REAL: ROBUSTNESS-ENHANCED ACTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Active learning saves labels by selecting a small part of an unlabeled pool. For adversarial training, a useful batch should represent both the original learning signals and how they change under input attacks. We propose REAL, which describes every candidate by its clean last-layer gradient and its attack-induced gradient change. Repeated k-means++ seeding selects representatives of these signals, and cluster-size weights preserve the number of candidates each representative replaces. We connect the same coverage potential used for selection to approximation of the pool's robust objective. Under an explicit condition linking representation distance to true-label robust loss, its square root bounds the weighted objective's error and its minimizer's excess pool risk. This realized-batch bound holds for every selection; independent restarts additionally provide a high-probability comparison with the best achievable coverage. Across CIFAR-10, CIFAR-100, and Tiny-ImageNet with ResNet-18 and WRN-28-10, REAL improves measured adversarial robustness over the evaluated baselines in most settings, while revealing clean-accuracy and seed-variability trade-offs.

1 INTRODUCTION

Active learning seeks to reduce annotation cost by selecting informative examples from a large unlabeled pool (Settles, 2009). Common acquisition criteria use predictive uncertainty, representation coverage, or gradient diversity (Lewis & Gale, 1994; Houlsby et al., 2011; Sener & Savarese, 2018; Ash et al., 2020). These criteria are usually constructed for clean prediction. When the selected labels are instead used for adversarial training, their usefulness also depends on behavior within an adversarial neighborhood.

Prior work on robust active learning shows that acquisition rules for clean accuracy are not always equally effective for adversarial robustness (Guo et al., 2021). The mismatch is not only about which individual samples are difficult; under a batch labeling budget, several high-scoring candidates may convey similar learning signals, leaving other regions of the pool poorly represented. Clean-gradient diversity addresses redundancy in the learning signal at the original input, but does not explicitly distinguish candidates whose gradients respond differently to adversarial perturbations. This motivates a batch geometry that represents both the clean learning signal and its perturbation-induced change.

We introduce REAL (Robustness-Enhanced Active Learning) for this purpose. For each candidate, REAL computes a clean gradient $g_{c,t}(x)$ and an acquisition-time adversarial gradient $g_{a,t}(x)$ using the same pseudo-label, and represents the candidate by $\phi_t(x) := [g_{c,t}(x);\, g_{a,t}(x) - g_{c,t}(x)]$. This representation separates the clean baseline from its adversarial correction in the acquisition geometry. Multi-start k-means++ then selects representatives with low average squared distance to the unlabeled pool, and each queried representative receives a cluster-size weight reflecting the pool mass it represents in the subsequent robust-training update. Under a representation-loss regularity condition, the realized coverage potential bounds both the uniform approximation error of the cluster-weighted robust objective and the excess robust risk of its minimizer on the current unlabeled pool. Combining this with the classical k-means++ approximation result (Arthur & Vassilvitskii, 2007) yields a high-probability comparison between the practical multi-start selector and the best achievable coverage under the same query budget. Across CIFAR-10, CIFAR-100, and Tiny-ImageNet with ResNet-18 and WRN-28-10, REAL improves measured adversarial robustness over the evaluated baselines in most settings, while also exhibiting clean-accuracy and seed-variability trade-offs.

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.

2 Related Work

Uncertainty- and Gradient-based Batch Acquisition. Uncertainty-based methods select examples on which the current model is least certain, using criteria such as confidence, margin, or entropy (Settles, 2009; Lewis & Gale, 1994). Bayesian approaches instead measure information about model parameters; BALD uses mutual information between predictions and parameters (Houlsby et al., 2011). For batch acquisition, uncertainty alone can select redundant examples. BADGE represents candidates by last-layer pseudo-label gradients and uses k-means++-style selection to combine gradient magnitude with batch diversity (Ash et al., 2020). These criteria capture complementary aspects of informativeness and redundancy, with acquisition signals constructed from clean inputs.

Robustness-aware and Training-aware Acquisition. Adversarial training optimizes prediction under bounded input perturbations (Madry et al., 2018), motivating acquisition criteria that account for more than clean predictive uncertainty. DFAL uses adversarial perturbations to estimate proximity to decision boundaries (Ducoffe & Precioso, 2018). DRE studies label acquisition for adversarial training and selects a batch that matches the unlabeled pool's predictive-entropy distribution (Guo et al., 2021). JIN combines prediction inconsistencies across clean and attacked inputs and across standard and robust models to improve robust minimax fairness under limited labels (Wu et al., 2023). RAISE studies robust batch acquisition for sequential uncertainty reduction in robot perception (Heo, 2026). While related in its use of robustness-aware batch selection, it addresses a robot-perception setting; REAL instead studies pool-based adversarially robust classification and connects a clean–secant gradient geometry and cluster-weighted coverage to approximation of the robust pool objective. RoAL uses elastic weight consolidation to address forgetting under changing attacks (Fajri et al., 2024), while TARL iteratively acquires challenging examples and retrains the robust model (Ghosh et al., 2025). In a related training-aware direction, SAAL selects examples using loss sharpness under parameter perturbations (Kim et al., 2023). Parameter-space sharpness and input-space adversarial sensitivity are distinct acquisition signals, even though both use perturbations to assess a candidate's learning value.

Coreset and Coverage-based Active Learning. Representative-subset methods seek a small batch that captures the unlabeled pool rather than merely selecting individually uncertain points. The core-set approach of Sener & Savarese (2018) uses coverage in a learned feature space, formulating batch selection through a k-center objective that accounts for previously labeled examples. Pinsler et al. (2019) develop a Bayesian formulation based on sparse subset approximation to the expected complete-data log posterior, accounting for unknown labels through the current predictive distribution. Related coverage methods use other notions of representativeness: TypiClust combines clustering with the selection of typical, high-density examples in low-budget regimes (Hacohen et al., 2022), whereas ProbCover seeks to cover probability mass through neighborhoods around selected examples (Yehuda et al., 2022). These approaches emphasize worst-case feature coverage, posterior approximation, or density and probability-mass coverage; they need not share the same objective-approximation guarantee.

Related but Distinct Settings. Subset selection also appears in robust training when labels are already available. Adversarial coreset selection uses gradient approximation to reduce the training set and accelerate robust optimization (Dolatabadi et al., 2023); margin-based selection has likewise been used to choose examples within adversarial training (Mirsadeghi, 2026). These methods select training data rather than which unknown labels to acquire. ADROIT instead uses adversarial representation learning within a self-supervised active-learning framework (Banerjee & Verma, 2025). Such representation-learning objectives differ from optimizing robustness to a specified set of bounded input attacks.

3 Problem Setup

We consider pool-based classification with input domain $\mathcal{X}$ and label set $\mathcal{Y}$. At round t, let S_t be the labeled set, $U_t \subseteq \mathcal{X}$ the unlabeled pool, and θ_t the current model parameters. Write $n_t = |S_t|$ and $N_t = |U_t|$. A query batch $B_t \subseteq U_t$ contains q distinct candidates, where $1 \le q \le N_t$. After obtaining their true labels y_b, we update

$$S_{t+1} = S_t \cup \{(b, y_b) : b \in B_t\}, \qquad U_{t+1} = U_t \setminus B_t. \tag{1}$$

For T feasible rounds with a fixed batch size, the additional label budget is $Q = Tq$, and the final labeled set has $n_0 + Tq$ examples, where $n_0 = |S_0|$ is the initial labeled-set size.

The downstream training objective distinguishes this setting from clean active learning. For a classification loss ℓ and a nonempty labeled set S, the empirical clean objective is

$$\mathcal{L}_{\text{clean}}(\theta; S) = \frac{1}{|S|} \sum_{(x,y)\in S} \ell(\theta; x, y). \tag{2}$$

Fix a norm $\|\cdot\|_p$ and an attack radius $\varepsilon \geq 0$. The admissible perturbations at x are $\mathcal{D}_\varepsilon(x) = \{\delta : \|\delta\|_p \leq \varepsilon,\ x + \delta \in \mathcal{X}\}$. Thus, image attacks must also remain in the valid pixel domain. Define the pointwise robust loss as $\ell_{\text{adv}}(\theta; x, y) := \max_{\delta\in\mathcal{D}_\varepsilon(x)} \ell(\theta; x + \delta, y)$. Adversarial training uses its empirical average (Madry et al., 2018):

$$\mathcal{L}_{\text{adv}}(\theta; S) = \frac{1}{|S|} \sum_{(x,y)\in S} \ell_{\text{adv}}(\theta; x, y). \tag{3}$$

The queried labels are used to reduce this robust objective, rather than only the clean objective.

To express the acquisition goal, let y_x denote the latent true label of $x \in U_t$. Define its robust loss $r_\theta(x)$ and the current-pool robust risk by

$$r_\theta(x) = \ell_{\text{adv}}(\theta; x, y_x), \qquad \mathcal{R}_{U_t}(\theta) = \frac{1}{N_t} \sum_{x\in U_t} r_\theta(x). \tag{4}$$

This full-information risk is unavailable at acquisition time because the pool labels are unknown. It provides the target that a small queried, weighted set would ideally represent.

An ideal acquisition rule would compare the models obtained after querying different batches. Let $\mathcal{T}_t$ denote a fixed training procedure that uses the current state (θ_t, S_t), a candidate batch, and its queried labels. This procedure includes its weighting rule, optimization settings, and fixed training randomness. Its output for a batch B is $\theta_{t+1}(B) = \mathcal{T}_t\big(B, \{y_b : b \in B\}\big)$. For REAL, this procedure uses the cluster-weighted objective specified in (15); the weights are computed without pool labels. An oracle with all candidate labels could then seek

$$B_t^\star \in \arg\min_{\substack{B\subseteq U_t \\ |B|=q}} \mathcal{R}_{U_t}\big(\theta_{t+1}(B)\big). \tag{5}$$

This rule is unavailable because the labels are unknown, and its direct evaluation would require combinatorial batch search and repeated robust training. We therefore use a label-free proxy for how well a batch represents the pool. The analysis will bound objective approximation under an explicit regularity condition, not the approximation ratio to this post-training oracle.

At acquisition time, the current predictive distribution $p_{\theta_t}(y \mid x)$ supplies the pseudo-label $\hat{y}_t(x) \in \arg\max_{y\in\mathcal{Y}} p_{\theta_t}(y \mid x)$. The clean and attacked acquisition quantities in Section 4 use this same pseudo-label. Their difference then reflects the input perturbation rather than a changed target label. Subsequent training uses queried true labels, and unqueried labels remain hidden. All ties are resolved by fixed rules.

4 METHOD

REAL operates on the entire unlabeled pool U_t. It first constructs a compact learning-signal representation for every candidate, then selects q representatives that cover this space, and finally weights each queried representative by the number of candidates it replaces.

4.1 GRADIENT REPRESENTATION

Features describe an input, whereas gradients also describe the model's current learning signal. To avoid storing and comparing full-network gradients for every candidate, we use the final linear layer. Let h_ϑ be the feature extractor, W the final layer with d entries, and $\theta = (W, \vartheta)$. Define

$$f_\theta(x) = W h_\vartheta(x), \tag{6a}$$

$$g_{c,t}(x) = \text{vec}(\nabla_W \ell(\theta_t; x, \hat{y}_t(x))) \in \mathbb{R}^d, \tag{6b}$$

where vec flattens a matrix. For softmax cross-entropy, with probability vector $p_t(x)$ and one-hot label vector e_y, $\nabla_W \ell(\theta_t; x, \hat{y}_t(x)) = \big(p_t(x) - e_{\hat{y}_t(x)}\big) h_{\vartheta_t}(x)^\top$. This combines features and prediction residuals and is obtained by an outer product once both are available (Ash et al., 2020). It is an exact gradient for this layer, not a complete summary of full-network learning. Attack generation and subsequent training are not restricted to this layer. Distances can be evaluated in chunks without changing the candidate pool.

4.2 Adversarial Secant Embedding

For each candidate $x \in U_t$, compute

$$\delta_t^\star(x) \in \arg\max_{\delta \in \mathcal{D}_\varepsilon(x)} \ell(\theta_t; x + \delta, \hat{y}_t(x)), \tag{7a}$$

$$g_{a,t}(x) = \mathrm{vec}(\nabla_W \ell(\theta_t; x + \delta_t^\star(x), \hat{y}_t(x))) \,. \tag{7b}$$

The attacked input is fixed when differentiating with respect to W. A feasible approximate attack may replace $\delta_t^\star(x)$ throughout; all selection results condition on the computed embeddings.

The attacked gradient alone does not distinguish whether two candidates were already similar at the clean input or became similar through different attack-induced changes. For example, changes from 1 to 3 and from 2 to 3 have identical final values but different baselines and changes. We preserve this distinction through

$$\Gamma_t(x) = g_{a,t}(x) - g_{c,t}(x), \qquad \phi_t(x) = \begin{bmatrix} g_{c,t}(x) \\ \Gamma_t(x) \end{bmatrix} \in \mathbb{R}^{2d}. \tag{8}$$

The resulting squared distance decomposes as

$$\|\phi_t(x) - \phi_t(b)\|_2^2 = \|g_{c,t}(x) - g_{c,t}(b)\|_2^2 + \|\Gamma_t(x) - \Gamma_t(b)\|_2^2. \tag{9}$$

Unlike $[g_c; g_a]$, this penalizes baseline and response differences separately rather than mixing them in the second block. Thus, clean-signal differences and attack-response differences contribute separately to the acquisition geometry. Since $g_{a,t} = g_{c,t} + \Gamma_t$, it also gives

$$\|g_{a,t}(x) - g_{a,t}(b)\|_2^2 \le 2\|\phi_t(x) - \phi_t(b)\|_2^2. \tag{10}$$

In summary, the representations $[g_c; g_a]$ and $[g_c; g_a - g_c]$ contain the same underlying pair of gradients up to an invertible linear transformation, but they induce different Euclidean metrics. The secant form makes the baseline and perturbation-response discrepancies explicit.

4.3 Batch Selection

Our goal is not to spread selected points apart, but to give each candidate a nearby representative of both gradient components. For nonempty $B \subseteq U_t$, measure this replacement error by

$$\Psi_t(B) := \frac{1}{N_t} \sum_{x \in U_t} D_t(x; B)^2, \quad D_t(x; B)^2 := \min_{b \in B} \|\phi_t(x) - \phi_t(b)\|_2^2. \tag{11}$$

For candidates at scalar coordinates $0, 1, 2, 8, 9, 10$, selecting 1 and 9 represents both separated groups, whereas selecting 0 and 1 leaves the right group far away. Smaller Ψ_t means better average representation, not per-class coverage.

This is the k-means objective with $k = q$ representatives, not classes. We use k-means++ seeding (Arthur & Vassilvitskii, 2007): sample the first candidate uniformly, then choose each unselected candidate with probability $\mathbb{P}_t(x \mid B) := D_t(x; B)^2 / \sum_{x' \in U_t \setminus B} D_t(x'; B)^2$. After each selection, update distances and repeat until q distinct inputs are chosen. Thus, poorly represented candidates receive higher selection probability. If all remaining distances vanish, sample uniformly. We do not relocate centers to means that may lack queryable inputs; the seeding stage already has a coverage guarantee (Arthur & Vassilvitskii, 2007).

Run this procedure independently $M \ge 1$ times on the same embeddings and keep

$$m_t^\star \in \arg\min_{1 \le m \le M} \Psi_t(B_t^{(m)}), \qquad B_t = B_t^{(m_t^\star)}. \tag{12}$$

Restarts require no new attacks or labels; only the final batch is queried.

Algorithm 1 REAL at round t

Require: $\theta_t, S_t, U_t, q, \mathcal{D}_\varepsilon(\cdot), M$
Ensure: Queried batch B_t, true labels, and weights w_t
1: Compute $\hat{y}_t(x)$, $g_{c,t}(x)$, $g_{a,t}(x)$, $\Gamma_t(x)$, and $\phi_t(x)$ for all $x \in U_t$
2: **for** $m = 1, \ldots, M$ independently **do**
3: Select q representatives by the seeding rule in Section 4.3; evaluate $\Psi_t(B_t^{(m)})$
4: **end for**
5: Choose B_t by (12); compute a_t, w_t
6: Query B_t and update the pools by (1)
7: Train using (15) with the pre-query set S_t and the computed weights

4.4 CLUSTER-WEIGHTED UPDATE

We next assign candidates to their nearest representatives and count their masses:

$$a_t(x) \in \arg\min_{b \in B_t} \|\phi_t(x) - \phi_t(b)\|_2, \tag{13a}$$

$$w_t(b) = |\{x \in U_t : a_t(x) = b\}|, \quad \sum_{b \in B_t} w_t(b) = N_t. \tag{13b}$$

For coincident embeddings, assign each selected representative to itself; resolve all other ties by the fixed rule. After querying labels, use

$$\widehat{R}_t(\theta; B_t) = \frac{1}{N_t} \sum_{b \in B_t} w_t(b) \ell_{\mathrm{adv}}(\theta; b, y_b) \tag{14}$$

as the new batch's robust training contribution. These weights preserve group proportions rather than treating every representative equally. For a concrete update with the previously labeled data, use

$$F_{t,B_t}(\theta) = \frac{1}{n_t + q} \left[\sum_{(x,y) \in S_t} \ell_{\mathrm{adv}}(\theta; x, y) + q \widehat{R}_t(\theta; B_t) \right]. \tag{15}$$

Thus, old examples have unit weight and the newly queried batch has total weight q, with relative weights given by its cluster masses. These weights are *round-local*: they approximate the contribution of the current unlabeled pool during the immediate post-query update rather than assigning permanent multiplicities to labeled examples. Because both the model and the remaining pool change after each acquisition round, cluster weights are recomputed for each new batch. Previously queried examples enter S_t once each with unit weight. Labels are never propagated to unqueried candidates.

5 THEORETICAL ANALYSES

We now analyze two questions suggested by the construction of REAL: 1) what information the adversarial secant captures locally, and 2) whether covering the resulting embedding makes the queried, weighted batch a good surrogate for the full unlabeled pool. We then connect this deterministic coverage guarantee to the randomized multi-start k-means++ selector.

Throughout this section, we fix a query round and condition on its full unlabeled pool, current model, and computed embeddings. That is, we write $U = U_t$, $N = N_t$, $\phi = \phi_t$, and $\Psi = \Psi_t$.

5.1 LOCAL MEANING OF THE SECANT

Before analyzing batch coverage, we first clarify what the secant component $\Gamma_t(x)$ measures. We show that because it is the difference between the attacked and clean gradients, it can be viewed locally as the accumulated change of the learning signal along the acquisition attack.

For a fixed target label y, the input Jacobian at input z describes how the last-layer gradient changes with the input:

$$J_t(z; y) = \mathrm{D}_z[\mathrm{vec}(\nabla_W \ell(\theta_t; z, y))] .$$

If the gradient map is continuously differentiable near the attack segment,

$$\Gamma_t(x) = \left[\int_0^1 J_t(x + s\delta_t^\star(x); \hat{y}_t(x))\, ds\right] \delta_t^\star(x). \tag{16}$$

If J_t is also locally Lipschitz, $\Gamma_t(x) = J_t(x; \hat{y}_t(x))\delta_t^\star(x) + O(\|\delta_t^\star(x)\|_2^2)$. Thus, the secant records sensitivity along the actual acquisition attack, rather than sensitivity in an arbitrary input direction. This interpretation motivates the representation but is not needed for the coverage guarantees below.

5.2 Coverage and Robust Coreset Error

We now turn to the main theoretical question. Since REAL selects a batch by reducing the coverage potential $\Psi(B)$, when does small coverage error imply that the selected cluster-weighted representatives faithfully approximate the robust objective of the full unlabeled pool?

Let Θ be a nonempty parameter region on which the robust losses are finite. Retain the true-label loss r_θ from (4) and write $R(\theta) = \mathcal{R}_{U_t}(\theta)$ for the full-pool objective. For any nonempty batch $B \subseteq U$, apply the nearest-representative assignment and cluster weights of (13a)–(13b) to B in place of B_t, denoting them by a_B and w_B. Write $R_B(\theta) = \widehat{R}_t(\theta; B)$ for the resulting weighted objective (14). Unlike R, R_B requires only the queried labels in B. The embedding, assignments, and weights remain fixed as θ varies over Θ.

Assumption 1 (Local robust regularity)**.** *There is a finite $L \geq 0$ such that, for all $\theta \in \Theta$ and $x, x' \in U$, $|r_\theta(x) - r_\theta(x')| \leq L\|\phi(x) - \phi(x')\|_2$.*

Nearby representations must therefore have similar true-label robust losses throughout Θ. This is an additional assumption, not a consequence of gradient similarity; it does not require convexity. The condition must hold on the full current pool, not only on queried points.

Theorem 1 (Coverage controls robust coreset error)**.** *Under Assumption 1, every nonempty $B \subseteq U$ satisfies*

$$\sup_{\theta\in\Theta} |R(\theta) - R_B(\theta)| \leq L\sqrt{\Psi(B)}. \tag{17}$$

If the minima $\theta^\star \in \arg\min_{\theta\in\Theta} R(\theta)$ and $\theta_B \in \arg\min_{\theta\in\Theta} R_B(\theta)$ are attained, then

$$R(\theta_B) - R(\theta^\star) \leq 2L\sqrt{\Psi(B)}. \tag{18}$$

The realized potential controls objective replacement and the coreset minimizer's excess risk for every batch. A smaller potential tightens the bounds under the same embedding and L; it does not guarantee monotone actual-risk improvement or convergence to the stated minima.

5.3 Multi-start Coverage Guarantee

Theorem 1 applies to any realized batch, but does not yet say how small a potential the randomized REAL selector returns. We therefore compare its realized coverage with the best coverage obtainable by any q representatives from the current pool, $\Psi_q^\star = \min_{B\subseteq U, |B|=q} \Psi(B)$. The classical k-means++ guarantee (Arthur & Vassilvitskii, 2007) gives $\mathbb{E}[\Psi(B^{(m)})] \leq 8(\ln q + 2)\Psi_q^\star$. Its arbitrary-center comparator also implies this bound for the restricted pool-subset optimum. Let $\widehat{B}$ be the best of M independent runs. Combining this expectation bound with Markov's inequality and Theorem 1 gives the following corollary.

Corollary 1 (Multi-start coverage and robust-risk comparison)**.** *For $1 \leq q \leq N$ and $M \geq 1$, with probability at least $1 - 2^{-M}$,*

$$\Psi(\widehat{B}) \leq 16(\ln q + 2)\Psi_q^\star. \tag{19}$$

Under Assumption 1, if $\theta^\star \in \arg\min_{\theta\in\Theta} R(\theta)$ and $\theta_{\widehat{B}} \in \arg\min_{\theta\in\Theta} R_{\widehat{B}}(\theta)$ are attained, the same event gives

$$R(\theta_{\widehat{B}}) - R(\theta^\star) \leq 2L\sqrt{\Psi(\widehat{B})} \leq 8L\sqrt{(\ln q + 2)\Psi_q^\star}. \tag{20}$$

The first inequality holds for every returned batch; only the second requires this event. For $\xi \in (0, 1)$, choosing $M \geq \lceil \log_2(1/\xi) \rceil$ gives probability at least $1 - \xi$.

Proofs of Theorem 1 and Corollary 1 are given in Appendix B. The observable $\Psi(\widehat{B})$ evaluates the returned batch; $\Psi_q^\star$ benchmarks coverage, not optimal risk. Probability concerns only seeding randomness conditional on the fixed pool and embeddings. The geometric comparison needs no loss-regularity assumption.

Cluster weights are essential: even zero potential need not preserve pool proportions without them. Adding the same previously labeled objective to R and R_B leaves their difference unchanged. For the normalized update (15), the bounds acquire the factor $q/(n_t + q)$ when compared with the corresponding objective using the oracle pool loss. The guarantees concern the full unlabeled pool at the current round. They do not establish population generalization, cumulative gains across rounds, or superiority over other representations. Since the selector does not provide L, the risk bound remains conditional. Appendix B.3 makes the normalized-update comparison explicit, including a possible optimization error.

6 EXPERIMENTS

We evaluate label efficiency on CIFAR-10 and CIFAR-100 with ResNet-18. Appendix A changes the architecture or dataset, reports acquisition cost, and provides the final-checkpoint APGD evaluation. Every image accuracy is measured on the complete held-out set; no value from another paper is substituted for a run.

6.1 EXPERIMENTAL PROTOCOL

Data and paired comparisons. CIFAR-10 and CIFAR-100 contain 50,000 training and 10,000 test images (Krizhevsky, 2009). We use a small-image ResNet-18 (He et al., 2016) with a 3×3 stride-one stem, no initial max-pool, and a bias-free classifier. For seeds $0, 1, 2$, a class-balanced initial set contains 500 examples. Each method acquires 500 labels in each of nine rounds, ending at 5,000 labels. Methods share the initial labeled IDs and trained model within each seed. Later candidate labels remain hidden until the batch is committed; test images never enter selection or weight construction.

Training and selection. Training uses SGD with momentum 0.9, weight decay 5×10^{-4}, batch size 128, and cosine decay from LR0.1 to zero. We train for 100 initial epochs and 30 epochs after every query, retaining the model and BatchNorm state while resetting the optimizer. PGD adversarial training uses true-label cross-entropy, ten steps, step size $2/255$, a uniform random start, and ℓ_∞ radius $8/255$ in raw $[0, 1]$ pixels. REAL acquisition uses five PGD steps with a fixed clean pseudo-label, the unnormalized embedding $[g_c; g_a - g_c]$, eight independent k-means++ seeding runs, and multipliers $qw_t(b)/N_t$. We compare with Random sampling (Settles, 2009), Core-set (Sener & Savarese, 2018), BADGE (Ash et al., 2020), DRE (Guo et al., 2021), and SAAL (Kim et al., 2023). All methods use the same adversarial-training architecture, optimization schedule, and update budget; the baselines use unit training weights, whereas REAL uses the cluster-weighted update in (15). The main comparison should be interpreted as an end-to-end comparison of the resulting active-learning procedures, rather than as an isolated comparison of acquisition scores. Appendix A.3 records the implemented acquisition rules and their adaptations.

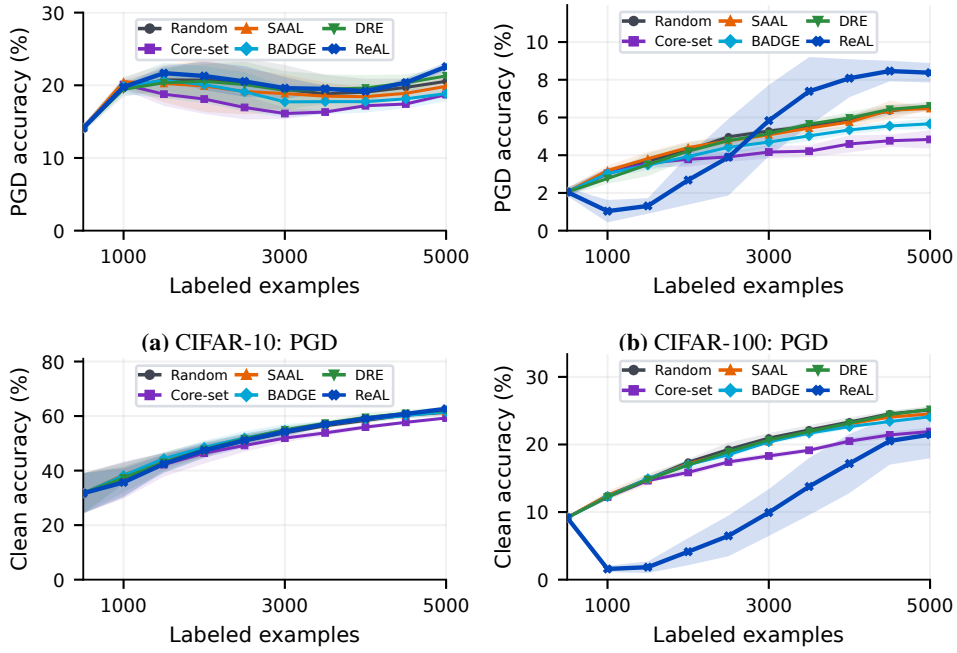
Evaluation and reporting. We evaluate the final epoch of every round on all 10,000 test images. Throughout the figures and tables, *PGD* denotes the projected-gradient attack of Madry et al. (2018), evaluated using the true labels of the held-out test set. For each of five independent restarts, we initialize uniformly in the ℓ_∞ ball of radius $\varepsilon = 8/255$ and apply 20 updates

$$x^{(k+1)} = \Pi_{\mathcal{B}_\infty(x,\varepsilon)\cap[0,1]^{d_{\text{in}}}}\left(x^{(k)} + \alpha \operatorname{sign} \nabla_{x^{(k)}} \ell_{\text{CE}}(f_\theta(x^{(k)}), y)\right), \qquad \alpha = 2/255. \tag{21}$$

Here $f_\theta : [0, 1]^{d_{\text{in}}} \to \mathbb{R}^C$ is the classifier's logit map, $x \in [0, 1]^{d_{\text{in}}}$ and y are a held-out image and its true label, d_{in} is the number of input coordinates, C is the number of classes, ℓ_{CE} is cross-entropy, and $x^{(k)}$ is the attacked image after k updates. We use $\mathcal{B}_\infty(x, \varepsilon) = \{z : \|z - x\|_\infty \le \varepsilon\}$. The gradient is with respect to the current attacked input, sign acts coordinatewise, and α is the fixed step size. The operator $\Pi_{\mathcal{B}_\infty(x,\varepsilon)\cap[0,1]^{d_{\text{in}}}}$ projects back to the intersection of the perturbation set and valid pixel box; equivalently, it clips each coordinate to both constraints. We suppress the restart

index in (21); each restart begins from $x^{(0)} = \Pi_{\mathcal{B}_\infty(x,\varepsilon)\cap[0,1]^{d_{\text{in}}}}(x+u)$, with u sampled uniformly from $[-\varepsilon,\varepsilon]^{d_{\text{in}}}$. We check the random initialization and every iterate. A test example is counted as PGD-robust only if it is classified correctly before attack and no iterate from any restart causes misclassification. PGD accuracy is this survival rate over the full 10,000-example test set, rather than an average over restarts.

At the final label budget we additionally evaluate Auto-PGD (APGD), a first-order projected attack designed to reduce sensitivity to a manually chosen step size (Croce & Hein, 2020). In contrast to the fixed-α update in (21), APGD uses momentum, retains the strongest point encountered, and reduces its step size at scheduled checkpoints when the attack loss oscillates or makes insufficient progress (Croce & Hein, 2020). We run untargeted APGD-CE, which maximizes cross-entropy, for 100 iterations. We then run untargeted APGD-DLR, which uses the scale-invariant difference-of-logits-ratio loss introduced by Croce & Hein (2020), for 100 iterations on the CE survivors, starting again from their original clean inputs. Both components use one random start and the same ℓ_∞ radius $8/255$. The reported APGD accuracy uses the full test-set denominator and requires an example to be clean-correct and to survive both components. Thus it is a sequential CE+DLR ensemble, not an average of two accuracies and not the full AutoAttack suite (Appendix A.6). Reported uncertainties are sample standard deviations over three training seeds. PGD AUC is the trapezoidal integral of PGD accuracy over label count divided by the observed label interval; it includes the common initial checkpoint and has percentage-point units.



(a) CIFAR-10: PGD **(b)** CIFAR-100: PGD

(c) CIFAR-10: clean accuracy **(d)** CIFAR-100: clean accuracy

Figure 1: **ResNet-18 label efficiency.** Curves show the mean across three seeds, and translucent bands show ± one sample SD. Markers denote the measured label budgets; lines only connect these checkpoints.

6.2 Empirical Results

Figure 1 and Table 1 show the central empirical pattern. On CIFAR-10, ReAL has the strongest final clean, PGD, and APGD means and the largest PGD AUC, although the robust curves remain close for much of the trajectory and are not monotone in label count. On CIFAR-100, ReAL again has the strongest final PGD and APGD means, but this robustness is accompanied by lower clean accuracy and noticeably larger variation across seeds; its trajectory-level advantage is also much less pronounced than its endpoint advantage. Thus the evidence supports improved measured endpoint

Table 1: **ResNet-18 results.** Mean ± sample SD over three seeds. Clean, PGD, and APGD are final-checkpoint accuracies (%) at 5,000 labels. PGD AUC is the normalized robust trajectory area defined above. Bold marks the largest mean within each dataset and metric.

	CIFAR-10				CIFAR-100			
Method	Clean	PGD	APGD	PGD AUC	Clean	PGD	APGD	PGD AUC
Random	61.34 ± 1.22	20.55 ± 1.95	20.18 ± 1.88	19.53 ± 1.84	25.13 ± 0.46	6.60 ± 0.26	6.35 ± 0.20	4.81 ± 0.16
Core-set	59.28 ± 0.30	18.72 ± 0.38	18.42 ± 0.41	17.49 ± 0.81	21.92 ± 0.59	4.84 ± 0.48	4.52 ± 0.49	3.94 ± 0.31
SAAL	61.68 ± 0.44	19.86 ± 1.18	19.49 ± 1.05	19.05 ± 1.90	24.53 ± 0.34	6.47 ± 0.08	6.25 ± 0.06	4.79 ± 0.25
BADGE	61.60 ± 1.21	18.85 ± 1.06	18.46 ± 0.99	18.62 ± 1.75	24.10 ± 0.23	5.66 ± 0.29	5.43 ± 0.31	4.37 ± 0.21
DRE	62.09 ± 1.30	21.29 ± 1.01	20.90 ± 0.88	19.65 ± 1.74	$\mathbf{25.19 \pm 0.49}$	6.60 ± 0.14	6.33 ± 0.16	4.75 ± 0.24
ReAL	$\mathbf{62.67 \pm 0.43}$	$\mathbf{22.56 \pm 0.59}$	$\mathbf{21.77 \pm 0.55}$	$\mathbf{20.03 \pm 0.84}$	21.46 ± 3.50	$\mathbf{8.37 \pm 0.52}$	$\mathbf{7.35 \pm 0.13}$	$\mathbf{4.88 \pm 0.99}$

robustness, not uniform dominance at every acquisition round or a significance claim from three seeds.

The favorable robustness trend also persists beyond ResNet-18. After replacing the model with WRN-28-10, ReAL has the largest mean PGD endpoint, APGD endpoint, and PGD AUC on both CIFAR-10 and CIFAR-100. After changing the dataset to Tiny-ImageNet while retaining ResNet-18, ReAL again has the largest mean PGD endpoint and PGD AUC, with an APGD mean only marginally above Random. These extensions support a recurring robust-accuracy trend across model and dataset changes, while also exposing lower clean accuracy or larger seed variation in several settings. The complete trajectories and endpoint statistics are reported in Appendix A.4, Figure 2, and Table 5. The acquisition-cost measurements in Appendix A.5 place ReAL between inexpensive clean-gradient baselines and the substantially more expensive SAAL procedure. Together, these results motivate reporting clean accuracy, PGD, APGD, PGD AUC, and acquisition cost rather than reducing the comparison to a single endpoint number.

7 Conclusion and Limitations

We introduced REAL to connect robustness-aware acquisition geometry with the robust objective used after labels are queried. The method represents each candidate by both its clean last-layer gradient and the change induced by an input attack. Multi-start k-means++ then selects representatives of this joint signal, while cluster masses determine how strongly the queried representatives enter subsequent adversarial training. These components form one construction: the representation defines what must be covered, the selector reduces the corresponding potential, and the weights preserve the pool mass represented by each selected example. Under a uniform local regularity condition linking representation distance to true-label robust loss, the realized coverage potential controls both the uniform error of the weighted pool approximation and the excess robust risk of its minimizer. Independent seeding runs additionally yield a high-probability comparison with the best attainable coverage at the same query budget. The result is conditional on a fixed round and embedding; it does not by itself establish population generalization, optimization convergence, or that the required regularity constant is small for a trained network. Empirically, REAL shows stronger measured robust endpoints under PGD and APGD-CE+DLR across the evaluated ResNet-18 and WRN-28-10 settings, with PGD AUC indicating that trajectory-level gains can be more modest than final-checkpoint improvements. At the same time, lower clean accuracy, larger seed variability in several settings, low absolute Tiny-ImageNet robustness, and nontrivial acquisition cost prevent an across-the-board superiority claim.

Several directions follow naturally. Larger-scale studies could test more architectures, label schedules, and stronger complete attack suites with more training seeds. On the methodological side, estimating or diagnostically testing representation-loss regularity could connect the conditional bound more directly to observed runs. Faster approximations to full-pool attack embeddings and seeding could reduce acquisition cost, while objectives that explicitly balance clean accuracy, robust accuracy, and seed stability may address the trade-offs observed here. Extending the fixed-round analysis to adaptive multi-round selection and population-level guarantees remains the main theoretical challenge.

AI USE STATEMENT

We used generative AI tools to assist with manuscript writing and editing, code development and debugging, experimental design and implementation, analysis and interpretation of experimental results, and the discussion and refinement of the theoretical formulation. Generative AI tools were also used to assist with the development and presentation of mathematical arguments and proofs. The authors independently reviewed and verified the mathematical proofs, experimental results, and final manuscript, and take full responsibility for the content of this work.

REFERENCES

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, 2007.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

Soumya Banerjee and Vinay Kumar Verma. ADROIT: A self-supervised framework for learning robust representations for active learning. *arXiv preprint arXiv:2503.07506*, 2025.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2206–2216, 2020.

Hadi M. Dolatabadi, Sarah M. Erfani, and Christopher Leckie. Adversarial coreset selection for efficient robust training. *International Journal of Computer Vision*, 131(12):3307–3331, 2023.

Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: A margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

Ricky Maulana Fajri, Yulong Pei, Lu Yin, and Mykola Pechenizkiy. Robust active learning (RoAL): Countering dynamic adversaries in active learning with elastic weight consolidation. *arXiv preprint arXiv:2408.07364*, 2024.

Susmita Ghosh, Abhiroop Chatterjee, and Lance Fiondella. An active learning framework for adversarial training of deep neural networks. *Neural Computing and Applications*, 37:6849–6876, 2025.

Yuejun Guo, Qiang Hu, Maxime Cordy, Mike Papadakis, and Yves Le Traon. Robust active learning: Sample-efficient training of robust deep learning models. *arXiv preprint arXiv:2112.02542*, 2021.

Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8175–8195, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Doyoung Heo. RAISE: Robust batch acquisition for sequential uncertainty reduction in robot perception. IROS 2026 Workshop on Rethinking Uncertainty for Modern Robotics Paradigms, 2026. URL `https://openreview.net/forum?id=AbpLqvCqrr`. Non-archival workshop paper.

Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Mate Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Yoon-Yeong Kim, Youngjae Cho, Joonho Jang, Byeonghu Na, Yeongmin Kim, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. SAAL: Sharpness-aware active learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 16424–16440, 2023.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–12, 1994.

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

Seyed Mohammad Hadi Mirsadeghi. Active machine learning approach to adversarial training improves trade-off between natural accuracy and adversarial robustness. *Scientific Reports*, 16: 19726, 2026.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and Jose Miguel Hernandez-Lobato. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

Burr Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison, 2009.

Tsung-Han Wu, Hung-Ting Su, Shang-Tse Chen, and Winston H. Hsu. Fair robust active learning by joint inconsistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 3624–3633, 2023.

Ofer Yehuda, Avihu Dekel, Guy Hacohen, and Daphna Weinshall. Active learning through a covering lens. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

A Additional Experiments and Details

This appendix documents the experiments behind the reported numbers, including negative and mixed results. It complements rather than changes the protocol distinctions in the main text. Every budget has the PGD evaluation defined in Section 6.1 on all 10,000 held-out examples. The 90 reported final checkpoints additionally have APGD-CE+DLR results; we report neither a full AutoAttack score nor an intermediate-budget APGD curve. This appendix provides additional figures, extended measurements, implementation details, and reproducibility information.

A.1 Data, Models, and Reproducibility

Datasets and indexing. CIFAR-10/100 use their official 50,000-image training and 10,000-image test splits at 32×32 resolution. Tiny-ImageNet uses 100,000 training images and the labeled 10,000-image validation split over 200 classes at 64×64; its unlabeled public test split is not treated as labeled data. CIFAR IDs follow official batch order, while Tiny IDs are fixed by sorted class identifier and relative path. Images are converted from uint8 to raw $[0, 1]$ pixels. Channelwise mean and standard deviation $(0.5, 0.5, 0.5)$ are applied inside the model, so attack radii are specified in raw pixel space.

Table 2: Dataset and label schedules. The final column includes the initial set, and all three seeds use every listed budget.

Dataset	Classes	Train pool	Held out	Initial	Query $\times$ rounds	Final
CIFAR-10	10	50,000	10,000	500	500×9	5,000
CIFAR-100	100	50,000	10,000	500	500×9	5,000
Tiny-ImageNet	200	100,000	10,000	1,000	$1{,}000 \times 9$	10,000

Initialization and label access. Initial sets are sampled class-balanced: 50 images per class on CIFAR-10 and five per class on CIFAR-100/Tiny. Within a setting and seed, all methods share the initial IDs and model/BatchNorm tensors. Later selection receives only candidate images, IDs, and model outputs. A selected batch is validated and committed before its labels are retrieved; unqueried candidates are never pseudo-label-trained.

Architectures. The ResNet-18 adaptation uses residual block counts $(2, 2, 2, 2)$, widths $(64, 128, 256, 512)$, a 3×3 stride-one stem, no initial max-pooling, adaptive global pooling, and a bias-free classifier. Tiny uses the same stem at 64×64 without resizing. WRN-28-10 (Zagoruyko & Komodakis, 2016) has a 16-channel stem and three groups of four preactivation blocks with widths $(160, 320, 640)$ and strides $(1, 2, 2)$, followed by BN/ReLU, global average pooling, and a bias-free 640-dimensional classifier input. Dropout is zero. All initial models are trained from scratch without pretrained weights. Subsequent acquisition rounds warm-start from the preceding model and BatchNorm state, as described in Appendix A.2.

Numerics and hardware. Production training and evaluation use PyTorch 2.5.1+cu124, CUDA 12.4, and cuDNN 9.8 on 24-GiB RTX 4090 GPUs. Computation is FP32; AMP, TF32, channels-last, compilation, and cuDNN benchmarking are disabled, while deterministic cuDNN is enabled. A worker uses one GPU and two CPU threads with no parallel data-loader workers. These controls pin the recorded environment but do not promise bitwise identity on arbitrary hardware. Exclusive acquisition timing is separated from concurrent campaign wall time in Appendix A.5.

A.2 Training, Acquisition, and Evaluation Attacks

Optimizer and batches. The initial model is trained for 100 epochs, and every acquisition appends distinct samples and trains for 30 epochs. Model and BN state warm-start; momentum buffers and the cosine schedule reset at each stage. Except where an explicit setting-specific schedule is stated, SGD uses initial LR0.1, momentum 0.9, dampening zero, weight decay 0.0005, no Nesterov momentum, and cosine decay to zero. Batch size is 128 and the last incomplete batch is kept. Every epoch shuffles each distinct labeled ID once. Thus a CIFAR trajectory uses $100\lceil 500/128 \rceil + 30 \sum_{j=1}^{9} \lceil (500 +$

$500j)/128\rceil = 6{,}880$ updates for every method, independent of acquisition weights. Training uses random padded crops and horizontal flips; acquisition and evaluation do not. There is no early stopping or test-selected checkpoint.

Table 3: Attack definitions in the completed image experiments. Every attack uses the raw-pixel ℓ_∞ radius $8/255$ and clips to $[0, 1]$. Fixed-step PGD uses step size $2/255$; APGD adapts its step size.

Purpose	Objective and label	Steps	Restarts
ReAL acquisition	Cross-entropy, fixed clean argmax pseudo-label	5	1
PGD-AT training	Cross-entropy, acquired true label	10	1
Held-out PGD	Cross-entropy, held-out true label	20	5
Final APGD-CE	Cross-entropy, held-out true label	100	1
Final APGD-DLR	Untargeted DLR on CE survivors	100	1

Weighted robust update. Let ω_i denote the training multiplier of example i. An old labeled example has multiplier $\omega_i = 1$, while a newly queried representative $b \in B_t$ has multiplier

$$\omega_b = \frac{q\, w_t(b)}{N_t}.$$

Across the distinct examples used in the post-query update, these multipliers sum to $n_t + q$. For a minibatch $\mathcal{B}$, let ℓ_i denote the adversarial training loss of example i. The minibatch objective is

$$\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \omega_i \ell_i,$$

rather than a loss normalized by the minibatch's total weight. We neither replicate representatives nor use a weighted sampler, and we do not clip large cluster weights. At the next round, previously queried examples become old examples and receive unit multipliers. Baseline multipliers are one.

Attack and BatchNorm state. The inner PGD loop freezes model parameters, uses evaluation mode, and differentiates only with respect to the input; BN statistics do not change. PGD-AT then switches to training mode for the outer true-label cross-entropy update. Acquisition fixes the clean model's argmax pseudo-label before its attack and does not replace that target with the attacked prediction. Random starts are uniform in the perturbation box, and every iterate is projected to the intersection of that box and the valid pixel range.

Per-example evaluation and statistics. Evaluation proceeds in deterministic chunks, normally 32 images, recording canonical IDs, clean correctness, robust survival, and maximum observed CE. A robust flag is cleared by a clean error or any successful attack at a restart's initialization or iterate. Completed chunks are reused after an interruption only if data, checkpoint, evaluator, RNG, and content hashes match. Mean and sample SD are computed over three independent training seeds, not over examples or budgets. Let $b_0 < \cdots < b_T$ denote the measured label budgets, and let $A_s(b_j)$ denote the PGD accuracy in percent for seed s at budget b_j. The PGD AUC for seed s is

$$\mathrm{AUC}_s^{\mathrm{PGD}} = \frac{1}{b_T - b_0} \sum_{j=1}^{T} (b_j - b_{j-1}) \frac{A_s(b_j) + A_s(b_{j-1})}{2}. \tag{22}$$

This is a descriptive interpolation between measured budgets, not a claim about unmeasured budgets.

A.3 Baseline Implementations

We hold the trainer, architecture, data view, label schedule, and update count fixed. The comparison is among acquisition mechanisms under one robust trainer, not among published headline numbers or heterogeneous pipelines.

DRE and SAAL. Our DRE implementation follows Guo et al. (2021) and preserves the strict upper histogram endpoint. A bin of mass p_j receives quota $\lceil qp_j \rceil$; sampled bins are concatenated in order and the last q entries are retained. If fewer than q unique IDs remain, an independently seeded uniform draw fills the deficit. Following Kim et al. (2023), SAAL uses the pseudo-label gradient g and parameter perturbation

$$\Delta\theta = \rho \frac{\theta^2 \odot g}{\| \, |\theta| \odot g \, \|_2 + 10^{-12}}, \qquad \rho = 0.05, \tag{23}$$

over all network parameters, restoring the model after each example and then applying diversity selection to the scalar sharpness score. BN/dropout are frozen. These details matter: this is neither top-q scoring nor an input-PGD selector.

ReAL numerical consistency. Let u_c and u_a denote the clean and attacked prediction-residual vectors, and let h_c and h_a denote their corresponding feature vectors. Each last-layer gradient therefore has the outer-product form $uh^\top$. We retain the factor pairs (u_c, h_c) and (u_a, h_a) and evaluate exact Euclidean products algebraically using

$$\langle uh^\top, u'h'^\top \rangle = (u^\top u')(h^\top h').$$

No random projection, separate block normalization, or metric change is introduced. Final-center distances are recomputed before restart selection; nearest-center ties use the smallest global ID and restart ties use the smallest restart index. Distinct IDs are returned even when some candidates have zero distance.

Table 4: Implemented acquisition rules. Each method selects unique examples from the complete remaining pool. Main baseline trajectories use unit training weights.

Method	Implementation used
Random	Uniform sampling without replacement (Settles, 2009).
Core-set	Greedy Euclidean farthest-first in penultimate features, initialized with all labeled examples as centers (Sener & Savarese, 2018).
BADGE	Clean pseudo-label last-layer gradients; maximum-gradient-norm first center and one D^2 seeding run (Ash et al., 2020).
DRE	Base-2 predictive-entropy histogram matching with ceiling quotas and unique selection (Guo et al., 2021).
SAAL	Full-parameter sharpness scoring followed by diversity selection in score space (Kim et al., 2023).
ReAL	Fixed-pseudo-label PGD5; exact unnormalized $[g_c; g_a - g_c]$ distances; uniform first center; best potential among eight seedings; cluster-weighted update.

A.4 Additional Architectures and Dataset

We evaluate all six acquisition methods with seeds $0, 1, 2$ after changing either the model or the dataset: CIFAR-10/WRN-28-10, CIFAR-100/WRN-28-10, and Tiny-ImageNet/ResNet-18. CIFAR uses the main label schedule. Tiny starts with 1,000 labels, queries 1,000 per round, and ends at 10,000. CIFAR-100/WRN applies LR0.01 after initialization to every method; the remaining setting-specific schedules are held fixed within a panel.

CIFAR-10/WRN-28-10. ReAL reaches $26.50 \pm 3.29\%$ PGD and $24.79 \pm 1.82\%$ APGD, versus DRE's $23.21 \pm 0.50\%$ and $23.01 \pm 0.49\%$. PGD AUC is 21.79 ± 0.32, compared with DRE's 19.96 ± 0.18. However, ReAL's clean score is $60.07 \pm 8.23\%$, versus DRE's $66.19 \pm 0.56\%$, and endpoint variability is much larger. This is a measured robustness/clean-accuracy trade-off, not an across-the-board gain.

Tiny-ImageNet. ReAL reaches $2.63 \pm 1.15\%$ PGD and $2.18 \pm 1.01\%$ APGD, versus Random's $2.42 \pm 0.09\%$ and $2.17 \pm 0.05\%$. PGD AUC is 1.70 ± 0.86, versus Random's 1.52 ± 0.05. Clean accuracy is lower ($11.35 \pm 8.12\%$ versus $16.93 \pm 0.36\%$) and variability is substantial. Absolute robust accuracies remain low, so the tiny difference in APGD means does not establish a meaningful advantage.

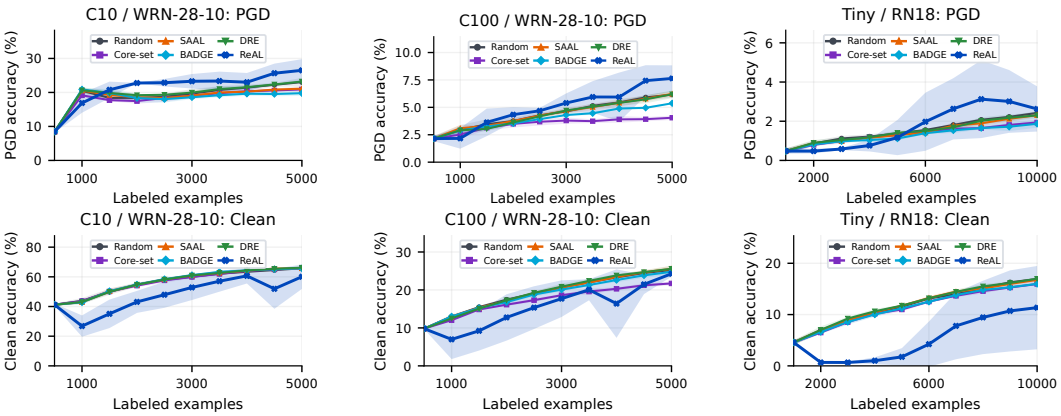


Figure 2: **Architecture and dataset changes.** PGD and clean-accuracy trajectories for CIFAR-10/WRN-28-10, CIFAR-100/WRN-28-10, and Tiny-ImageNet/RN18. Curves show three-seed means, and translucent bands show ± one sample SD. Colors and markers match Figure 1.

Table 5: Architecture/dataset extensions, mean ± sample SD over three seeds. Clean, PGD, and APGD are final accuracies (%); PGD AUC summarizes the complete robust trajectory. CIFAR uses 5,000 labels and Tiny uses 10,000. Bold marks the largest mean in each setting and metric.

Setting	Method	Clean	PGD	APGD	PGD AUC
C10 / WRN	Random	65.74 ± 0.84	23.04 ± 1.25	22.80 ± 1.23	19.53 ± 0.64
C10 / WRN	Core-set	65.77 ± 0.58	20.89 ± 0.66	20.66 ± 0.67	18.57 ± 0.54
C10 / WRN	SAAL	65.72 ± 0.34	21.09 ± 0.86	20.86 ± 0.83	19.07 ± 0.31
C10 / WRN	BADGE	65.69 ± 0.38	19.77 ± 0.80	19.54 ± 0.85	18.68 ± 0.71
C10 / WRN	DRE	$\mathbf{66.19 \pm 0.56}$	23.21 ± 0.50	23.01 ± 0.49	19.96 ± 0.18
C10 / WRN	ReAL	60.07 ± 8.23	$\mathbf{26.50 \pm 3.29}$	$\mathbf{24.79 \pm 1.82}$	$\mathbf{21.79 \pm 0.32}$
C100 / WRN	Random	25.25 ± 0.29	6.17 ± 0.31	5.87 ± 0.23	4.41 ± 0.23
C100 / WRN	Core-set	21.72 ± 0.49	4.05 ± 0.20	3.77 ± 0.24	3.48 ± 0.26
C100 / WRN	SAAL	25.54 ± 0.87	6.23 ± 0.22	5.94 ± 0.31	4.39 ± 0.10
C100 / WRN	BADGE	24.78 ± 0.79	5.37 ± 0.04	5.11 ± 0.08	4.00 ± 0.07
C100 / WRN	DRE	$\mathbf{25.59 \pm 0.75}$	6.20 ± 0.35	5.93 ± 0.36	4.31 ± 0.17
C100 / WRN	ReAL	24.27 ± 0.78	$\mathbf{7.64 \pm 1.19}$	$\mathbf{6.78 \pm 0.72}$	$\mathbf{4.94 \pm 0.27}$
Tiny / RN18	Random	16.93 ± 0.36	2.42 ± 0.09	2.17 ± 0.05	1.52 ± 0.05
Tiny / RN18	Core-set	15.96 ± 0.17	1.94 ± 0.07	1.64 ± 0.08	1.34 ± 0.04
Tiny / RN18	SAAL	16.76 ± 0.40	2.31 ± 0.08	2.06 ± 0.12	1.44 ± 0.08
Tiny / RN18	BADGE	15.99 ± 0.29	1.87 ± 0.20	1.62 ± 0.23	1.27 ± 0.10
Tiny / RN18	DRE	$\mathbf{16.99 \pm 0.25}$	2.30 ± 0.16	2.06 ± 0.12	1.48 ± 0.08
Tiny / RN18	ReAL	11.35 ± 8.12	$\mathbf{2.63 \pm 1.15}$	$\mathbf{2.18 \pm 1.01}$	$\mathbf{1.70 \pm 0.86}$

CIFAR-100/WRN-28-10 with a common low-LR schedule. Each seed trains a fresh initial model for 100 epochs at LR0.1. Every later 30-epoch stage of every method starts at LR0.01 with otherwise unchanged optimizer, batch, ten-step PGD training, and PGD evaluation settings. This is one fixed schedule on another architecture, not a tuned LR grid. ReAL reaches $7.64 \pm 1.19\%$ PGD, $6.78 \pm 0.72\%$ APGD, and 4.94 ± 0.27 PGD AUC. The strongest corresponding baseline means are $6.23 \pm 0.22\%$, $5.94 \pm 0.31\%$, and 4.41 ± 0.23, respectively. ReAL's clean accuracy, $24.27 \pm 0.78\%$, remains below Random, BADGE, DRE, and SAAL.

A.5 Acquisition Cost

We measure acquisition exclusively on one RTX 4090 using the actual seed-0 initial RN18 checkpoint for each CIFAR dataset, all 49,500 remaining candidates, and $q = 500$. Each method has three independent cold factor-cache repetitions with extraction batch size 128. CUDA synchronization surrounds factor extraction/attack, seeding, and assignment; total time also includes selection and I/O outside these regions. These are repeated acquisitions at one reference model, not the three training seeds used in accuracy tables.

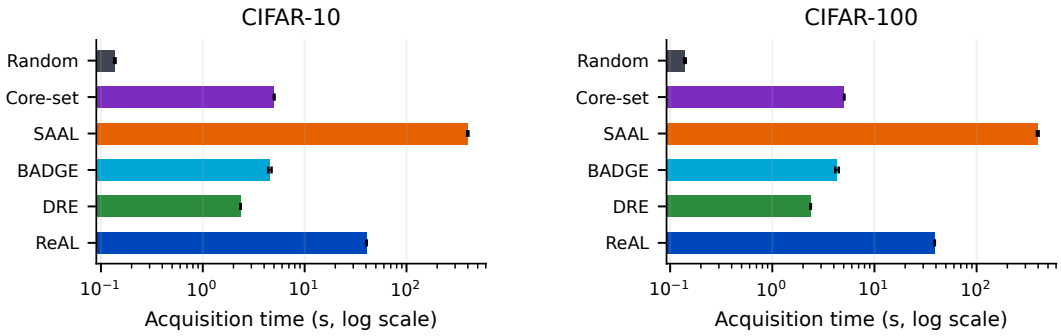


Figure 3: **Full-pool acquisition cost.** Mean total seconds with one sample SD across three exclusive repetitions. Horizontal axes are logarithmic and bar colors match the algorithm colors in the accuracy figures. Training and held-out evaluation are excluded.

Table 6: Acquisition time (seconds) and peak reserved GPU memory (MiB), mean ± sample SD over three exclusive repetitions at one checkpoint. Factors include input attacks when present. A zero seeding entry means no instrumented factor-seeding call, not zero selection work.

Data	Method	Total (s)	Factors (s)	Seeding (s)	Reserved MiB
C10	Random	0.14 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	64.00 ± 0.00
C10	Core-set	5.02 ± 0.04	2.55 ± 0.02	0.00 ± 0.00	520.00 ± 0.00
C10	SAAL	399.77 ± 5.38	399.35 ± 5.38	0.28 ± 0.00	918.00 ± 0.00
C10	BADGE	4.56 ± 0.17	2.61 ± 0.03	1.77 ± 0.16	912.00 ± 0.00
C10	DRE	2.35 ± 0.01	2.21 ± 0.01	0.00 ± 0.00	912.00 ± 0.00
C10	ReAL	40.61 ± 0.08	33.60 ± 0.09	6.64 ± 0.01	1156.00 ± 0.00
C100	Random	0.14 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	64.00 ± 0.00
C100	Core-set	5.05 ± 0.02	2.56 ± 0.02	0.00 ± 0.00	520.00 ± 0.00
C100	SAAL	396.94 ± 7.60	396.52 ± 7.60	0.28 ± 0.00	918.00 ± 0.00
C100	BADGE	4.30 ± 0.17	2.66 ± 0.02	1.45 ± 0.14	912.00 ± 0.00
C100	DRE	2.37 ± 0.01	2.23 ± 0.01	0.00 ± 0.00	912.00 ± 0.00
C100	ReAL	38.93 ± 0.10	33.86 ± 0.10	4.76 ± 0.00	1156.00 ± 0.00

ReAL takes approximately 40.61 seconds on CIFAR-10: factor extraction and attacks dominate, while seeding takes 6.64 seconds and assignment 0.18. CIFAR-100 takes 38.93 seconds, including 4.76 seconds of seeding. BADGE takes about 4.6/4.3 seconds and SAAL 399.77/396.94 seconds. The factorization avoids materializing all high-dimensional gradients, but input attacks remain more expensive than clean-prediction baselines. Peak reserved memory reflects the allocator state and is not a minimal-memory claim. No end-to-end speedup follows from these acquisition-only measurements.

A.6 Final-Checkpoint APGD Ensemble

Coverage and checkpoint rule. Random, Core-set, SAAL, BADGE, DRE, and ReAL are evaluated in five settings: CIFAR-10/100 and Tiny-ImageNet with RN18, plus CIFAR-10/100 with WRN-28-10. Each method has all three seeds, giving 90 final checkpoints. The endpoint is the last epoch at 5,000 CIFAR labels or 10,000 Tiny labels; no best-epoch checkpoint is selected. Every PGD/APGD pair is matched by checkpoint SHA-256.

Attack implementation. We use the official `APGDAttack` implementation (Croce & Hein, 2020), pinned to commit `a39220048b3c9f2cca9a4d3a54604793c68eca7e`. Untargeted APGD-CE is followed by untargeted APGD-DLR; each uses 100 iterations, one restart, `Linf`, $\varepsilon = 8/255$, `eot_iter=1`, `rho=0.75`, `use_largereps=False`, and `best_loss=False`. Inputs are raw $[0, 1]$ pixels with normalization inside the model. The model/BN state is fixed, parameter gradients are disabled, and input gradients remain enabled. This is not full AutoAttack: targeted APGD, FAB, and Square are absent.

Sequential survivors. A clean error is immediately nonrobust. CE attacks the clean-correct subset; DLR attacks only CE survivors, always from their original clean input. Hence

$$\widehat{\mathrm{Acc}}_{\mathrm{ens}} = \frac{100}{10{,}000} \sum_{i=1}^{10{,}000} \mathbf{1}\{\text{clean-correct and survives CE and DLR}\}. \tag{24}$$

Here $\mathbf{1}\{\cdot\}$ denotes the indicator function. The denominator remains 10,000. We neither average component accuracies nor multiply aggregate survival rates. APGD-CE uses the same denominator. A standalone APGD-DLR accuracy is unavailable because DLR is not run on samples already broken by CE.

Validation. The held-out set is traversed in canonical chunks of 32. For chunk offset s, CE uses seed $20260921 + s$ and DLR uses $20260921 + s + 100000$. Completed chunks are reused only after identity/hash checks. Each result contains 10,000 unique canonical IDs. We verify finite, valid-range perturbations within the declared radius; unchanged model and BN state; clean-correctness agreement with the paired PGD record; and ensemble survival no greater than CE or clean survival. All 90 reported evaluations pass these checks.

Table 7: **RN18 endpoint robustness and trajectory summary.** Mean ± sample SD over three seeds. Clean, PGD, and APGD are accuracies (%); PGD AUC uses every measured label budget. All endpoint columns in a row use the same checkpoint and complete held-out set.

Setting	Method	Clean	PGD	APGD	PGD AUC
C10 / RN18	Random	61.34 ± 1.22	20.55 ± 1.95	20.18 ± 1.88	19.53 ± 1.84
C10 / RN18	Core-set	59.28 ± 0.30	18.72 ± 0.38	18.42 ± 0.41	17.49 ± 0.81
C10 / RN18	SAAL	61.68 ± 0.44	19.86 ± 1.18	19.49 ± 1.05	19.05 ± 1.90
C10 / RN18	BADGE	61.60 ± 1.21	18.85 ± 1.06	18.46 ± 0.99	18.62 ± 1.75
C10 / RN18	DRE	62.09 ± 1.30	21.29 ± 1.01	20.90 ± 0.88	19.65 ± 1.74
C10 / RN18	ReAL	$\mathbf{62.67 \pm 0.43}$	$\mathbf{22.56 \pm 0.59}$	$\mathbf{21.77 \pm 0.55}$	$\mathbf{20.03 \pm 0.84}$
C100 / RN18	Random	25.13 ± 0.46	6.60 ± 0.26	6.35 ± 0.20	4.81 ± 0.16
C100 / RN18	Core-set	21.92 ± 0.59	4.84 ± 0.48	4.52 ± 0.49	3.94 ± 0.31
C100 / RN18	SAAL	24.53 ± 0.34	6.47 ± 0.08	6.25 ± 0.06	4.79 ± 0.25
C100 / RN18	BADGE	24.10 ± 0.23	5.66 ± 0.29	5.43 ± 0.31	4.37 ± 0.21
C100 / RN18	DRE	$\mathbf{25.19 \pm 0.49}$	6.60 ± 0.14	6.33 ± 0.16	4.75 ± 0.24
C100 / RN18	ReAL	21.46 ± 3.50	$\mathbf{8.37 \pm 0.52}$	$\mathbf{7.35 \pm 0.13}$	$\mathbf{4.88 \pm 0.99}$
Tiny / RN18	Random	16.93 ± 0.36	2.42 ± 0.09	2.17 ± 0.05	1.52 ± 0.05
Tiny / RN18	Core-set	15.96 ± 0.17	1.94 ± 0.07	1.64 ± 0.08	1.34 ± 0.04
Tiny / RN18	SAAL	16.76 ± 0.40	2.31 ± 0.08	2.06 ± 0.12	1.44 ± 0.08
Tiny / RN18	BADGE	15.99 ± 0.29	1.87 ± 0.20	1.62 ± 0.23	1.27 ± 0.10
Tiny / RN18	DRE	$\mathbf{16.99 \pm 0.25}$	2.30 ± 0.16	2.06 ± 0.12	1.48 ± 0.08
Tiny / RN18	ReAL	11.35 ± 8.12	$\mathbf{2.63 \pm 1.15}$	$\mathbf{2.18 \pm 1.01}$	$\mathbf{1.70 \pm 0.86}$

Table 8: **WRN-28-10 endpoint robustness and PGD AUC**, with the same aggregation as Table 7. CIFAR-100 applies query-stage LR0.01 to every method; CIFAR-10 uses LR0.1.

Setting	Method	Clean	PGD	APGD	PGD AUC
C10 / WRN	Random	65.74 ± 0.84	23.04 ± 1.25	22.80 ± 1.23	19.53 ± 0.64
C10 / WRN	Core-set	65.77 ± 0.58	20.89 ± 0.66	20.66 ± 0.67	18.57 ± 0.54
C10 / WRN	SAAL	65.72 ± 0.34	21.09 ± 0.86	20.86 ± 0.83	19.07 ± 0.31
C10 / WRN	BADGE	65.69 ± 0.38	19.77 ± 0.80	19.54 ± 0.85	18.68 ± 0.71
C10 / WRN	DRE	$\mathbf{66.19 \pm 0.56}$	23.21 ± 0.50	23.01 ± 0.49	19.96 ± 0.18
C10 / WRN	ReAL	60.07 ± 8.23	$\mathbf{26.50 \pm 3.29}$	$\mathbf{24.79 \pm 1.82}$	$\mathbf{21.79 \pm 0.32}$
C100 / WRN	Random	25.25 ± 0.29	6.17 ± 0.31	5.87 ± 0.23	4.41 ± 0.23
C100 / WRN	Core-set	21.72 ± 0.49	4.05 ± 0.20	3.77 ± 0.24	3.48 ± 0.26
C100 / WRN	SAAL	25.54 ± 0.87	6.23 ± 0.22	5.94 ± 0.31	4.39 ± 0.10
C100 / WRN	BADGE	24.78 ± 0.79	5.37 ± 0.04	5.11 ± 0.08	4.00 ± 0.07
C100 / WRN	DRE	$\mathbf{25.59 \pm 0.75}$	6.20 ± 0.35	5.93 ± 0.36	4.31 ± 0.17
C100 / WRN	ReAL	24.27 ± 0.78	$\mathbf{7.64 \pm 1.19}$	$\mathbf{6.78 \pm 0.72}$	$\mathbf{4.94 \pm 0.27}$

On CIFAR-10/RN18, ReAL obtains 21.77% APGD versus DRE's 20.90%; the 0.87-point margin is smaller than the 1.27-point PGD margin. On CIFAR-100/RN18, ReAL changes from 8.37% PGD to 7.35% APGD, while the largest baseline APGD mean is Random's 6.35%. ReAL reaches 24.79% APGD on CIFAR-10/WRN versus DRE's 23.01%, and 6.78% on CIFAR-100/WRN versus SAAL's 5.94%.

On Tiny, ReAL reaches 2.18% APGD versus Random's 2.17%, a negligible difference relative to ReAL's approximately 1.01-point sample SD. With three seeds, none of these mean differences is a significance claim. PGD and APGD-CE+DLR have different optimization procedures; neither is assumed to dominate in general, and neither certifies robustness.

A.7 Evidence Inventory and Scope

The reported six-method comparison contains 825 checkpoint evaluations over five settings: 165 per setting, comprising three common initial checkpoints and $6 \times 3 \times 9 = 162$ queried-round checkpoints. Ninety final-checkpoint APGD evaluations cover every reported setting, method, and seed. Shared observations are not counted as new independent runs.

For every image evaluation, we verify all canonical IDs, the raw per-example file hash, clean and robust counts, and the implication from robust to clean correctness. Checkpoint identity agrees between the training and evaluation records. Interrupted stages are not counted as successful, and discarded numerical-validation attempts do not enter the reported results.

The evidence does not establish population robustness, a verified uniform regularity constant for image models, statistical significance from three seeds, or universal superiority over acquisition baselines. Full AutoAttack and independently tuned baselines are absent. CIFAR-100/WRN evaluates one common later-stage schedule rather than a learning-rate grid. These limits are part of the empirical result, not qualifications to be omitted.

B Theoretical Proofs and Details

All arguments below condition on one active-learning round: the finite pool U, its cardinality N, the embedding ϕ, and the deterministic nearest-center tie-breaking rule are fixed. Randomness enters only through the independent k-means++ seeding runs in Corollary 1. The objective-approximation argument is a deterministic weighted-coreset calculation, related in motivation to coverage- and subset-based active learning (Sener & Savarese, 2018; Pinsler et al., 2019), but the bounds below follow directly from Assumption 1.

It is important to keep three objects separate. First, ϕ is computed at acquisition time from the frozen model and pseudo-labels and remains fixed throughout each argument. Second, $r_\theta(x)$ is the true-label adversarial loss at a variable parameter θ, corresponding to the robust inner maximization used in adversarial training (Madry et al., 2018). Third, the constant L is required to hold uniformly for every $\theta \in \Theta$ and every pair of pool points. The assignment a_B and weights w_B depend on B and the fixed embedding, but not on the later value of θ.

B.1 Proof of Theorem 1

Fix a nonempty batch $B \subseteq U$ and its nearest-representative assignment a_B. For each $b \in B$, define the assigned cluster

$$C_B(b) = \{x \in U : a_B(x) = b\}, \qquad w_B(b) = |C_B(b)|. \tag{25}$$

The deterministic tie rule makes $\{C_B(b) : b \in B\}$ a disjoint partition of U. Therefore the weighted representative sum expands cluster by cluster:

$$\begin{aligned} NR_B(\theta) &= \sum_{b\in B} |C_B(b)| r_\theta(b) = \sum_{b\in B} \sum_{x\in C_B(b)} r_\theta(b) \\ &= \sum_{x\in U} r_\theta(a_B(x)). \end{aligned} \tag{26}$$

This identity is precisely where cluster-mass weighting enters the proof. Write $d_B(x) = \|\phi(x) - \phi(a_B(x))\|_2$. For every $\theta \in \Theta$, the scalar triangle inequality and Assumption 1 give

$$\begin{aligned} |R(\theta) - R_B(\theta)| &= \left| \frac{1}{N} \sum_{x\in U} \big(r_\theta(x) - r_\theta(a_B(x))\big) \right| \\ &\le \frac{1}{N} \sum_{x\in U} |r_\theta(x) - r_\theta(a_B(x))| \\ &\le \frac{L}{N} \sum_{x\in U} d_B(x). \end{aligned} \tag{27}$$

The remaining average is controlled explicitly by Cauchy–Schwarz:

$$\begin{aligned} \left(\frac{1}{N} \sum_{x\in U} d_B(x) \right)^2 &= \frac{1}{N^2} \left(\sum_{x\in U} 1 \cdot d_B(x) \right)^2 \\ &\le \frac{1}{N^2} \left(\sum_{x\in U} 1^2 \right) \left(\sum_{x\in U} d_B(x)^2 \right) \\ &= \frac{1}{N} \sum_{x\in U} d_B(x)^2 = \Psi(B). \end{aligned} \tag{28}$$

Combining (27) and (28) yields

$$|R(\theta) - R_B(\theta)| \le L \left(\frac{1}{N} \sum_{x\in U} d_B(x)^2 \right)^{1/2} = L\sqrt{\Psi(B)}. \tag{29}$$

The equality uses the nearest-representative definition of a_B. Since the right-hand side is independent of θ, taking the supremum over Θ proves (17). Finiteness of U makes all sums finite; no differentiability or convexity of r_θ in θ is used.

For the second claim, set $\Delta_B = L\sqrt{\Psi(B)}$. The uniform result implies both $R(\theta) \le R_B(\theta) + \Delta_B$ and $R_B(\theta) \le R(\theta) + \Delta_B$ for every $\theta \in \Theta$. When the stated minima are attained, optimality of θ_B for R_B gives

$$\begin{aligned} R(\theta_B) &\le R_B(\theta_B) + \Delta_B \\ &\le R_B(\theta^\star) + \Delta_B \\ &\le R(\theta^\star) + 2\Delta_B. \end{aligned} \tag{30}$$

Subtracting $R(\theta^\star)$ proves (18). The factor two appears because the uniform approximation is used once at the returned parameter and once at the oracle parameter; it is not a probability or union-bound factor.

The same reasoning separates coreset error from optimization error. Suppose $\widetilde{\theta}_B \in \Theta$ satisfies $R_B(\widetilde{\theta}_B) \le \inf_{\theta\in\Theta} R_B(\theta) + \eta$ for a finite $\eta \ge 0$. Then

$$\begin{aligned} R(\widetilde{\theta}_B) &\le R_B(\widetilde{\theta}_B) + \Delta_B \\ &\le \inf_{\theta\in\Theta} R_B(\theta) + \eta + \Delta_B \\ &\le \inf_{\theta\in\Theta} R(\theta) + \eta + 2\Delta_B. \end{aligned} \tag{31}$$

The last line uses $\inf_\Theta R_B \le \inf_\Theta R + \Delta_B$. Thus, when the infima are finite, $R(\widetilde{\theta}_B) - \inf_\Theta R \le 2L\sqrt{\Psi(B)} + \eta$ even if the infima are not attained. This exposes a possible optimization error; it does not assert that SGD achieves any specified η.

B.2 Proof of Corollary 1

Write $z_x = \phi(x)$ and, for any q centers $C = \{c_1, \dots, c_q\}$, define the unnormalized Euclidean potential

$$\Phi(C) = \sum_{x\in U} \min_{c\in C} \|z_x - c\|_2^2. \tag{32}$$

For a selected batch B, the center set is $\phi(B)$ and $\Psi(B) = \Phi(\phi(B))/N$. For comparison with the classical result, define the unrestricted q-center optimum

$$\Psi_q^{\text{free}} = \min_{c_1,\dots,c_q\in\mathbb{R}^{2d}} \frac{1}{N} \sum_{x\in U} \min_{1\le j\le q} \|\phi(x) - c_j\|_2^2. \tag{33}$$

Allowing centers anywhere in $\mathbb{R}^{2d}$ rather than only at embedded pool points enlarges the feasible set, hence $\Psi_q^{\text{free}} \le \Psi_q^\star$. Each ReAL seeding run draws its first center uniformly and every later center with probability proportional to its squared distance from the current center set. This is exactly the D^2 sampling rule analyzed for k-means++ by Arthur & Vassilvitskii (2007). Their expected-potential guarantee applies to Φ; dividing by N gives, for every run m,

$$\mathbb{E}[\Psi(B^{(m)})] \le 8(\ln q + 2)\Psi_q^{\text{free}} \le 8(\ln q + 2)\Psi_q^\star. \tag{34}$$

The logarithm and constant are inherited from that seeding analysis, whereas the second inequality only changes the comparator from arbitrary centers to the more restrictive pool-subset optimum. All expectations and probabilities below condition on the fixed pool and computed embeddings.

Assume first that $\Psi_q^\star > 0$ and set $\tau = 16(\ln q + 2)\Psi_q^\star$. Because $\Psi(B^{(m)})$ is nonnegative, Markov's inequality and (34) give

$$\Pr\Big(\Psi(B^{(m)}) > \tau\Big) \le \frac{\mathbb{E}[\Psi(B^{(m)})]}{\tau} \le \frac{1}{2}. \tag{35}$$

The algorithm returns $\widehat{B} \in \arg\min_{1\le m\le M} \Psi(B^{(m)})$. Consequently, failure of the threshold for $\widehat{B}$ occurs if and only if every run fails:

$$\{\Psi(\widehat{B}) > \tau\} = \bigcap_{m=1}^{M} \{\Psi(B^{(m)}) > \tau\}, \tag{36}$$

$$\Pr\Big(\Psi(\widehat{B}) > \tau\Big) = \prod_{m=1}^{M} \Pr\Big(\Psi(B^{(m)}) > \tau\Big) \le 2^{-M}. \tag{37}$$

The product uses independence of the M seeding random streams conditional on U and ϕ; identical marginal distributions are not otherwise required.

If $\Psi_q^\star = 0$, (34) instead implies $\mathbb{E}[\Psi(B^{(m)})] = 0$. A nonnegative random variable with zero expectation equals zero almost surely, so the claimed geometric bound holds without dividing by $\Psi_q^\star$. If all remaining candidates have zero distance during seeding, completing the batch with distinct candidates leaves the potential equal to zero.

On the event $\Psi(\widehat{B}) \le \tau$, Theorem 1 yields

$$\begin{aligned} R(\theta_{\widehat{B}}) - R(\theta^\star) &\le 2L\sqrt{\Psi(\widehat{B})} \\ &\le 2L\sqrt{16(\ln q + 2)\Psi_q^\star} = 8L\sqrt{(\ln q + 2)\Psi_q^\star}. \end{aligned} \tag{38}$$

The same substitution in (31) shows that an η-approximate coreset minimizer obeys, on this event,

$$R(\widetilde{\theta}_{\widehat{B}}) - \inf_{\theta\in\Theta} R(\theta) \le 8L\sqrt{(\ln q + 2)\Psi_q^\star} + \eta. \tag{39}$$

Finally, $M \ge \lceil \log_2(1/\xi) \rceil$ implies $2^{-M} \le \xi$, hence success probability at least $1-\xi$. This probability statement applies to one fixed round. A simultaneous statement over several rounds would require allocating failure probabilities across rounds (for example, by a union bound); no such cross-round claim is made here.

B.3 Details of the Cluster-weighted Training Update

Fix round t and a batch B. For this subsection, write $R(\theta) = \mathcal{R}_{U_t}(\theta)$ and $R_B(\theta) = \widehat{R}_t(\theta; B)$, with assignments and weights recomputed for B in the fixed embedding. Define the old-data loss sum $A_t(\theta) = \sum_{(x,y)\in S_t} \ell_{\mathrm{adv}}(\theta; x, y)$ and the relative new-batch weight $\beta_t = q/(n_t + q)$. Since the assignment clusters partition the N_t candidates,

$$\sum_{b\in B} w_B(b) = N_t, \qquad \sum_{b\in B} \frac{q\, w_B(b)}{N_t} = q. \tag{40}$$

Moreover, the total weighted loss of the newly queried representatives is

$$\sum_{b\in B} \frac{q\, w_B(b)}{N_t} \ell_{\mathrm{adv}}(\theta; b, y_b) = q\, R_B(\theta). \tag{41}$$

Thus the implemented objective and its full-pool comparator can be written as

$$F_{t,B}(\theta) = \frac{A_t(\theta) + qR_B(\theta)}{n_t + q}, \qquad F_{t,U}(\theta) = \frac{A_t(\theta) + qR(\theta)}{n_t + q}. \tag{42}$$

Their common old-data term cancels exactly:

$$\begin{aligned} F_{t,B}(\theta) - F_{t,U}(\theta) &= \frac{qR_B(\theta) - qR(\theta)}{n_t + q} \\ &= \beta_t\big(R_B(\theta) - R(\theta)\big). \end{aligned} \tag{43}$$

Applying Theorem 1 to the final factor gives

$$\sup_{\theta\in\Theta} |F_{t,B}(\theta) - F_{t,U}(\theta)| \le \beta_t L\sqrt{\Psi_t(B)}. \tag{44}$$

The coefficient β_t is therefore not an additional assumption: it is the exact fraction of normalized objective mass allocated to the new batch. Suppose a returned parameter $\widetilde{\theta}_{t,B} \in \Theta$ satisfies $F_{t,B}(\widetilde{\theta}_{t,B}) \le \inf_{\theta\in\Theta} F_{t,B}(\theta) + \eta_t$ for a finite $\eta_t \ge 0$, and both objective infima are finite. For every $\theta \in \Theta$,

$$\begin{aligned} F_{t,U}(\widetilde{\theta}_{t,B}) &\le F_{t,B}(\widetilde{\theta}_{t,B}) + \beta_t L\sqrt{\Psi_t(B)} \\ &\le F_{t,B}(\theta) + \eta_t + \beta_t L\sqrt{\Psi_t(B)} \\ &\le F_{t,U}(\theta) + \eta_t + 2\beta_t L\sqrt{\Psi_t(B)}. \end{aligned} \tag{45}$$

Taking the infimum over θ gives

$$F_{t,U}(\widetilde{\theta}_{t,B}) - \inf_{\theta\in\Theta} F_{t,U}(\theta) \le 2\beta_t L\sqrt{\Psi_t(B)} + \eta_t. \tag{46}$$

Equivalently, the two approximation terms contribute one $\beta_t L\sqrt{\Psi_t(B)}$ at the returned parameter and one at the oracle parameter, while η_t accounts only for optimization of $F_{t,B}$. This compares the combined training objective with the corresponding full-pool comparator. It does not give the same excess-risk bound for R alone at a minimizer of $F_{t,B}$, nor does it compare the returned batch with the post-training oracle in (5).

The cluster weights are necessary even in the zero-potential case. For example, suppose two representatives b_1, b_2 have cluster sizes $N_1 \neq N_2$, $N_1 + N_2 = N$, and every point has zero embedding distance to its representative. Assumption 1 then makes the robust loss constant within each cluster, so the pool objective is

$$R(\theta) = \frac{N_1 r_\theta(b_1) + N_2 r_\theta(b_2)}{N}. \tag{47}$$

The cluster-weighted coreset reproduces this expression exactly, whereas an unweighted average gives $\frac{1}{2}(r_\theta(b_1) + r_\theta(b_2))$, which generally differs when the cluster masses and representative losses differ. Previously queried examples have unit weight, as specified in the method; weights from earlier rounds are not accumulated.

B.4 Details of Acquisition-gradient Approximation

Fix the model, embeddings, and batch. Write $g_a = g_{a,t}$, $g_c = g_{c,t}$, and $\Gamma = \Gamma_t$. Define $\Delta_c = g_c(x) - g_c(b)$ and $\Delta_\Gamma = \Gamma(x) - \Gamma(b)$. Then $2\langle\Delta_c, \Delta_\Gamma\rangle \le \|\Delta_c\|_2^2 + \|\Delta_\Gamma\|_2^2$, which gives

$$\begin{aligned}
\|g_a(x) - g_a(b)\|_2^2 &= \|\Delta_c + \Delta_\Gamma\|_2^2 \\
&= \|\Delta_c\|_2^2 + 2\langle\Delta_c, \Delta_\Gamma\rangle + \|\Delta_\Gamma\|_2^2 \\
&\le 2\|g_c(x) - g_c(b)\|_2^2 + 2\|\Gamma(x) - \Gamma(b)\|_2^2 \\
&= 2\|\phi(x) - \phi(b)\|_2^2.
\end{aligned} \tag{48}$$

Using the same mass-counting identity as in (26), define the representative error $e_x = g_a(x) - g_a(a_B(x))$. The triangle inequality for the Euclidean norm, Cauchy–Schwarz, and (48) yield

$$\begin{aligned}
&\left\| \frac{1}{N}\sum_{x\in U} g_a(x) - \frac{1}{N}\sum_{b\in B} w_B(b) g_a(b) \right\|_2 \\
&= \left\| \frac{1}{N}\sum_{x\in U} e_x \right\|_2 \\
&\le \frac{1}{N}\sum_{x\in U} \|e_x\|_2 \\
&\le \left(\frac{1}{N}\sum_{x\in U} \|e_x\|_2^2 \right)^{1/2} \\
&\le \left(\frac{2}{N}\sum_{x\in U} \|\phi(x) - \phi(a_B(x))\|_2^2 \right)^{1/2} = \sqrt{2\Psi(B)}.
\end{aligned} \tag{49}$$

The second inequality is the root-mean-square bound $N^{-1}\sum_x \|e_x\|_2 \le (N^{-1}\sum_x \|e_x\|_2^2)^{1/2}$. This result is deterministic once the model, pseudo-labels, attacks, and batch are fixed, and it requires no relationship between pseudo-labels and true labels. It controls only the mean acquisition-gradient approximation. It therefore does not imply the uniform true-label loss regularity in Assumption 1 and cannot replace that assumption in Theorem 1.