DYNAMIC SKETCHING FOR EFFICIENT SGD VIA KERNEL MEAN APPROXIMATION WITH RANDOM FEATURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale deep learning is often trained on highly redundant datasets, where many examples provide similar learning signals. This raises a natural question: can we reduce training compute by replacing full-data SGD with a small coreset that adapts as the model representation evolves?

We propose DySk (Dynamic Sketching), a weighted dynamic coreset method based on loss-aware kernel mean approximation. At periodic refresh steps, DySk computes current representations and losses, builds a loss-weighted random Fourier feature sketch of the representation distribution, and selects a compact support by greedy residual matching. It then fits nonnegative simplex weights on the selected support and trains on the resulting weighted coreset until the next refresh. This yields a forward-pass-only refresh mechanism that tracks the evolving training distribution without requiring per-example backward gradients for selection.

Empirically, DySk preserves accuracy while substantially reducing the amount of data processed by SGD, yielding a favorable accuracy–runtime trade-off against gradient-matching and other coreset baselines. We further provide ablations and a conditional sketch-to-gradient analysis supporting the empirical and theoretical behavior of DySk.

1 INTRODUCTION

Modern deep learning often relies on training over increasingly large datasets, yet many datasets contain substantial redundancy: many examples provide similar learning signals once represented by the current model. This raises a simple question: *can we reduce the number of training examples processed by SGD while preserving most of the final accuracy?*

We study this question through the lens of dynamic coreset training. Throughout the paper, we measure training compute by the number of processed training examples, or equivalently by the number of SGD updates at fixed batch size. This provides a hardware-agnostic view of the compute–accuracy trade-off. A small coreset is useful only if it preserves the relevant training signal of the full dataset. Since neural representations evolve during training, this coreset should also be refreshed dynamically rather than selected once at initialization.

Existing data-selection methods use losses, forgetting statistics, gradient norms, or gradient-matching objectives to identify useful examples (Toneva et al., 2019; Paul et al., 2021; Mirzasoleiman et al., 2020; Killamsetty et al., 2021a;b; Hosseini et al., 2025). However, gradient-based methods can be expensive, static methods may become stale as representations change, and many approaches do not explicitly preserve distributional coverage in the learned representation space. We seek a method that is dynamic, forward-pass-only at refresh time, and able to balance *coverage* with *difficulty*.

We introduce **DySk** (**Dy**namic **Sk**etching), a weighted dynamic coreset method for efficient SGD. At periodic refresh steps, we extract current representations and losses for all training examples using forward passes only. We then build a loss-weighted random Fourier feature sketch of the representation distribution, select a sparse support through a loss-density-aware greedy residual-matching

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.

Table 1: **Summary of DySk improvements on CIFAR.** Accuracy gains are reported in percentage points and averaged over budgets 5%–30%. Runtime reductions are measured relative to the fastest non-random dynamic coreset baseline at the same budget.

Dataset	vs. Random	vs. best coreset	Runtime reduction
CIFAR-10	$+3.8$ pp	$+1.4$ pp	46.3%
CIFAR-100	$+6.1$ pp	$+4.3$ pp	47.2%

procedure, and fit nonnegative simplex weights on the selected support. Training until the next refresh uses the resulting weighted coreset distribution. In this sense, DySk is an online sketching procedure that tracks a compact summary of the evolving training distribution in representation space.

Table 1 summarizes the main empirical takeaways on CIFAR. Across CIFAR-10 and CIFAR-100, DySk improves accuracy over random selection and other coreset baselines, while being substantially faster than gradient- and bilevel-based dynamic coreset methods in our controlled implementation. We further evaluate DySk on ImageNet and language-model pretraining, where it maintains strong low-budget performance, demonstrating its usefulness across tasks and scales. Our main contributions are:

- We propose **DySk**, a weighted dynamic coreset method based on random-feature kernel mean sketches in evolving representation spaces.
- We introduce a forward-pass-only refresh procedure that matches a random-feature representation sketch, uses loss and density signals to guide support construction, and refits simplex weights on the selected support.
- We prove a conditional guarantee: when gradients are locally predictable from sketch features, sketch matching controls the weighted-gradient error.
- We evaluate DySk across image classification and language-model pretraining settings, with accuracy-compute and wall-clock comparisons, dynamic coreset baselines, and ablations of the main design choices.

2 Related Work

Data-selection methods differ in both the information they use to select examples and the quantity they attempt to preserve during training. At one extreme, score-based methods assign an independent utility to each example and retain those deemed most informative. At the other, coreset and matching methods explicitly optimize a set-level objective so that training on the selected subset approximates some property of the full dataset. A separate but related line of work preserves geometric or distributional structure in a feature space. DySk lies at the intersection of the latter two views: it constructs a dynamic coreset by matching a compact representation of the evolving feature distribution, while requiring only forward-pass quantities at refresh time.

Score-based and training-aware selection. A broad family of methods ranks training examples using quantities such as loss, confidence, prediction error, forgetting events, or gradient-related surrogates. Forgetting-based selection tracks whether examples transition from correct to incorrect predictions during training (Toneva et al., 2019), while GraNd and EL2N use early-training gradient norms or prediction errors to identify examples expected to have a larger influence on optimization (Paul et al., 2021). RHO-LOSS instead prioritizes examples according to their estimated reducible holdout loss (Mindermann et al., 2022), and CSReL uses reducible loss to estimate the value of adding an example to a coreset (Tong et al., 2025). Such scores can provide lightweight signals for prioritization and can be recomputed as training evolves. However, ranking examples individually does not directly enforce that the selected set preserves the diversity or distributional structure of the full training population. This distinction is particularly relevant at small budgets, where selecting many individually difficult but mutually redundant examples can leave substantial regions of the representation space uncovered.

Optimization-matching and dynamic coresets. Coreset methods instead define a set-level objective intended to preserve the optimization behavior of full-data training. CRAIG formulates subset selection through a facility-location objective in gradient space (Mirzasoleiman et al., 2020), while

GradMatch directly matches the aggregate gradient of the selected subset to the full-data gradient (Killamsetty et al., 2021a). GLISTER selects examples through a bilevel objective based on their effect on a validation loss (Killamsetty et al., 2021b). More recent work also emphasizes the need to update or reuse coresets as the optimization state changes. CREST constructs and reuses coresets while controlling their approximation quality (Yang et al., 2023), whereas ERACS monitors a representativeness criterion derived from gradient signal-to-noise and triggers reselection when the current subset degrades (Cheng et al., 2025). These methods provide a direct connection between subset selection and optimization, but gradient-based matching or bilevel objectives can make refresh steps substantially more expensive. DySk targets a related set-level approximation problem while avoiding per-example backward gradients during coreset construction.

Representation and distribution matching. A complementary line of work seeks to preserve the geometry or distribution of the data in a feature space rather than directly matching gradients. Kernel herding and its conditional-gradient interpretations construct sparse approximations of kernel mean embeddings (Welling, 2009; Bach et al., 2012). Random Fourier features make such kernel representations explicit and finite-dimensional for shift-invariant kernels (Rahimi & Recht, 2007), while dataset sketching studies compact moment- and random-feature representations of large empirical distributions (Gribonval et al., 2021). These ideas provide the closest conceptual basis for DySk. The main difference is that the feature distribution in deep learning is not fixed: the representation map itself evolves throughout training. DySk therefore performs sketch matching repeatedly in the current model representation space, using the resulting residual objective to construct a sparse support and subsequently refitting nonnegative weights on that support.

This perspective also clarifies the distinction from purely score-based and gradient-matching approaches. DySk explicitly targets set-level coverage in representation space rather than ranking examples independently, while its refresh procedure requires current representations and losses but no per-example backward gradients. Loss and density information can additionally bias the sketch target or guide support search, but the underlying matching objective is defined in the representation sketch space. A broader discussion of static pruning, training-aware selection, importance sampling, distribution matching, and data selection for language models is provided in Appendix C.

3 DySk: Dynamic Representation Sketching for Weighted Coreset Training

We propose *DySk* (**Dy**namic **Sk**etching), a training-time coreset method that accelerates SGD by periodically replacing full-data training with a small weighted subset of examples. At each refresh, DySk performs a forward pass over the training set, extracts current representations and losses, builds a random-feature sketch of the current representation distribution, selects a sparse support by greedy residual matching, and fits nonnegative simplex weights on the selected support. Between refreshes, training uses weighted sampling from this coreset rather than uniform sampling from the full dataset. Our goal is to preserve *representation-space coverage* while separating support selection from training weights. The support is chosen so that its random-feature atoms approximate the current sketch target, optionally using loss and density information to guide the search. Final training probabilities are then refit on the selected support to match the sketch target. This avoids using greedy residual coefficients directly as training weights and makes fallback or carry-over examples part of the same final weighted approximation.

3.1 Weighted dynamic coreset training

Let $\mathcal{D} = \{z_i = (x_i, y_i)\}_{i=1}^n$ be the training set. We denote by f_θ the neural network with trainable parameters θ, and write the per-example loss as $\ell_i(\theta) := \ell(f_\theta(x_i), y_i)$. The full empirical objective is $L(\theta) = \frac{1}{n}\sum_{i=1}^n \ell_i(\theta)$. DySk replaces this objective between refreshes by a weighted coreset objective. At refresh time t, it constructs a support $I_t \subseteq \{1, \dots, n\},\ |I_t| \le k$, and a probability vector $q^{(t)} \in \Delta_n$ supported on I_t. Training until the next refresh uses $L_{I_t,q_t}(\theta) = \sum_{i \in I_t} q_i^{(t)} \ell_i(\theta)$. In practice, we implement this objective by weighted sampling from I_t with replacement. Alternatively, one may optimize the same weighted objective using per-example weighted losses.

DySk can begin with a full-data warm start. Let E be the total number of epochs and let $\rho = k/n$ be the nominal coreset fraction. In our experiments, the default warm-start length is $E_{\text{warm}} =$

$\mathrm{round}(\kappa_{\mathrm{warm}} E\rho)$, unless an explicit value is provided. This convention keeps the warm-start length proportional to the nominal amount of coreset training. After warm-up, DySk refreshes the coreset every P epochs. At refresh time t, with current parameters θ_t, DySk computes for every example $h_i^{(t)} = g_{\theta_t}(x_i) \in \mathbb{R}^d$, $\ell_i^{(t)} = \ell_i(\theta_t)$, where g_{θ_t} is the representation map induced by the current network. In the vision experiments, we use the penultimate-layer representation. This refresh pass is forward-only: DySk does not compute per-example backward gradients during coreset construction.

3.2 Random-feature sketch target

DySk summarizes the current representation distribution through a finite-dimensional approximation of a kernel mean embedding. This follows the dataset-sketching viewpoint of Gribonval et al. (2021): instead of explicitly matching all examples, DySk matches a compact sketch of the distribution in representation space.

We use a Gaussian kernel $\kappa_\sigma(h, h') = \exp\left(-\frac{\|h-h'\|_2^2}{2\sigma^2}\right)$, where the bandwidth σ controls locality. Smaller σ emphasizes local neighborhoods in representation space, while larger σ produces smoother summaries. We approximate this kernel with random Fourier features, $\Phi_\sigma(h) = \sqrt{\frac{2}{m}}\left(\cos\left(\frac{\omega_1^\top h}{\sigma} + b_1\right), \ldots, \cos\left(\frac{\omega_m^\top h}{\sigma} + b_m\right)\right)$, with $\omega_r \sim \mathcal{N}(0, I_d)$, $b_r \sim \mathrm{Unif}[0, 2\pi]$. Thus, $\Phi_\sigma(h)^\top \Phi_\sigma(h') \approx \kappa_\sigma(h, h')$. The random-feature map is sampled once and reused across refreshes. At refresh t, each example is represented by a sketch atom $v_i^{(t)} = \Phi_\sigma(h_i^{(t)}) \in \mathbb{R}^m$.

DySk defines the current sketch target as a weighted mean of the sketch atoms, $\mu_t = \sum_{i=1}^n a_i^{(t)} v_i^{(t)}$, $a^{(t)} \in \Delta_n$. The uniform choice $a_i^{(t)} = 1/n$ directly approximates the empirical representation distribution. Our default implementation additionally biases this target toward examples with larger current loss. Let $\bar{\ell}^{(t)} = \frac{1}{n}\sum_{j=1}^n \ell_j^{(t)}$. We define clipped relative losses $\tilde{a}_i^{(t)} = \mathrm{clip}\left(\frac{\ell_i^{(t)}}{\bar{\ell}^{(t)}+\varepsilon}, 0, c_{\max}\right)$, $a_i^{(t)} = \frac{\tilde{a}_i^{(t)}}{\sum_{j=1}^n \tilde{a}_j^{(t)}}$. The clipping constant prevents a small number of extreme-loss examples from dominating the target.

In the experiments considered here, we use

$$s_t = \mu_t,$$

and denote by s_t the sketch target passed to the support-selection procedure.

3.3 Support selection

The sketch s_t defines the target that DySk wants the coreset to approximate. Our default support-search procedure additionally uses two signals to guide greedy residual matching: a density signal that favors representative regions of the current embedding space, and a difficulty signal that favors examples with nontrivial loss.

The raw sketch-density score is $\bar{d}_i^{(t)} = \langle v_i^{(t)}, s_t \rangle$. We standardize these scores over the training set and map them to $[0, 1]$: $d_i^{(t)} = \mathrm{sigmoid}\left(\frac{\bar{d}_i^{(t)} - \mathrm{mean}_j\, \bar{d}_j^{(t)}}{\mathrm{std}_j\, \bar{d}_j^{(t)} + \varepsilon}\right)$. Similarly, the difficulty score is computed from the current losses: $b_i^{(t)} = \mathrm{sigmoid}\left(\frac{\ell_i^{(t)} - \mathrm{mean}_j\, \ell_j^{(t)}}{\mathrm{std}_j\, \ell_j^{(t)} + \varepsilon}\right)$. The final support-search score is $\pi_i^{(t)} = d_i^{(t)} b_i^{(t)} \mathbf{1}_{\mathrm{loss}}^{(t)}(i) \mathbf{1}_{\mathrm{dens}}^{(t)}(i)$. The loss gate removes trivially easy examples and extreme loss outliers: $\mathbf{1}_{\mathrm{loss}}^{(t)}(i) = \mathbf{1}\left\{\ell_i^{(t)} \in [Q_\ell(q_{\mathrm{easy}}), Q_\ell(q_{\mathrm{out}})]\right\}$, where $Q_\ell(q)$ is the empirical q-quantile of the current losses. The density gate removes very low-density points, $\mathbf{1}_{\mathrm{dens}}^{(t)}(i) = \mathbf{1}\left\{d_i^{(t)} \geq Q_d(q_{\mathrm{dens}})\right\}$. These gates are not used to define the final weights; they only guide the greedy support search.

Starting from $r_0 = s_t$ and $I_0 = \emptyset$, DySk greedily selects atoms in sketch space. At greedy iteration

Algorithm 1 Weighted Dynamic Sketching (DySk)

Require: Dataset $\mathcal{D}$, model f_θ, budget k, refresh period P, sketch map Φ_σ, mixing parameter λ_{mix}

Train on the full dataset for epochs $e = 1, \ldots, E_{\text{warm}}$, if $E_{\text{warm}} > 0$.
for epoch $e = E_{\text{warm}} + 1, \ldots, E$ **do**
  if e is a refresh epoch **then**
    Compute $h_i^{(t)}$, $\ell_i^{(t)}$, and $v_i^{(t)} = \Phi_\sigma(h_i^{(t)})$ for all i.
    Build the random-feature sketch target s_t (clipped loss-weighted by default).
    Compute density scores $d_i^{(t)}$, difficulty scores $b_i^{(t)}$, and support scores $\pi_i^{(t)}$.
    Select a support by greedy residual matching.
    Complete the support with carry-over and fallback if enabled.
    Fit simplex weights $w^{(t)}$ on the final support I_t.
    Form training probabilities $q^{(t)}$ by optional mixing with a uniform floor.
  end if
  Train one epoch using weighted sampling from I_t with probabilities $q^{(t)}$.
end for

j, it chooses $i_j \in \arg\max_{i \notin I_{j-1}} \pi_i^{(t)} \frac{\langle v_i^{(t)}, r_{j-1} \rangle}{\|v_i^{(t)}\|_2 + \epsilon}$. The selected atom is then projected out using

$$\eta_j = \frac{\langle v_{i_j}^{(t)}, r_{j-1} \rangle}{\|v_{i_j}^{(t)}\|_2^2 + \epsilon}, \qquad r_j = r_{j-1} - \eta_j v_{i_j}^{(t)}.$$

Thus, support selection uses normalized residual alignment, whereas the residual update uses the corresponding one-dimensional least-squares projection coefficient. The coefficient η_j is used only to update the residual and is not used as a training weight. The greedy procedure stops when the budget is reached, the relative residual is below a tolerance, or no remaining candidate has positive alignment with the residual.

3.4 Fallback, carry-over, and simplex weights

The greedy residual procedure may stop before reaching the nominal budget k. This can happen when the residual is already small or when no remaining atom has positive alignment. In the forced-budget setting used in the final experiments, DySk still returns a coreset of the requested size by filling the remaining slots with a fallback policy. Our default fallback policy, `top_importance`, adds the remaining examples with the largest support-search scores $\pi_i^{(t)}$. This fallback only completes the support; it does not assign final training probabilities.

DySk can also retain a fraction ρ_{keep} of the previous support to reduce abrupt changes between refreshes. Carried-over examples are treated as part of the candidate support at the new refresh. They do not keep their old weights. After the final support I_t is formed, all weights are refit from the current sketch. Given I_t, DySk fits the actual coreset weights by simplex least squares: $w^{(t)} \in \arg\min_{w \in \Delta(I_t)} \left\| s_t - \sum_{i \in I_t} w_i v_i^{(t)} \right\|_2^2$, where $\Delta(I_t) = \left\{ w \in \mathbb{R}^n : w_i \geq 0,\ \sum_{i \in I_t} w_i = 1,\ w_i = 0 \text{ for } i \notin I_t \right\}$. We solve this small convex problem by projected-gradient steps on the simplex. The resulting distribution is then optionally smoothed toward a small uniform floor: $q_i^{(t)} = \frac{\lambda_{\text{mix}} w_i^{(t)} + (1 - \lambda_{\text{mix}})/n}{\sum_{j \in I_t} \left(\lambda_{\text{mix}} w_j^{(t)} + (1 - \lambda_{\text{mix}})/n \right)}$, $i \in I_t$, and $q_i^{(t)} = 0$ for $i \notin I_t$. This floor reduces sampling variance and prevents the fitted distribution from becoming overly concentrated on a very small number of selected examples. Importantly, fallback and carry-over examples are not uniform padding: after refitting, they may receive large, small, or zero probability depending on whether they help approximate the current sketch target s_t.

3.5 Computational cost

A refresh consists of four main steps. First, DySk performs one full forward pass to compute representations and losses, with cost $O(nC_{\text{fwd}})$. Second, it computes random features, with cost $O(nm)$. Third, greedy residual matching requires $O(k_{\text{gr}} nm)$, where $k_{\text{gr}} \leq k$ is the number of actual greedy

iterations. Finally, simplex refitting costs $O(T_{\text{fit}}km)$, where T_{fit} is the number of projected-gradient iterations. Thus, the refresh cost is $O(nC_{\text{fwd}} + nm + k_{\text{gr}}nm + T_{\text{fit}}km)$. Between refreshes, each epoch processes k sampled examples rather than n. DySk therefore trades periodic forward-only refresh overhead for fewer backward updates. Unlike gradient-matching coreset methods, it does not require per-example backward gradients during refresh. This distinction is important in practice: the expensive part of training is usually the backward pass, and DySk avoids running it over the full dataset during selection.

4 From Sketch Matching to Gradient Approximation

We study what the DySk sketch controls. We first show that, if per-example gradients are locally predictable from the sketch features at refresh time, then matching the sketch target approximates the corresponding target-weighted gradient. We then propagate this approximation between refreshes and obtain a conditional stationarity bound for vanilla SGD. Proofs and additional results are deferred to Appendix D.

4.1 Sketch matching and gradient approximation

Fix a refresh time t, and let $v_i^{(t)} = \Phi(h_i^{(t)}) \in \mathbb{R}^m$. For a target distribution $a^{(t)} \in \Delta_n$, define $\mu_t = \sum_{i=1}^n a_i^{(t)} v_i^{(t)}$, and let $q^{(t)} \in \Delta_n$ denote the DySk training distribution. Its sketch mismatch is $\Delta_t(q) := \left\| \mu_t - \sum_{i=1}^n q_i^{(t)} v_i^{(t)} \right\|_2$. For any $p \in \Delta_n$, write $G_p(\theta) := \sum_{i=1}^n p_i \nabla \ell_i(\theta)$.

Assumption 4.1 (Local sketch-gradient predictability)**.** At refresh time t, there exist $A_t \in \mathbb{R}^{p \times m}$ and residuals $\rho_i^{(t)}$ such that $\nabla \ell_i(\theta_t) = A_t v_i^{(t)} + \rho_i^{(t)}, \qquad \|\rho_i^{(t)}\|_2 \le \zeta_t$.

Theorem 4.2 (Sketch-to-gradient approximation)**.** *Under Assumption 4.1,*

$$\|G_a(\theta_t) - G_q(\theta_t)\|_2 \le \|A_t\|_{\text{op}} \Delta_t(q) + \zeta_t \|a^{(t)} - q^{(t)}\|_1.$$

For the uniform target $a^{(t)} = u$, $u_i = 1/n$, $G_a = \nabla L$, so sketch matching directly controls the full-data gradient: $\|\nabla L(\theta_t) - G_q(\theta_t)\|_2 \le \|A_t\|_{\text{op}} \Delta_t(q) + 2\zeta_t$. For a nonuniform target, including our clipped loss-weighted target, an additional target-weighting bias $\|G_u(\theta_t) - G_a(\theta_t)\|_2$ appears. The complete decomposition, including sketch smoothing, is given in Appendix D.

4.2 Optimization consequence

The previous bound holds at a refresh point. Between refreshes, DySk keeps its training distribution q_r fixed while the parameters evolve. Let τ_r denote refresh r. Under local Lipschitz regularity of the gradient discrepancy, for $t \in [\tau_r, \tau_{r+1})$,

$$\|G_{q_r}(\theta_t) - \nabla L(\theta_t)\|_2 \le M_r \delta_r + \zeta_r \|q_r - u\|_1 + \Lambda_r \|\theta_t - \theta_{\tau_r}\|_2,$$

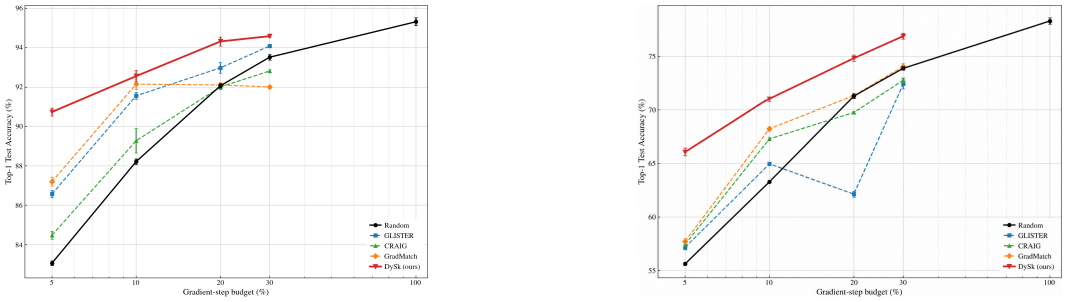
where $M_r = \|A_{\tau_r}\|_{\text{op}}$ and δ_r is the sketch mismatch at refresh time. The last term captures fixed-support staleness. Assuming L is L_s-smooth, minibatch gradients are conditionally unbiased under q_r with variance v_t, and $\eta_t \le 1/(4L_s)$, vanilla SGD satisfies

$$\frac{1}{S_T} \sum_{t<T} \eta_t \, \mathbb{E}\|\nabla L(\theta_t)\|_2^2 \le \frac{4\Delta_0}{S_T} + \frac{3}{S_T} \sum_{t<T} \eta_t \mathbb{E}[B_t^2] + \frac{2L_s}{S_T} \sum_{t<T} \eta_t^2 v_t,$$

where $B_t = M_r \delta_r + \zeta_r \|q_r - u\|_1 + \Lambda_r \|\theta_t - \theta_{\tau_r}\|_2$. Thus, the optimization error separates into sketch approximation, local sketch-to-gradient residual, fixed-support staleness, and minibatch variance. This is a conditional guarantee for vanilla SGD; it does not establish global convergence of the deep networks used in our experiments.

5 Experimental Results

We evaluate DySk across image classification and language-model pretraining. Our main image-classification experiments use ResNet-18 on CIFAR-10, CIFAR-100, and ImageNet-1k. We additionally evaluate the same sketch-based selection mechanism in Transformer pretraining. Unless



(a) CIFAR-10 (b) CIFAR-100

Figure 1: **Data selection on CIFAR-10 and CIFAR-100.** Top-1 test accuracy of ResNet-18 as a function of the nominal coreset budget. Results are reported as mean ± standard deviation over 5 random seeds. The horizontal axis is logarithmic. All coreset methods use the same warm-start protocol, whose full-data epochs are included in the compute accounting. The 100% point corresponds to standard full-data training. Complete numerical results and wall-clock runtimes are reported in Appendix E.1.

stated otherwise, methods within each controlled comparison use the same preprocessing, optimization protocol, and training horizon. DySk uses the weighted dynamic coreset procedure from Section 3; implementation details and default hyperparameters are given in Appendix A.

Protocol. For CIFAR-10 and CIFAR-100, all methods are trained for 300 epochs with ResNet-18 and evaluated at matched nominal coreset budgets. We report Top-1 test accuracy and wall-clock training time. Runtimes are measured on a single A100 GPU and include the full training procedure. For DySk, this includes full-data warm-start epochs, refresh forward passes, sketch construction, support selection, fallback and carry-over processing, simplex weight fitting, and weighted coreset training. Full-data warm-start epochs are included in all reported compute budgets.

5.1 CIFAR-10/100 Data Selection

Figure 1 compares DySk with Random-Warm, GLISTERPB-Warm, CRAIGPB-Warm, and GradMatchPB-Warm. Across both datasets, DySk provides a favorable accuracy-budget trade-off in the 5%–30% regime. On CIFAR-10, DySk reaches 90.74% at 5% budget, compared with 83.06% for Random-Warm, and reaches 94.59% at 30%, within 0.73 points of full-data training. On CIFAR-100, DySk reaches 65.82% at 5%, compared with 55.63% for Random-Warm, and reaches 76.88% at 30%, within 1.41 points of full-data training.

DySk also remains substantially cheaper than gradient- and bilevel-based dynamic coreset methods in our controlled implementation. Complete wall-clock comparisons are reported in Appendix E.1. Random selection has negligible selection overhead, but generally yields lower accuracy at aggressive budgets.

5.2 What Drives DySk? Representation Coverage vs. Difficulty

The central design question is whether DySk benefits from representation-space coverage itself, rather than merely from prioritizing difficult examples. We therefore compare against two matched forward-pass selectors on CIFAR-100: Loss-only, which retains DySk's current loss signal and gates but removes representations, random features, residual matching, and simplex fitting; and Dynamic EL2N, which recomputes a prediction-error score at each refresh and selects the highest-scoring examples. All methods use the same optimization protocol, warm start, refresh period, budgets, and five random seeds. DySk improves over both Loss-only and Dynamic EL2N. This isolates a benefit from matching representation-space coverage beyond difficulty or prediction-error ranking. We also test whether the loss-weighted sketch target itself is necessary. In a matched target ablation, replacing the default loss-weighted target by the uniform target $a_i = 1/n$ gives $65.80 \pm 0.31\%$ and $70.83 \pm 0.21\%$ at 5% and 10%, respectively. These values are numerically similar to the corresponding loss-weighted runs and lie within the observed run-to-run variation. A

Table 2: **Matched forward-pass controls on CIFAR-100.** Top-1 test accuracy (%, mean ± standard deviation over 5 seeds).

Method	5%	10%
DYNAMIC EL2N	60.79 ± 0.29	68.93 ± 0.19
LOSS-ONLY	61.88 ± 0.53	69.33 ± 0.47
DYSK	$\mathbf{65.82 \pm 0.16}$	$\mathbf{70.76 \pm 0.39}$

Table 3: **ImageNet-1k results with ResNet-18.** Top-1 validation accuracy (%, mean ± standard deviation over 3 seeds).

Method	5%	10%
RANDOM-WARM	56.23 ± 0.06	60.93 ± 0.05
DYSK-DIRECT	58.75 ± 0.12	62.60 ± 0.05
DYSK-CORE	$\mathbf{59.14 \pm 0.01}$	$\mathbf{62.95 \pm 0.16}$
FULL	70.36 ± 0.05	

matched comparison with dynamic k-center coverage is reported in Appendix B.1. Together with the forward-pass controls, this points to representation-space coverage as the more robust component of DySk; loss weighting acts as an additional bias rather than the sole source of the gain.

5.3 SCALING TO IMAGENET-1K

We evaluate DySk on ImageNet-1k with ResNet-18 for 90 epochs at nominal 5% and 10% subset fractions. We consider two instantiations. DYSK-DIRECT applies the weighted selector used on CIFAR directly at ImageNet scale, whereas DYSK-CORE limits residual matching to a fixed 256-example coverage core, completes the support by class-balanced uniform sampling, and trains with uniform weights. Random and both DySk variants use the same budget-specific full-data warm start.

Both DySk variants improve over random subset training. The scalable core construction is also slightly more accurate than the direct weighted implementation at both budgets, while decoupling the sketch-matching cost from the nominal subset size. Detailed protocols and refresh-time scaling measurements are reported in Appendix E.2.

5.4 TRANSFORMER PRETRAINING

We next evaluate DySk in autoregressive Transformer pretraining using a nanochat-style model with approximately 286M parameters. We compare the direct weighted implementation, DYSK-DIRECT, with the scalable DYSK-CORE construction, which selects a small representation-coverage core and completes the support by uniform sampling. All methods use matched backward-token budgets.

Both DySk variants improve over loss-only and random subset training, and the direct and core implementations obtain nearly identical BPB. Thus, restricting sketch matching to a small coverage core preserves the observed benefit in this experiment while reducing the size of the greedy selection problem. Detailed protocol information is reported in Appendix E.3.

5.5 SENSITIVITY AND COMPUTATIONAL SCALING

Detailed hyperparameter sweeps are reported in Appendix B. On CIFAR-100, accuracy varies modestly across random-feature dimensions from $m \in [64, 1024]$, while refresh cost increases substantially with m. We use $m = 256$ as an empirical accuracy–cost compromise. Single-seed sweeps further characterize sensitivity to refresh frequency, carry-over, and residual tolerance. Warm-start duration also changes the total optimization budget. Appendix B.2 reports both accuracy and the corresponding full-data-equivalent epochs, including all full-data warm-start updates. Appendix E.4 reports published ERACS (Cheng et al., 2025) CIFAR accuracies alongside DySk runs using the same nominal refresh period. The values come from different codebases and training protocols and therefore do not constitute a controlled comparison.

Table 4: **Transformer pretraining.** Validation bits-per-byte (BPB; lower is better). Results are from one run per configuration.

Method	5%	10%
RANDOM	1.1611	0.9829
LOSS-ONLY	0.9474	0.9197
DYSK-DIRECT	0.9316	0.9069
DYSK-CORE	**0.9309**	**0.9057**

6 DISCUSSION

The empirical results highlight DySk as an effective middle ground between random subset training and more expensive gradient-based coreset selection. On CIFAR-10 and CIFAR-100, DySk consistently improves the accuracy–budget trade-off over standard dynamic coreset baselines, with the largest gains appearing on CIFAR-100 and at smaller budgets. This is the regime where subset quality matters most: random selection is inexpensive, but its accuracy can degrade sharply, whereas gradient- and bilevel-based methods improve selection quality at a higher computational cost.

DySk achieves this trade-off by maintaining an explicit representation-space coverage objective using forward-pass statistics. The mechanism controls in Section 5.2 show that representation matching contributes beyond loss-only and prediction-error ranking. Uniform-target and coverage-only variants remain close to the default configuration, indicating that loss weighting and difficulty-aware support guidance act as secondary biases. The final simplex fit converts the selected support into a weighted coreset and assigns current weights to all selected examples, including fallback and carry-over points.

The ablations further suggest that DySk behaves predictably under practical design choices. Warm-start sweeps show that accuracy depends on the length of the initial full-data phase, but longer warm starts also increase the full-data-equivalent optimization budget. We therefore report this dependence jointly with compute. The internal hyperparameter sweeps show that strong performance does not depend on a single fragile setting: moderate random-feature dimension, small carry-over, and periodic refreshes are sufficient to obtain competitive results. Published ERACS Cheng et al. (2025) results are reported in Appendix E.4; differences in implementation and training protocol prevent a controlled algorithmic comparison.

7 CONCLUSION

We introduced DySk, a dynamic weighted coreset method based on random-feature sketches of the evolving representation distribution. At each refresh, DySk constructs a compact support from current forward-pass representations, optionally uses loss and density information to guide selection, fits nonnegative simplex weights on the selected support, and trains on the resulting coreset until the next refresh. This avoids per-example backward-gradient computations during selection while maintaining an explicit representation-space coverage objective.

On CIFAR-10 and CIFAR-100 with ResNet-18, DySk achieves a strong accuracy–budget trade-off across nominal budgets from 5% to 30%. The gains are largest on CIFAR-100 and in low-budget regimes, where random selection and existing dynamic coreset baselines are less reliable. DySk also compares favorably with the published CIFAR results of ERACS under a matched refresh-period protocol, although runtime comparisons to ERACS remain indirect because its code is not publicly available. Our ablations further show that warm-starting is an important stabilizer and that DySk performs robustly across reasonable choices of its main hyperparameters.

The main remaining limitations concern scaling and representation dependence. Kernel bandwidth must be adapted when representation scale changes, the sketch dimension required for increasingly complex representation spaces is not characterized theoretically, and full-corpus forward scans can become expensive when refreshes cannot be sufficiently amortized. These limitations motivate adaptive kernel choices, structured or more efficient sketches, and cheaper refresh strategies.

AI USE STATEMENT

Generative AI tools were used for language editing and debugging support. All AI-assisted text were reviewed by the authors. The authors take full responsibility for the paper's claims, results, citations, and final content.

ETHICS STATEMENT

This work studies computationally efficient training through dynamic data selection and does not involve human subjects or the collection of new personal or sensitive data. As authors, we thus do not think this submission raises questions regarding the Code of Ethics. However, as with other data-selection methods, coreset construction changes which training examples influence the optimization process and may therefore interact with biases or under-represented groups present in a dataset. Such effects are not evaluated in this work and should be considered when applying the method in domains where representation or subgroup fairness is important.

REPRODUCIBILITY STATEMENT

The algorithmic specification is given in Section 3, with implementation details, default hyperparameters, optimization settings, and sampling procedures documented in Appendix A. Complete proofs for the theoretical results are provided in Appendix D, and additional experimental protocols, ablations, and scaling measurements are reported in the appendices. The code will be released upon publication.

REFERENCES

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in SGD by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

Francis Bach, Simon Lacoste-Julien, and Guillaume Obozinski. On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, pp. 1355–1362, 2012.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4750–4759, 2022.

Yicheng Chen, Yining Li, Kai Hu, Ma Zerun, Ye Haochen, and Kai Chen. MIG: Automatic data selection for instruction tuning by maximizing information gain in semantic space. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9902–9915, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.515. URL https://aclanthology.org/2025.findings-acl.515/.

Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 109–116, 2010.

Zihao Cheng, Binrui Wu, Zhiwei Li, Yuesen Liao, Su Zhao, Shuai Chen, Yuan Gao, and WEIZHONG ZHANG. Efficient representativeness-aware coreset selection. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=CrdrVdXCui.

Rémi Gribonval, Antoine Chatalic, Nicolas Keriven, Vincent Schellekens, Laurent Jacques, and Philip Schniter. Sketching Data Sets for Large-Scale Learning: Keeping only what you need. *IEEE Signal Processing Magazine*, 38(5):12–36, September 2021. doi: 10.1109/MSP.2021.3092574. URL https://inria.hal.science/hal-03350599. For an extended version of this article that contains additional references and more in-depth discussions on a variety of topics, see the long version published on HAL (https://hal.inria.fr/hal-02909766) and arXiv (2008.01839). See also the associated python notebook https://gitlab.inria.fr/SketchedLearning/spm-notebook.

Sayed Mohammadreza Tayaranian Hosseini, Seyyed Hasan Mozafari, Brett H Meyer, James J Clark, and Warren J Gross. Automatic pruning of fine-tuning datasets for transformer-based language models. In *Conference on Lifelong Learning Agents*, pp. 1–28. PMLR, 2025.

Ferenc Huszár and David Duvenaud. Optimally-weighted herding is bayesian quadrature. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 377–386, 2012.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018.

Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pp. 5464–5474. PMLR, 2021a.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8110–8118, 2021b.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14255–14273, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.769. URL https://aclanthology.org/2024.acl-long.769/.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BTKAeLqLMw.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15630–15649. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/mindermann22a.html.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020.

Brian B Moser, Arundhati S Shanbhag, Stanislav Frolov, Federico Raue, Joachim Folz, and Andreas Dengel. A coreset selection of coreset selection literature: Introduction and recent advances. *arXiv preprint arXiv:2505.17799*, 2025.

Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. *Advances in neural information processing systems*, 27, 2014.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607, 2021.

Omead Pooladzandi, David Davini, and Baharan Mirzasoleiman. Adaptive second order coresets for data-efficient machine learning. In *International Conference on Machine Learning*, pp. 17848–17869. PMLR, 2022.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

Seungjae Shin, Heesun Bae, Donghyeok Shin, Weonyoung Joo, and Il-Chul Moon. Loss-curvature matching for dataset selection and condensation. In *International Conference on Artificial Intelligence and Statistics*, pp. 8606–8628. PMLR, 2023.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

Ruilin Tong, Yuhang Liu, Javen Qinfeng Shi, and Dong Gong. Coreset selection via reducible loss in continual learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mAztx8QO3B.

Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12196–12205, 2022.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1121–1128, 2009.

Weiwei Xiao, Yongyong Chen, Qiben Shan, Yaowei Wang, and Jingyong Su. Feature distribution matching by optimal transport for effective and robust coreset selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 9196–9204, 2024.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Yu Yang, Hao Kang, and Baharan Mirzasoleiman. Towards sustainable learning: Coresets for data-efficient deep learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 39314–39330. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yang23g.html.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6514–6523, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=mSAKhLYLSsl.

A DySk Implementation Details

This appendix specifies the implementation choices used for DySk in the experiments. The method itself is described in Section 3; here we focus on experimental defaults, numerical safeguards, and sampling details. Unless otherwise stated, the final CIFAR experiments use the configuration in Table 5.

Table 5: **Default DySk hyperparameters.** Default values used in the final CIFAR experiments unless otherwise stated.

Hyperparameter	**Default value**
Random-feature dimension m	256
Gaussian bandwidth σ	0.5
Refresh period P	10 epochs
Residual tolerance τ_{res}	0.1
Weight mixing λ_{mix}	0.75
Carry-over fraction ρ_{keep}	0.25
Minimum effective fraction	1.0
Fallback policy	`top_importance`
Warm-up multiplier κ_{warm}	0.5
Easy-loss quantile q_{easy}	0.30
Outlier-loss quantile q_{out}	0.97
Minimum-density quantile q_{dens}	0.20
Loss clipping constant $c_{\max}$	5.0
Simplex fit iterations	64
Simplex fit tolerance	10^{-7}

Refresh schedule and warm start. Let E denote the total number of training epochs and let $\rho = k/n$ denote the nominal coreset fraction. DySk optionally begins with a full-data warm start. In the final experiments, the number of warm-up epochs is

$$E_{\text{warm}} = \text{round}\left(\kappa_{\text{warm}} E \rho\right).$$

If an explicit value of E_{warm} is provided, it overrides this rule. After warm-up, DySk refreshes the coreset every P epochs. At each refresh, the model performs a forward pass over the full training set to compute representations and losses. Between refreshes, the support and weights remain fixed.

We distinguish the nominal coreset fraction from the resulting optimization compute. Ignoring forward-only refresh scans, a run with E_{warm} full-data epochs followed by coreset training at fraction ρ processes

$$E_{\text{FDE}} = E_{\text{warm}} + \rho(E - E_{\text{warm}})$$

full-data-equivalent epochs (FDE). Full-data warm-start epochs are therefore included in all reported optimization-compute budgets. Forward-only refresh scans are not counted as SGD updates, but are included in all wall-clock measurements.

Representations and preprocessing. At refresh time t, DySk computes the representation $h_i^{(t)} = g_{\theta_t}(x_i)$ and loss $\ell_i^{(t)} = \ell(f_{\theta_t}(x_i), y_i)$ for every example. In the vision experiments, g_{θ_t} is the penultimate-layer representation of ResNet-18. The refresh pass is forward-only and does not compute per-example backward gradients.

Unless otherwise stated, DySk applies the random-feature map directly to the extracted representations, without explicit ℓ_2-normalization, whitening, or classwise normalization. The Gaussian bandwidth σ therefore controls similarity in the raw representation scale. We study the sensitivity to this choice in Appendix B.

Random Fourier features. For the Gaussian kernel, DySk uses random Fourier features

$$\Phi_\sigma(h) = \sqrt{\frac{2}{m}} \left(\cos\left(\frac{\omega_1^\top h}{\sigma} + b_1\right), \ldots, \cos\left(\frac{\omega_m^\top h}{\sigma} + b_m\right) \right),$$

where

$$\omega_r \sim \mathcal{N}(0, I_d), \qquad b_r \sim \mathrm{Unif}[0, 2\pi].$$

Equivalently, the effective frequencies ω_r/σ are distributed as $\mathcal{N}(0, \sigma^{-2} I_d)$. The random-feature layer is initialized once before coreset training and reused across refreshes. At each refresh, only the current representations, losses, and sketch statistics are recomputed.

Sketch target. DySk supports any probability distribution over examples as the sketch target. The default CIFAR implementation uses clipped relative-loss weights, as described in Section 3.2. We additionally evaluate the uniform target $a_i = 1/n$ as a controlled ablation. The clipping constant is $c_{\max} = 5.0$, which prevents extreme-loss examples from dominating the default target. The numerical constant ϵ is used only for stability. All reported experiments use the current target directly, $s_t = \mu_t$.

Density, difficulty, and quantile gates. DySk uses a density score $d_i^{(t)}$, a difficulty score $b_i^{(t)}$, and two quantile gates to construct the support-search score $\pi_i^{(t)}$. The density score is computed from the alignment between each atom and the current sketch target, then standardized and mapped to $[0, 1]$. The difficulty score is computed analogously from current losses.

The loss gate keeps examples whose current loss lies between the empirical q_{easy}- and q_{out}-quantiles. The density gate removes examples below the empirical q_{dens}-quantile of the density score. The default values are

$$q_{\mathrm{easy}} = 0.30, \qquad q_{\mathrm{out}} = 0.97, \qquad q_{\mathrm{dens}} = 0.20.$$

Degenerate gates and fallback scores. If the gated support-search scores have zero total mass, the default configuration falls back to the clipped relative-loss importance scores. In coverage-only mode, the corresponding fallback uses the ungated density factor. These safeguards affect only support search; the final training weights are still obtained by simplex refitting on the completed support.

Forced budget and fallback. Let $k = \lceil \rho n \rceil$. In the forced-budget setting, the final support must contain at least

$$k_{\min} = \lceil \mathrm{min_effective_frac} \cdot k \rceil$$

examples. The final experiments use

$$\mathrm{min_effective_frac} = 1.0,$$

so the selected support has the nominal coreset size k.

If greedy residual matching stops with fewer than $k_{\min}$ examples, DySk fills the remaining slots with a fallback policy. The default policy is `top_importance`, which selects the remaining examples with the largest support-search scores $\pi_i^{(t)}$. Fallback examples are not assigned uniform weights by default; after the support is completed, all weights are refit from the current sketch.

Carry-over. DySk can retain part of the previous support to reduce abrupt changes between refreshes. After constructing the current candidate support, we retain up to $\rho_{\mathrm{keep}} k$ examples from the previous support, selecting those with the largest current support-search scores. Duplicate examples are removed, and the remaining slots are filled by newly selected points so that the final support preserves the requested size. The default CIFAR configuration uses

$$\rho_{\mathrm{keep}} = 0.25.$$

After the final support I_t is formed, all simplex weights are refit from the current sketch; carried-over examples do not retain their previous weights.

Simplex weight fitting. Given the final support I_t, DySk fits nonnegative weights by solving the simplex least-squares problem in Section 3.4. We solve this problem by projected-gradient steps on the simplex. The default maximum number of iterations is 64, with stopping tolerance 10^{-7}. The resulting weights are clipped to be nonnegative and renormalized. If the fitted weights are non-finite or collapse to zero mass after clipping, the distribution is reset to uniform on the selected support.

Weight smoothing and sampling. The directly fitted weighted coreset distribution is $q^{(t)} = w^{(t)}$. In practice, we smooth the fitted distribution toward a small uniform floor using $\lambda_{\text{mix}} = 0.75$, as described in Section 3.4. In the implementation, this is done by first mixing the fitted coreset distribution with the full-dataset uniform distribution, then restricting and renormalizing the probabilities on I_t.

Between refreshes, each coreset epoch draws $|I_t|$ examples from the selected support using a weighted random sampler with replacement. Thus, high-probability examples may appear multiple times in one epoch, while low-probability selected examples may be skipped.

CIFAR training protocol. All controlled CIFAR experiments use a CIFAR-adapted torchvision ResNet-18 trained for 300 epochs with batch size 128. We use SGD with learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, and a MultiStepLR schedule with milestones $60/120/160$ and decay factor 0.2. Training augmentation consists of random cropping with four-pixel padding and random horizontal flipping, followed by standard dataset normalization. The same training transform is used during CIFAR refresh scans, so representations and losses are computed from one augmented view of each training example at each refresh. Refresh forward passes use batch size 256.

The CORDS comparisons use the PerBatch variants of CRAIG, GLISTER, and GradMatch. We therefore denote them CRAIGPB-WARM, GLISTERPB-WARM, and GRADMATCHPB-WARM throughout.

ImageNet-1k protocol. We train ResNet-18 on ImageNet-1k for 90 epochs. For scalable support construction, DySk first selects a global sketch-based core and then completes the nominal 5% or 10% support by class-balanced uniform sampling. Training on the completed support uses uniform sampling. Random and DySk use the same budget-specific full-data warm start (5 epochs at 5% and 7 epochs at 10%).

Transformer-pretraining protocol. We additionally evaluate DySk in autoregressive Transformer pretraining with a nanochat-style model of approximately 286M parameters. The experiment processes 5,000 candidate batches and uses 500 initial full-data candidate steps before subset training. Before applying the random-feature map, Transformer representations are ℓ_2-normalized. DySk uses a uniform sketch target to select a coverage core, completes the requested support by uniform sampling, and uses uniform training weights on the completed support. Selected candidate batches trigger optimizer updates according to the 5% or 10% budget. We report validation bits-per-byte (BPB).

Diagnostics. For diagnostic runs, DySk logs the number of greedy-selected and fallback-filled examples, the final residual ratio, the greedy stopping reason, and the fitted-weight residual. These diagnostics are not used during training, but help verify that the selected support is not dominated by fallback points and that the fitted simplex weights approximate the sketch target.

B ABLATIONS AND SENSITIVITY

This section collects the controlled mechanism ablations and sensitivity studies used to characterize DySk. We first isolate representation-space coverage from loss-based difficulty signals, then study sensitivity to the warm start and the main internal hyperparameters. Unless stated otherwise, hyperparameter sweeps without reported standard deviations use a single seed. Multi-seed experiments are reported with their corresponding standard deviations.

B.1 MECHANISM ABLATIONS: COVERAGE VERSUS DIFFICULTY

We provide controlled experiments to isolate the contribution of representation-space coverage from forward-pass difficulty scoring and loss weighting. All comparisons use CIFAR-100 with ResNet-18 under the same 300-epoch optimization protocol, warm start, refresh period, nominal budgets, and five random seeds.

Table 6: **Matched forward-pass controls on CIFAR-100.** Top-1 test accuracy (%, mean ± standard deviation over five random seeds). LOSS-ONLY retains the current loss signal but removes representation sketching, while DYNAMIC EL2N-STYLE recomputes a prediction-error score at every refresh.

Method	**5%**	**10%**
DYNAMIC EL2N-STYLE	60.794 ± 0.291	68.930 ± 0.187
LOSS-ONLY	61.882 ± 0.526	69.328 ± 0.465
DYSK	65.818 ± 0.160	70.758 ± 0.387
UNIFORM-TARGET DYSK	65.802 ± 0.312	70.830 ± 0.214

Forward-pass selection controls. LOSS-ONLY retains DySk's current clipped relative-loss signal and loss gates but removes representations, random Fourier features, sketch construction, residual matching, and simplex fitting. DYNAMIC EL2N-STYLE instead recomputes prediction error at every refresh and selects the current highest-scoring examples. DySk improves over LOSS-ONLY by 3.936 and 1.430 percentage points at 5% and 10%, respectively, and over DYNAMIC EL2N-STYLE by 5.024 and 1.828 points. This comparison isolates a contribution from representation-space coverage beyond loss or prediction-error ranking.

Geometric coverage control. We additionally compare DYSK-CORE with a matched dynamic K-CENTER-CORE baseline. Both methods use the current penultimate-layer representations, a 256-example core budget, uniform completion to the nominal support size, and uniform training weights. K-CENTER-CORE uses Euclidean farthest-first selection, whereas DYSK-CORE uses random-feature kernel-mean residual matching. At 5%, DYSK-CORE reaches $66.750 \pm 0.229\%$, compared with $61.396 \pm 0.290\%$ for K-CENTER-CORE; at 10%, the corresponding values are $70.752 \pm 0.237\%$ and $67.542 \pm 0.269\%$. Thus, sketch-based distribution matching provides a gain under the matched core construction, operating in a 256-dimensional random-feature space rather than the original 512-dimensional representation space.

Uniform versus loss-weighted target. To separate representation coverage from the choice of target weights, we replace the default clipped loss-weighted target by the uniform target $a_i = 1/n$, while leaving the remaining DySk procedure unchanged. The resulting accuracies are $65.802 \pm 0.312\%$ and $70.830 \pm 0.214\%$ at 5% and 10%, respectively, compared with $65.818 \pm 0.160\%$ and $70.758 \pm 0.387\%$ for the default target. Thus, target loss weighting is not necessary to obtain the observed gains.

We additionally evaluate a coverage-only variant that uses the uniform target and removes the loss-difficulty term and easy/outlier loss gates from support search. Relative to default DySk, the paired change in final accuracy is

$$-0.297 \pm 0.238 \qquad \text{and} \qquad -0.460 \pm 0.782$$

percentage points at 5% and 10%, respectively. Together, these controls indicate that representation-space distribution coverage is the more robust component of DySk, while loss weighting and difficulty-aware support guidance act as secondary biases.

B.2 WARM-START SENSITIVITY

We study sensitivity to the warm-start multiplier κ using single-seed sweeps. Increasing κ lengthens the initial full-data phase and increases the total number of full-data SGD updates. The sweep is therefore not compute-matched; the corresponding full-data-equivalent optimization budgets are reported below.

For total training length $E = 300$ and nominal coreset fraction ρ, the warm-start length follows

$$E_{\text{warm}} = \text{round}(\kappa E \rho),$$

and the corresponding optimization budget in equivalent full-data epochs is

$$E_{\text{FDE}} = E_{\text{warm}} + \rho(E - E_{\text{warm}}).$$

Table 7: **Warm-start sensitivity at 10% budget.** Best observed Top-1 test accuracy (%) over training in single-seed runs as the warm-start multiplier κ varies. CORDS baselines use the PerBatch variants with selection every 20 epochs, while DySk uses its default refresh protocol. Increasing κ also increases full-data-equivalent optimization compute, so columns should be interpreted as a sensitivity sweep rather than as compute-matched comparisons.

Dataset	Method	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 0.5$	$\kappa = 0.7$
CIFAR-10	RANDOM-WARM	79.34	85.07	90.06	91.21
	CRAIGPB-WARM	89.18	90.00	92.07	92.06
	GLISTERPB-WARM	90.00	91.00	92.42	92.00
	GRADMATCHPB-WARM	91.00	91.00	92.00	92.01
	DYSK	90.41	90.85	92.61	92.55
CIFAR-100	RANDOM-WARM	38.42	56.00	66.49	69.22
	CRAIGPB-WARM	60.32	63.42	69.15	71.47
	GLISTERPB-WARM	64.00	67.00	71.00	72.28
	GRADMATCHPB-WARM	64.41	67.00	70.27	72.09
	DYSK	64.86	68.41	70.72	72.66

Table 8: **DySk warm-start sensitivity at 5% and 10%.** Best observed Top-1 test accuracy (%) over training in single-seed runs. Larger κ values use more full-data optimization compute and should therefore not be interpreted as compute-free improvements.

Dataset	Budget	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 0.5$	$\kappa = 0.7$
CIFAR-10	5%	66.93	85.98	90.54	90.74
	10%	90.41	90.85	92.61	92.55
CIFAR-100	5%	53.13	60.03	66.75	66.57
	10%	64.86	68.41	70.72	72.66

At 10% budget, the values $\kappa \in \{0, 0.2, 0.5, 0.7\}$ correspond to $E_{\text{warm}} \in \{0, 6, 15, 21\}$ and $E_{\text{FDE}} \in \{30.0, 35.4, 43.5, 48.9\}$.

For the $\kappa = 0$ pilot configuration, the implementation performs one initial full-data epoch before the first support is available. All main DySk experiments use a positive warm start and are unaffected by this initialization detail.

To examine the lower-budget regime separately, we also vary κ for DySk at both 5% and 10%. At 5%, the same sweep corresponds to $E_{\text{warm}} \in \{0, 3, 8, 10\}$ and $E_{\text{FDE}} \in \{15.0, 17.85, 22.6, 24.5\}$.

The sweep confirms that the amount of initial full-data training can have a substantial effect, particularly at low subset fractions. However, later subset construction and additional full-data optimization change simultaneously as κ increases. For example, moving from $\kappa = 0.5$ to 0.7 increases the full-data-equivalent optimization budget from 22.6 to 24.5 epochs at 5%, and from 43.5 to 48.9 epochs at 10%. We therefore treat κ as an accuracy–compute sensitivity parameter and include all full-data warm-start epochs in the reported training-compute budgets.

B.3 INTERNAL HYPERPARAMETER SENSITIVITY

We next study the main internal hyperparameters of DySk. Unless otherwise specified, these experiments use CIFAR-10 with ResNet-18 at a 10% nominal coreset budget. Values reported without $\pm$ correspond to single-seed sweeps and should be interpreted as sensitivity diagnostics rather than precise estimates of expected performance. The final CIFAR configuration uses $\sigma = 0.5$, $\lambda_{\text{mix}} = 0.75$, refresh period $P = 10$, carry-over fraction $\rho_{\text{keep}} = 0.25$, residual tolerance 0.1, and $m = 256$ random Fourier features.

Overall, the sweeps indicate that performance is reasonably stable over a range of internal settings, while some parameters expose clear computational trade-offs. In particular, increasing the sketch dimension m provides little systematic accuracy improvement in the evaluated range but substan-

Table 9: **Bandwidth sensitivity.** Single-seed CIFAR-10 sweeps at 10% budget, together with the final five-seed result at $\sigma = 0.5$.

Setting	Bandwidth σ	Top-1 (%)
Wide sweep	0.5	92.35
	1.0	92.09
	1.5	92.08
	2.0	91.87
	3.0	91.83
	4.0	91.54
Local sweep	0.25	92.64
	0.50	92.66
	0.75	92.57
Five-seed final run	0.50	92.56 ± 0.29

Table 10: **Mixing-coefficient sensitivity.** The upper blocks report the original single-seed sweep. The lower block reports a five-seed comparison of the two most relevant settings at $\sigma = 0.5$.

Setting	λ_{mix}	Top-1 (%)
Single seed, $\sigma = 0.5$	0.50	92.57
	0.75	92.14
	0.90	92.20
	1.00	92.29
Single seed, $\sigma = 2.0$	0.50	92.50
	0.75	92.06
	0.90	92.17
	1.00	92.39
Five paired seeds, $\sigma = 0.5$	0.50	91.996 ± 0.188
	0.75	92.254 ± 0.286

tially increases greedy-selection cost. We therefore treat the reported defaults as empirical operating points rather than universally optimal hyperparameters.

Kernel bandwidth. Table 9 shows sensitivity to the Gaussian bandwidth. On this CIFAR-10 configuration, the wider sweep favors the smaller tested bandwidths, while the local sweep shows little variation around $\sigma = 0.5$. Since the random-feature map is applied directly to raw penultimate-layer representations, the appropriate bandwidth depends on representation scale. The appropriate bandwidth therefore depends on representation scale and must be retuned when the architecture or domain changes.

Weight mixing. At $\sigma = 0.5$, the single-seed sweep gives 92.57% for $\lambda_{\mathrm{mix}} = 0.5$ and 92.14% for $\lambda_{\mathrm{mix}} = 0.75$. A five-seed paired comparison gives $91.996 \pm 0.188\%$ and $92.254 \pm 0.286\%$, respectively. We use $\lambda_{\mathrm{mix}} = 0.75$ as the default in this configuration.

Residual tolerance and support carry-over. The residual tolerance controls when greedy matching stops, while ρ_{keep} determines how much of the previous support is retained across refreshes. Tables 11 and 12 show that the final settings $\tau_{\mathrm{res}} = 0.1$ and $\rho_{\mathrm{keep}} = 0.25$ lie in a well-performing region of the sweeps.

Refresh period. More frequent refreshes track the evolving representation distribution more closely but require more full-dataset forward scans. In the single-seed experiment below, $P = 5$ is slightly more accurate than $P = 10$, while approximately doubling the number of refreshes. We use $P = 10$ as an empirical accuracy–refresh-cost compromise.

Random-feature dimension. We additionally evaluate the sketch dimension over three seeds on CIFAR-100 at 5% and 10% budgets. Table 14 reports the change in Top-1 accuracy relative to the

Table 11: **Residual-tolerance sensitivity.** Single-seed CIFAR-10 runs at 10% budget.

Bandwidth	Residual tolerance	Top-1 (%)
$\sigma = 0.5$	0.02	92.26
	0.05	92.30
	0.10	92.73
	0.20	92.36
$\sigma = 1.0$	0.02	92.28
	0.05	91.96
	0.10	92.40
	0.20	92.05
$\sigma = 2.0$	0.02	91.83
	0.05	91.60
	0.10	91.25

Table 12: **Carry-over sensitivity.** Single-seed CIFAR-10 runs at 10% budget and $\sigma = 0.5$.

$\rho_{\mathbf{keep}}$	Top-1 (%)
0.00	92.58
0.25	92.75
0.50	92.17
0.75	92.02

default $m = 256$. Accuracy varies only modestly over the tested 16-fold range, with no systematic benefit from increasing m beyond 256.

The computational picture is different. Table 15 profiles one CIFAR refresh while varying m. Although random-feature construction itself remains inexpensive, the greedy stage becomes substantially more costly as the sketch dimension increases.

From $m = 256$ to $m = 1024$, total refresh time nearly doubles while greedy-selection time increases by roughly an order of magnitude. Since the accuracy sweep does not show a corresponding systematic gain, we retain $m = 256$ as an empirical accuracy–cost compromise. Whether a fixed m remains sufficient as representation complexity or domain changes is not established by these experiments and remains open.

Minimum effective support size. Finally, we vary the minimum fraction of the nominal support that is enforced after greedy residual matching. The sweep shows modest variation across intermediate settings. The final experiments use 1.0, ensuring that the requested nominal budget is filled consistently across methods.

C Additional Related Work

This appendix expands the related-work discussion from Section 2. Our work connects dataset pruning and coreset selection, training-time data selection, importance sampling and kernel quadrature. We focus on approaches that reduce training compute by selecting or reweighting a small subset while preserving the effect of full-data optimization. These approaches differ both in the information available at selection time and in the quantity that the selected subset is intended to preserve.

Static pruning and training-free coresets. A classical line of work constructs a subset *once*, prior to training or independently of model dynamics. Recent surveys highlight that simple baselines remain strong at moderate pruning rates (Moser et al., 2025). Geometry-based selection methods such as k-center greedy, facility-location objectives, and herding aim to cover the dataset in a feature space (Welling, 2009; Sener & Savarese, 2018; Mirzasoleiman et al., 2020). These methods admit a kernel interpretation: herding can be seen as building a quadrature rule that matches the empirical kernel mean embedding in an RKHS (Chen et al., 2010; Bach et al., 2012; Huszár & Duvenaud, 2012). Recent fixed or mostly static coreset methods also revisit the scoring criterion

Table 13: **Refresh-period sensitivity.** Single-seed CIFAR-10 runs at 10% budget with $\sigma = 0.5$ and $\lambda_{\text{mix}} = 0.75$. The last column reports the relative number of refresh operations compared with $P = 10$.

Refresh period P	**Top-1 (%)**	**Relative refresh count**
5	92.57	$2.0\times$
10	92.05	$1.0\times$
20	91.81	$0.5\times$

Table 14: **Sensitivity to random-feature dimension.** Mean change in Top-1 accuracy (percentage points) relative to $m = 256$, with standard deviation over three seeds, on CIFAR-100. All other settings are held fixed.

RFF dimension m	**5%**	**10%**
64	-0.397 ± 0.424	$+0.003 \pm 0.463$
128	-0.297 ± 0.480	$+0.037 \pm 0.540$
256	0	0
512	$+0.403 \pm 0.527$	$+0.500 \pm 0.370$
1024	$+0.350 \pm 0.370$	$+0.183 \pm 0.725$

used to identify useful samples. For instance, CSReL uses reducible loss to estimate the performance gain of adding a sample to a coreset, with the goal of selecting examples that are informative while avoiding ambiguous or noisy high-loss samples (Tong et al., 2025). This is related to our use of forward-pass difficulty information, but CSReL is primarily designed for continual-learning rehearsal and data summarization, whereas our setting is dynamic refresh during standard training. Our approach inherits the mean-approximation perspective, but departs from static pruning by using an *epoch-dependent* representation and periodically refreshing the subset throughout training.

Training-aware selection via scores and dynamics. Training-time selection exploits signals produced during optimization, including loss values, prediction confidence, or gradient-related scores. Methods such as GraNd and EL2N rank examples using early-training loss/gradient surrogates (Paul et al., 2021), while forgetting-based criteria track prediction flips and interpret high-forgetting points as informative (Toneva et al., 2019). RHO-LOSS prioritizes examples according to their estimated reducible holdout loss (Mindermann et al., 2022). These score-based approaches primarily estimate the utility or difficulty of individual examples, rather than explicitly matching a global representation of the training distribution. Dynamic coreset methods further update the selected subset during training, using the current model state to avoid stale selections. GradMatch and GLISTER periodically refresh subsets using gradient matching or bilevel validation objectives (Killamsetty et al., 2021a;b). More recently, ERACS identifies the degradation of coreset representativeness over training and uses a gradient signal-to-noise ratio criterion to decide when the current coreset should be refreshed (Cheng et al., 2025). In contrast, we optionally incorporate a loss-based difficulty signal while keeping selection constrained by a representation-level mean approximation objective, which helps preserve global coverage in the evolving g_t space. DySk injects difficulty bias while explicitly preserving distributional coverage via mean-embedding approximation.

Gradient matching and submodular subset selection. Another family of methods chooses subsets that approximate full-data training updates. CRAIG (Mirzasoleiman et al., 2020) uses a facility-location-style objective to cover gradient diversity in a gradient embedding space, while GradMatch (Killamsetty et al., 2021a) selects points whose aggregated gradient matches the full gradient; GLISTER (Killamsetty et al., 2021b) formulates selection as bilevel optimization to improve validation performance. CREST constructs and reuses coresets during stochastic optimization while controlling their approximation quality (Yang et al., 2023). These approaches directly target optimization behavior, but can be computationally heavy due to repeated gradient computations or expensive objective evaluations. Our method is not gradient matching: it maintains a low-dimensional kernel sketch of the representation distribution via random Fourier features (Rahimi & Recht, 2007), and builds a coreset by greedy pursuit in sketch space, using only forward passes at refresh time (and optionally per-sample losses). Unlike gradient matching, DySk requires no per-example backward gradients at refresh time.

Table 15: **Refresh cost versus random-feature dimension.** Single-refresh profiling on CIFAR with 50,000 candidate examples. RFF/target denotes random-feature and sketch-target operations; greedy/fit includes greedy support construction and the initial fitting stage.

m	Total	RFF/target	Greedy/fit	Greedy/total
128	6.49 s	0.10 s	0.13 s	2.0%
256	6.97 s	0.19 s	0.64 s	9.2%
512	9.04 s	0.36 s	2.26 s	25.0%
1024	13.60 s	0.72 s	6.46 s	47.5%

Table 16: **Minimum effective support fraction.** Single-seed CIFAR-10 runs at 10% nominal budget.

Minimum effective fraction	Top-1 (%)
0.25	91.55
0.50	92.15
0.75	92.39
1.00	92.20

Second-order and distribution-matching coresets. Beyond first-order matching, recent work incorporates curvature information or stronger distributional objectives. Pooladzandi et al. (2022) propose AdaCore, which matches Hessian-preconditioned gradients using exponentially averaged curvature estimates and provides convergence guarantees for first- and second-order optimization. Shin et al. (2023) introduce LCMat, a sharpness-aware objective that matches loss curvatures in a local neighborhood, improving robustness at the cost of additional second-order computations. Orthogonally, Xiao et al. (2024) propose FDMat, which matches feature distributions via optimal transport. Compared to these approaches, we avoid Hessian estimates and OT solvers, trading stronger matching objectives for substantially lower refresh cost while still preserving representativeness through kernel mean approximation.

Data distillation and synthetic coresets. Dataset distillation/condensation replaces real datasets with compact *synthetic* surrogates. Dataset distillation (Wang et al., 2018) learns synthetic examples whose training dynamics approximate those of the full dataset. More recent methods improve performance by matching gradients (Zhao et al., 2021), distributions (Zhao & Bilen, 2023), features (Wang et al., 2022), or training trajectories (Cazenavette et al., 2022). While extremely compact, these methods often require bilevel optimization and can be sensitive to architecture and training choices. Our focus is different: we select *real* examples online and maintain representativeness in an evolving feature space via kernel mean approximation.

Importance sampling and quadrature viewpoints. Importance sampling adapts the sampling distribution to reduce gradient variance or focus compute on influential examples, with classical guarantees based on gradient norms or Lipschitz-type scores (Needell et al., 2014; Alain et al., 2015) and practical schemes for deep networks (Katharopoulos & Fleuret, 2018). Our method can be interpreted as importance sampling *constrained by representativeness*: we bias training toward a coreset whose random-feature embeddings approximate the empirical kernel mean under g_t, linking training-time sampling to kernel quadrature and herding ideas (Welling, 2009; Bach et al., 2012; Rahimi & Recht, 2007).

Data selection and filtering for LLMs. Recent work shows that selecting or reweighting training data can significantly improve the compute–quality trade-off for large language models. DoReMi optimizes data mixtures to accelerate pretraining (Xie et al., 2023). For instruction tuning, several approaches filter examples to reduce redundancy and improve alignment quality, including weak-to-strong filtering (Li et al., 2024) and systematic studies of automatic selection criteria (Liu et al., 2024). More structured objectives have also been proposed, e.g., maximizing information gain in semantic space (Chen et al., 2025). We additionally evaluate DySk in language-model pretraining, where the same mechanism can be applied using model representations and forward-pass loss information while maintaining representativeness through kernel mean approximation.

Positioning of DySk. DySk combines several ideas from the above lines of work in a specific training-time coreset mechanism. Like dynamic coreset methods, it refreshes the subset as the model evolves; like importance sampling, it trains from a nonuniform distribution; and like kernel herding or quadrature, it selects points by approximating a distributional summary. Its main distinction is to perform this approximation in the model's current representation space using a low-dimensional random-feature sketch, while incorporating loss information through a weighted sketch target and difficulty-aware support search. Thus, DySk is neither a purely static pruning rule, nor a pure loss-based sampler, nor a gradient-matching method: it is a dynamic, loss-aware, representation-sketching approach to weighted coreset training.

D Additional Theory

This appendix provides proofs and additional details for the theoretical results in Section 4. We first relate the operational sketch used by DySk to the ideal target sketch, then prove the sketch-to-gradient bound and its full-data decomposition. We next quantify fixed-support staleness between refreshes and prove the conditional stationarity guarantee. Finally, we give a simple existence result for sparse sketch approximation.

D.1 Operational sketch error and MMD interpretation

Fix a refresh time t, and let

$$v_i^{(t)} = \Phi(h_i^{(t)}) \in \mathbb{R}^m$$

denote the sketch feature of example i. For a target distribution $a^{(t)} \in \Delta_n$, define

$$\mu_t = \sum_{i=1}^{n} a_i^{(t)} v_i^{(t)}.$$

For any training distribution $q^{(t)} \in \Delta_n$, define the ideal target-sketch mismatch

$$\Delta_t(q) := \left\| \mu_t - \sum_{i=1}^{n} q_i^{(t)} v_i^{(t)} \right\|_2.$$

DySk may use a smoothed operational sketch s_t instead of μ_t. Define

$$m_t := \|\mu_t - s_t\|_2$$

and

$$\varepsilon_t(q) := \left\| s_t - \sum_{i=1}^{n} q_i^{(t)} v_i^{(t)} \right\|_2.$$

Then, by the triangle inequality,

$$\Delta_t(q) \le m_t + \varepsilon_t(q).$$

Thus, approximation of the operational sketch controls approximation of the ideal target sketch up to the smoothing mismatch m_t.

The sketch mismatch also admits a kernel interpretation. Define the finite-dimensional kernel

$$k_t(z_i, z_j) := \left\langle v_i^{(t)}, v_j^{(t)} \right\rangle.$$

Then

$$\Delta_t(q) = \mathrm{MMD}_{k_t}\left(\sum_{i=1}^{n} a_i^{(t)} \delta_{z_i}, \sum_{i=1}^{n} q_i^{(t)} \delta_{z_i} \right).$$

Hence DySk's sketch-fitting objective is exactly a finite-dimensional MMD objective under the kernel induced by the random-feature map.

D.2 Proof of the sketch-to-gradient bound

Recall that, for any $p \in \Delta_n$,

$$G_p(\theta) := \sum_{i=1}^{n} p_i \nabla_\theta \ell_i(\theta).$$

Under Assumption 4.1, at refresh time t,

$$\nabla_\theta \ell_i(\theta_t) = A_t v_i^{(t)} + \rho_i^{(t)}, \qquad \|\rho_i^{(t)}\|_2 \le \zeta_t.$$

Proof of Theorem 4.2. We have

$$\begin{aligned} G_a(\theta_t) - G_q(\theta_t) &= \sum_{i=1}^{n} \left(a_i^{(t)} - q_i^{(t)}\right) \nabla_\theta \ell_i(\theta_t) \\ &= \sum_{i=1}^{n} \left(a_i^{(t)} - q_i^{(t)}\right) \left(A_t v_i^{(t)} + \rho_i^{(t)}\right) \\ &= A_t \left(\sum_{i=1}^{n} a_i^{(t)} v_i^{(t)} - \sum_{i=1}^{n} q_i^{(t)} v_i^{(t)}\right) \\ &\quad + \sum_{i=1}^{n} \left(a_i^{(t)} - q_i^{(t)}\right) \rho_i^{(t)}. \end{aligned}$$

Taking norms and applying the triangle inequality gives

$$\begin{aligned} \|G_a(\theta_t) - G_q(\theta_t)\|_2 \le& \|A_t\|_{\mathrm{op}} \left\| \sum_i a_i^{(t)} v_i^{(t)} - \sum_i q_i^{(t)} v_i^{(t)} \right\|_2 \\ &+ \sum_i \left| a_i^{(t)} - q_i^{(t)} \right| \|\rho_i^{(t)}\|_2 \\ \le& \|A_t\|_{\mathrm{op}} \Delta_t(q) + \zeta_t \|a^{(t)} - q^{(t)}\|_1. \end{aligned}$$

Since $a^{(t)}$ and $q^{(t)}$ are probability distributions,

$$\|a^{(t)} - q^{(t)}\|_1 \le 2,$$

which gives

$$\|G_a(\theta_t) - G_q(\theta_t)\|_2 \le \|A_t\|_{\mathrm{op}} \Delta_t(q) + 2\zeta_t.$$

Finally, if the operational sketch s_t is used,

$$\Delta_t(q) \le m_t + \varepsilon_t(q),$$

which yields

$$\|G_a(\theta_t) - G_q(\theta_t)\|_2 \le \|A_t\|_{\mathrm{op}} \big(m_t + \varepsilon_t(q)\big) + 2\zeta_t.$$

□

D.3 Uniform and nonuniform targets

The uniform target

$$u_i = \frac{1}{n}$$

is of particular interest because

$$G_u(\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell_i(\theta) = \nabla L(\theta).$$

Therefore, when $a^{(t)} = u$, Theorem 4.2 directly gives

$$\|\nabla L(\theta_t) - G_q(\theta_t)\|_2 \le \|A_t\|_{\mathrm{op}} \Delta_t(q) + \zeta_t \|u - q^{(t)}\|_1.$$

In particular,

$$\|\nabla L(\theta_t) - G_q(\theta_t)\|_2 \le \|A_t\|_{\mathrm{op}} \Delta_t(q) + 2\zeta_t.$$

For a nonuniform target $a^{(t)}$, including the clipped loss-weighted target used by our default implementation, we instead write

$$\begin{aligned} G_u(\theta_t) - G_q(\theta_t) = \big(G_u(\theta_t) - G_a(\theta_t)\big) \\ + \big(G_a(\theta_t) - G_q(\theta_t)\big). \end{aligned}$$

Hence

$$\begin{aligned} \|G_u(\theta_t) - G_q(\theta_t)\|_2 \le \|G_u(\theta_t) - G_a(\theta_t)\|_2 \\ + \|A_t\|_{\mathrm{op}} \Delta_t(q) + \zeta_t \|a^{(t)} - q^{(t)}\|_1. \end{aligned}$$

Using $\|a^{(t)} - q^{(t)}\|_1 \le 2$,

$$\|G_u(\theta_t) - G_q(\theta_t)\|_2 \le \|G_u(\theta_t) - G_a(\theta_t)\|_2 + \|A_t\|_{\mathrm{op}} \Delta_t(q) + 2\zeta_t.$$

If additionally

$$\|\nabla_\theta \ell_i(\theta_t)\|_2 \le G_t \qquad \text{for all } i,$$

then

$$\begin{aligned} \|G_u(\theta_t) - G_a(\theta_t)\|_2 &= \left\| \sum_i \left(u_i - a_i^{(t)}\right) \nabla_\theta \ell_i(\theta_t) \right\|_2 \\ &\le G_t \|u - a^{(t)}\|_1. \end{aligned}$$

Consequently,

$$\|G_u(\theta_t) - G_q(\theta_t)\|_2 \le G_t \|u - a^{(t)}\|_1 + \|A_t\|_{\mathrm{op}} \Delta_t(q) + 2\zeta_t.$$

If a smoothed operational sketch is used, $\Delta_t(q)$ can throughout be replaced by

$$m_t + \varepsilon_t(q).$$

This decomposition separates the deliberate change in training distribution induced by a nonuniform target from the approximation error introduced by representing that target with a sparse coreset.

D.4 Fixed-support staleness between refreshes

The preceding results hold at refresh time. DySk, however, reuses the same training distribution for several optimization steps before refreshing it.

Let

$$\tau_0 < \tau_1 < \cdots$$

denote refresh iterations, and let q_r denote the training distribution constructed at τ_r and held fixed on

$$t \in [\tau_r, \tau_{r+1}).$$

For clarity, we first consider the uniform target $a^{(\tau_r)} = u$.

Define the gradient approximation error

$$e_t := G_{q_r}(\theta_t) - \nabla L(\theta_t), \qquad t \in [\tau_r, \tau_{r+1}).$$

At refresh time, let

$$M_r := \|A_{\tau_r}\|_{\mathrm{op}}, \qquad \zeta_r := \zeta_{\tau_r},$$

and define

$$\delta_r := \left\| \sum_i (q_{r,i} - u_i)\, v_i^{(\tau_r)} \right\|_2.$$

The refresh-time bound gives

$$\|e_{\tau_r}\|_2 \le M_r \delta_r + \zeta_r \|q_r - u\|_1.$$

We next impose a local regularity condition on how the gradient discrepancy evolves after the refresh.

Assumption D.1 (Local gradient staleness)**.** For every refresh block r, there exists $\Lambda_r \geq 0$ such that, along the iterates visited within the block,

$$\begin{aligned}&\left\|\left(G_{q_r}(\theta) - G_u(\theta)\right) - \left(G_{q_r}(\theta_{\tau_r}) - G_u(\theta_{\tau_r})\right)\right\|_2 \\ &\quad \leq \Lambda_r \|\theta - \theta_{\tau_r}\|_2.\end{aligned}$$

This condition follows from local Lipschitz regularity of the per-example gradients. In particular, suppose

$$\|\nabla \ell_i(\theta) - \nabla \ell_i(\theta')\|_2 \leq L_r \|\theta - \theta'\|_2$$

for all i and all relevant θ, θ' in refresh block r. Then

$$\begin{aligned}&\left\|\left(G_{q_r}(\theta) - G_u(\theta)\right) - \left(G_{q_r}(\theta') - G_u(\theta')\right)\right\|_2 \\ &= \left\|\sum_i (q_{r,i} - u_i)\left[\nabla \ell_i(\theta) - \nabla \ell_i(\theta')\right]\right\|_2 \\ &\leq L_r \|q_r - u\|_1 \|\theta - \theta'\|_2.\end{aligned}$$

Hence one may take

$$\Lambda_r = L_r \|q_r - u\|_1 \leq 2L_r.$$

Proposition D.2 (Gradient error within a refresh block)**.** *Under Assumption 4.1 at refresh time τ_r and Assumption D.1 within the block, for every $t \in [\tau_r, \tau_{r+1})$,*

$$\|e_t\|_2 \leq M_r \delta_r + \zeta_r \|q_r - u\|_1 + \Lambda_r \|\theta_t - \theta_{\tau_r}\|_2.$$

Proof. By the refresh-time sketch-to-gradient result,

$$\|e_{\tau_r}\|_2 \leq M_r \delta_r + \zeta_r \|q_r - u\|_1.$$

Furthermore, Assumption D.1 gives

$$\|e_t - e_{\tau_r}\|_2 \leq \Lambda_r \|\theta_t - \theta_{\tau_r}\|_2.$$

Therefore,

$$\begin{aligned}\|e_t\|_2 &\leq \|e_{\tau_r}\|_2 + \|e_t - e_{\tau_r}\|_2 \\ &\leq M_r \delta_r + \zeta_r \|q_r - u\|_1 + \Lambda_r \|\theta_t - \theta_{\tau_r}\|_2.\end{aligned}$$

□

If the operational sketch is smoothed, then

$$\delta_r \leq m_{\tau_r} + \varepsilon_{\tau_r}(q_r),$$

so the same result holds with this upper bound.

The final term is a fixed-support staleness term: even if the selected distribution closely approximates the desired gradient at the refresh point, the approximation can deteriorate as the parameters move while q_r remains fixed.

D.5 Conditional stationarity guarantee

We now propagate the preceding approximation error through vanilla SGD.

Assume that L is L_s-smooth and bounded below by $L_\star$. Let

$$\theta_{t+1} = \theta_t - \eta_t \widehat{g}_t,$$

where, for $t \in [\tau_r, \tau_{r+1})$,

$$\mathbb{E}[\widehat{g}_t \mid \theta_t] = G_{q_r}(\theta_t).$$

Assume also that

$$\mathbb{E}\left[\|\widehat{g}_t - G_{q_r}(\theta_t)\|_2^2 \mid \theta_t\right] \leq v_t.$$

Define

$$B_t := M_r \delta_r + \zeta_r \|q_r - u\|_1 + \Lambda_r \|\theta_t - \theta_{\tau_r}\|_2.$$

By Proposition D.2,

$$\|e_t\|_2 \leq B_t.$$

Theorem D.3 (Conditional stationarity bound)**.** *Suppose the preceding assumptions hold in every refresh block and*

$$\eta_t \le \frac{1}{4L_s}.$$

Let

$$S_T := \sum_{t=0}^{T-1} \eta_t$$

and

$$\Delta_0 := L(\theta_0) - L_\star.$$

Then

$$\begin{aligned}\frac{1}{S_T}\sum_{t=0}^{T-1} \eta_t \mathbb{E}\left[\|\nabla L(\theta_t)\|_2^2\right] \le & \frac{4\Delta_0}{S_T} \\ & + \frac{3}{S_T}\sum_{t=0}^{T-1} \eta_t \mathbb{E}[B_t^2] \\ & + \frac{2L_s}{S_T}\sum_{t=0}^{T-1} \eta_t^2 v_t.\end{aligned}$$

Proof. For brevity, write

$$g_t := \nabla L(\theta_t).$$

Since

$$\mathbb{E}[\widehat{g}_t \mid \theta_t] = g_t + e_t,$$

L_s-smoothness gives

$$\begin{aligned}\mathbb{E}[L(\theta_{t+1}) \mid \theta_t] \le & L(\theta_t) - \eta_t \langle g_t, g_t + e_t\rangle \\ & + \frac{L_s\eta_t^2}{2}\mathbb{E}[\|\widehat{g}_t\|_2^2 \mid \theta_t].\end{aligned}$$

Using the conditional variance assumption,

$$\mathbb{E}[\|\widehat{g}_t\|_2^2 \mid \theta_t] \le \|g_t + e_t\|_2^2 + v_t.$$

Therefore,

$$\begin{aligned}\mathbb{E}[L(\theta_{t+1}) \mid \theta_t] \le & L(\theta_t) - \eta_t\|g_t\|_2^2 - \eta_t\langle g_t, e_t\rangle \\ & + \frac{L_s\eta_t^2}{2}\|g_t + e_t\|_2^2 + \frac{L_s\eta_t^2}{2}v_t.\end{aligned}$$

Since $L_s\eta_t \le 1/4$, Young's inequality yields the convenient bound

$$\begin{aligned}\mathbb{E}[L(\theta_{t+1}) \mid \theta_t] \le & L(\theta_t) - \frac{\eta_t}{4}\|g_t\|_2^2 \\ & + \frac{3\eta_t}{4}\|e_t\|_2^2 + \frac{L_s\eta_t^2}{2}v_t.\end{aligned}$$

Using

$$\|e_t\|_2 \le B_t,$$

taking expectations, and summing from $t = 0$ to $T - 1$ gives

$$\begin{aligned}\frac{1}{4}\sum_{t=0}^{T-1} \eta_t \mathbb{E}[\|g_t\|_2^2] \le & L(\theta_0) - \mathbb{E}[L(\theta_T)] \\ & + \frac{3}{4}\sum_{t=0}^{T-1} \eta_t \mathbb{E}[B_t^2] \\ & + \frac{L_s}{2}\sum_{t=0}^{T-1} \eta_t^2 v_t.\end{aligned}$$

Because $L(\theta_T) \geq L_\star$,

$$\frac{1}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}[\|g_t\|_2^2] \leq \Delta_0 + \frac{3}{4}\sum_{t=0}^{T-1}\eta_t\mathbb{E}[B_t^2] + \frac{L_s}{2}\sum_{t=0}^{T-1}\eta_t^2 v_t.$$

Multiplying by $4/S_T$ proves the result. □

The bound separates four effects: (i) refresh-time sketch approximation through δ_r; (ii) the local sketch-to-gradient residual through ζ_r; (iii) fixed-support staleness through $\Lambda_r\|\theta_t - \theta_{\tau_r}\|_2$; and (iv) stochastic minibatch variance through v_t.

For a nonuniform target, the refresh-time target-weighting bias from Section D.3 can be added to the corresponding bound on B_t.

D.6 Sparse approximation in sketch space

The preceding results explain what follows when the sketch mismatch is small. We next give a simple existence result showing that any target sketch admits a sparse approximation.

Proposition D.4 (Sparse sketch approximation)**.** *Fix a refresh time t, and suppose*

$$\|v_i^{(t)}\|_2 \leq R \qquad \text{for all } i.$$

For any target distribution $a^{(t)} \in \Delta_n$ and any integer $k \geq 1$, there exists a probability distribution $q \in \Delta_n$, supported on at most k examples, such that

$$\left\|\sum_{i=1}^n a_i^{(t)} v_i^{(t)} - \sum_{i=1}^n q_i v_i^{(t)}\right\|_2 \leq \frac{R}{\sqrt{k}}.$$

Proof. Draw indices

$$J_1, \ldots, J_k \overset{\text{i.i.d.}}{\sim} a^{(t)}$$

and define the empirical distribution

$$q_i := \frac{1}{k}\sum_{j=1}^k \mathbf{1}\{J_j = i\}.$$

Its support contains at most k examples. Let

$$\widehat{\mu} := \sum_i q_i v_i^{(t)} = \frac{1}{k}\sum_{j=1}^k v_{J_j}^{(t)}.$$

Since

$$\mathbb{E}[\widehat{\mu}] = \mu_t,$$

independence gives

$$\begin{aligned}\mathbb{E}\left[\|\widehat{\mu} - \mu_t\|_2^2\right] &= \frac{1}{k}\mathbb{E}_{J\sim a^{(t)}}\left[\|v_J^{(t)} - \mu_t\|_2^2\right] \\ &= \frac{1}{k}\left(\sum_i a_i^{(t)}\|v_i^{(t)}\|_2^2 - \|\mu_t\|_2^2\right) \\ &\leq \frac{R^2}{k}.\end{aligned}$$

Therefore at least one realization satisfies

$$\|\widehat{\mu} - \mu_t\|_2^2 \leq \frac{R^2}{k},$$

which proves the claim. □

For the cosine random-feature map used by DySk,

$$v_i = \sqrt{\frac{2}{m}} \left(\cos(\cdot), \ldots, \cos(\cdot)\right),$$

and therefore

$$\|v_i\|_2^2 \le \frac{2}{m} \sum_{j=1}^{m} 1 = 2.$$

Thus one may take

$$R = \sqrt{2},$$

which gives

$$\Delta_t(q) \le \sqrt{\frac{2}{k}}$$

for at least one k-sparse distribution.

Under exact local sketch-gradient predictability ($\zeta_t = 0$), Theorem 4.2 then implies the existence of a k-sparse distribution satisfying

$$\|G_a(\theta_t) - G_q(\theta_t)\|_2 \le \|A_t\|_{\mathrm{op}} \sqrt{\frac{2}{k}}.$$

This proposition applies to sparse approximation in the fixed sketch space. It provides neither a convergence rate for DySk's gated greedy solver nor a uniform guarantee that a fixed random-feature dimension approximates the Gaussian kernel along an evolving representation trajectory.

D.7 Optional consequence under a PL condition

Under an additional Polyak-Lojasiewicz (PL) condition, the biased-SGD analysis also yields a standard restricted-path consequence. This result is purely conditional and is not assumed to hold globally for the neural-network objectives in our experiments.

Suppose that along the iterates visited by the algorithm,

$$\|\nabla L(\theta)\|_2^2 \ge 2\mu\big(L(\theta) - L_\star\big)$$

for some $\mu > 0$. Assume a constant step size

$$\eta \le \frac{1}{L_s},$$

and uniform bounds

$$\|e_t\|_2 \le B, \qquad v_t \le v.$$

Using the same smoothness argument as above, one obtains

$$\begin{aligned}\mathbb{E}[L(\theta_{t+1}) - L_\star] \le &(1 - \mu\eta)\mathbb{E}[L(\theta_t) - L_\star] \\ &+ \frac{\eta}{2} B^2 + \frac{L_s \eta^2}{2} v.\end{aligned}$$

Iterating gives

$$\begin{aligned}\mathbb{E}[L(\theta_T) - L_\star] \le &(1 - \mu\eta)^T \Delta_0 \\ &+ \frac{B^2}{2\mu} + \frac{L_s \eta v}{2\mu}.\end{aligned}$$

Thus, under this additional PL assumption, the iterates converge linearly to a neighborhood controlled by the coreset-gradient approximation error and stochastic-gradient variance.

This consequence assumes the stated condition only along the visited optimization path and does not establish global convergence for the deep networks used in the experiments.

Table 17: **Complete CIFAR data-selection results.** Top-1 test accuracy and wall-clock training time for ResNet-18 on CIFAR-10 and CIFAR-100 at matched nominal coreset budgets. Results follow the controlled protocol described in Section 5 and Appendix A. Entries are reported as mean ± standard deviation over five random seeds.

Data Selection Results									
		Top-1 Test Accuracy (%)				Training Time (hrs)			
	Budget	5%	10%	20%	30%	5%	10%	20%	30%
Dataset	Selection Strategy								
CIFAR-10	FULL	95.32 ±0.20	95.32 ±0.20	95.32 ±0.20	95.32 ±0.20	0.609 ±0.005	0.609 ±0.005	0.609 ±0.005	0.609 ±0.005
	RANDOM-WARM	83.06 ±0.12	88.22 ±0.14	92.07 ±0.11	93.52 ±0.14	0.114 ±0.003	0.148 ±0.002	0.188 ±0.005	0.242 ±0.003
	GLISTERPB-WARM	86.57 ±0.19	91.56 ±0.18	92.98 ±0.27	94.09 ±0.09	0.323 ±0.002	0.400 ±0.005	0.598 ±0.002	0.787 ±0.022
	CRAIGPB-WARM	84.48 ±0.20	89.28 ±0.62	92.01 ±0.14	92.82 ±0.07	0.354 ±0.010	0.422 ±0.004	0.571 ±0.005	0.759 ±0.008
	GRADMATCHPB-WARM	87.20 ±0.23	92.15 ±0.27	92.11 ±0.12	92.01 ±0.08	0.346 ±0.008	0.423 ±0.009	0.579 ±0.012	0.733 ±0.003
	DYSK (ours)	**90.74** ±0.20	**92.56** ±0.29	**94.32** ±0.23	**94.59** ±0.06	0.186 ±0.003	0.221 ±0.002	0.283 ±0.005	0.384 ±0.0062
CIFAR-100	FULL	78.29 ±0.29	78.29 ±0.29	78.29 ±0.29	78.29 ±0.29	0.621 ±0.020	0.621 ±0.020	0.621 ±0.020	0.621 ±0.020
	RANDOM-WARM	55.63 ±0.14	63.29 ±0.11	71.29 ±0.22	73.88 ±0.17	0.116 ±0.004	0.147 ±0.003	0.195 ±0.004	0.240 ±0.005
	GLISTERPB-WARM	57.17 ±0.22	64.95 ±0.16	62.14 ±0.29	72.43 ±0.46	0.338 ±0.007	0.411 ±0.008	0.571 ±0.013	0.738 ±0.006
	CRAIGPB-WARM	57.44 ±0.31	67.30 ±0.18	69.76 ±0.11	72.77 ±0.23	0.374 ±0.010	0.434 ±0.007	0.578 ±0.006	0.736 ±0.014
	GRADMATCHPB-WARM	57.72 ±0.27	68.23 ±0.18	71.34 ±0.15	74.06 ±0.27	0.341 ±0.009	0.401 ±0.010	0.551 ±0.003	0.785 ±0.010
	DYSK (ours)	**65.82** ±0.16	**70.99** ±0.11	**74.83** ±0.31	**76.88** ±0.28	0.187 ±0.007	0.219 ±0.007	0.291 ±0.015	0.356 ±0.009

Table 18: **Single-refresh scaling on ImageNet-1k.** Wall-clock cost and peak memory of one DySk refresh as the number of candidate examples increases, using $m = 256$ random features. The forward/loss column includes the full candidate-set scan used to extract representations and losses. k_{gr} denotes the number of greedy iterations executed before residual stopping. Total time includes minor bookkeeping costs not shown separately. Profiling is performed at random initialization and is intended to characterize computational cost rather than selection quality.

Candidates	**Forward/loss**	**RFF**	**Greedy**	k_{gr}	**Total**	**Host mem.**	**GPU mem.**
100k	101.345 s	0.157 s	1.295 s	150	102.961 s	9.25 GB	7.00 GB
500k	458.408 s	0.661 s	5.949 s	151	465.515 s	12.51 GB	7.00 GB
1.281M	1246.200 s	1.924 s	18.789 s	142	1268.094 s	17.24 GB	7.00 GB

E ADDITIONAL RESULTS

E.1 COMPLETE CIFAR RESULTS

Table 17 expands the budget–accuracy curves in Figure 1 by reporting both Top-1 test accuracy and wall-clock training time under the controlled CIFAR protocol. Across both datasets, DYSK achieves the highest accuracy among the evaluated non-full-data methods at every reported budget. On CIFAR-10, it reaches 94.59% at 30%, within 0.73 points of full-data training. On CIFAR-100, it reaches 76.88% at 30%, within 1.41 points of full-data training.

Random selection remains the cheapest baseline because it has negligible selection overhead. Among the non-random dynamic coreset methods evaluated in our controlled implementation, DYSK combines higher accuracy with substantially lower wall-clock cost than the gradient- and bilevel-based baselines. Full-data warm-start epochs and all selection overheads are included in the reported wall-clock times.

E.2 IMAGENET-1K PROTOCOL AND SCALING DETAILS

We train ResNet-18 on ImageNet-1k for 90 epochs at nominal 5% and 10% subset fractions. We evaluate two DySk instantiations. DYSK-DIRECT applies the weighted selector used in the CIFAR experiments directly at ImageNet scale. DYSK-CORE selects a global sketch-based core of 256 examples and then completes the nominal support by class-balanced uniform sampling. Training on the completed support uses uniform sampling. Random uses the same budget-specific full-data warm start as both DySk variants. The corresponding accuracy results are reported in Table 3.

For DYSK-CORE, the greedy budget is therefore $k_{\mathrm{gr}} = 256$ at each refresh, independently of the much larger nominal support size.

To assess refresh-time scalability independently of selection quality, we additionally profile a single DySk refresh as the number of candidate examples increases. These profiling runs use random initialization and therefore support computational-cost and memory claims only.

At the full ImageNet scale, the forward/loss scan accounts for 98.3% of refresh time, whereas greedy support construction accounts for only 1.5%. The random-feature computation itself remains a small fraction of the total cost, and peak GPU memory remains approximately 7 GB across the evaluated candidate-set sizes. Thus, at this scale, the dominant refresh cost is the forward pass over the candidate set rather than sketch construction or greedy matching.

For DYSK-CORE, the greedy budget is $k_{\mathrm{gr}} = 256$ at each refresh, independently of the much larger nominal support size.

At the evaluated ImageNet scale, refresh computation remains feasible. Its overhead increases with random-feature dimension, representation-extraction cost, and the number of examples scanned per refresh.

E.3 TRANSFORMER-PRETRAINING PROTOCOL AND ADDITIONAL DETAILS

We additionally evaluate DySk in autoregressive Transformer pretraining using a nanochat-style model with approximately 286M parameters. The experiment processes $5{,}000$ candidate batches and uses 500 initial full-data candidate steps before subset training. Performance is measured using validation bits-per-byte (BPB), for which lower is better. The corresponding accuracy-quality results are reported in Section 5.

We evaluate two DySk instantiations. DYSK-DIRECT applies the weighted selector to the complete selected support, whereas DYSK-CORE selects a small representation-coverage core and completes the support by uniform sampling. The greedy-core budget is 32 sequences at 5% and 52 sequences at 10%. Residual stopping may terminate earlier; in the reported runs, the effective number of greedy iterations averages 4.15 at 5% and 3.80 at 10%.

The comparison with LOSS-ONLY uses identical backward-token budgets. The latter retains forward-pass difficulty information but removes representation sketching and coverage-based support selection. At 5% and 10% budgets, DYSK-DIRECT reaches validation BPB values of 0.931628 and 0.906927, respectively, while DYSK-CORE reaches 0.930890 and 0.905722. For comparison, LOSS-ONLY reaches 0.947432 and 0.919664. Each configuration is evaluated with one run, so run-to-run variability is not estimated.

Refresh cost. In this setting, scoring the $5{,}000$ candidate batches requires approximately 147-150 minutes, whereas all subsequent support selection calls together require only a few seconds. Thus, as in the ImageNet profiling study, the dominant cost of a refresh is extracting the current model-dependent representations and losses rather than the sketch-matching procedure itself.

E.4 CONTEXTUAL COMPARISON WITH ERACS

ERACS (Cheng et al., 2025) is a dynamic coreset method based on a gradient signal-to-noise criterion. No official public implementation is available at date of submission, so Table 19 reports its published CIFAR accuracies alongside DySk results obtained in our codebase. DySk uses refresh period $P = 20$, matching the nominal refresh cadence used in the published ERACS evaluation.

These values are included as literature context only. The ERACS and DySk values use different codebases and training protocols. Numerical differences can therefore reflect optimization, augmentation, warm start, or implementation choices in addition to the selection method.

E.5 ADDITIONAL GLUE TRANSFER RESULTS

We additionally evaluate DistilBERT fine-tuning on GLUE using a nominal 10% DySk coreset after a short full-data warm start. This experiment tests the refresh-and-sketch mechanism on sequence-classification and regression tasks. No NLP-specific data-selection baselines are included.

We fine-tune DistilBERT using a nominal 10% DySk coreset after a short full-data warm start. Table 20 reports development set performance. CoLA uses Matthews correlation, STS-B uses Spearman correlation, and all remaining tasks use accuracy.

Table 19: **Contextual comparison with published ERACS results.** Top-1 test accuracy (%) on CIFAR-10 and CIFAR-100. ERACS values are taken from its published evaluation, whereas DySk is evaluated in our codebase with refresh period $P = 20$. DySk values are mean $\pm$ standard deviation over five seeds. Because the methods use different implementations and training protocols, this table should not be interpreted as a controlled head-to-head comparison.

Dataset	**Budget**	**ERACS**	**DySk** ($P = 20$)
CIFAR-10	5%	87.27	89.58 ± 0.24
	10%	91.37	92.25 ± 0.18
	20%	92.75	93.76 ± 0.33
CIFAR-100	5%	46.86	62.99 ± 0.41
	10%	69.67	69.71 ± 0.29
	20%	71.03	74.36 ± 0.17

Table 20: **DistilBERT results on GLUE.** Development-set performance for full-data fine-tuning and DySk with a nominal 10% coreset. CoLA reports Matthews correlation, STS-B reports Spearman correlation, and all other tasks report accuracy.

Method	**CoLA**	**SST-2**	**MRPC**	**STS-B**	**QQP**	**MNLI**	**QNLI**	**RTE**
FULL	51.3	91.3	87.5	86.5	88.5	82.2	89.2	59.9
DYSK (10%)	49.9	90.7	86.7	86.3	88.1	80.9	88.6	59.6

Across these tasks, DySk remains close to full-data fine-tuning despite using a substantially smaller nominal training subset. The comparison therefore measures transfer relative to full-data fine-tuning rather than performance against NLP-specific selectors.

E.6 COMPUTE-ACCURACY DYNAMICS

Figure 2 provides an additional view of the CIFAR-10 training dynamics. Instead of reporting only final accuracy at each budget, it plots test accuracy as a function of equivalent full-dataset epochs, i.e., the number of SGD updates normalized by the number of updates in one full-data epoch. Full-data warm-start updates are included in this compute axis. This diagnostic view therefore visualizes how quickly different DySk budgets recover accuracy as a function of optimization compute.

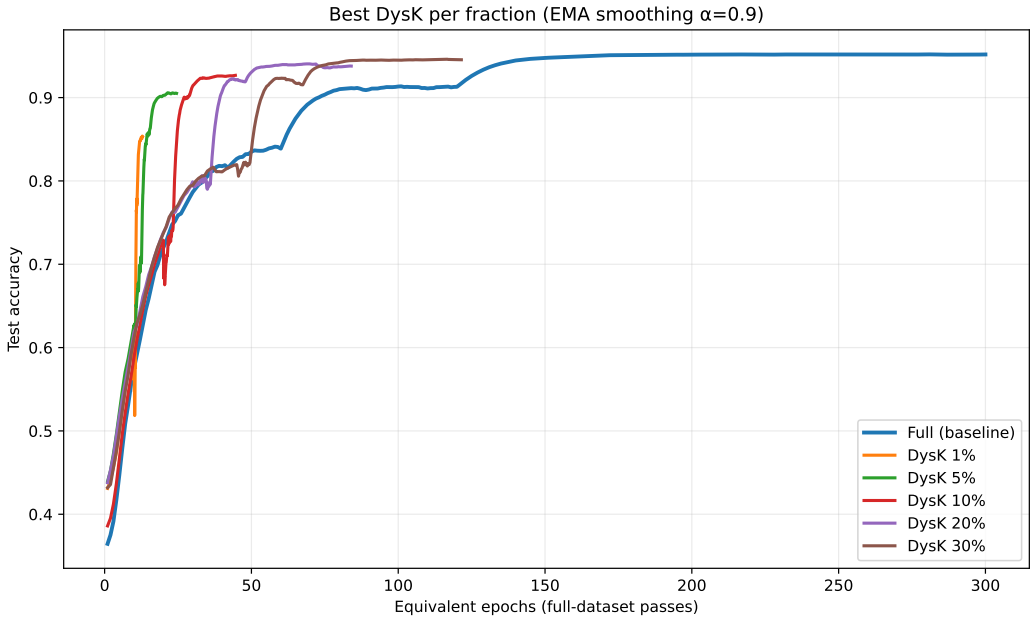


Figure 2: **Compute–accuracy dynamics on CIFAR-10 with ResNet-18.** Test accuracy is plotted against training compute, expressed as equivalent full-dataset epochs, i.e., the number of SGD updates normalized by the number of updates in one full-data epoch. Full-data warm-start updates are included. The black curve shows full-data SGD, while colored curves show DySk at different coreset fractions. This view compares training trajectories at matched SGD-update budgets rather than matched wall-clock time. Curves are smoothed with an exponential moving average for readability; raw values are used for the numerical results reported in the tables.