0.4**	**71.8**	**13.5**	**22.7**	**33.0**	**3.3**	**6.0**

G Limitations and Future Work

The feature paths model a white-box insider adversary with write access to intermediate activations, so joint robustness is complementary to, not a replacement for, pixel-space robustness. Our evaluation attacks are also method-agnostic, using identical fixed budgets across all compared methods for a fair comparison. We apply RAB, and the joint pixel–feature threat model more broadly, to CLIP's vision encoder alone: our measurement confirms substantial scale differences across layers but not strict monotonicity in depth, and since the prompts we tune are inserted into both encoders, this vision-only scope leaves the text encoder's activation geometry, and whether the same per-layer calibration transfers to it, unexamined even in a limited ablation. JA-PGD is also a finite-step approximation with no global-optimality guarantee. Finally, our path-attribution analysis is currently a correlational account, from allocation and leave-one-path-out deletion necessity at a single converged solution; moving it toward causal intervention that perturbs individual paths at each JA-PGD iteration, to separate paths that merely correlate with vulnerability from paths that causally mediate it, is left to future work.