EFFICIENT TEST-TIME FINETUNING OF LLMS VIA CONVEX RECONSTRUCTION AND GRADIENT CACHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time finetuning (TTFT) is a rapidly evolving paradigm that adapts a language model to each prompt by retrieving related sequences, updating the model on them, and then evaluating the prompt. However, TTFT is only practical if it is *fast*: selection and finetuning both happen per query, making each a direct bottleneck. Existing methods trade speed for quality: fast retrieval is often redundant, while stronger diversity-aware selection adds prohibitive per-query cost. We introduce **HullFT**, a geometric approach to TTFT that addresses both bottlenecks. Given a query, HullFT first represents the query embedding as a sparse convex combination of few training sequences, using efficient projection-free Frank–Wolfe optimization. This favors a support set that is both potentially *relevant and diverse*, which we confirm empirically. We then convert the fractional convex weights into an exact integer multiset for finetuning through a geometric integerization procedure. The resulting multiplicities naturally create repeated examples, which we exploit with **Gradient Reuse** to amortize forward–backward computation across repeated finetuning steps. Our experiments show that HullFT improves the quality–efficiency tradeoff over current state-of-the-art TTFT methods, achieving lower bits-per-byte at substantially lower total runtime.

1 INTRODUCTION

Large language models are trained on web-scale corpora and optimized for broad, general-purpose performance across a vast distribution of text (Raffel et al., 2020; Penedo et al., 2023). As a result, their weights represent a global optimum that is unlikely to be locally optimal for any specific input. Test-time training (TTT) (Sun et al., 2020; Gandelsman et al., 2022; Hardt & Sun, 2024; Sun et al., 2025; Akyürek et al., 2024), also referred to as test-time finetuning (TTFT) in recent influential work (Hübotter et al., 2025), addresses this directly: at inference time, given a prompt q, we retrieve the N most relevant sequences from a large corpus, take one gradient step per retrieved sequence, and evaluate the finetuned model on q. Retrieving as few as 20 neighbors is enough to substantially close the gap between models differing by more than an order of magnitude in parameter count (Hardt & Sun, 2024).

Data selection for TTFT runs per prompt at inference time, making it a direct contributor to user-facing latency. Current methods sit at opposite ends of a quality–efficiency tradeoff. Pure nearest-neighbor retrieval (Hardt & Sun, 2024) returns the top-N candidates (nearest neighbors in some embedding space) via a FAISS index (Johnson et al., 2021): fast, but blind to redundancy. On large corpora, duplicate content is common (Wenzek et al., 2020; Abbas et al., 2023); without accounting for redundancy, the top-N neighbors can collapse to near-identical sequences, causing every subsequent gradient step to repeat the same signal. Diversity-aware selection methods address this by explicitly penalizing redundancy when choosing examples (Hübotter et al., 2025), producing more informative batches. The improvement in bits-per-byte (bpb) (Gao et al., 2020) over pure retrieval is significant, though such greedy selection introduces overhead, especially at lower N values.

Novelty. In this work, we take a different route, drawing on provable sparse convex approximation. The key idea is that directions in the embedding space carry meaning: sequences spanning different directions cover a broader range of semantic features, while sequences pointing in nearly the same direction are largely redundant (Carbonell & Goldstein, 1998; Kulesza & Taskar, 2012). Thus, framing data selection as a *convex approximation* of the prompt q encourages diversity through the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

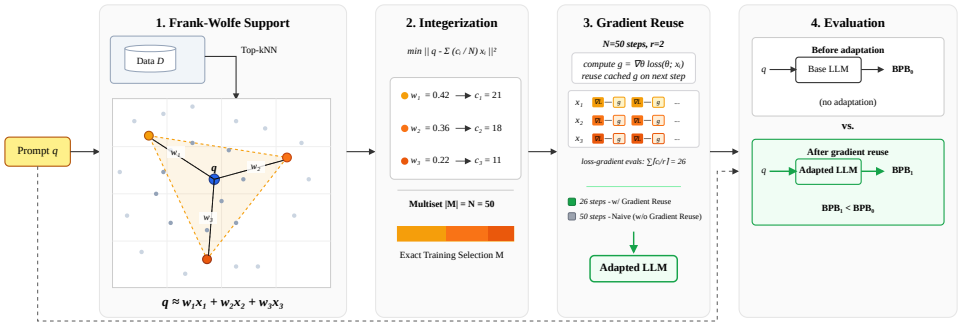

Figure 1: Our test-time finetuning pipeline. **1. Frank-Wolfe Support:** Given a prompt q, we retrieve a candidate pool from the corpus, then approximate q as a sparse convex combination to select a support set. **2. Integerization:** Fractional weights are converted to integer counts forming an exact N-point multiset. **3. Finetuning & Inference:** The base LLM is finetuned on this multiset before evaluating q via gradient reusing on repeated samples.

geometry itself, without explicit penalty terms, greedy heuristics, or expensive search and heavy gradient descent based optimization methods. Once a candidate is selected to help reconstruct q, near-duplicates pointing in the same direction are naturally down-weighted, as they offer little additional value. The optimization therefore tends to turn to unexplored directions instead, covering the prompt's context (in the embedding space) more broadly (Sener & Savarese, 2018; Mirzasoleiman et al., 2020). The objective itself contains no diversity term, so this is an intuition rather than a guarantee. We measure it directly in Sec. 4 (Tab. 6), where HullFT's support is as non-redundant as SIFT's.

Approximating q as a convex combination of candidates is precisely the *approximate Carathéodory problem* (Barman, 2015; Carathéodory, 1907; Combettes & Pokutta, 2023): find a small support set with positive weights summing to 1 whose weighted centroid is close to q. The approximate Carathéodory theorem guarantees that an ε-accurate solution (in squared ℓ_2 error) using $O(1/\varepsilon)$ points always exists, regardless of ambient dimension (Barman, 2015). Notably, convex reconstruction alone does not yet specify a TTFT update sequence. It returns fractional weights over selected examples, while TTFT requires an explicit set of N samples. We therefore convert the weighted solution into integer multiplicities that preserve the geometric approximation as closely as possible, see more details in Sec. D.2.

Crucially, this discrete reconstruction does more than make the method executable. Once the fractional weights are converted into integers, some selected examples appear multiple times in the final multiset. These repetitions create an opportunity for efficient finetuning: rather than recomputing a fresh forward–backward pass for every repeated occurrence, we can reuse cached gradients across short runs of identical examples. Thus, integerization not only bridges geometry and TTFT, but also induces the structure that enables Gradient Reuse. Figure 1 summarizes the HullFT pipeline.

Our contributions. We introduce **HullFT**, an efficient test-time finetuning method that improves the quality–efficiency tradeoff of TTFT through three coupled ideas.

1. We formulate query/prompts-specific data selection as a sparse convex approximation problem over a kNN-retrieved candidate pool, which favors a support set that is both relevant and diverse. Empirically, its diversity matches SIFT's (Tab. 6). We suggest solving it using the **Frank–Wolfe** algorithm. This efficient projection-free convex approximation procedure selects a small, diverse, and relevant support set from the candidate pool (Alg. 3 in the appendix).

2. We introduce a **geometric integerization** procedure (Alg. 1) that turns fractional convex weights into an exact N-example training multiset while preserving approximation quality.

3. We show that the resulting multiset (from 2) enables **Gradient Reuse**, which amortizes gradient computation across repeated training examples and reduces finetuning cost (Alg. 4).

Across 12 subsets of **The Pile** (Gao et al., 2020), HullFT strictly outperforms our competing methods (kNN (Hardt & Sun, 2024) and SIFT (Hübotter et al., 2025)) on quality–efficiency tradeoffs, with the largest gains in the latency-critical regime. We see the same pattern with **Pythia-410M** (Biderman et al., 2023) as the base model.

2 Related work

Test-time training and finetuning. Instance-specific adaptation traces back to transductive learning (Joachims, 1999) and has recently re-emerged as test-time training (TTT), where deep networks adapt at inference time using self-supervised objectives (Sun et al., 2020; Gandelsman et al., 2022; Sun et al., 2025; Akyürek et al., 2024). For language models, nearest-neighbor TTT finetunes on retrieved examples (Hardt & Sun, 2024), while SIFT adds an information-theoretic selection criterion that accounts for redundancy (Hübotter et al., 2025). Alternatively, retrieval-augmented models place retrieved text in context (Lewis et al., 2020; Borgeaud et al., 2022), but self-attention cost limits context size and full benefits often require retrieval-aware training (Hardt & Sun, 2024). Related selection methods balance relevance and diversity via maximal marginal relevance or determinantal point processes (Carbonell & Goldstein, 1998; Kulesza & Taskar, 2012). Influence-based methods estimate example utility (Hampel, 1974; Koh & Liang, 2017), including through gradient similarity for LLM finetuning (Pruthi et al., 2020; Kwon et al., 2024; Xia et al., 2024), but typically score points independently and thus miss redundancy (Hübotter et al., 2025). Active learning selects diverse subsets to cover the data manifold (Sener & Savarese, 2018; Mirzasoleiman et al., 2020), but is not query-conditioned.

Data selection. Early transfer-learning work showed that an additional finetuning stage on in-domain text produces large gains across tasks (Howard & Ruder, 2018; Gururangan et al., 2020). The approach has since been validated in biomedicine (Lee et al., 2020; Alsentzer et al., 2019; Huang et al., 2019), scientific text (Beltagy et al., 2019), legal language (Chalkidis et al., 2020), and clinical question answering (Singhal et al., 2023), establishing domain-specific adaptation as a default recipe. Beyond the choice of domain, the composition of the training subset itself can be decisive: a handful of curated examples sometimes matches full-corpus finetuning (Zhou et al., 2023), parameter-efficient methods make per-task adaptation practical even for very large models (Hu et al., 2022), and recent work explicitly optimizes which data to include for specialization (Hulkund et al., 2025; Sreeram et al., 2025). Adaptation benefits extend beyond language; for instance, they have been shown to govern out-of-distribution robustness in vision-based driving (Mallak & Maalouf, 2026; Wang et al., 2024). At the pre-training scale, a parallel effort in data curation tackles redundancy through deduplication, quality filtering, and domain reweighting (Wenzek et al., 2020; Abbas et al., 2023; Sachdeva et al., 2024; Wettig et al., 2024; Raffel et al., 2020; Penedo et al., 2023). Our approach pushes this logic to its extreme: rather than curating a training set once, we curate query-specific subsets at inference time, using convex geometry to jointly enforce relevance and diversity within a single optimization.

Convex approximation, Frank–Wolfe, and coresets. Carathéodory's theorem (Carathéodory, 1907) guarantees that any point in the convex hull of a set in $\mathbb{R}^d$ can be written as a convex combination of at most $d+1$ points, requiring $O(n^2 d)$ time for constructing such a set. Lately, Maalouf et al. (2019) proposed a fast construction running in $O(nd + d^4 \log n)$ time. The approximate variant relaxes this to $O(1/\varepsilon^2)$ points independent of dimension (Maurey, 1972-1973; Barman, 2015; Mirrokni et al., 2017). The Frank–Wolfe (conditional gradient) algorithm (Frank & Wolfe, 1956; Jaggi, 2013) provides a constructive proof of this result, with each iteration being a sparse convex combination that grows by at most one point (Combettes & Pokutta, 2023). Other variants achieve faster convergence under favorable geometry (Guélat & Marcotte, 1986; Lacoste-Julien, 2016). The connection between Frank–Wolfe and coresets (small weighted subsets that provably approximate a cost function over the full data) was established by Clarkson (2010) and developed in a rich body of work on geometric data summarization (Agarwal et al., 2005; Har-Peled, 2011; Phillips, 2017). In machine learning, coreset methods have been applied to regression (Maalouf et al., 2019; 2021), data-efficient model training and subset selection for deep learning (Mirzasoleiman et al., 2020; Sener & Savarese, 2018; Tukan et al., 2023; Killamsetty et al., 2021b;a; Coleman et al., 2020), model compression and network pruning (Tukan et al., 2022; Liebenwein et al., 2020; Mussay et al., 2022; Baykal et al., 2022), and general-purpose loss approximation under unified coreset frameworks (Maalouf et al., 2023; 2024; Feldman & Langberg, 2011; Braverman et al., 2016; Bachem et al., 2017; Munteanu &

Algorithm 1 INTEGERIZE$(q, \{p_1, \ldots, p_K\}, w, N, T)$

Input: query q, candidate pool $\{p_1, \ldots, p_K\}$, sparse weights $w \in \Delta^K$, budget N, swap passes T (default 2)

Output: support set $S = \{s_1, \ldots, s_{|S|}\} \subseteq \{p_1, \ldots, p_K\}$ and counts $c \in \mathbb{Z}_{\geq 0}^{|S|}$ with $\sum_j c_j = N$

Let $\{i_1, \ldots, i_{|S|}\} \leftarrow \{i : w_i > 0\}$; $s_j \leftarrow p_{i_j}$, $\tilde{w}_j \leftarrow w_{i_j}$; $S \leftarrow \{s_1, \ldots, s_{|S|}\}$
$c_j \leftarrow \lfloor N \tilde{w}_j \rfloor$ for all j
for $N - \sum_j c_j$ steps **do** ▷ greedy fill
    $j^\star \leftarrow \arg\min_j \left\| q - \sum_\ell \frac{c_\ell + \mathbf{1}[\ell = j]}{N} s_\ell \right\|_2^2$; $c_{j^\star} \leftarrow c_{j^\star} + 1$
for $t = 1$ **to** T **do** ▷ local-swap refinement
    improved $\leftarrow$ false
    for each (j, k) with $c_j > 0$, $j \neq k$ **do**
        if swapping one unit from j to k strictly reduces Eq. (2) **then**
            $c_j \leftarrow c_j - 1$; $c_k \leftarrow c_k + 1$; improved $\leftarrow$ true
    if not improved **then break**
return S, c

Schwiegelshohn, 2018). Our selection stage instantiates Frank–Wolfe on the ℓ_2-squared objective over the probability simplex, yielding a coreset of the candidate pool. Integerization then converts into a multiset for finetuning.

3 METHOD

Notation. Let $\langle u, v \rangle = u^\top v$ be the dot product on $\mathbb{R}^d$, $\mathbf{1}[\cdot]$ the indicator function ($\mathbf{1}[A] = 1$ if predicate A is true, 0 otherwise), Δ^K the probability simplex in $\mathbb{R}^K$, and $e_i \in \mathbb{R}^K$ the i-th standard basis vector. We use $q \in \mathbb{R}^d$ to denote a query, $\{p_1, \ldots, p_K\} \subset \mathbb{R}^d$ to denote a candidate training pool, and $P \in \mathbb{R}^{d \times K}$ to denote the matrix whose i-th column is p_i, so $Pw = \sum_i w_i p_i$ for any $w \in \mathbb{R}^K$ (all in the embedding space). The final selected support set is $S = \{s_1, \ldots, s_{|S|}\} \subseteq \{p_1, \ldots, p_K\}$, with associated integer counts $c \in \mathbb{Z}_{\geq 0}^{|S|}$, where c_j is the count of s_j. Throughout, N is the finetuning budget (total multiset size), ε is the Frank–Wolfe tolerance, and m is the support cap. The base model has parameters θ, learning rate η, and per-sequence loss $\mathcal{L}(\theta; s)$.

3.1 DATA SELECTION VIA CONVEX APPROXIMATION

Frank–Wolfe support set. From the preselected pool of K candidates, we solve the approximate Carathéodory problem by running the Frank–Wolfe (conditional gradient) algorithm (Frank & Wolfe, 1956; Jaggi, 2013) on the ℓ_2-squared objective over the probability simplex (Alg. 3). Selection reduces to finding a sparse $w \in \Delta^K$ whose convex combination Pw approximates q:

$$\min_{w \in \Delta^K} \|q - Pw\|_2^2. \tag{1}$$

The support of w is the selected subset: p_i is picked iff $w_i > 0$. The method is projection-free, requires only inner-product evaluations, and its iterates are sparse convex combinations by design (Clarkson, 2010). Starting from the single vertex $w = e_{v^\star}$, where $v^\star = \arg\max_{i \in [K]} \langle q, p_i \rangle$, each iteration forms the residual $r = q - Pw$, picks the vertex index $v = \arg\max_{i \in [K]} \langle r, p_i \rangle$, and takes an exact line search along the edge from w to e_v. A step adds at most one new nonzero to w, so iterates stay sparse throughout, and near-duplicates barely move the residual and are naturally skipped. We stop as soon as $\|q - Pw\|_2^2 \leq \varepsilon$ or $|\{i : w_i > 0\}| = m$, whichever comes first. The cap m is the binding constraint in practice: left alone, FW keeps adding vertices and w eventually fills in over most of the pool. The algorithm is defined in the appendix, Sec. C.1 as Alg. 3; we spell out Frank–Wolfe for completeness.

Geometric integerization. Frank–Wolfe (Alg. 3) returns fractional weights, which cannot be used directly for finetuning. Training on a uniformly weighted subset can outperform loss reweighting, which is prone to variance amplification and overfitting to noisy samples (Byrd & Lipton, 2019). We

Algorithm 2 HULLFTSELECT$(q, \{p_1, \ldots, p_K\}, N, \varepsilon, m, T)$

Input: query q, candidate pool $\{p_1, \ldots, p_K\}$, budget N, FW tolerance ε, support cap m, swap passes T
Output: support points S, counts c
1: $w \leftarrow$ FRANKWOLFE$(q, \{p_1, \ldots, p_K\}, \varepsilon, m)$
2: $S, c \leftarrow$ INTEGERIZE$(q, \{p_1, \ldots, p_K\}, w, N, T)$
3: **return** S, c

therefore convert the fractional weights into integer counts that define an equally weighted multiset, preserving the geometric approximation quality (Alg. 1).

Write the indices of the nonzero entries of w as $\{i_1, \ldots, i_{|S|}\} := \{j \mid w_j > 0\}$, let $S := \{s_1, \ldots, s_{|S|}\} = \{p_j \mid w_j > 0\}$ be the corresponding chosen points from Alg. 3, and for every $j \in \{1, \ldots, |S|\}$, let $\tilde{w}_j := w_{i_j}$. We look for integer counts $c \in \mathbb{Z}_{\geq 0}^{|S|}$ with $\sum_j c_j = N$ minimizing the reconstruction error

$$\left\| q - \sum_{j=1}^{|S|} \tfrac{c_j}{N} \, s_j \right\|_2^2, \tag{2}$$

i.e. the N-point multiset over S whose uniform mean is closest to q. Turning the continuous weights into discrete counts is tricky because the rounded multiset must satisfy two constraints at once: it must contain exactly N examples, and its empirical mean should remain close to the Frank–Wolfe convex combination. We therefore view integerization as a small discrete reconstruction problem around the continuous solution. The three steps below progressively restore feasibility and reduce the same geometric objective in Eq. (2):

1. **Floor allocation:** We start by setting $c_j = \lfloor N\tilde{w}_j \rfloor$. This gives each point the number of copies suggested by its Frank–Wolfe weight, while making sure we do not exceed the budget. Very small-weight points naturally get zero copies at this stage.
2. **Greedy fill:** Rounding down leaves a few unallocated copies. We add them one at a time to the point that most reduces the reconstruction error in Eq. (2), so the leftover budget is used to repair the geometric shift introduced by flooring.
3. **Local-swap refinement:** Finally, we check whether moving one copy from one support point to another improves the reconstruction. This keeps the budget fixed, but corrects small rounding mistakes left by the greedy fill. We run this pairwise check for two passes.

After this, each support point s_j is included c_j times to form the final multiset of exactly N sequences.

Full selection algorithm. Combining both steps, the selection pipeline (Alg. 2) takes the candidate pool and budget and returns the final training multiset.

3.2 EFFICIENT FINETUNING VIA GRADIENT REUSE

The integerized multiset contains repeated sequences: a support point s_j with count c_j appears c_j times in the finetuning batch. A naïve implementation treats these copies as independent samples and performs a separate forward-backward pass for each one, causing the computational cost to scale linearly with N regardless of duplication. We exploit the integer multiplicities produced by geometric integerization to amortize gradient computation across repeated optimizer steps (Alg. 4). Rather than recomputing the gradient at every repeated step on s_j, we *cache and reuse gradients*: we refresh $\tilde{g}$ once every r steps and apply the cached value for the intermediate $r-1$ updates. Let $t = 0, 1, \ldots, c_j - 1$ index the repeated steps on s_j:

$$\tilde{g}_t = \begin{cases} \nabla_\theta \mathcal{L}(\theta_t;\, s_j) & \text{if } t \bmod r = 0, \\ \tilde{g}_{t-1} & \text{otherwise,} \end{cases} \qquad \theta_{t+1} = \text{ADAMSTEP}(\theta_t,\, \tilde{g}_t,\, \eta). \tag{3}$$

This reduces forward-backward passes from N to $\lceil N/r \rceil$. For small refresh intervals r, the parameter drift between consecutive steps on identical input is mild enough that reusing a cached gradient preserves convergence quality (see Sec. 4). The full gradient-reuse procedure is defined in Sec. C.1

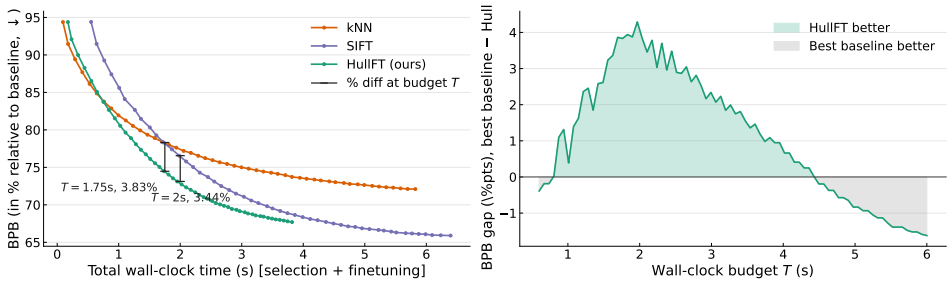

Figure 2: *Left*: BPB% vs total runtime (selection + finetuning) as we sweep $N \in [1, 50]$ for each method. Our method (green) is Pareto-dominant for every budget $T \lesssim 4s$. Vertical lines at T=1.75s and T=2.0s mark the BPB% gap to the best baseline: our method is 3.83% and 3.44% lower than the best baseline at those budgets, respectively. *Right*: Quality gap (best-baseline BPB% − ours) in % as a function of the total-runtime budget T, where the best baseline is the lower-BPB% method among kNN and SIFT at that budget. Positive values (green) mean our method wins at that budget; negative (grey) means the best baseline wins. Our method dominates for every $T \leq 4.5s$ with largest margins at tight budgets, which is exactly the regime a latency-sensitive deployment cares about.

as Alg. 4. For gradient reuse to work, all copies of a given sequence must appear *consecutively* in the training order. Integerization gives this property by construction: multiplicities are fixed upfront, so each unique text is trained for all its c_j steps in a single consecutive block. We test whether this structure can be imposed on SIFT in Sec. 4.

4 Experimental results

The setup. We evaluate on 12 subsets of The Pile (Gao et al., 2020) using GPT-2 (Radford et al., 2019). For each subset, we average the results over 150 test queries. All methods share a K=200 nearest-neighbor candidate pool retrieved via a precomputed FAISS index. Following Hübotter et al. (2025), we use the normalized RoBERTa encoder for embeddings and finetune with Adam (lr=5×10^{-5}). Experiments run on a single NVIDIA A100 GPU, except for the CPU evaluation in Fig. 6. Our metric is Bits-Per-Byte (BPB%) relative to a non-finetuned baseline. Since TTFT happens at inference time, we report the BPB% achievable within a given *total-runtime budget* T (selection + finetuning), sweeping $N \in [1, 50]$. Protocol details are in Sec. A; FW and Carathéodory variants and kNN pool sizing are ablated in Sec. D.

Quality–runtime tradeoff. We compare HullFT against SIFT (Hübotter et al., 2025), which selects examples to reduce response uncertainty by optimizing information gain while accounting for information duplication, and kNN (Hardt & Sun, 2024), which finetunes on top-N nearest neighbors, across a range of total-runtime budgets T. We compare against these two because they are the methods that fit the TTFT pipeline studied here: select training sequences for a given query from an embedding space, finetune on them, then evaluate the query. The other selection methods in Sec. 2 do not fit it as published: influence-based methods score examples independently and so miss redundancy between them (Hübotter et al., 2025), and coreset and active-learning methods are not conditioned on a query. Figure 2 reports the average BPB% across all datasets as a function of total runtime, together with the BPB% gap between HullFT and the best baseline available at each budget. Table 1 reports the corresponding per-subset BPB% values at the two representative budgets T=1.75s and T=2.0s. Across every total-runtime budget $T \leq 4s$, our method achieves a strictly lower BPB% than both SIFT and kNN. The gap over the best baseline is largest at tight budgets: **6.4**% lower at T=0.75s, **3.8**% lower at T=1.75s, **3.4**% lower at T=2.0s, and closes only once our method saturates its N budget near 3.8s. In the latency-critical TTFT regime, where every additional second of total runtime is visible in user-facing latency, our method is Pareto-dominant. The advantage is even stronger on the newer Pythia-410M (Biderman et al., 2023): there, HullFT's selection alone, without Gradient Reuse, already beats SIFT at matched N for every $N \leq 30$, and with Gradient Reuse HullFT is 2.54% and 2.00% lower than the best baseline at T=6.4s and T=9.5s (Sec. B, Fig. 7).

Per-subset breakdown. We test whether the aggregate runtime advantage is driven by a few favorable subsets or appears broadly across The Pile. Figure 3 shows the full per-subset quality–runtime curves

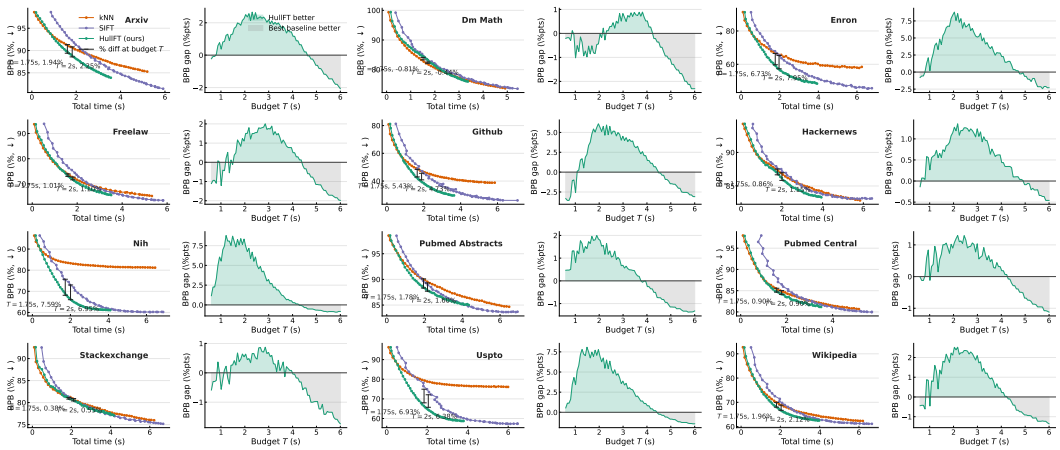

Figure 3: Per-subset breakdown for all 12 Pile subsets. For each subset we show, *left*: BPB% vs. wall-clock time, and *right*: the quality gap (best baseline − HullFT, where the best baseline is the better of kNN and SIFT) as a function of the total-runtime budget T.

Table 1: Per-subset BPB% at time budgets T=1.75s (left) and T=2.0s (right), using each method's average selection size. Δ shows the difference between the best baseline (kNN or SIFT) and HullFT.

	T=1.75s				T=2.0s			
Subset	**kNN**	**SIFT**	**Ours**	**Δ**	**kNN**	**SIFT**	**Ours**	**Δ**
ArXiv	91.33	91.80	**89.39**	**1.94** ↓	90.82	90.84	**88.57**	**2.25** ↓
DM Math.	**83.31**	85.29	84.12	**−0.81** ↑	**82.14**	83.97	82.58	**−0.44** ↑
Enron	66.47	67.84	**59.74**	**6.73** ↓	65.67	65.26	**57.32**	**7.94** ↓
FreeLaw	73.99	76.32	**72.98**	**1.01** ↓	72.73	74.40	**71.63**	**1.10** ↓
GitHub	48.42	48.95	**42.98**	**5.44** ↓	46.97	45.61	**40.88**	**4.73** ↓
HackerNews	87.52	87.92	**86.66**	**0.86** ↓	86.98	87.51	**85.83**	**1.15** ↓
NIH	83.46	75.72	**68.13**	**7.59** ↓	83.08	73.01	**66.05**	**6.95** ↓
PubMed Abs.	90.11	90.27	**88.33**	**1.78** ↓	89.54	89.30	**87.70**	**1.60** ↓
PubMed Cent.	85.57	86.47	**84.67**	**0.90** ↓	85.03	85.31	**84.13**	**0.90** ↓
StackEx.	81.16	81.74	**80.78**	**0.38** ↓	80.91	81.03	**80.37**	**0.54** ↓
USPTO	79.69	74.92	**67.99**	**6.93** ↓	78.86	72.08	**65.70**	**6.38** ↓
Wikipedia	69.96	72.54	**68.01**	**1.95** ↓	68.83	70.41	**66.71**	**2.12** ↓
Avg.	78.42	78.31	**74.48**	**3.83** ↓	77.63	76.56	**73.12**	**3.44** ↓

and best-baseline gaps. Table 1 shows per-subset BPB% for kNN, SIFT, and HullFT at T=1.75s (kNN $N^\star$=15, SIFT $N^\star$=11, HullFT $N^\star$=19) and at T=2.0s (kNN $N^\star$=17, SIFT $N^\star$=13, HullFT $N^\star$=22), using the $N^\star$ each method attains in our sweep for that time budget. At T=1.75s, our method improves over the best performing baseline on 11 of 12 subsets, with an average margin of 3.83%; at T=2.0s the same holds, with an average margin of 3.44% (as in Fig. 2). The quality gap is largest on subsets where finetuning signal continues to improve with larger multisets (Enron, GitHub, USPTO, NIH), which is exactly where our method's ability to reach a higher $N^\star$ within the same time budget matters most. The mechanism is a two-stage speedup: our selection stage is $\mathbf{8.8}\times$ faster than SIFT at N=50 (0.059s vs 0.524s, the saturation point our method reaches near $T\approx$3.8s) and over $\mathbf{12}\times$ faster averaged across $N \in [1, 50]$, and Gradient Reuse further cuts finetuning by $\mathbf{1.48}\times$ (Tab. 4). These speedups compound so that, inside a given time budget, our method can afford a substantially larger $N^\star$, and the extra multiset mass more than offsets matched-N quality differences relative to the baselines (at matched N, SIFT is ahead of HullFT by 0.02–1.17 BPB%; Sec. E.2). Against kNN at matched N (Fig. 8), our BPB% is lower at every $N \geq 10$: pure nearest-neighbour retrieval saturates because its top-k is redundant under the query embedding, whereas the convex-hull support set keeps adding signal.

Gradient reuse ablation. Table 4 in the appendix isolates the effect of Gradient Reuse on our method at N=50. Averaged over 12 subsets (Tab. 2), GR yields a $\mathbf{1.48}\times$ finetuning speedup at a cost of only 0.64% of average BPB%. On subsets where integerization produces few repeats (GitHub, Enron) GR

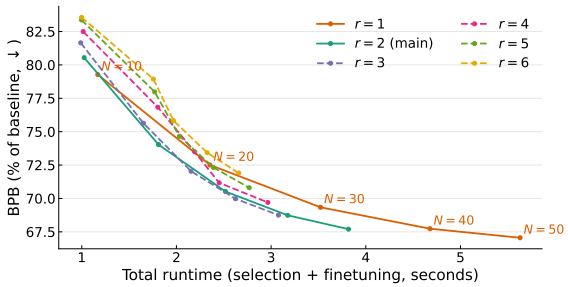

Figure 4: HullFT BPB% vs. total runtime for gradient-reuse depths $r \in \{1, \ldots, 6\}$ at $N \in \{10, \ldots, 50\}$ (r=1: no reuse).

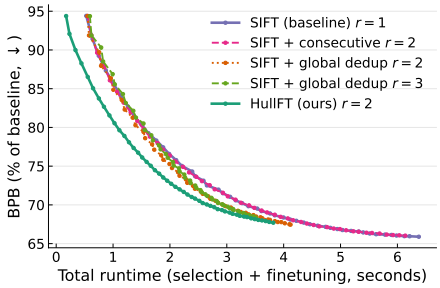

Figure 5: Comparison of SIFT gradient-reuse variants across selection budgets and HullFT.

Table 2: Gradient-reuse variants applied to SIFT-derived schedules at N=50, averaged over 12 Pile subsets, with HullFT without (r=1) and with (r=2) reuse for reference. Time is selection + finetuning.

	SIFT	SIFT + consec.	SIFT + global dedup					HullFT (ours)	
r	1	2	2	3	4	5	6	1	2
BPB% ↓	65.89	66.03	67.45	68.37	69.30	70.26	71.20	67.06	67.70
Time (s) ↓	6.38	6.14	4.16	3.51	3.49	3.31	3.21	5.63	3.82

is effectively a no-op both in runtime and quality, while on subsets with many repeats the speedup is substantially larger. Figure 4 ablates the gradient refresh interval r, where r=1 is the standard finetuning without GR; r=2 already delivers a substantial runtime reduction at a very low BPB% cost, while r=3 saves further wall time at a more noticeable quality cost. We adopt r=2 as our default for the best speedup–accuracy tradeoff.

SIFT gradient reuse experiment. We investigate whether Gradient Reuse can be applied to other selection methods to improve their efficiency. While kNN retrieval (Hardt & Sun, 2024) returns N distinct sequences without repetitions, SIFT (Hübotter et al., 2025) can revisit the same sequence at non-consecutive steps. To enable Gradient Reuse for SIFT, we evaluate two modified schedules: *global deduplication*, which collapses all occurrences of each unique text into a single block with aggregated counts, and *consecutive grouping*, which merges only adjacent duplicates. Figure 5 compares these variants with HullFT across selection budgets, and Tab. 2 at N=50. The results show that Gradient Reuse accelerates SIFT, presenting a clear quality–efficiency tradeoff. Global deduplication provides a substantial speedup but increases the BPB%, while consecutive grouping yields a smaller runtime gain with minimal quality degradation. This cost is unlikely to come from the new order itself: training on a random permutation of SIFT's schedule, without reuse, leaves its BPB% unchanged (See appendix in Sec. E.1, Tab. 7), so departing from SIFT's greedy order is not in itself harmful. Unlike SIFT, HullFT inherently produces repeated contiguous blocks during integerization, allowing it to fully exploit Gradient Reuse without post-hoc reordering.

CPU-only evaluation. We repeat the N=20 comparison on CPU only over 6 Pile subsets (ArXiv, DM Mathematics, Enron, FreeLaw, GitHub, HackerNews). Figure 6 reports mean CPU selection time, finetuning time, total runtime, and BPB% for kNN, SIFT, and HullFT. HullFT's FW selector is $\mathbf{25.8}\times$ faster than SIFT's on CPU ($0.036s$ vs. $0.934s$). Because Gradient Reuse reduces the number of forward-backward passes, finetuning is also faster: $228.4s$ vs. $317.4s$ for kNN and $318.4s$ for SIFT, saving roughly **89s** end-to-end. On BPB%, HullFT (72.05%) is close to kNN (72.78%) and within 2.3% of SIFT (69.78%), while being substantially faster end-to-end (Fig. 6).

Selection diversity. Figure 16 visualizes, for a representative ArXiv query at N=20, the support sets chosen by kNN, SIFT, and HullFT in a 3D t-SNE projection of the candidate pool. kNN concentrates its budget in a tight neighborhood of the query, while SIFT places its points more broadly around the query to reduce redundancy. HullFT yields a comparably diverse support, obtained directly from the geometry of the convex approximation in Sec. 3 rather than from any explicit diversity term. The redundancy statistics in Tab. 6 confirm this across all 12 subsets: at N=50, HullFT and

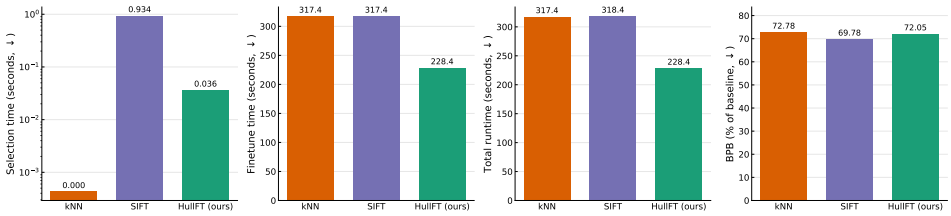

Figure 6: CPU-only comparison at $N{=}20$ over 6 Pile subsets. **Left**: mean selection time on CPU. **Middle-left**: mean finetuning time. **Middle-right**: total runtime (selection + finetuning). **Right**: BPB% relative to the non-finetuned baseline. HullFT is $25.8\times$ faster than SIFT at selection and saves about $89s$ end-to-end, at a modest BPB% cost relative to SIFT.

SIFT are statistically tied in mean nearest-neighbor distance (26.9 vs. 25.8, $\times 10^{-3}$), while kNN's is about $15\times$ smaller (1.8). This diversity is achieved at a fraction of SIFT's per-query selection cost (Sec. 4), consistent with our central thesis that geometric reconstruction recovers a comparably diverse selection at substantially lower latency.

More results. The appendix gives the full Pythia-410M results (Sec. B), ablations and per-subset breakdowns (Sec. D), the diversity analysis (Sec. D.9), qualitative examples (Sec. D.10), and the training-order and matched-N analyses (Sec. E).

5 Conclusion and Limitations

We introduced **HullFT**, a test-time finetuning pipeline that frames data selection as sparse convex approximation of the query embedding over a candidate pool. Across 12 subsets of The Pile, HullFT attains a strictly lower BPB% than both kNN retrieval and SIFT at every total-runtime budget $T \leq 4s$, with the margin widening as the budget tightens. Beyond that regime the curves converge, and at matched N SIFT remains slightly ahead in quality. The same picture holds with Pythia-410M as the base model, and the gains do not depend on the order in which the multiset is trained (Sec. E.1). The gains come from a compounded two-stage speedup (selection is $12\times$ faster than SIFT on average and Gradient Reuse cuts finetuning by $1.48\times$), which lets HullFT reach a substantially larger effective N inside a fixed latency budget. These results suggest that convex geometry is a natural vehicle for latency-constrained test-time adaptation, and that the integer structure it induces opens further room for optimizer-level amortization.

Limitations. Similarly to SIFT (Hübotter et al., 2025), HullFT operates over a kNN-preselected candidate pool rather than the full corpus: the convex approximation is only as expressive as the directions present in this pool, so the quality of the support set ultimately inherits the recall ceiling of the upstream retriever and embedding model. Second, selection runs in a fixed embedding space decoupled from the base LLM: this keeps the method general and model-agnostic, but the geometry is not tailored to the loss landscape of the model being adapted, leaving potential per-model gains on the table. Third, all our experiments use a single embedding model, the RoBERTa encoder used by SIFT (Hübotter et al., 2025). We held it fixed for comparability with the baselines; since selection only treats the embeddings as points in a vector space, we expect the method to carry over to other encoders, but we have not tested this.

AI Use Statement

We used large language models (LLMs) solely as writing assistants, to improve the grammar, clarity, and readability of the manuscript. LLMs were not used for research ideation, method design, experiment design or execution, data generation, code implementation, or interpretation of results. All text was reviewed and verified by the authors, who take full responsibility for the content of this paper.

REFERENCES

Amro Abbas, Kushal Tirumala, Daniel Simig, Surya Ganguli, and Ari S Morcos. SemDeDup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. In Jacob E Goodman, János Pach, and Emo Welzl (eds.), *Combinatorial and Computational Geometry*. Cambridge University Press, 2005.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for abstract reasoning. *arXiv preprint arXiv:2411.07279*, 2024.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pp. 72–78, 2019.

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

Siddharth Barman. Approximating Nash equilibria and dense bipartite subgraphs via an approximate version of Carathéodory's theorem. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pp. 361–369, 2015.

Cenk Baykal, Lucas Liebenwein, Igor Gilitschenski, Dan Feldman, and Daniela Rus. Sensitivity-informed provable pruning of neural networks. *SIAM Journal on Mathematics of Data Science*, 4 (1):26–45, 2022.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3615–3620, 2019.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 2206–2240, 2022.

Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889*, 2016.

Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *Proceedings of the 36th International Conference on Machine Learning*, pp. 872–881, 2019.

Constantin Carathéodory. Über den Variabilitätsbereich der Koeffizienten von Potenzreihen, die gegebene Werte nicht annehmen. *Mathematische Annalen*, 64(1):95–115, 1907.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336, 1998.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2898–2904, 2020.

Kenneth L Clarkson. Coresets, sparse greedy approximation, and the Frank–Wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):63:1–63:30, 2010.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020.

Cyrille W. Combettes and Sebastian Pokutta. Revisiting the approximate carathéodory problem via the frank–wolfe algorithm. *Mathematical Programming*, 197(1):191–214, 2023.

Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, pp. 569–578, 2011.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110, 1956.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time training with masked autoencoders. In *Advances in Neural Information Processing Systems*, 2022.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Jacques Guélat and Patrice Marcotte. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35:110–119, 1986.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, 2020.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69:383–393, 1974.

Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2011.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, 2018.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. ClinicalBERT: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. Efficiently learning at test-time: Active fine-tuning of LLMs. In *International Conference on Learning Representations*, 2025.

Neha Hulkund, Alaa Maalouf, Levi Cai, Daniel Yang, Tsun-Hsuan Wang, Abigail O'Neil, Timm Haucke, Sandeep Mukherjee, Vikram Ramaswamy, Judy Hansen Shen, Gabriel Tseng, Mike Walmsley, Daniela Rus, Ken Goldberg, Hannah Kerner, Irene Chen, Yogesh Girdhar, and Sara Beery. DataS^3: Dataset subset selection for specialization. *arXiv preprint arXiv:2504.16277*, 2025.

Martin Jaggi. Revisiting Frank–Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 427–435, 2013.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 200–209, 1999.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. GRAD-MATCH: Gradient matching based data subset selection for efficient deep model training. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 5464–5474, 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.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1885–1894, 2017.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. DataInf: Efficiently estimating data influence in LoRA-tuned LLMs and diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

Simon Lacoste-Julien. Convergence rate of Frank–Wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2020.

Alaa Maalouf, Ibrahim Jubran, and Dan Feldman. Fast and accurate least-mean-squares solvers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, volume 32, 2019.

Alaa Maalouf, Ibrahim Jubran, Murad Tukan, and Dan Feldman. Coresets for the average case error for finite query sets. *Sensors*, 21(19), 2021.

Alaa Maalouf, Murad Tukan, Vladimir Braverman, and Daniela Rus. AutoCoreset: An automatic practical coreset construction framework. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 23451–23466, 2023.

Alaa Maalouf, Gilad Eini, Ben Mussay, Dan Feldman, and Margarita Osadchy. A unified approach to coreset learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5):6893–6905, 2024.

Amir Mallak and Alaa Maalouf. Robustness is a function, not a number: A factorized comprehensive study of OOD robustness in vision-based driving. *arXiv preprint arXiv:2602.09018*, 2026.

B. Maurey. Théorèmes de factorisation pour les opérateurs linéaires à valeurs dans un espace $l^p(\omega, \mu), 0 < p \leq +\infty$. *Séminaire Maurey-Schwartz*, pp. 1–8, 1972-1973.

Vahab Mirrokni, Renato Paes Leme, Adrian Vladu, and Sam Chiu-wai Wong. Tight bounds for approximate Carathéodory and beyond. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2440–2448, 2017.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 6950–6960, 2020.

Alexander Munteanu and Chris Schwiegelshohn. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *KI - Künstliche Intelligenz*, 32(1): 37–53, 2018.

Ben Mussay, Dan Feldman, Samson Zhou, Vladimir Braverman, and Margarita Osadchy. Data-independent structured pruning of neural networks via coresets. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7829–7841, 2022.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

Jeff M Phillips. Coresets and sketches. In *Handbook of Discrete and Computational Geometry*. CRC Press, 2017.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, volume 33, pp. 19920–19930, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Technical Report, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed H Chi, James Caverlee, Julian McAuley, and Derek Zhiyuan Cheng. How to train data-efficient LLMs. *arXiv preprint arXiv:2402.09668*, 2024.

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

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, Blaise Agüera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

Shiva Sreeram, Alaa Maalouf, Pratyusha Sharma, and Daniela Rus. Compress to impress: Efficient LLM adaptation using a single gradient step on 100 samples. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*, 2025.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9229–9248, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Forty-second International Conference on Machine Learning*, 2025.

Murad Tukan, Loay Mualem, and Alaa Maalouf. Pruning neural networks via coresets and convex geometry: Towards no assumptions. In *Advances in Neural Information Processing Systems*, 2022.

Murad Tukan, Samson Zhou, Alaa Maalouf, Daniela Rus, Vladimir Braverman, and Dan Feldman. Provable data subset selection for efficient neural networks training. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 34533–34555, 2023.

Tsun-Hsuan Wang, Alaa Maalouf, Wei Xiao, Yutong Ban, Alexander Amini, Guy Rosman, Sertac Karaman, and Daniela Rus. Drive anywhere: Generalizable end-to-end autonomous driving with multi-modal foundation models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6687–6694. IEEE, 2024.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 4003–4012, 2020.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. QuRating: Selecting high-quality data for training language models. In *International Conference on Machine Learning*, 2024.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning*, 2024.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

A EXPERIMENTAL PROTOCOL DETAILS

The main experiments use 12 Pile subsets and 150 test queries per subset. This protocol is a compute-conscious version of the evaluation style used by TTT-NN (Hardt & Sun, 2024) and SIFT (Hübotter et al., 2025), which evaluate approximately 1% of the full Pile test set (1812 points) and report only on subsets with more than 10 test examples. We use the larger fixed per-subset sample because our goal is not only to report a single aggregate number, but also to understand whether HullFT's runtime–quality behavior is stable across domains with different redundancy patterns. A balanced 150-query sample gives each subset enough weight for per-domain comparisons while keeping the full sweep over methods, budgets, and ablations feasible on our available A100 compute.

We build and use precomputed per-subset retrieval pools rather than a single live global index. This keeps the comparison focused on the TTFT components studied in the paper: query-conditioned selection from a shared candidate pool and the subsequent finetuning schedule. All methods receive the same K precomputed nearest neighbors for each query, so retrieval differences cannot explain the reported gaps. We also exclude pool retrieval time from the main runtime measurements because it is identical across methods once the candidate pool is fixed; selection and finetuning are the parts changed by HullFT. For context, Hardt & Sun (2024) report mean distributed Pile retrieval time of $1.35s$ (std. $0.12s$).

Unless otherwise stated, all main experiments use normalized RoBERTa embeddings from *socialfoundations/roberta-large-pile-lr2e-5-bs16-8gpu-1700000*, GPT-2 as the adapted model, Adam with learning rate 5×10^{-5}, K=200 preselected candidates, and selection budgets $N \in \{1, \ldots, 50\}$.

B A SECOND BASE MODEL: PYTHIA-410M

The setup matches GPT-2 except for $N \in \{5, 10, \ldots, 50\}$ and the learning rate, 10^{-6}. Table 3 gives the full results. HullFT is strong even without Gradient Reuse: at matched N it is ahead of SIFT for $N \leq 30$ and within 0.27 BPB% of it up to N=50, and under a runtime budget it leads the best baseline at every $3s \leq T \leq 10.5s$, by up to 2.81%. kNN saturates near 88%. Gradient Reuse costs 0.08 BPB% at N=50 and cuts finetuning by $1.54\times$ ($14.34s \rightarrow 9.32s$), and FW selection is $5.7\times$ faster than SIFT's ($0.13s$ vs. $0.72s$). HullFT trails the best baseline only for $2.15s \leq T < 2.64s$ (by 0.08, where both afford only N=5) and past its saturation point.

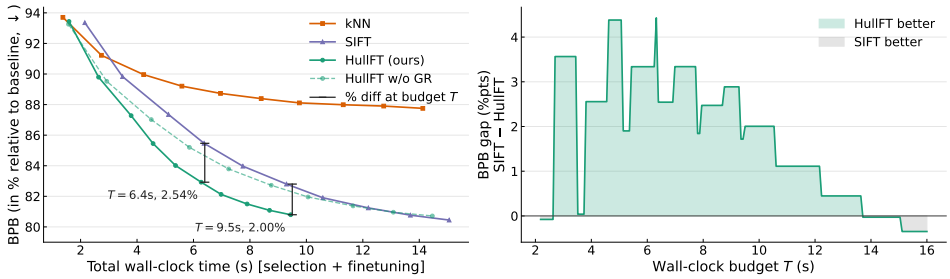

Figure 7: Pythia-410M as the base model, averaged over 12 Pile subsets. *Left*: BPB% vs. total runtime (selection + finetuning) as we sweep $N \in \{5, 10, \ldots, 50\}$; HullFT (ours) uses Gradient Reuse (r=2), and the dashed line shows it without reuse. Vertical lines at T=$6.4s$ and T=$9.5s$ mark the gap to SIFT at those budgets. *Right*: quality gap (SIFT BPB% − HullFT) as a function of the total-runtime budget T.

Table 3: Pythia-410M as the base model, averaged over 12 Pile subsets (150 queries each). Top: BPB% (lower is better). Bottom: total runtime per query in seconds (selection + finetuning). HullFT is shown without and with Gradient Reuse (r=2). Bold marks the best BPB% and the fastest runtime in each column.

		N=5	N=10	N=15	N=20	N=25	N=30	N=35	N=40	N=45	N=50
BPB% ↓	kNN	93.71	91.22	89.96	89.21	88.73	88.39	88.11	87.99	87.90	87.75
	SIFT	93.36	89.83	87.35	85.47	83.97	82.80	**81.91**	**81.24**	**80.77**	**80.45**
	HullFT w/o GR	**93.26**	**89.52**	**87.02**	**85.20**	**83.79**	**82.72**	81.96	81.37	80.96	80.72
	HullFT w/ GR	93.44	89.79	87.28	85.45	84.02	82.92	82.13	81.50	81.08	80.80
Time (s) ↓	kNN	**1.37**	2.74	4.24	5.59	6.96	8.40	9.75	11.31	12.74	14.13
	SIFT	2.15	3.49	5.12	6.38	7.75	9.30	10.58	12.20	13.69	15.06
	HullFT w/o GR	1.57	2.93	4.51	5.85	7.25	8.76	10.06	11.65	13.08	14.46
	HullFT w/ GR	1.59	**2.64**	**3.79**	**4.57**	**5.36**	**6.27**	**6.98**	**7.90**	**8.70**	**9.44**

C Algorithmic details

This section provides pseudocode for the core procedures, algorithmic variants, and baselines referenced in Sec. 3. The variants modify individual stages of the HullFT pipeline.

C.1 Core HullFT procedures

Algorithm 3 and Alg. 4 give the procedures referenced in Sec. 3. Frank–Wolfe is a classical conditional-gradient method (Frank & Wolfe, 1956; Jaggi, 2013); we include its specialization to our objective for completeness, not as a new algorithmic contribution.

Algorithm 3 FrankWolfe$(q, \{p_1, \dots, p_K\}, \varepsilon, m)$

Input: query $q \in \mathbb{R}^d$, candidate pool $\{p_1, \dots, p_K\} \subset \mathbb{R}^d$, tolerance ε, support cap m
Output: sparse weights $w \in \Delta^K$
Form $P \in \mathbb{R}^{d\times K}$ with p_i as the i-th column
$v^\star \leftarrow \arg\max_{i\in[K]} \langle q, p_i\rangle$; $\quad w \leftarrow e_{v^\star}$
while $\|q - Pw\|_2^2 > \varepsilon$ **and** $|\{i : w_i > 0\}| < m$ **do**
  $r \leftarrow q - Pw$
  $v \leftarrow \arg\max_{i\in[K]} \langle r, p_i\rangle$
  $\gamma^\star \leftarrow \arg\min_{\gamma\in[0,1]} \|r - \gamma(p_v - Pw)\|_2^2$ ▷ line search
  $w \leftarrow (1-\gamma^\star)w + \gamma^\star e_v$
return w

Algorithm 4 GradReuse$(\theta_0, S, c, r, \eta)$

Input: Base model θ_0, support points $S = \{s_1, \dots, s_{|S|}\}$, counts $c \in \mathbb{Z}_{\geq 0}^{|S|}$, refresh interval r, learning rate η
Output: Adapted model θ
$\theta \leftarrow \theta_0$
for $j = 1$ **to** $|S|$ with $c_j > 0$ **do**
  for $t = 0$ **to** $c_j - 1$ **do**
    if $t \bmod r = 0$ **then**
      $\tilde{g} \leftarrow \nabla_\theta \mathcal{L}(\theta;\, s_j)$ ▷ forward-backward pass
    $\theta \leftarrow$ AdamStep$(\theta,\, \tilde{g},\, \eta)$
return θ

C.2 EXACT CARATHÉODORY REDUCTION

The exact Carathéodory theorem guarantees that any point in the convex hull of a set in $\mathbb{R}^d$ can be expressed as a convex combination of at most $d+1$ points. Algorithm 5 implements the standard constructive procedure: it iteratively finds an affine dependency among the current support points and eliminates the one that drives a weight to zero, repeating until the size bound $d+1$ is met.

Algorithm 5 CARATHÉODORY$(\{p_1, \ldots, p_k\},\, w)$

Input: Set of points $\{p_1, \ldots, p_k\} \subset \mathbb{R}^d$, weights $w \in \Delta^k$
Output: Support points $S \subseteq \{p_1, \ldots, p_k\}$, new weights w' with $|S| \le d+1$
$A \leftarrow \{1, \ldots, k\}$ ▷ active index set
while $|A| > d+1$ **do**
  Find non-trivial coefficients $(\alpha_i)_{i \in A}$ such that $\sum_{i \in A} \alpha_i\, p_i = 0$ and $\sum_{i \in A} \alpha_i = 0$
  $i^\star \leftarrow \arg\min_{i \in A\,:\,\alpha_i > 0} \frac{w_i}{\alpha_i}$; $\quad \gamma \leftarrow w_{i^\star} / \alpha_{i^\star}$
  $w_i \leftarrow w_i - \gamma \alpha_i$ for all $i \in A$
  $A \leftarrow A \setminus \{i^\star\}$
$S \leftarrow \{p_i : i \in A\}$; $\; w' \leftarrow (w_i)_{i \in A}$
return S, w'

Figure 10 (see Sec. D.3) compares exact Carathéodory selection against HullFT with different integerization strategies. The key finding is that geometric integerization is essential: padding Carathéodory weights directly is unstable as N grows, while FW with geometric integerization (HullFT) matches or beats the best Carathéodory variant at lower selection cost.

C.3 PCA-REDUCED CONVEX APPROXIMATION

When the embedding dimension d is large, inner-product evaluations inside Frank–Wolfe dominate selection cost. Algorithm 6 mitigates this by projecting the query and candidate pool into a lower-dimensional subspace via PCA before running the geometric selection, then mapping the chosen support back to the original space for integerization.

Algorithm 6 PCA_SELECTION$(q, \{p_1, \ldots, p_K\}, N, \varepsilon, m, T, d')$

Input: Query q, candidate pool $\{p_1, \ldots, p_K\}$, budget N, FW tolerance ε, support cap m, swap passes T, target dimension d'
Output: Support points S, integer counts c
Fit PCA on $\{p_1, \ldots, p_K\} \cup \{q\}$ to reduce dimensionality to d'
$q' \leftarrow$ TRANSFORM(q); $\quad p'_i \leftarrow$ TRANSFORM(p_i) for $i = 1, \ldots, K$
$w \leftarrow$ FRANKWOLFE$(q', \{p'_1, \ldots, p'_K\}, \varepsilon, m)$ ▷ or CARATHÉODORY via Alg. 5
$S', c \leftarrow$ INTEGERIZE$(q', \{p'_1, \ldots, p'_K\}, w, N, T)$
Map S' back to the original pool: $S \leftarrow \{p_i : p'_i \in S'\}$ (preserving order)
return S, c

C.4 WEIGHT-PROPORTIONAL PADDING

As a simpler baseline to the full geometric integerization (Alg. 1), Alg. 7 allocates the discrete budget N proportionally to the fractional weights via largest-remainder rounding, without the subsequent local-swap refinement step.

Algorithm 7 PADBYWEIGHTS(w, N)

Input: Fractional weights $w \in \Delta^k$, budget N
Output: Integer counts $c \in \mathbb{Z}^k_{\geq 0}$ such that $\sum_i c_i = N$
$c_i \leftarrow \lfloor N w_i \rfloor$ for all $i \in \{1, \ldots, k\}$
$\rho \leftarrow N - \sum_i c_i$ ▷ remaining budget to allocate
Let I be the list of indices sorted descending by fractional remainder $N w_i - c_i$
for $j = 1$ **to** ρ **do**
    $i \leftarrow I[j]$
    $c_i \leftarrow c_i + 1$
return c

Figure 9 (see Sec. D.2) compares this simpler baseline against the full geometric integerization used by HullFT.

D ABLATIONS

This section isolates the effect of individual design choices. Each ablation sweeps one axis while holding the rest of the pipeline fixed to the configuration used in Sec. 4.

For completeness, Fig. 8 reports BPB% as a function of the selection budget N across the three main methods.

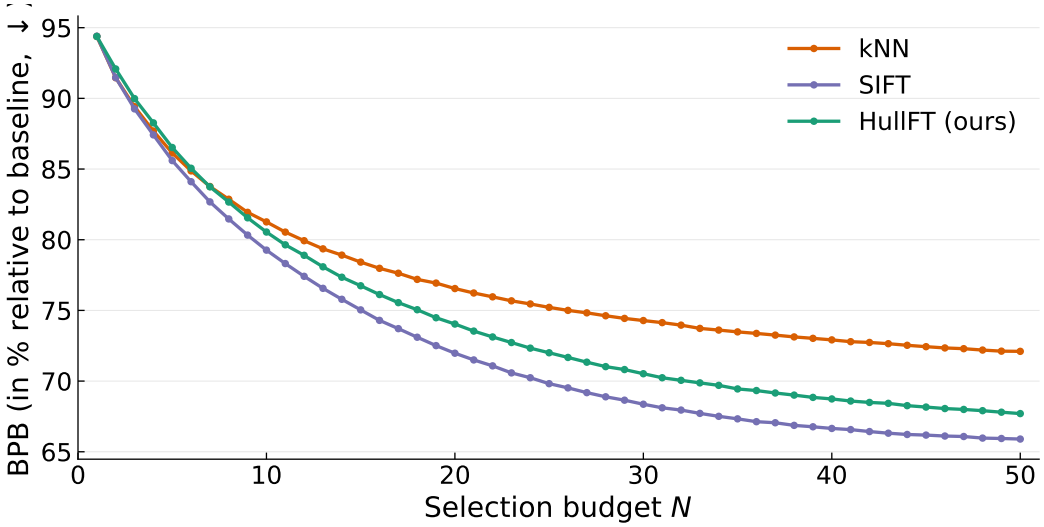

Figure 8: BPB% averaged over 12 Pile subsets as a function of selection budget N for kNN, SIFT, and ours.

D.1 PER-SUBSET GRADIENT REUSE ABLATION

Table 4: Ablation: our method with and without Gradient Reuse (at r=2) at N=50. GR yields a **1.48**× average finetuning speedup at a cost of 0.64% average BPB%; on subsets where the integerized multiset has few repeats (GitHub, Enron) GR is a no-op both in time and quality, while on subsets with more repeats the speedup is larger.

		ArXiv	DM Math	Enron	FreeLaw	GitHub	HackerNews	NIH	PubMed Abs.	PubMed Central	StackExchange	USPTO	Wikipedia	Average
BPB% ↓	w/o	83.19	74.08	48.70	64.32	29.64	82.83	60.71	84.42	80.59	76.28	58.08	61.82	67.06
	w/	83.90	76.00	48.42	65.58	29.64	83.31	61.08	85.00	81.16	77.08	58.72	62.52	67.70
FT (s)	w/o	5.40	5.04	3.87	5.47	3.58	6.03	6.62	6.62	5.31	6.38	6.16	6.37	5.57
	w/	3.49	3.30	3.88	3.45	3.60	3.87	4.04	4.28	3.43	4.10	3.75	3.92	3.76
Speedup		**1.54**×	**1.53**×	**1.00**×	**1.59**×	**1.00**×	**1.56**×	**1.64**×	**1.55**×	**1.55**×	**1.56**×	**1.64**×	**1.63**×	**1.48**×

D.2 INTEGERIZATION: PAD-BY-WEIGHTS VS. GEOMETRIC (FW FAMILY)

We compare the fractional FW solution (no integerization), FW followed by padding by weights (Alg. 7), and FW followed by the geometric integerization used by HullFT (Alg. 1). Figure 9 reports BPB% for these three FW-based variants both as a function of selection budget N and as a function of total runtime. The right panel shows the same comparison on a total-runtime x-axis: since all three variants share the same FW selection stage, the runtime difference is negligible and quality vs time differences are the dominant effect.

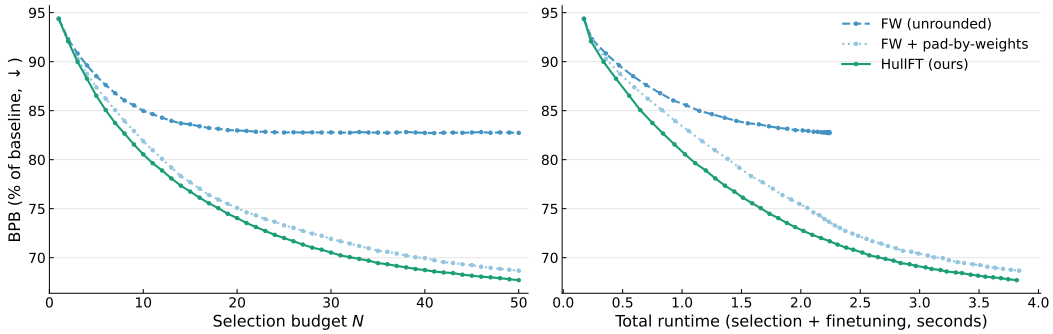

Figure 9: BPB% for the three FW integerization strategies. **Left**: BPB% vs. selection budget N. **Right**: BPB% vs. total runtime in seconds.

D.3 CARATHÉODORY SELECTION AND ITS INTEGERIZATIONS

Replacing FW with exact Carathéodory (Alg. 5) yields a fractional solution with at most $d{+}1$ points; we compare its pad-by-weights and geometric integerizations against HullFT. Figure 10 reports BPB% for these Carathéodory-based variants and HullFT, plotted against both selection budget and total runtime. Geometric integerization is essential: padding Carathéodory weights directly is unstable as N grows. The right panel makes the runtime cost of exact Carathéodory visible: its selection time is roughly $14\times$ higher than FW, so even when its BPB% at matched N is competitive, HullFT dominates on the total-runtime axis.

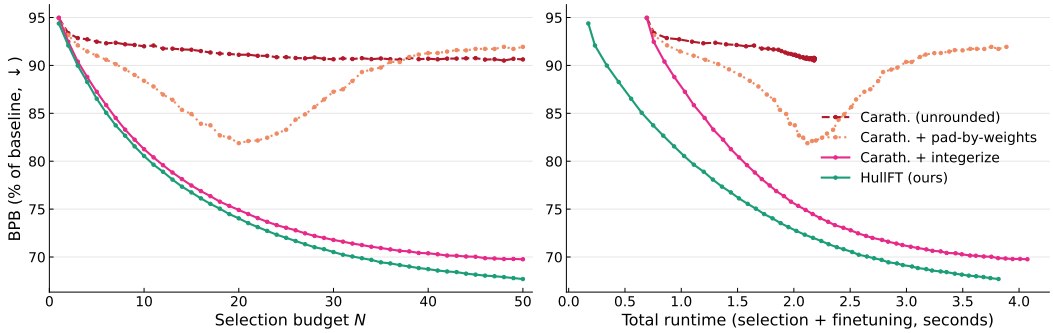

Figure 10: BPB% for Carathéodory selection (Alg. 5) with three integerization strategies, with HullFT (FW + geometric integerization) plotted for reference. **Left**: BPB% vs. selection budget N. **Right**: BPB% vs. total runtime (selection + finetuning, seconds); the Carathéodory curves shift right relative to HullFT, reflecting its higher selection cost.

D.4 FRANK–WOLFE TOLERANCE ε

We sweep the FW stopping tolerance $\varepsilon \in \{0, 10^{-8}, 10^{-5}, 10^{-2}, 10^{-1}\}$ for HullFT and for the Carathéodory-integerized variant. Figure 11 reports BPB% under this tolerance sweep at $N{=}20$ and $N{=}50$. The x-axis is *inverted*: the left end is the loosest tolerance ($\varepsilon{=}10^{-1}$) and the right end is the exact solution ($\varepsilon{=}0$), so moving right means strictly tighter convergence. HullFT's BPB% is flat

across the entire range up through $\varepsilon{=}10^{-2}$, indicating that the support cap m, not the tolerance, is the binding constraint in practice. Meanwhile, the Carathéodory variant degrades noticeably once $\varepsilon{>}10^{-5}$, justifying the moderate tolerance used in the main experiments.

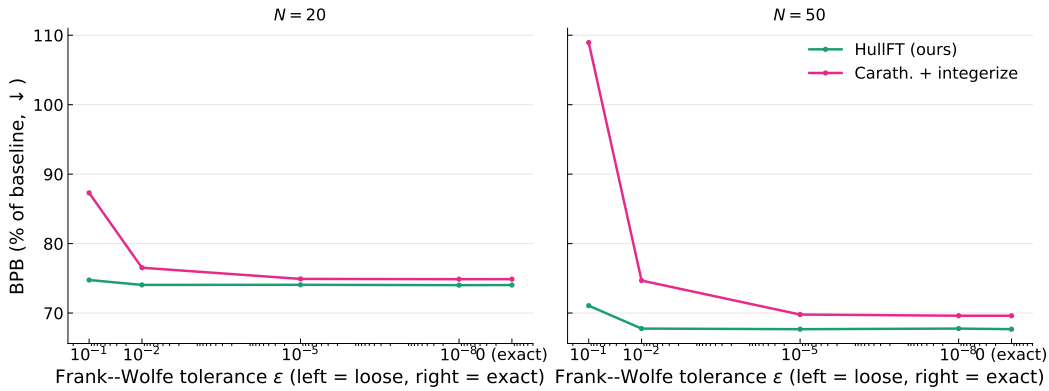

Figure 11: BPB% vs. FW tolerance ε at $N{=}20$ (left) and $N{=}50$ (right). The x-axis is inverted so that the plot reads left-to-right as loose-to-exact; $\varepsilon{=}0$ (exact solution) is placed one decade beyond 10^{-8} for visibility.

D.5 FW VARIANT: FORCED-UNIQUE SELECTION WITHOUT ε EARLY-STOP

We ask whether the ε early-stop and the allowance for point revisits in standard HullFT are important. The `fw_no_epsilon` variant disables the ε early-stop and forces each FW iteration to select a previously unchosen point. When FW keeps adding distinct points, the weights assigned to them become spread thinly, so many of the selected points carry negligible weight and contribute little information. Geometric integerization handles this gracefully: by selecting only the most important points from the support, it recovers performance on par with standard HullFT. Variants that use the full point set without this selection step do not benefit from the additional points and perform noticeably worse. Table 5 reports total runtime and BPB% relative to the non-finetuned baseline at $N \in \{20, 25, 30, 35\}$; all four variants are within $0.15s$ of each other.

Table 5: Total runtime (seconds) and BPB% relative to the non-finetuned baseline (lower is better) for forced-unique Frank–Wolfe variants across different selection budgets N.

Method	$N = 20$	$N = 25$	$N = 30$	$N = 35$
FW no-epsilon	$1.93s$ (81.36%)	$2.27s$ (80.80%)	$2.58s$ (80.22%)	$2.88s$ (79.98%)
FW no-epsilon + pad-by-weights	$1.98s$ (81.60%)	$2.32s$ (80.82%)	$2.63s$ (80.38%)	$2.93s$ (80.14%)
FW no-epsilon + integerize	$1.78s$ (74.03%)	$2.13s$ (71.93%)	$2.47s$ (70.44%)	$2.81s$ (69.44%)
HullFT (ours)	$1.81s$ (74.04%)	$2.17s$ (72.00%)	$2.52s$ (70.52%)	$2.84s$ (69.45%)

D.6 kNN PRE-SELECTION POOL SIZE k

We vary the pre-selection pool k fed to the three selectors at two representative budgets $N \in \{20, 50\}$. Figure 12 reports BPB% as the pre-selection pool size varies at each of these budgets. All methods improve slightly with larger k, with returns saturating by $k{=}500$; HullFT stays strictly between kNN and SIFT at matched N and remains Pareto-dominant on total runtime at every k.

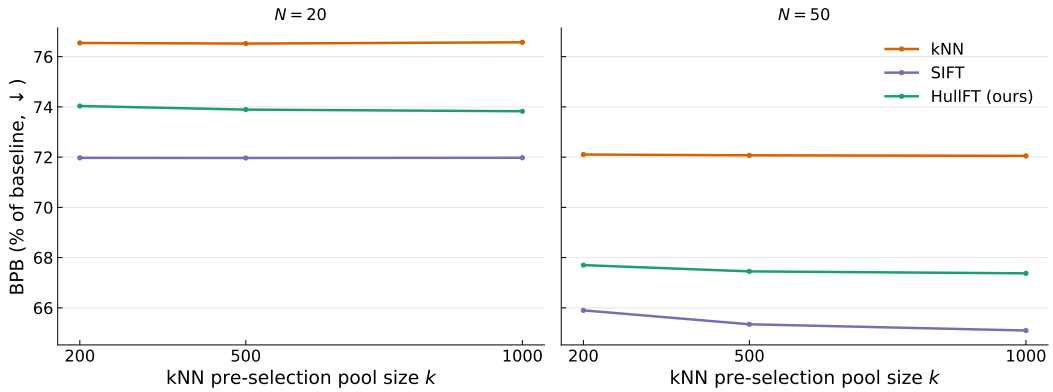

Figure 12: Effect of the kNN pre-selection pool size k on BPB% at N=20 (left) and N=50 (right).

D.7 Per-subset breakdown against each baseline individually

Figure 3 in the main text reports the per-subset gap against the *best* of kNN and SIFT at each budget. For completeness, Figs. 13 and 14 show the same breakdown against each baseline in isolation, so the reader can see where HullFT's advantage comes from kNN vs. SIFT separately. At each anchor T, $N^\star$ is taken from the global aggregate (matching Tab. 1) and per-subset BPB% is read at those fixed $N^\star$ values.

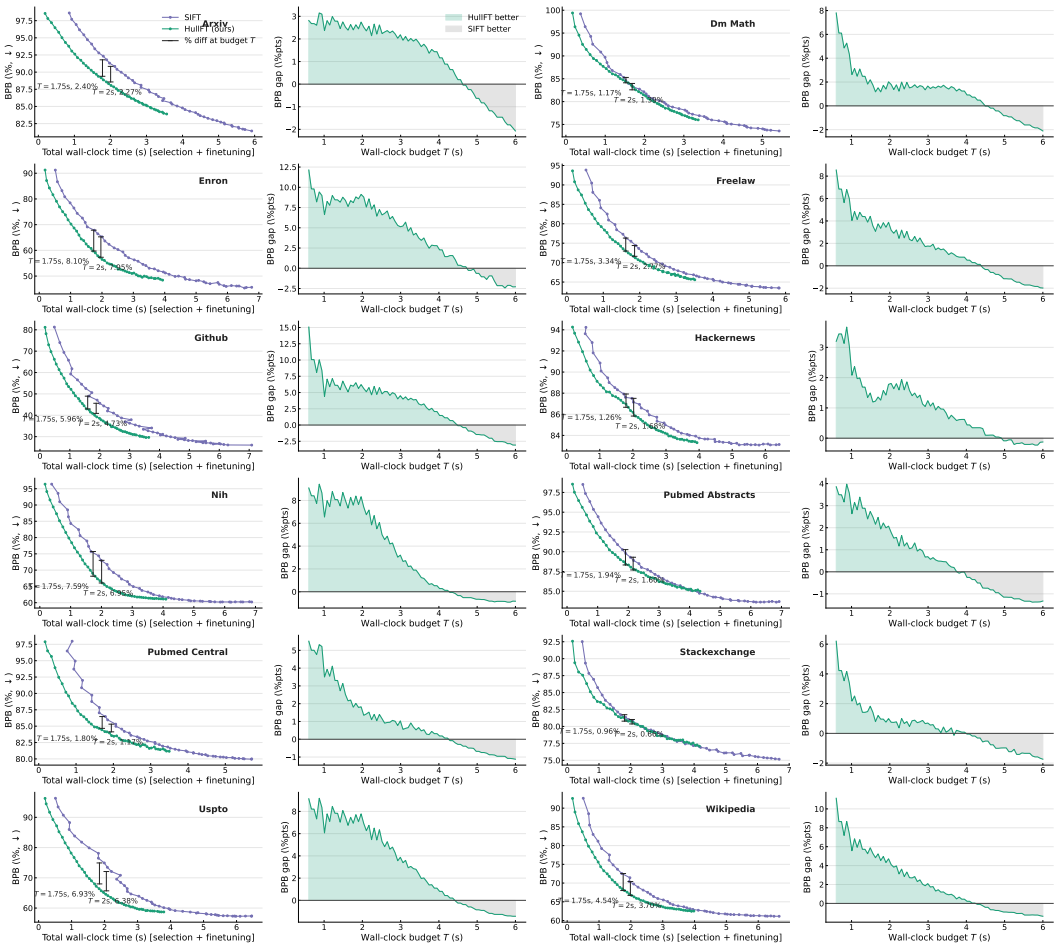

Figure 13: Per-subset breakdown against **SIFT** for all 12 Pile subsets (2 subsets per row). For each subset we show, *left*: BPB% vs. wall-clock time, and *right*: the quality gap (SIFT − HullFT) as a function of the total-runtime budget T.

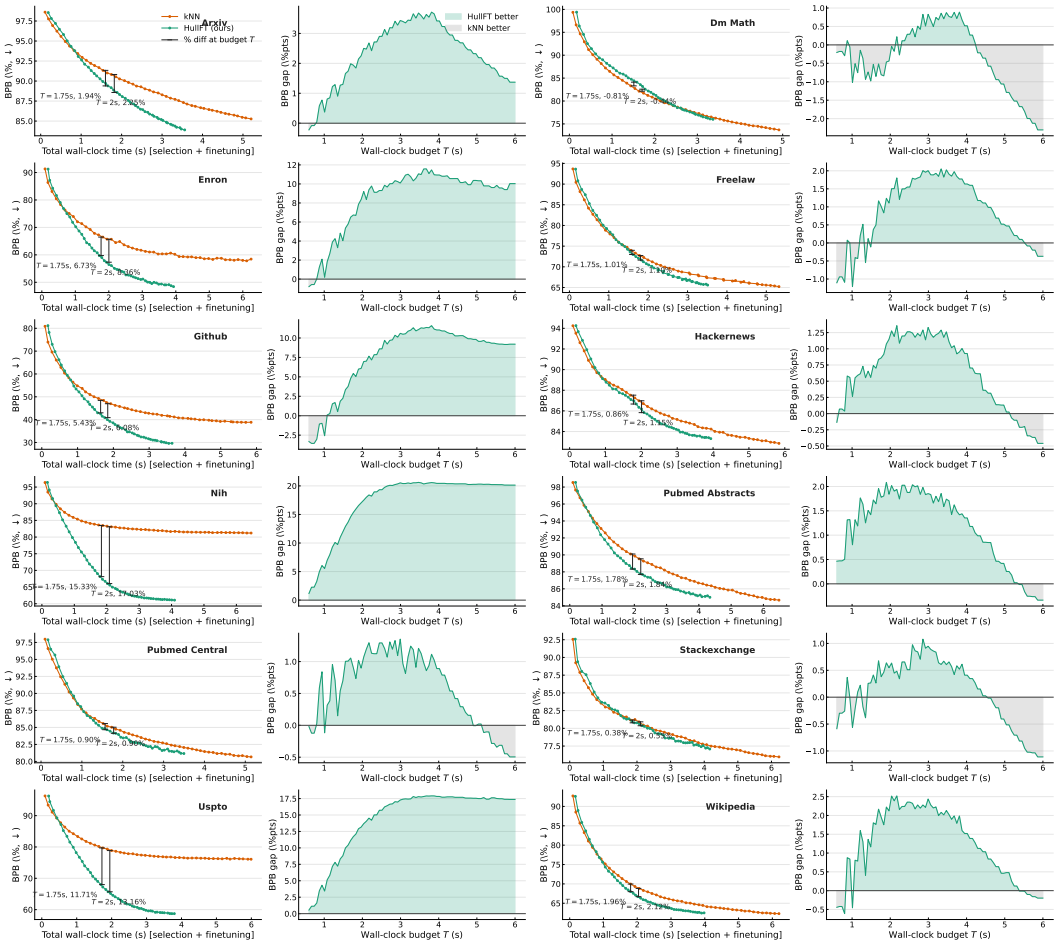

Figure 14: Per-subset breakdown against **kNN** for all 12 Pile subsets (2 subsets per row). For each subset we show, *left*: BPB% vs. wall-clock time, and *right*: the quality gap (kNN − HullFT) as a function of the total-runtime budget T.

D.8 Fidelity of geometric integerization to Frank–Wolfe weights

Frank–Wolfe produces fractional weights on a sparse support; geometric integerization converts these into an exact N-tuple of pool indices with multiplicities c_i, so the induced empirical weights are $\hat{w}_i = c_i/N$ on that same support. We quantify how much the discrete allocation departs from the pre-integerization FW vector by the per-query distance $\|\mathbf{w} - \hat{\mathbf{w}}\|_2$, averaged over all test queries in each subset. Fig. 15 reports the mean of these averages across the twelve Pile subsets for which we precomputed embeddings (shaded band: $\pm$ one standard deviation across subsets). The distance stays in a moderate range for every $N \leq 50$, which supports the view that integerization preserves the FW weight profile rather than inducing an unrelated discrete selection.

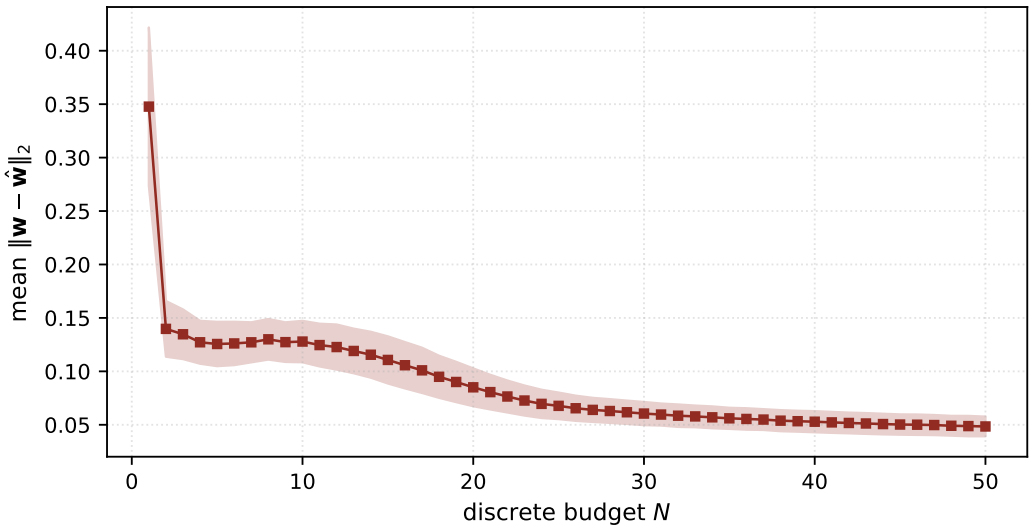

Figure 15: Fidelity of geometric integerization to Frank–Wolfe weights. Mean ℓ_2 distance between FW weights $\mathbf{w}$ on the support and empirical weights $\hat{\mathbf{w}}$ with $\hat{w}_i = c_i/N$. Each point averages over test queries within a subset, then over twelve Pile subsets; the band shows $\pm$ one standard deviation across subsets.

D.9 Diversity

We extend the qualitative analysis of the main text to two additional subsets, GitHub and FreeLaw, which exhibit different candidate-pool structure. The pattern observed for ArXiv recurs: kNN clusters tightly around the query, SIFT places its points more broadly around the query to reduce redundancy, and HullFT produces a comparably diverse support obtained directly from the convex approximation in Sec. 3. This diversity is achieved at a fraction of SIFT's per-query selection cost, reinforcing the headline efficiency–quality tradeoff reported in the main text.



Figure 16: 3D t-SNE projections of the candidate pool (gray) and the $N{=}20$ points selected by kNN, SIFT, and HullFT (colored) for a representative test query (pink star) from ArXiv (top), GitHub (middle) and FreeLaw (bottom). HullFT delivers a selection geometry comparable to SIFT through convex reconstruction, at substantially lower selection latency.

Because a single projection is only illustrative, Tab. 6 measures redundancy directly on the distinct selected support across all 12 subsets and 150 queries each. HullFT and SIFT are statistically tied on every spread metric at the same similarity to the query, with HullFT marginally ahead on all of them. We read this as parity rather than a win: HullFT's distinct support is slightly smaller (14.5 vs. 15.9 points at $N{=}50$), and spread metrics tend to favor smaller sets. kNN shows the near-duplicate failure described in Sec. 1: its mean nearest-neighbor distance is about $15\times$ smaller than HullFT's, and its closest selected pair is over $50\times$ closer. Since the objective has no diversity term, this is an empirical property rather than a guarantee.

Table 6: Redundancy of the distinct selected support, mean $\pm$ std across 12 Pile subsets (150 queries each), on unit-normalized RoBERTa embeddings. $|S|$: number of distinct points. Pair / NN / min NN: mean pairwise, mean nearest-neighbor, and smallest nearest-neighbor cosine distance ($\times 10^{-3}$, higher is less redundant). Radius: mean distance to the support centroid. $\cos(\cdot, q)$: mean similarity to the query.

Method	$\lvert S\rvert$	**Pair**	**NN**	**Min NN**	**Radius**	$\cos(\cdot, q)$
$N = 20$						
kNN	20.0 ± 0.0	25.7 ± 9.1	1.9 ± 2.7	0.4 ± 0.8	0.146	0.976
SIFT	12.0 ± 2.6	39.3 ± 13.7	27.0 ± 9.7	19.6 ± 7.5	0.178	0.974
HullFT (ours)	11.3 ± 2.4	$\mathbf{39.9} \pm 13.9$	$\mathbf{27.8} \pm 10.0$	$\mathbf{20.8} \pm 7.9$	**0.179**	0.974
$N = 50$						
kNN	50.0 ± 0.0	29.4 ± 10.2	1.8 ± 2.7	0.2 ± 0.4	0.160	0.973
SIFT	15.9 ± 3.4	39.0 ± 13.4	25.8 ± 9.1	17.6 ± 6.8	0.180	0.973
HullFT (ours)	14.5 ± 3.2	$\mathbf{39.8} \pm 13.8$	$\mathbf{26.9} \pm 9.7$	$\mathbf{19.3} \pm 7.4$	**0.181**	0.973

D.10 Qualitative examples of selected contexts

For two representative test queries in each of ArXiv, GitHub, and PubMed Central, we show the test query alongside the two candidates with the largest weight in the HullFT support at N=20. All text below is verbatim from the Pile candidate pool; we truncate each excerpt to roughly 400 characters for readability.

ArXiv.

prompt	The operational meaning of quantum conditional information. Information might be regarded as an answer to a question. To know if it is raining, one need only look outside. However, a person living in a desert would expect a different answer than one living in a climate where on average, it rains every other day. After looking outside, who gains "more" information? The answer to this question has nothing to do with weather . . .
Example 1	Every forecasting system is noncalibrated for uncountably many data sequences that it might see. This result is strengthened here: from a topological point of view, failure of calibration is typical and calibration rare. Meanwhile, Bayesian forecasters are certain that they are calibrated—this invites worries about the connection between Bayesianism and rationality. . . .
Example 2	Landauer's erasure principle exposes an intrinsic relation between thermodynamics and information theory: the erasure of information stored in a system, S, requires an amount of work proportional to the entropy of that system. This entropy, $H(S\|O)$, depends on the information that a given observer, O, has about S . . .
prompt	Nanoscale particles embedded in a metallic matrix are of considerable interest as a route towards identifying and tailoring material properties. We present a detailed investigation of the electronic structure, and in particular the Fermi surface, of a nanoscale phase ($L1_2$ Al_3Li) that has so far been inaccessible with conventional techniques . . .
Example 1	Point defect species and concentrations in metastable Fe–C alloys are determined using density functional theory and a constrained free-energy functional. Carbon interstitials dominate unless iron vacancies are in significant excess, whereas excess carbon causes greatly enhances vacancy concentration . . .
Example 2	The Fermi surface topology of the shape-memory alloy $Ni_{0.62}Al_{0.38}$ has been determined using Compton scattering. A large area of this Fermi surface can be made to nest with other areas by translation through a vector of $\approx 0.18\,[1, 1, 0](2\pi/a)$, which corresponds to the wavevector associated with martensitic precursor phenomena . . .

GitHub.

prompt	`// Debugger.Object referents can be transparent wrappers of objects in the debugger compartment. var g = newGlobal(); g.f = function (a, b) { return a + "/" + b; }; var dbg = Debugger(g); var hits = 0; dbg.onDebuggerStatement = function (frame) { var f = frame.eval("f").return; assertEq(f.call(null, "a", "b").return, "a/b"); hits++; }; g.eval("debugger;"); assertEq(hits, 1);`
Example 1	`// setVariable can set variables and arguments in functions. var g = newGlobal(); var dbg = new Debugger(g); dbg.onDebuggerStatement = function (frame) { frame.environment.setVariable("a", 100); frame.environment.setVariable("b", 200); }; g.eval("function f(a) { var b = a + 1; debugger; return a + b; }"); assertEq(g.f(1), 300);`
Example 2	`// Referents of Debugger.Objects in other compartments always survive per-compartment GC. var g = newGlobal(); var dbg = Debugger(g); var arr = []; dbg.onDebuggerStatement = function (frame) { arr.push(frame.eval("[]").return); }; g.eval("for (var i = 0; i < 10; i++) debugger;"); assertEq(arr.length, 10); gc(g);`
prompt	`class Foo { // function fBar (x,y); fOne(argA, argB, argC, argD, argE, argF, argG, argH) { Array<string> numbers = ['one','two','three','four','five','six']; var x = ("" + argA) + argB + argC + argD + argE + argF + argG + argH; try { this.fTwo(argA, argB, argC, this.fThree("", argE, argF, argG, argH)); } catch (e,s) {} ...`
Example 1	`class Foo { // function fBar (x,y); fOne(argA, argB, argC, argD, argE, argF, argG, argH) { Array<string> numbers = ['one','two','three','four','five','six']; var x = ("" + argA) + argB + argC + argD + argE + argF + argG + argH; try { this.fTwo(argA, argB, argC, this.fThree("", argE, argF, argG, argH)); } catch (error, stack) {} ...`
Example 2	`import com.Foo; using com.utils.MathUtil; class Foo { // function fBar (x,y); function fOne(argA:Int, argB, argC, argD, argE, argF, argG, argH) { var numbers:Array<String> = ['one','two','three','four','five','six']; var x = ("" + argA) + argB + argC + argD + argE + argF + argG + argH; ...`

PubMed Central.

prompt	Childhood psychiatric disorders are of concern because of the associated negative impact on general wellbeing across the lifespan, for example, lower educational achievements. The findings in several studies suggest that the adverse impact of childhood onset psychiatric disorders on educational attainment is largely accounted for by problems of inattention and conduct . . .
Example 1	The costs of primary and lower secondary schooling in Denmark are among the highest in OECD. This is in part due to large amounts being spent on children in special needs education; about one third of all the costs of primary and lower secondary education . . .
Example 2	Juvenile idiopathic arthritis (JIA) is an umbrella term for chronic childhood arthritis and a significant contributor to chronic disease in children and adolescents. Even in the era of modern treatment with improved outcome on biologic therapies, many children with JIA still experience flares and difficulties to attend daily life activities . . .
prompt	Dengue fever (DF) is an insect-borne disease caused by four different dengue viruses (DENV 1–4), which are mainly transmitted by *Ae. aegypti* and *Ae. Albopictus*. DF is endemic to more than 100 countries in tropical and subtropical areas, especially in Southeast Asia, the Americas, the Western Pacific, Africa . . .
Example 1	Dengue is a vector-borne tropical disease caused by the four dengue virus serotypes (DENV 1–4) that is mainly transmitted by *Aedes aegypti* and *Aedes albopictus* mosquitoes. Dengue is an endemic infectious disease in most of the tropical and subtropical regions, especially in Southeast Asia, the Western Pacific, Latin America . . .
Example 2	Dengue fever, one of the most prevalent mosquito-borne diseases in humans, is mainly transmitted by *Aedes aegypti* and *Aedes albopictus*. There are four distinct serotypes for dengue virus, namely DENV 1, 2, 3 and 4. Dengue fever is endemic in more than 100 countries of Southeast Asia . . .

E Additional experiments

E.1 Does the training order matter?

HullFT trains its multiset in a fixed order (decreasing convex weight, each point's copies in one block), kNN and SIFT use the order their selectors return. We finetune each method on a random per-query permutation of its $N{=}50$ multiset instead, with Gradient Reuse off, over three seeds (Tab. 7). Shuffling leaves SIFT unchanged ($p \geq 0.44$) and does not significantly move kNN ($p \geq 0.11$). For HullFT it lowers BPB% by 0.63 (same sign on 11 of 12 subsets, $p < 0.01$ in every seed), so the gains come from the multiset's content, not its order.

Table 7: Native vs. shuffled finetuning order at $N{=}50$ on GPT-2, averaged over 12 Pile subsets and three seeds, Gradient Reuse off. Δ = shuffled $-$ native (negative means shuffling lowers BPB%), mean $\pm$ std over seeds. Native values come from this experiment's own runs and differ slightly from Tabs. 2 and 4.

Method	**Native**	**Shuffled**	Δ
kNN	72.07	71.78	-0.29 ± 0.04
SIFT	65.90	65.90	-0.01 ± 0.03
HullFT (ours)	66.77	66.15	-0.63 ± 0.02

E.2 Matched-N comparison

A fixed runtime budget lets a cheaper selector afford more gradient steps, so the gains in Sec. 4 combine selection quality with step count. Table 8 separates the two: every method uses the same N and the same finetuning (one gradient step per selected sequence, no Gradient Reuse), so the selected multiset is the only difference. HullFT is ahead of kNN at every N, while SIFT is ahead of HullFT by 0.02 at $N{=}10$, growing to 1.17 at $N{=}50$. We therefore do not claim better selection than SIFT at matched N. What HullFT offers is close-to-SIFT selection quality at a fraction of the cost, from

two independent sources: the selector itself (Tab. 8, right) and Gradient Reuse on finetuning (Tab. 4). Within a fixed budget, these savings buy a larger N, which is where the gains in Fig. 2 come from.

Table 8: Matched-N comparison on GPT-2, averaged over 12 Pile subsets (150 queries each). All methods use identical finetuning (one gradient step per selected sequence, no Gradient Reuse), so only the selected multiset differs. Δ = best baseline $-$ HullFT (negative means the best baseline, SIFT, is ahead). Right: per-query selection time.

	BPB% ↓				**Selection time (s) ↓**		
N	kNN	SIFT	HullFT	Δ	kNN	SIFT	HullFT
10	81.24	**79.26**	79.28	−0.02	0.001	0.458	0.007
20	76.56	**71.97**	72.51	−0.54	0.001	0.511	0.036
30	74.25	**68.37**	69.34	−0.96	0.001	0.480	0.057
40	72.92	**66.64**	67.74	−1.10	0.001	0.495	0.059
50	72.11	**65.89**	67.06	−1.17	0.001	0.524	0.059