ONLINE LABEL-FREE GRADIENT MATCHING FOR CONTINUAL LANGUAGE MODEL TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models in customer-facing products must be continually updated, since the data they serve shifts over time. As retraining on the full accumulated data is impractical, each update should instead run on a small set of examples carrying the training signal of all data seen so far. The strongest current methods build this set by gradient matching, which synthesizes examples whose training gradients match those of the full data. However, they require ground-truth labels, run once on a fixed snapshot rather than on a growing stream, and are slow on large data. We introduce **OLGA**, which lifts these limitations. It distills unlabeled data by drawing the gradient matching targets from the model's own next-token predictions. Across five text classification benchmarks with phi-1.5, it improves accuracy by 10 percentage points over the strongest selection baseline using as few as five examples. OLGA also distills online data streams by keeping a running rehearsal memory of synthetic examples, which is re-distilled together with each new batch as it arrives. In the online setting, it outperforms the strongest selection-based continual baseline by 6.6 stream-AUC percentage points. Finally, our method scales to large data by pairing distillation with selection in a hybrid pipeline, letting selection handle the mild stage of compression and distillation the extreme one. It cuts runtime by 30% versus distillation alone at similar accuracy, and beats the best selection-only pipeline by up to 12.6 percentage points.

1 INTRODUCTION

Language models are increasingly deployed in customer-facing products, where the traffic they serve keeps changing. User preferences move, new topics appear, and safety requirements shift, so a model that was accurate at release gradually falls out of step with the data it now sees (Lazaridou et al., 2021). Keeping it current means updating it repeatedly rather than once, and the data available at each update grows with every cycle. Retraining on every example accumulated so far is the obvious remedy and also the least affordable one, because a full pass costs more as the data grows while the oldest examples contribute least. What a practical update loop needs instead is a small set of examples, rebuilt at each cycle, that still carries the training signal of everything seen so far.

Most existing methods build that set by selecting a well-chosen subset of the real examples. This works well when the budget is generous, but once only a few examples can be kept, it becomes hard to preserve a strong training signal. Dataset distillation with gradient matching addresses this by synthesizing examples whose training gradient matches the gradient of the real data (Zhao et al., 2021). Because a synthetic example does not have to be a real sentence, it can capture what many real examples teach the model, so the set can be kept to a few examples.

Three obstacles stand in the way of using gradient matching to keep a language model current. First, it needs ground-truth labels, which rules out the unlabeled data that makes up most of what a deployed system collects. For images, recent work replaces the labels with targets from a frozen model trained without them (Lee et al., 2024; Joshi et al., 2025), but text distillation methods still rely on labels (Ogawa et al., 2025). Second, it runs once over a fixed snapshot, so it never has to fold an arriving batch into an existing set. Methods that do this exist for images (Gu et al., 2024), but they need labels and nudge pixel values in small steps, which does not work for the discrete tokens of text. Finally, the cost of matching gradients grows with the amount of data it compresses, making it least affordable exactly when data is plentiful. For images, discarding part of the real data before

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.

distilling cuts this cost (Xu et al., 2024; Moser et al., 2025), but these methods rank examples by their loss against a label and have only been studied on a fixed snapshot. As a result, no existing method compresses an unlabeled text stream into a small training set at a cost that stays affordable as the stream grows.

We remove all three obstacles at once. To drop the labels, we keep them out of the objective and take the matching target from the model's own predictions instead. To move from a snapshot to a stream, we keep a running rehearsal memory of synthetic examples and merge each arriving batch into it with the same gradient matching objective. To control the cost, we introduce a hybrid pipeline in which cheap unsupervised selection runs ahead of expensive distillation, since the two are strong at different amounts of compression.

We make the following contributions.

- We introduce a label-free gradient matching objective whose compact distilled set trains a model 10 points more accurate than the best real subset of the same size.
- We extend text distillation to streaming data, updating a rehearsal memory as each batch arrives, and beat the best real memory of the same size by 6.6 stream-AUC points.
- We design a hybrid pipeline in which cheap selection shrinks each batch before distillation, cutting distillation runtime by 30% compared with distilling the full batch.

2 Background and Related Work

For much of the past decade, progress in deep learning came from scale, with larger models trained on more data consistently outperforming smaller ones. As models have grown to billions of parameters and corpora to trillions of tokens, however, the bottleneck has shifted from data availability to compute and time. Deployed models must also be updated ever more often as user preferences, knowledge, and safety requirements change, and retraining on the full corpus at every update is prohibitively expensive. The question is therefore no longer how to collect more data, but how to select or construct the smallest set of examples that preserves the training signal of the full corpus.

Coreset selection. Coreset selection keeps the most representative slice of the real training set without modifying any examples. Geometry-based algorithms formalize this as a coverage problem, where k-center greedy (Sener & Savarese, 2018) picks the point farthest from the selected set, while herding (Rebuffi et al., 2017) and prototypical selection (Snell et al., 2017) sequentially choose samples that best approximate the per-class feature centroid. Training-oriented variants use model-dependent signals such as gradient norms (GraNd) or L2 prediction error (EL2N) (Paul et al., 2021) to prioritize samples near the decision boundary rather than near the class mean. Despite this diversity, all coreset methods share a hard representational ceiling, since a text sequence is an atomic unit that cannot be perturbed below the token level and no selection rule can manufacture variance that does not already exist in the chosen sample. This ceiling becomes binding at extreme compression, the regime where synthetic examples become necessary.

Dataset distillation. Dataset distillation compresses a large training set into a handful of synthetic examples, optimized so that a model trained on them behaves like one trained on the original data. Wang et al. (2018) first framed this as bi-level meta-learning, updating the synthetic examples through an unrolled inner training loop, but the memory cost and numerical instability of unrolling restrict it to shallow networks trained for only a few steps. Later work replaces the unroll with cheaper surrogate objectives. *Trajectory matching* (Cazenavette et al., 2022) aligns the parameter path traced by training on synthetic data with paths recorded on real data, at the cost of precomputing and caching thousands of such expert trajectories. *Distribution matching* (Zhao & Bilen, 2023) instead matches per-class feature embeddings of real and synthetic examples, which is cheap but ignores the task loss and can leave class boundaries misaligned. *Gradient matching* (Zhao et al., 2021) targets the task loss directly by finding synthetic examples whose per-step parameter gradient matches the real gradient in cosine distance. It needs neither long unrolls nor stored trajectories and scales to larger architectures more readily, making it the dominant paradigm. All three directions, however, assume that gradients flow through a continuous input space, which holds for images but breaks down for discrete text tokens, where a gradient step on the input has no direct meaning.

Dataset distillation for language models. Early work on distilling text avoided discreteness by optimizing continuous word embeddings instead of tokens (Maekawa et al., 2023), but the results are model-specific, meaning a synthetic embedding created with BERT cannot be used to fine-tune RoBERTa, and projecting the optimized embeddings back to the nearest vocabulary token yields incoherent sequences that carry little training signal. Maekawa et al. (2024) addressed this with DiLM, which trains a proxy generative model to synthesize readable text, routing gradients through the discrete bottleneck via generation probabilities and producing sequences that transfer across architectures. Training a full proxy language model as a distillation scaffold is itself expensive, however, and the proxy may memorize sensitive training-set content. GradMM (Nguyen et al., 2025) avoids the proxy by applying the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011), in which gradient matching proceeds in continuous embedding space while a periodic projection step snaps each embedding back to its nearest vocabulary token, enforcing hard discreteness throughout optimization. This produces human-readable, cross-architecture synthetic text, and it is the offline supervised technique we build on here. Like every text distillation method so far, however, it places ground-truth labels in the matching objective, which ties the distilled data to a single task and excludes the unlabeled data that makes up most of what language models are trained on.

Continual learning and catastrophic forgetting. In continual learning, a model must keep absorbing a non-stationary stream of data without losing what it has already learned. The core difficulty is *catastrophic interference*, in which training on new material abruptly overwrites earlier knowledge (McCloskey & Cohen, 1989; French, 1999), and it arises whether the stream introduces new tasks, new domains, or new classes (van de Ven & Tolias, 2019). Architecture-based methods counter it by dedicating parameters to each task (Rusu et al., 2016; Mallya & Lazebnik, 2018), but presuppose known task boundaries and a growing parameter budget. Regularization-based methods penalize changes to weights that matter for past tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Li & Hoiem, 2018), but weaken over long streams as their accumulated constraints stiffen the network. The dominant remedy is *rehearsal*, which replays a small memory of past examples alongside incoming data, with iCaRL (Rebuffi et al., 2017) as its canonical instance and a strong baseline for nearly a decade. Because the memory *is* the method, much work concerns how to curate it, from fixed rules such as reservoir sampling, herding, or class-balanced storage (Prabhu et al., 2020) to gradient-based rules that keep the examples whose gradients best span (Aljundi et al., 2019) or best approximate (Tiwari et al., 2022) the gradient of all data seen so far. Every such scheme still chooses among real examples, so the memory inherits the representational ceiling of coreset selection. That ceiling binds hardest where rehearsal matters most: as the stream grows and the per-class budget shrinks, a few stored examples can no longer carry the training signal of the task they stand for.

Continual distillation as rehearsal memory. This ceiling motivates compressing the memory rather than selecting it, using dataset distillation to synthesize a handful of maximally informative rehearsal examples. Distilled Replay keeps as little as one synthetic pattern per class (Rosasco et al., 2022), Zhao et al. (2021) showed that gradient-matched synthetic memories alleviate forgetting in a task-incremental image setting, and later methods refine the buffer by carrying cross-task information (Liao et al., 2025), mixing distilled with real exemplars (Kong et al., 2024), or condensing prototypical exemplars (Nguyen et al., 2026). Two limitations run through this line of work. First, these methods optimize continuous pixel tensors with objectives that do not carry over to discrete tokens. Continual learning for text has therefore sidestepped distillation in favor of generative rehearsal, sampling pseudo-examples from an autoencoder (Zeng et al., 2024) or prompting the model to synthesize its own rehearsal set (Huang et al., 2024), which re-incurs the expense and memorization risk of a separate generator. Second, nearly all of them are offline, distilling each task once from a complete snapshot. The one exception, SSD (Gu et al., 2024), folds each arriving batch into a synthetic image memory by matching gradients against the labeled batch. How to fold an arriving batch into a synthetic memory of discrete tokens, with or without labels, has not been studied.

3 METHODOLOGY

To address these obstacles, we introduce Online Label-free Gradient Matching (**OLGA**), in which each component removes one obstacle (Figure 1). We begin in Section 3.1 with the offline supervised method that OLGA builds on. Section 3.2 then drops the need for labels, Section 3.3 extends the method to streaming data, and Section 3.4 speeds it up with a hybrid design.

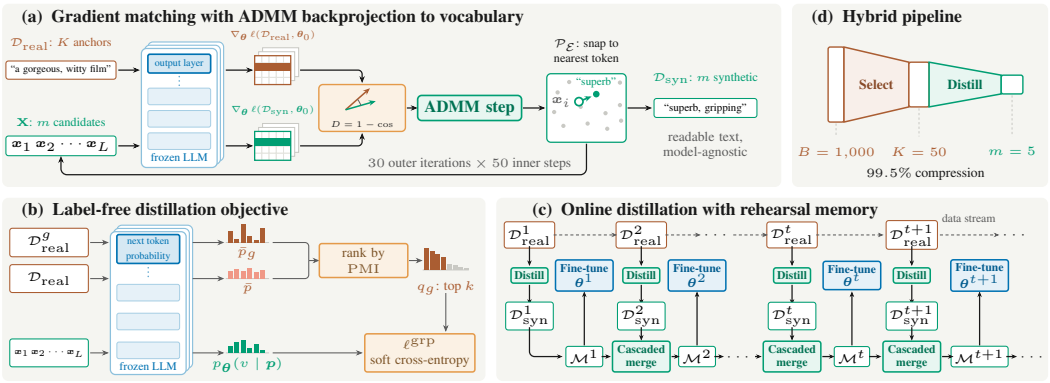

Figure 1: Overview of the components of OLGA. **(a)** Synthetic text is generated by matching its training gradient to that of real data, alternating with a step that maps it back to real tokens. **(b)** A frozen language model's next-token predictions replace ground-truth labels, so gradient matching works on unlabeled data. **(c)** A running rehearsal memory extends offline distillation to an online data stream. Each new batch is distilled together with the memory, which the model then retrains on. **(d)** Our hybrid design first uses cheap selection to shrink each batch, then distills what remains into the few examples kept, which cuts runtime on large data.

3.1 Gradient matching with the supervised objective

The method we start from is GradMM (Nguyen et al., 2025), which produces readable and architecture-agnostic examples by solving a discrete gradient matching optimization. Let $\boldsymbol{\theta}_0$ denote the parameters of a frozen pretrained language model and $\mathcal{D}$ a set of examples $s = (\boldsymbol{p}, \boldsymbol{r})$, each pairing a prompt with a response. The loss of the model on $\mathcal{D}$, written $\ell(\mathcal{D}, \boldsymbol{\theta})$, is the average of the individual example losses $\ell(s, \boldsymbol{\theta})$. The real examples that serve as a matching target are called *anchors* and written $\mathcal{D}_{\text{real}}$, and the synthetic set produced from them is $\mathcal{D}_{\text{syn}}$, held to a budget of m examples. GradMM searches for a synthetic set whose fine-tuning gradient matches the anchor gradient at the pretrained parameters,

$$\mathcal{D}^{\star}_{\text{syn}} = \underset{|\mathcal{D}_{\text{syn}}| \le m,\ \mathcal{D}_{\text{syn}} \subset \Gamma}{\arg\min} \; D\big(\nabla_{\boldsymbol{\theta}}\, \ell(\mathcal{D}_{\text{syn}}, \boldsymbol{\theta}_0),\ \nabla_{\boldsymbol{\theta}}\, \ell(\mathcal{D}_{\text{real}}, \boldsymbol{\theta}_0)\big)\,, \tag{1}$$

where $D(\boldsymbol{u}, \boldsymbol{v}) = 1 - \cos(\boldsymbol{u}, \boldsymbol{v})$ compares gradient vectors by direction rather than scale, and Γ is the set of readable token sequences. GradMM solves this with ADMM (Boyd et al., 2011), which alternates a few steps of first-order optimization in continuous embedding space with a projection back onto real vocabulary tokens (Figure 1(a)). We keep this solver unchanged and focus instead on making the optimization work without labels, on a data stream, and at lower runtime.

In the supervised setup, the model is asked to emit a verbalizer token v_y, a single word standing in for the class of the example, at a fixed position after the prompt, and the loss is the cross-entropy of that single token, $\ell^{\text{sup}}(s, \boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(v_y \mid \boldsymbol{p})$. Since every example of a class is trained toward the same token, their gradients align and add up rather than cancel. The anchor gradient thus lies in a few class directions, which a handful of synthetic examples can reproduce closely.

3.2 Gradient matching with a label-free objective

While the supervised loss $\ell^{\text{sup}}(s, \boldsymbol{\theta})$ needs each example's class to know which verbalizer token the model should emit, customer-facing platforms mostly receive unannotated examples for which a single class label is hard to pin down (Zhao et al., 2024). Such data is rarely unstructured, though, and can be partitioned into groups $\mathcal{D}^{g}_{\text{real}}$ by pre-logged metadata or through unsupervised clustering (Wan et al., 2024). We use this grouping, together with the model's internal next-token probabilities, to design a label-free loss $\ell^{\text{grp}}(s, \boldsymbol{\theta})$ that preserves inter-group separation (Figure 1(b)).

Let $p_{\boldsymbol{\theta}_0}(v \mid \boldsymbol{p})$ denote the probability the frozen model puts on a next token v after a prompt $\boldsymbol{p}$, and let $\bar{p}_g(v)$ and $\bar{p}(v)$ be its averages over one group and over all anchors. The pointwise mutual information between a token and a group,

$$\text{PMI}(v, g) = \log \bar{p}_g(v) - \log \bar{p}(v) = \log \frac{\frac{1}{|\mathcal{D}^{g}_{\text{real}}|} \sum_{s \in \mathcal{D}^{g}_{\text{real}}} p_{\boldsymbol{\theta}_0}(v \mid \boldsymbol{p})}{\frac{1}{|\mathcal{D}_{\text{real}}|} \sum_{s \in \mathcal{D}_{\text{real}}} p_{\boldsymbol{\theta}_0}(v \mid \boldsymbol{p})}, \tag{2}$$

measures how much more the model expects v inside one group than across the data as a whole. Ranking the vocabulary by equation 2 and keeping the k highest-scoring tokens gives a set $\mathcal{V}_g^k$ of the tokens that best separate group g from the rest, and k is the only hyperparameter the construction adds. Normalizing the group probabilities across $\mathcal{V}_g^k$ turns them into a target q_g, and for $s \in \mathcal{D}_{\text{real}}^g$ the group-distribution loss is the soft cross-entropy toward it,

$$\ell^{\text{grp}}(s, \boldsymbol{\theta}) = -\sum_{v \in \mathcal{V}_g^k} q_g(v)\, \log p_{\boldsymbol{\theta}}(v \mid \boldsymbol{p}), \qquad q_g(v) = \frac{\bar{p}_g(v)}{\sum_{u \in \mathcal{V}_g^k} \bar{p}_g(u)}. \tag{3}$$

Substituting ℓ^{grp} into equation 1 replaces the class label with a soft group distribution read off the model's own knowledge, while preserving the separation between groups. Appendix A9 gives the conditions under which this proxy induces the same class discrimination as the true label loss, and Section 4.5 compares it against several alternatives.

3.3 Online gradient matching with rehearsal memory

The objective of equation 1 suits an offline setup, where a single fixed dataset is available all at once. A deployed system rarely works this way, since its data arrives as a sequence of blocks $\mathcal{D}_{\text{real}}^1, \ldots, \mathcal{D}_{\text{real}}^T$, with $\mathcal{D}_{\text{real}}^{1:t}$ the data seen through step t (Li et al., 2025). In this online setting, we want a compact synthetic set of at most m examples that, at each step, best represents everything seen so far, so that fine-tuning $\boldsymbol{\theta}_0$ on it tracks the model trained on all data through step t. We write this set as ${}^\star\mathcal{D}_{\text{syn}}^{1:t}$, the one whose gradient at $\boldsymbol{\theta}_0$ matches that of the full history:

$${}^\star\mathcal{D}_{\text{syn}}^{1:t} = \underset{|\mathcal{D}_{\text{syn}}^{1:t}| \le m,\ \mathcal{D}_{\text{syn}}^{1:t} \subset \Gamma}{\arg\min} D\big(\nabla_{\boldsymbol{\theta}}\, \ell(\mathcal{D}_{\text{syn}}^{1:t}, \boldsymbol{\theta}_0),\ \nabla_{\boldsymbol{\theta}}\, \ell(\mathcal{D}_{\text{real}}^{1:t}, \boldsymbol{\theta}_0)\big)\,. \tag{4}$$

Computing ${}^\star\mathcal{D}_{\text{syn}}^{1:t}$ directly is impractical in a stream, since it requires keeping the entire history $\mathcal{D}_{\text{real}}^{1:t}$ and re-distilling it from scratch at every step, so both storage and per-step cost grow without bound as the stream lengthens. We approximate it by solving a separate distillation problem for each incoming block $\mathcal{D}_{\text{real}}^t$ and merging the results into a running rehearsal memory through the same gradient-matching objective. Each block is first compressed on its own into a synthetic set $\mathcal{D}_{\text{syn}}^t$ by the offline problem of equation 1, and the memory $\mathcal{M}^t$ carries these forward. It starts from the first distilled block and, at every later step, is rebuilt by distilling the union of the new set and the previous memory back down to m,

$$\mathcal{M}^t = \underset{|\cdot| \le m,\ \cdot \subset \Gamma}{\arg\min} D\big(\nabla_{\boldsymbol{\theta}}\, \ell(\cdot, \boldsymbol{\theta}_0),\ \nabla_{\boldsymbol{\theta}}\, \ell(\mathcal{D}_{\text{syn}}^t \cup \mathcal{M}^{t-1}, \boldsymbol{\theta}_0)\big)\,, \qquad \mathcal{M}^1 = \mathcal{D}_{\text{syn}}^1. \tag{5}$$

We call this a *cascaded merge* (Figure 1(c)). At every step we fine-tune on $\mathcal{M}^t$, which stands in for the ${}^\star\mathcal{D}_{\text{syn}}^{1:t}$ we cannot compute. Each step distills only the new block and the current memory instead of the whole history, so its cost is the same at every step and the total grows linearly with the stream length T. Since the merge is independent of the loss, it applies to both the supervised and the label-free objectives. We compare the cascaded merge against alternative merge strategies in Section 4.5, and in Appendix A10 we bound it against the intractable ideal of equation 4, showing its error stays bounded across the stream rather than growing with T.

3.4 Hybrid hierarchical compression

Selection and distillation work best in different regimes. When the budget is a large fraction of the data, cheap coreset selection is enough, since a well-chosen real subset still covers most of the training signal. As the budget shrinks toward a handful of examples, no real subset can carry that signal, whereas a single synthetic example can combine signal drawn from many real ones. Viewing compression as an information bottleneck (Tishby et al., 2000; Zhou et al., 2025), we expect selection to suffice while it is wide and distillation to be needed once it narrows.

We therefore compress in two stages (Figure 1(d)). At every compression step, offline on a static pool or at each step of the stream, a coreset selector first cuts the incoming batch of B examples down to a smaller pool of K, and gradient matching then distills that pool into the m examples actually kept, so $B \gg K \gg m$. We call this the hybrid pipeline and compare it against selection-only and distillation-only alternatives in Section 4.4. In Appendix A11 we further show that routing through selection leaves the pipeline a controlled approximation to gradient matching on the full batch, with a distillation cost independent of B.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Data. We use five text classification datasets: SST-2 (Socher et al., 2013), Rotten Tomatoes (Pang & Lee, 2005), RT-Polarity (Pang & Lee, 2005), IMDB (Maas et al., 2011), and TwitterEmotion (Saravia et al., 2018). They span two domains, movie reviews and tweets, inputs from a single sentence to a full review, and training pools from 1,000 examples to tens of thousands. Every experiment uses all five, with the exception of the streaming hybrid, where each of the ten steps draws 1,000 examples and only SST-2 and IMDB hold enough data for that.

Model. Following GradMM (Nguyen et al., 2025), we use phi-1.5 (Li et al., 2023) for our experiments. During distillation, we only match the gradients of the output layer to avoid recomputing the gradients of all 1.3B parameters at each step, and we run the ADMM solver for 30 outer iterations of 50 inner steps at a step size of 0.008. During fine-tuning, we train 200 steps on a linear schedule, evaluating every 50 steps and keeping the best checkpoint, with the learning rate picked per dataset from $\{7\times10^{-6}, 1\times10^{-5}, 1.5\times10^{-5}\}$. While we use phi-1.5 for both distillation and fine-tuning, two different models can be used in general, without any change to the method, because what distillation returns is plain text rather than weights.

Compared methods and metrics. In the offline setup, our main metric is accuracy after fine-tuning, and we compare against five competing methods: k-center greedy (Sener & Savarese, 2018) takes the example farthest from those already picked; herding (Rebuffi et al., 2017) takes the ones that keep the running mean closest to the average; prototypical (Snell et al., 2017) takes the ones closest to that average; uncertainty (Lewis & Gale, 1994) takes the ones the model is least sure about; and random picks uniformly. In the online setup, we focus on the area under the accuracy-versus-step curve (AUC), which rewards being right at every step rather than only the last, and we compare against four competing methods: k-center greedy, herding, and prototypical from above, and reservoir sampling (Vitter, 1985), which keeps a uniform sample of all data seen. In both settings we also report three reference points. An oracle fine-tunes on all available real data, and supervised GradMM distills with the ground-truth labels that our label-free method never sees. Together they give upper bounds, while the zero-shot model with no fine-tuning gives the lower bound.

4.2 OFFLINE LABEL-FREE GRADIENT MATCHING BEATS COMPETING METHODS

We benchmark OLGA, setting $k{=}100$ for its label-free group distribution objective, against the supervised objective (GradMM) and the five competing methods in the offline setup, where the full data pool is available at once. Each task starts from 50 real anchors, which we distill into a compact synthetic set of 5 examples and fine-tune phi-1.5 on, evaluating on the held-out validation split of each dataset. This is a 90% reduction, which creates the extreme-compression regime we target.

Figure 2(a) reports classification accuracy on each dataset. OLGA is the strongest label-free method, beating the strongest competing method on four of the five datasets and by 10.2 points on average. The gap is widest on IMDB, where OLGA scores 89.5 against 72.1 for the strongest competitor, and narrowest on Rotten Tomatoes, at 4.1 points (74.5 against 70.4). The one exception is TwitterEmotion, which is hard for every method, and where OLGA reaches just 56.5, on par with the competing methods. Overall, even without labels, gradient matching keeps enough signal to compress a dataset into a handful of synthetic examples that train a model better than any subset of real examples can. Table 1 shows two distilled examples next to the real examples they borrow their phrasing from, and Appendix A4 one for each of the other datasets. Appendix A1 lists exact values for Figure 2(a).

4.3 ONLINE GRADIENT MATCHING WITH MEMORY BEATS CONTINUAL SELECTION

We benchmark OLGA, setting $m{=}5$ for the rehearsal memory and cascaded merge budgets, across all five datasets in an online setup of $T{=}10$ steps (IMDB for 8 steps due to its size). At each step, 50 anchors arrive and are distilled into 5 synthetic examples, which are then folded into the memory by the cascaded merge. At each step, phi-1.5 is fine-tuned on the resulting memory and evaluated on the held-out validation set.

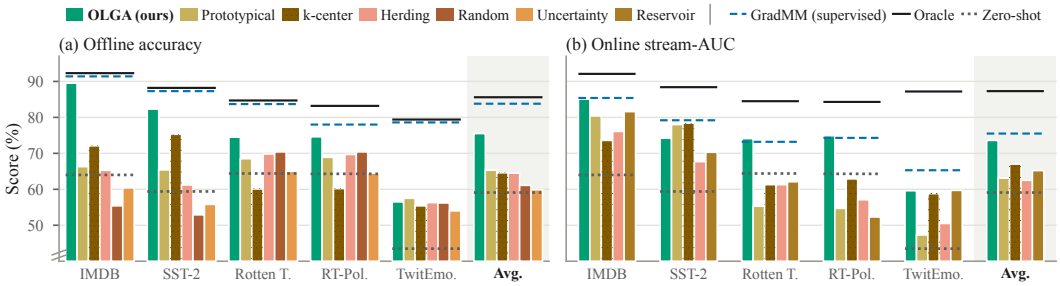

Figure 2: Across five datasets, OLGA beats every competing method on average in both the offline and online setups, and comes close to the supervised upper bound in the online one. **(a)** Offline methods reduce 50 real examples to 5, and we report the held-out accuracy of phi-1.5 fine-tuned on them. **(b)** Online data arrives over a sequence of steps, each method keeps a memory of only 5 examples, and we report stream-AUC, the area under the accuracy-versus-step curve.

Table 1: Two synthetic examples distilled by OLGA, one per dataset. Each is generated as a mix of phrases from real examples, three of which are listed below it. A shared phrase has the same color in both places, and its superscript gives the number of the real example it comes from.

	TwitterEmotion (joy)	**SST-2** (negative)
Synthetic	I am feeling[1] so much[2] better after my recent health surgery yesterday and I can feel[3] more	teacher was giving a long[1] and boring and[2] not interesting and[3] clear lecture to the students because the teacher was reading from a textbook
Real	1. i am feeling amazing mostly normal i am going to a pre thanksgiving celebration with our friends from that time we were in softball 2. i feel like we have so much to be thankful for 3. i can feel superior on that point	1. it ai n't art, by a long shot, 2. a shame that stealing harvard is too busy getting in its own way to be anything but frustrating, boring, and forgettable. 3. a more confused, less interesting and more sloppily made film could possibly come down the road in 2002

Figure 2(b) reports the online stream-AUC of the fine-tuned phi-1.5 on each dataset. Averaged over the five datasets, Online GradMM and OLGA finish 8.5 and 6.6 points above k-center, the strongest competing method overall at 67.0. The two are within two points of each other, so removing the labels costs little. Across datasets the biggest gain for both comes on RT-Polarity, roughly 10 points over the best competing method there. The shortfalls are all on the label-free side, where OLGA only draws level with the best competing method on TwitterEmotion and comes in 4.2 points behind it on SST-2. Appendix A2 lists exact values for Figure 2(b).

4.4 Hybrid compression pipeline

We explore whether a mix of selection and distillation exists that lets cheap selection take the early, easy part of compression and leaves only the extreme end to distillation, and how such a mix compares against alternatives. In our setup, a batch of B=1,000 real examples arrives, selection trims it to K=50, and distillation turns those into the m=5 examples we keep and train on. This achieves a 99.5% compression rate overall. We call this two-stage route **Hybrid**, and compare it against two alternatives. **Direct** drops the middle stage and goes from B straight to m in a single step. **Double** keeps both stages but runs the second one the same way as the first. Across all three routes, the distillation stage runs either GradMM or OLGA, and selection is done by uniform sampling.

Hybrid pipeline in offline setting. Table 2 compares the accuracy of the three routes in the offline setting, where the whole compression happens once over a static pool, and Table 2 compares what each route costs in time. With labels, the hybrid pipeline is the strongest route, averaging 86.0 percent across the five datasets, ahead of GradMM-Direct at 82.7 and GradMM-Double at 73.9, while costing less than distilling the whole batch directly or in two steps. This confirms that selecting early, while the compression is still easy, and distilling only at the extreme end is the more effective way through the bottleneck, and the cheaper one too. Without labels, the same funnel becomes a trade-off rather than a gain: OLGA-Hybrid falls short of OLGA-Direct by only 0.9 points on average while running 30% faster, which puts the two routes on par and makes the choice between

Table 2: Comparison of the hybrid, direct, and double routes in the offline and online settings across 5 datasets. At each time step, the data is compressed in stages from B=1,000 to K=50 to m=5. The offline setting runs a single step (T=1), and the online setting runs T=10 steps. Runtime is the wall-clock time of one offline distillation run on SST-2. On average, the hybrid route beats the direct and double routes for both objectives in both settings, except offline without labels, where OLGA-Hybrid is on par with OLGA-Direct. Bold marks the best result in each column, and underline marks the reference that each colored difference is measured against.

Method	Route	Offline accuracy						Online stream-AUC			Runtime (s) ↓
		SST-2 ↑	Rotten T. ↑	IMDB ↑	RT-Pol. ↑	TwitEmo. ↑	Avg. ↑	SST-2 ↑	IMDB ↑	Avg. ↑	
GradMM	Hybrid	88.2	83.9	**92.3**	82.1	**83.4**	**86.0** (+3.3)	**76.0**	**80.8**	**78.4** (+3.1)	436.1 (−29.9%)
	Direct	87.4	76.6	90.6	81.3	77.6	82.7	72.1	78.5	75.3	622.1
	Double	79.7	71.1	80.7	67.0	70.8	73.9 (−8.8)	—	—	—	2073.6 (+233.3%)
OLGA	Hybrid	87.8	81.2	89.0	**84.6**	57.3	80.0 (−0.9)	71.2	78.4	74.8 (+2.1)	447.5 (−30.1%)
	Direct	**88.6**	**85.0**	90.9	82.6	57.4	80.9	67.7	77.7	72.7	639.9
	Double	67.4	71.2	87.6	76.1	58.9	72.2 (−8.7)	—	—	—	2086.4 (+226.1%)
Selection	Double	82.5	73.6	89.8	52.9	68.0	73.4 (+6.8)	69.6	75.3	72.5 (+1.4)	5.5 (+14.6%)
	Direct	81.9	54.7	87.9	50.3	58.1	66.6	67.2	75.0	71.1	**4.8**

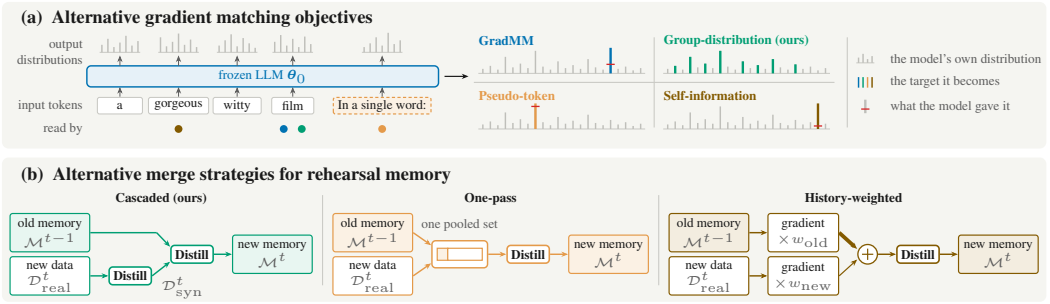

Figure 3: **(a)** How each gradient matching objective builds its loss from the model's next-token distribution; the dots mark which position it reads. **(b)** Three ways to fold a new batch into the memory: compress it first, pool it with the memory, or weight the two by hand.

them a balance of accuracy against time. What holds for both objectives is the lead over selection: GradMM-Hybrid and OLGA-Hybrid beat the selection-only routes on average, 86.0 and 80.0 against 73.4 for Selection-Double and 66.6 for Selection-Direct.

Hybrid pipeline in online setting. We extend the comparison to the online setting, where a new batch of B examples arrives at each of T=10 steps. After each distillation, the resulting m examples are merged into the running rehearsal memory by the cascaded merge. We report only SST-2 and IMDB, since they are the only datasets large enough to supply a stream of this length.

The online half of Table 2 shows that GradMM-Hybrid performs best, with an average stream-AUC of 78.4. GradMM-Hybrid beats GradMM-Direct by 3.1 points and OLGA-Hybrid beats OLGA-Direct by 2.1 points, while distilling only 50 examples at each step instead of 1,000. Every route that uses distillation also beats selection alone on average, and the best of them leads the strongest selection-only route (72.5) by 5.9 points.

4.5 Ablations

Alternative label-free objectives. The group-distribution loss ℓ^{grp} introduced in OLGA is not the only way to turn the model's own probabilities into a label-free signal. We compare it against two alternatives (Figure 3(a)). The *pseudo-token* signal appends the prompt "In a single word:" and takes the most likely next token as its target, letting the model label itself in one word. The *self-information* signal finds the token in each sequence that the frozen model predicted least well and takes it as the target, on the view that this token carries most of what makes the example distinctive. Appendix A5 describes both signals further and gives their exact formulations.

Figure 4 compares the three signals. At m=5, group-distribution averages 75.5 percent across the five datasets, ahead of pseudo-token (74.1) and self-information (72.2), and it is the steadiest as the

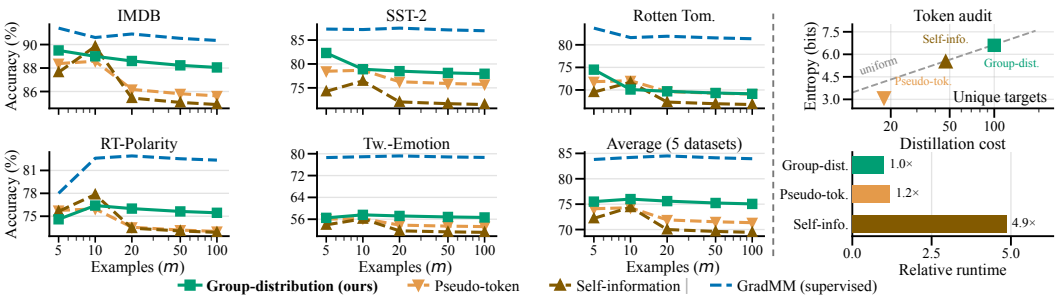

Figure 4: Comparison of alternative label-free objectives. **(Left)** Accuracy against the number of examples m, per dataset and on average. **(Right)** Number of distinct tokens each signal aims at, and runtime relative to group-distribution. Group-distribution leads at m=5, stays steady at larger budgets, and runs fastest.

Table 3: Effect of the selector on the offline hybrid pipeline, averaged over the five datasets.

Selector	Avg. accuracy ↑	SST-2 runtime (s) ↓
Uniform (ours)	**86.0**	**436.1**
k-center	82.1	564.8

Table 4: Comparison of online memory-merging strategies, averaged over the five datasets.

Merge strategy	Avg. stream-AUC ↑
Cascaded (ours)	**72.0**
One-pass	70.2
History-weighted	69.8

budget grows. The tokens each signal aims at help explain why. Pseudo-token aims at only 18 distinct tokens, spread over just 3.1 bits of entropy, so its training signal keeps pushing in the same few directions, while self-information spreads over 47 tokens (5.5 bits) and group-distribution over 100 (6.6 bits). The alternatives are also up to five times slower to run.

Alternative selectors for the hybrid pipeline. The hybrid routes use uniform sampling by default, but any selection method can take its place. We repeat the offline GradMM-Hybrid route with k-center, the strongest selection baseline of Figure 2(b). On average, uniform sampling is both more accurate and faster than k-center (Table 3). This indicates that naive sampling, which keeps the original distribution of the data, is preferable in the hybrid pipeline, since distillation is able to synthesize higher-quality text from it. Appendix A6 reports the per-dataset numbers.

Alternative merge strategies. The cascaded merge is only one way to use distillation to fold an arriving batch into the running memory. We compare it against two alternatives (Figure 3(b)). The *one-pass* merge appends the old memory to the raw new batch and distills the two together in a single step. The *history-weighted* merge does the same single pass, but weights the old and new gradients by how much data each one stands for.

Table 4 extends the experiment of Section 4.3 for OLGA with the alternative merge strategies, at T=5 steps. The cascaded merge beats both on average, at 72.0 against 70.2 for one-pass and 69.8 for history-weighted. Appendix A7 reports the per-dataset performance.

5 Conclusion

Keeping a deployed language model current calls for updates that run on a handful of examples carrying the training signal of all data seen so far. OLGA extends gradient matching for text in three directions: removing the need for labels, following a data stream through a running rehearsal memory, and cutting cost with a hybrid pipeline that selects before it distills. Across five text classification datasets, it outperforms every competing method on average in both the offline and online settings, and our experiments show that each of the three directions matters. Together, they make continual updating of a language model accurate, affordable, and independent of ground-truth labels, offering a practical path toward keeping deployed language model applications current.

AI USE STATEMENT

The research ideas and hypotheses, the method and its formulation in Section 3, the design of the experiments, the implementation of the methods, and the analysis and interpretation of the results are all our own work. Generative AI tools were used only in a supporting role. For the supplementary theoretical results of Appendices A9 to A11, we proposed how each result should be proved, and AI tools helped work out and write the detailed arguments under our direction. These results appear only in the appendix, and the main claims of the paper, which rest on the experiments, do not depend on them. AI coding agents also wrote secondary engineering code, namely experiment scripts, job launchers, and the plotting code behind our figures, and large language models were used to edit the paper for readability. No generative AI tools were used to generate, clean, or reformat datasets, for translation, or for qualitative data analysis. The synthetic examples reported in the paper are the output of our method run with phi-1.5, not text produced by an AI assistant. We have reviewed all AI-assisted work. We checked each proof step by step against the assumptions it relies on, and we reviewed every piece of code the agents produced. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All experiments use publicly available datasets (SST-2, Rotten Tomatoes, RT-Polarity, IMDB, and TwitterEmotion) and the open-source phi-1.5 model. We will release our full implementation publicly on GitHub upon acceptance, and it reproduces all results reported in the paper.

ETHICS STATEMENT

We have read and adhere to the ICLR Code of Ethics. Our experiments use only publicly available text classification datasets and an open-source language model, and involve no human subjects and no data collection of our own. Because the synthetic examples our method produces are built from phrases of the underlying data, they may carry over biases present in that data, which should be considered before deploying the method.

REFERENCES

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

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. 10718–10727, 2022.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Jianyang Gu, Kai Wang, Wei Jiang, and Yang You. Summarizing stream data for memory-constrained online continual learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(11):12217–12225, 2024.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1416–1428, 2024.

Siddharth Joshi, Jiayi Ni, and Baharan Mirzasoleiman. Dataset distillation via knowledge distillation: Towards efficient self-supervised pre-training of deep networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Jiangtao Kong, Jiacheng Shi, Ashley Gao, Shaohan Hu, Tianyi Zhou, and Huajie Shao. Hybrid memory replay: Blending real and distilled data for class incremental learning. *arXiv preprint arXiv:2410.15372*, 2024.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. Mind the gap: Assessing temporal generalization in neural language models. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29348–29363, 2021.

Dong Bok Lee, Seanie Lee, Joonho Ko, Kenji Kawaguchi, Juho Lee, and Sung Ju Hwang. Self-supervised dataset distillation for transfer learning. In *The Twelfth International Conference on Learning Representations*, 2024.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *SIGIR '94*, pp. 3–12. Springer, 1994.

Jeffrey Li, Mohammadreza Armandpour, Seyed Iman Mirzadeh, Sachin Mehta, Vaishaal Shankar, Raviteja Vemulapalli, Samy Bengio, Oncel Tuzel, Mehrdad Farajtabar, Hadi Pouransari, and Fartash Faghri. TiC-LM: A web-scale benchmark for time-continual LLM pretraining. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32231–32273, 2025.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need II: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

Wenyang Liao, Quanziang Wang, Yichen Wu, Renzhen Wang, and Deyu Meng. Data-DistillNet: A data distillation approach tailored for reply-based continual learning. *arXiv preprint arXiv:2505.20135*, 2025.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, 2011.

Aru Maekawa, Naoki Kobayashi, Kotaro Funakoshi, and Manabu Okumura. Dataset distillation with attention labels for fine-tuning BERT. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 119–127, 2023.

Aru Maekawa, Satoshi Kosugi, Kotaro Funakoshi, and Manabu Okumura. DiLM: Distilling dataset into language model for text-level dataset distillation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3138–3153, 2024.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Brian B. Moser, Federico Raue, Tobias C. Nauen, Stanislav Frolov, and Andreas Dengel. Distill the best, ignore the rest: A study in latent dataset distillation on core-sets. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2025.

Dang Nguyen, Zeman Li, Mohammadhossein Bateni, Vahab Mirrokni, Meisam Razaviyayn, and Baharan Mirzasoleiman. Synthetic text generation for training large language models via gradient matching. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 46008–46025. PMLR, 2025.

Minh-Duong Nguyen, Thien-Thanh Dao, Le-Tuan Nguyen, Dung D. Le, and Kok-Seng Wong. Memory-efficient continual learning with prototypical exemplar condensation. *arXiv preprint arXiv:2603.13804*, 2026.

Keith Ando Ogawa, Bruno Lopes Yamamoto, Lucas Lauton de Alcantara, Victor Zacarias, Edson Bollis, Lucas Pellicer, Rosimeire Pereira Costa, Anna Helena Reali Costa, and Artur Jordao. Technical report on text dataset distillation. *arXiv preprint arXiv:2512.03967*, 2025.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 115–124, 2005.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20596–20607, 2021.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A simple approach that questions our progress in continual learning. In *Computer Vision – ECCV 2020*, pp. 524–540. Springer, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Andrea Rosasco, Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Distilled replay: Overcoming forgetting through synthetic samples. In *Continual Semi-Supervised Learning*, pp. 104–117. Springer, 2022.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3687–3697, 2018.

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

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, 2013.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. GCR: Gradient coreset based replay buffer selection for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 99–108, 2022.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W. White, Longqi Yang, Reid Andersen, Georg Buscher, Dhruv Joshi, and Nagu Rangan. TnT-LLM: Text mining at scale with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5836–5847, 2024.

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

Yue Xu, Yong-Lu Li, Kaitong Cui, Ziyu Wang, Cewu Lu, Yu-Wing Tai, and Chi-Keung Tang. Distill gold from massive ores: Bi-level data pruning towards efficient dataset distillation. In *Computer Vision – ECCV 2024*, pp. 240–257. Springer, 2024.

Min Zeng, Haiqin Yang, Wei Xue, Qifeng Liu, and Yike Guo. Dirichlet continual learning: Tackling catastrophic forgetting in NLP. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, volume 244 of *Proceedings of Machine Learning Research*, pp. 4096–4108. PMLR, 2024.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017.

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.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. WildChat: 1M ChatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024.

Zheng Zhou, Wenquan Feng, Qiaosheng Zhang, Shuchang Lyu, Qi Zhao, and Guangliang Cheng. ROME is forged in adversity: Robust distilled datasets via information bottleneck. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78687–78713. PMLR, 2025.

APPENDIX

A1 EXACT VALUES FOR THE OFFLINE RESULTS

Bar charts make the offline comparison easy to scan but hard to read exactly, so here we list the values behind Section 4.2 in full. Each method reduces 50 real examples to 5, phi-1.5 is fine-tuned on the result, and Table 5 reports its accuracy on the held-out validation split of each dataset, together with its gap to the best selection method.

Table 5: Exact values for Figure 2(a), the offline accuracy of phi-1.5 on each dataset. The upper block gives the supervised upper bounds and the lower block the label-free methods. OLGA beats the best selection method on four of the five datasets and by 10.2 points on average. Bold marks the best label-free result in each column, and underline marks the reference that each colored difference is measured against.

Method	IMDB ↑	SST-2 ↑	Rotten Tom. ↑	RT-Polarity ↑	TwitEmo. ↑	Avg. ↑
Oracle	92.3 (+20.2)	88.2 (+12.9)	84.7 (+14.3)	83.2 (+12.8)	79.4 (+21.9)	85.6 (+20.3)
GradMM (Nguyen et al., 2025)	91.4 (+19.3)	87.3 (+12.0)	83.7 (+13.3)	78.0 (+7.6)	78.6 (+21.1)	83.8 (+18.5)
OLGA (ours)	**89.5** (+17.4)	**82.3** (+7.0)	**74.5** (+4.1)	**74.6** (+4.2)	56.5 (−1.0)	**75.5** (+10.2)
Prototypical (Snell et al., 2017)	66.3 (−5.8)	65.4 (−9.9)	68.5 (−1.9)	68.9 (−1.5)	**57.5**	65.3
k-center (Sener & Savarese, 2018)	72.1	75.3	60.1 (−10.3)	60.2 (−10.2)	55.4 (−2.1)	64.6 (−0.7)
Herding (Rebuffi et al., 2017)	65.3 (−6.8)	61.2 (−14.1)	69.8 (−0.6)	69.7 (−0.7)	56.3 (−1.2)	64.5 (−0.8)
Random	55.4 (−16.7)	52.9 (−22.4)	70.4	70.4	56.2 (−1.3)	61.1 (−4.2)
Uncertainty (Lewis & Gale, 1994)	60.4 (−11.7)	55.8 (−19.5)	65.0 (−5.4)	64.5 (−5.9)	54.0 (−3.5)	59.9 (−5.4)
Zero-shot	64.0 (−8.1)	59.4 (−15.9)	64.4 (−6.0)	64.3 (−6.1)	43.5 (−14.0)	59.1 (−6.2)

A2 EXACT VALUES FOR THE ONLINE RESULTS

As with the offline results, we list the exact values behind the online comparison of Section 4.3 here. Data arrives over $T{=}10$ steps (8 for IMDB), each method keeps a memory of 5 examples, and Table 6 reports the resulting stream-AUC on each dataset, together with each entry's gap to the best competing method.

Table 6: Exact values for Figure 2(b), the online stream-AUC of phi-1.5 on each dataset. Oracle-accumulate keeps all data seen so far and gives the upper bound. OLGA beats the best competing method by 6.6 points on average and comes within two points of supervised GradMM. Bold marks the best result in each column apart from the oracle, and underline marks the reference that each colored difference is measured against.

Method	IMDB ↑	SST-2 ↑	TwitEmo. ↑	Rotten Tom. ↑	RT-Polarity ↑	Avg. ↑
Oracle-accumulate	92.1 (+4.8)	88.4 (+10.0)	87.2 (+27.5)	84.5 (+20.1)	84.3 (+20.0)	87.3 (+18.2)
Online GradMM (ours)	**85.4** (+3.8)	**79.2** (+0.8)	**65.3** (+5.6)	73.2 (+8.8)	74.3 (+10.0)	**75.5** (+8.5)
OLGA (ours)	85.1 (+3.5)	74.2 (−4.2)	59.6 (−0.1)	**74.1** (+9.7)	**74.9** (+10.6)	73.6 (+6.6)
k-center (Sener & Savarese, 2018)	73.6 (−8.0)	78.4	58.8 (−0.9)	61.3 (−3.1)	62.9 (−1.4)	67.0
Reservoir (Vitter, 1985)	81.6	70.3 (−8.1)	59.7	62.1 (−2.3)	52.3 (−12.0)	65.2 (−1.8)
Herding (Rebuffi et al., 2017)	76.1 (−5.5)	67.7 (−10.7)	50.5 (−9.2)	61.3 (−3.1)	57.1 (−7.2)	62.5 (−4.5)
Prototypical (Snell et al., 2017)	80.4 (−1.2)	78.0 (−0.4)	47.3 (−12.4)	55.3 (−9.1)	54.7 (−9.6)	63.1 (−3.9)
Zero-shot	64.0 (−17.6)	59.4 (−19.0)	43.5 (−16.2)	64.4	64.3	59.1 (−7.9)

A3 EXTENDING THE OFFLINE RESULTS TO $m{=}10$ EXAMPLES

Distillation is designed for extreme compression, so it is worth checking how the picture changes when the budget is less tight. We repeat the offline comparison of Section 4.2 at $m{=}10$ examples, with the same methods and setup as Table 5. Table 7 gives the per-dataset results. The conclusions

of Section 4.2 carry over, as OLGA remains the strongest label-free method on average, 1.7 points ahead of the best selection method, and supervised GradMM stays close to the oracle. The margin is narrower than at m=5, however, since selection improves faster as the budget grows and already overtakes OLGA on three of the five datasets at m=10. Distillation is strongest under extreme compression while selection catches up as the budget grows, which is the split the hybrid pipeline is built on and is consistent with its results in Table 2.

Table 7: Offline accuracy of phi-1.5 on each dataset at m=10, the counterpart to Table 5. The upper block gives the supervised upper bounds and the lower block the label-free methods. OLGA stays ahead of the best selection method by 1.7 points on average, but leads on only two of the five datasets. Bold marks the best label-free result in each column, and underline marks the reference that each colored difference is measured against.

Method	**IMDB** ↑	**SST-2** ↑	**Rotten Tom.** ↑	**RT-Polarity** ↑	**TwitEmo.** ↑	**Avg.** ↑
Oracle	92.3 (+11.9)	88.2 (+6.2)	84.7 (+10.2)	83.2 (+9.1)	79.4 (+16.9)	85.6 (+12.9)
GradMM (Nguyen et al., 2025)	90.6 (+10.2)	87.2 (+5.2)	81.6 (+7.1)	82.6 (+8.5)	78.9 (+16.4)	84.2 (+11.5)
OLGA (ours)	**89.0** (+8.6)	78.9 (−3.1)	70.1 (−4.4)	**76.4** (+2.3)	57.6 (−4.9)	**74.4** (+1.7)
Prototypical (Snell et al., 2017)	78.8 (−1.6)	80.1 (−1.9)	72.2 (−2.3)	72.0 (−2.1)	60.2 (−2.3)	72.7
Herding (Rebuffi et al., 2017)	77.6 (−2.8)	78.8 (−3.2)	71.5 (−3.0)	71.8 (−2.3)	58.4 (−4.1)	71.6 (−1.1)
Uncertainty (Lewis & Gale, 1994)	70.4 (−10.0)	70.5 (−11.5)	73.0 (−1.5)	72.8 (−1.3)	**62.5**	69.8 (−2.9)
k-center (Sener & Savarese, 2018)	80.4	**82.0**	61.1 (−13.4)	61.6 (−12.5)	61.0 (−1.5)	69.2 (−3.5)
Random	68.3 (−12.1)	66.1 (−15.9)	**74.5**	**74.1**	62.1 (−0.4)	69.0 (−3.7)
Zero-shot	64.0 (−16.4)	59.4 (−22.6)	64.4 (−10.1)	64.3 (−9.8)	43.5 (−19.0)	59.1 (−13.6)

A4 What a distilled example looks like

Because OLGA returns plain text rather than weights, its synthetic examples can be read directly. OLGA optimizes each synthetic example so that its training gradient matches that of many real examples at once, and then maps it back to real tokens. The text it settles on therefore reads as a blend of those real examples, taking a phrase from one and another phrase from a different one, and joining them into a sentence that appears nowhere in the data. Tables 8 to 10 show one such example for each of the remaining datasets, in the style of Table 1. In each table, the synthetic example comes first and the real examples it draws from are listed below it. A phrase shared between the two is highlighted in both places, and its superscript gives the number of the real example it comes from.

Table 8: Synthetic example distilled by OLGA for **IMDB** (positive), with the seven real examples it borrows its phrasing from. A shared phrase is highlighted in both places, and its superscript gives the number of the real example it comes from.

Synthetic	IMDLEG is great[1] because it is a well made[2] and entertaining story that keeps you[3] engaged from beginning[4] onwards and the well done production and the talented and well[5] done people who[6] are involved in making the show are truly a great and[7] a
Real	1. This is just a short comment but I stumbled onto this movie by chance and I loved it. The acting is great, the story is simple and touching … Seek it out. 2. I thought this was one of those really great films to see with a bunch of close friends. … Although not a new concept this was a very well made film. 3. Powers Boothe turns in a stellar performance as 1970's cult figure Jim Jones of the Peoples Temple. … The movie keeps you riveted and is a must see for anyone. 4. A funny comedy from beginning to end! … Creative story line with a very talented cast. I thoroughly enjoyed it! 5. terribly underrated with matt dillon and tom skerritt, good backdrop for solid story … well acted and well cast, tommy lee jones and bruce dern make you hate them with passion 6. This movie will always be a Broadway and Movie classic, as long as there are still people who sing, dance, and act. 7. A great and truly independent film that hit most of my emotions and carried me into another world. …

Table 9: Synthetic example distilled by OLGA for **Rotten Tomatoes** (negative), with the four real examples it borrows its phrasing from. A shared phrase is highlighted in both places, and its superscript gives the number of the real example it comes from.

Synthetic	The new movie was boring and had no interesting or[1] well developed action or a good[2] one of the actors[3] in the story[4]
Real	1. what the director can't do is make either of val kilmer's two personas interesting or worth caring about. 2. the techno tux is good for a few laughs, as are chan and hewitt, but when such a good design turns out to be a cheap knockoff . . . 3. sadly, though many of the actors throw off a spark or two when they first appear, they can't generate enough heat in this cold vacuum of a comedy . . . 4. the story drifts so inexorably into cliches about tortured (and torturing) artists . . .

Table 10: Synthetic example distilled by OLGA for **RT-Polarity** (positive), with the three real examples it borrows its phrasing from. A shared phrase is highlighted in both places, and its superscript gives the number of the real example it comes from.

Synthetic	Amazing movie review provided by critics is a great[1] example of how[2] a single positive sentence can make a[3] significant impact on the
Real	1. morton is a great actress portraying a complex character, but morvern callar grows less compelling the farther it meanders from its shocking start. 2. a truly moving experience, and a perfect example of how art – when done right – can help heal, clarify, and comfort. 3. australian actor/director john polson and award-winning english cinematographer giles nuttgens make a terrific effort at disguising the obvious with energy and innovation.

A5 Alternative label-free gradient signals

The group-distribution loss equation 3 is one of several ways to read a matching target off the frozen model. Here we give the exact definitions of the two alternatives compared against it in Section 4.5, pseudo-token and self-information. Both are built like ℓ^{sup} and ℓ^{grp}, as a per-example cross-entropy toward a target the data supplies. Each substitutes into equation 1 in place of ℓ, and the same loss forms both the real and the synthetic gradient.

Pseudo-token. Let $\boldsymbol{u}$ be the fixed suffix " In a single word:" and write $\boldsymbol{p} \oplus \boldsymbol{u}$ for the prompt with that suffix appended. The loss is the cross-entropy toward the token the frozen model finds most likely to come next,

$$\ell^{\mathrm{pt}}(s, \boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(\hat{v}(s) \mid \boldsymbol{p} \oplus \boldsymbol{u}), \qquad \hat{v}(s) = \arg\max_{v} \; p_{\boldsymbol{\theta}_0}(v \mid \boldsymbol{p} \oplus \boldsymbol{u}).$$

The model is asked to name the example in one word and then trained to keep saying that word. Like ℓ^{sup} this puts all the weight on a single token, but the token is whatever the model volunteers rather than a class name we supply.

Self-information. No suffix is added here. Writing $s = (\boldsymbol{p}, \boldsymbol{r})$ with response tokens $r_1, \ldots, r_n$, the quantity $-\log p_{\boldsymbol{\theta}_0}(r_i \mid \boldsymbol{p}, r_{<i})$ says how unexpected the frozen model found the token at position i. The loss is the cross-entropy at the position where it is largest,

$$\ell^{\mathrm{si}}(s, \boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(r_{i^\star} \mid \boldsymbol{p}, r_{<i^\star}), \qquad i^\star(s) = \arg\max_{1 \le i \le n} \big[-\log p_{\boldsymbol{\theta}_0}(r_i \mid \boldsymbol{p}, r_{<i})\big].$$

The reasoning is that the token the model predicts least well is the one that carries most of what makes the example distinctive.

Two differences from equation 3 follow from these definitions. Both targets above are single tokens, where the group-distribution target spreads over the k tokens of $\mathcal{V}_g^k$; and both depend on the individual example, where the group-distribution target depends only on the group it belongs to.

For the real anchors neither difference costs anything, since every target is computed once under θ_0 before matching starts. For the synthetic set, the second one does, because the synthetic sequences keep changing while the solver runs, so a target read off a sequence has to be found again as it changes, whereas the group-distribution target is fixed from the outset. This is what separates the three runtimes in Figure 4, and self-information is the most expensive of them because finding its target means scoring every position of every sequence inside the matching loop.

A6 ALTERNATIVE SELECTORS FOR THE HYBRID PIPELINE

The first stage of the hybrid pipeline only has to thin out the incoming batch, so any selection method can run it. As in Section 4.5, we replace uniform sampling in the first stage of the offline GradMM-Hybrid route with k-center, the strongest selection method of Figure 2(b), and keep everything else fixed. Table 11 gives the per-dataset results. Uniform sampling is ahead of k-center on all five datasets, from 2.0 points on IMDB to 8.3 on TwitterEmotion, and is also faster to run. This supports keeping the original distribution of the data at the selection stage and leaving the rest of the compression to distillation.

Table 11: Per-dataset version of Table 3, the offline accuracy of the GradMM-Hybrid route with uniform sampling or k-center as its selector. Uniform sampling is more accurate on every dataset and faster to run. Bold marks the better selector in each column.

Selector	SST-2 ↑	Rotten Tomatoes ↑	IMDB ↑	RT-Polarity ↑	TwitterEmotion ↑	Avg. ↑	SST-2 runtime (s) ↓
Uniform (ours)	**88.2**	**83.9**	**92.3**	**82.1**	**83.4**	**86.0**	**436.1**
k-center	84.2	80.9	90.3	79.8	75.1	82.1	564.8

A7 ALTERNATIVE MERGE STRATEGIES

The running memory can absorb each arriving batch in more than one way. As in Section 4.5, we compare the cascaded merge against two alternatives for folding each batch into the memory of OLGA, the one-pass and the history-weighted merge, over $T{=}5$ steps. Table 12 gives the per-dataset stream-AUC. The cascaded merge scores highest on SST-2, Rotten Tomatoes, and RT-Polarity, while one-pass leads on TwitterEmotion and history-weighted on IMDB. Each alternative wins on one dataset at the cost of the others, which leaves the cascaded merge ahead on average, at 72.0 against 70.2 and 69.8.

Table 12: Per-dataset version of Table 4, the online stream-AUC of OLGA under each merge strategy. The cascaded merge leads on three of the five datasets and on average. Bold marks the best result in each column.

Merge strategy	SST-2 ↑	Rotten Tomatoes ↑	IMDB ↑	RT-Polarity ↑	TwitterEmotion ↑	Avg. ↑
Cascaded (ours)	**61.8**	**85.4**	69.1	**69.5**	73.9	**72.0**
One-pass	60.4	78.7	69.3	62.5	**80.1**	70.2
History-weighted	57.2	85.3	**72.4**	68.1	66.0	69.8

A8 MEMORY LEARNABILITY DIAGNOSTICS

This section asks whether a cheap property of the memory can tell in advance how well the model will do on the stream, without fine-tuning on it. We measure two such properties of each method's memory on every dataset. Feature discriminability measures how well the memory's examples separate the classes in the model's feature space, and gradient contrast measures how different the training gradients of the classes are. Figure 5 plots each against stream-AUC. Neither one predicts stream-AUC on its own. The least-squares fits explain almost none of the variation, with R^2 of 0.03 for feature discriminability and 0.08 for gradient contrast, and memories with nearly the same score

on either property end up tens of points apart in stream-AUC. The gradient-matching memories and the oracle do sit in the high-accuracy region, but so do some selection memories whose scores are very different from theirs. How well a memory trains the model therefore cannot be read off a single cheap property, and the choice of method, rather than a proxy score, has to drive how the memory is built.

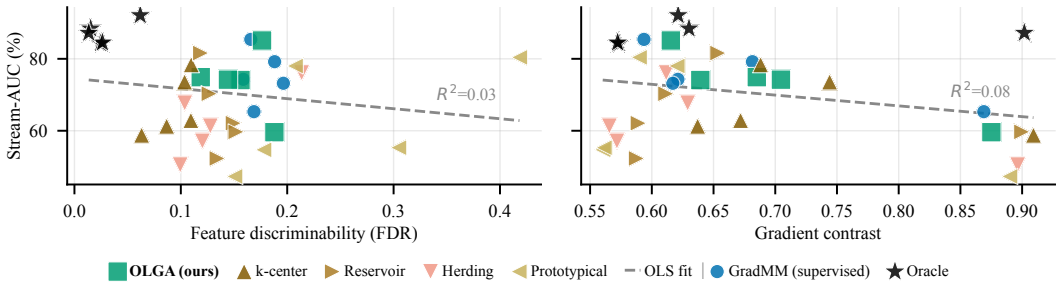

Figure 5: Feature discriminability (left) and gradient contrast (right) of each memory against its stream-AUC, one point per dataset and method. Neither property predicts how well a memory trains the model.

A9 Discrimination of the group-distribution proxy

OLGA matches gradients of the group-distribution loss ℓ^{grp} of equation 3 instead of the label loss ℓ^{sup}, but the model is evaluated by class accuracy. This section gives the condition under which a synthetic set distilled with ℓ^{grp} trains the model as well as one distilled with labels. We first show that gradient matching works for any loss that meets a few standard conditions, and then measure how far ℓ^{grp} is from ℓ^{sup}. Throughout, the loss we match on is used for both gradients in equation 1, the real one and the synthetic one.

Setup. Fine-tuning is short, so we only need the loss to behave well over the small region of parameters it visits.

Assumption 1. For the loss being matched, the following hold over the fine-tuning region: (i) the real and synthetic losses are β-smooth; (ii) the real loss obeys $\|\nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_{\text{real}},\boldsymbol{\theta})\|^2 \geq 2\mu\,\ell(\mathcal{D}_{\text{real}},\boldsymbol{\theta})$ with minimum zero; (iii) gradients along fine-tuning are bounded by G; (iv) the parameters move by at most δ, so $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\| \leq \delta$; and (v) the solver reaches anchor error $\|\nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_{\text{real}},\boldsymbol{\theta}_0) - \nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_{\text{syn}},\boldsymbol{\theta}_0)\| \leq \epsilon$.

Condition (v) measures the match as a distance, while equation 1 compares gradients by direction. The two agree once the scale is fixed. If both gradients have norm about R, their distance is $R\sqrt{2D}$, so lowering the matching objective D also lowers ϵ. We use the distance because it lets errors from different sources be added.

The match holds during fine-tuning. The synthetic set is matched once, at $\boldsymbol{\theta}_0$, but fine-tuning then moves the parameters away from $\boldsymbol{\theta}_0$. Write $\xi = 2\beta\delta + \epsilon$. The following lemma shows that the two gradients stay close along the whole run, and the theorem turns this into a bound on the loss.

Lemma 2. *Under Assumption 1, at every fine-tuning step t,*

$$\left\|\nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_{real},\boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_{syn},\boldsymbol{\theta}_t)\right\| \leq 2\beta\delta + \epsilon = \xi.$$

Proof. Add and subtract both gradients at $\boldsymbol{\theta}_0$, which splits the gap into three terms. The middle one is the anchor error, at most ϵ by (v). Each of the other two is the change of a β-smooth gradient over a distance of at most δ, so at most $\beta\delta$. □

Theorem 3. *Under Assumption 1 with step size $\eta \leq 1/\beta$, fine-tuning on the synthetic set tracks fine-tuning on the real set, with*

$$\left|\ell(\mathcal{D}_{real},\boldsymbol{\theta}_t) - \ell(\mathcal{D}_{syn},\boldsymbol{\theta}_t)\right| \leq \frac{\xi\,(2G-\xi)}{2\mu} \qquad \textit{for every } t.$$

Proof. With $\eta \leq 1/\beta$, smoothness makes every step cut the synthetic loss by at least $\frac{\eta}{2}\|\nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_{\text{syn}}, \boldsymbol{\theta}_t)\|^2$, and Lemma 2 lets us replace the synthetic gradient with the real one at a cost of ξ. Applying (iii) and then (ii) gives $\ell(\mathcal{D}_{\text{real}}, \boldsymbol{\theta}_{t+1}) \leq (1-\eta\mu)\,\ell(\mathcal{D}_{\text{real}}, \boldsymbol{\theta}_t) - \frac{\eta}{2}(\xi^2 - 2\xi G)$. Unrolling this recursion leaves a term that decays with t plus the stated residual. □

Neither proof uses a label, a verbalizer token, or the form of a target distribution; they use only conditions (i)–(v). Both results therefore hold unchanged with ℓ^{grp} in place of ℓ. The synthetic set reproduces the training signal of the anchors, as measured by the loss we matched on.

From the matched loss to the label loss. What remains is the gap between ℓ^{grp}, which we match, and ℓ^{sup}, which determines accuracy. One quantity captures it.

Definition 4. The *misalignment* of the group-distribution loss is

$$\Delta = \sup_{\|\boldsymbol{\theta}-\boldsymbol{\theta}_0\|\leq\delta} \big\|\nabla_{\boldsymbol{\theta}}\,\ell^{\text{grp}}(\mathcal{D}_{\text{real}}, \boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\,\ell^{\text{sup}}(\mathcal{D}_{\text{real}}, \boldsymbol{\theta})\big\|,$$

the largest distance between the two gradients over the region fine-tuning visits. It is zero for the supervised loss itself.

Theorem 5. *Let $\mathcal{D}_{syn}$ be distilled by matching ℓ^{grp} to anchor error ϵ, and write $\boldsymbol{\theta}_t$ for the parameters after fine-tuning on $\mathcal{D}_{real}$ and $\boldsymbol{\theta}_t^{\text{syn}}$ for those after fine-tuning on $\mathcal{D}_{syn}$. Under Assumption 1 for ℓ^{sup}, with $\xi' = 2\beta\delta + \epsilon + \Delta$,*

$$\big\|\nabla_{\boldsymbol{\theta}}\,\ell^{\text{sup}}(\mathcal{D}_{real}, \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\,\ell^{\text{grp}}(\mathcal{D}_{syn}, \boldsymbol{\theta}_t)\big\| \leq \xi',$$

$$\big|\ell^{\text{sup}}(\mathcal{D}_{real}, \boldsymbol{\theta}_t) - \ell^{\text{sup}}(\mathcal{D}_{real}, \boldsymbol{\theta}_t^{\text{syn}})\big| \leq \frac{\xi'\,(2G - \xi')}{2\mu}.$$

Proof. Add and subtract $\nabla_{\boldsymbol{\theta}}\,\ell^{\text{grp}}(\mathcal{D}_{\text{real}}, \boldsymbol{\theta}_t)$. The first difference, between the two losses on real data, is at most Δ by Definition 4, since $\boldsymbol{\theta}_t$ stays within δ of $\boldsymbol{\theta}_0$. The second, between real and synthetic data under ℓ^{grp}, is at most ξ by Lemma 2. The synthetic set thus tracks the supervised gradient with total error ξ', and Theorem 3, with ℓ^{sup} as the reference, gives the loss bound. □

Corollary 6. *With $\Delta = 0$ the bound collapses to Theorem 3, recovering the supervised case exactly. For $\xi' \leq G$ the right-hand side is strictly increasing in ξ', so the guarantee tightens as Δ falls.*

The condition is therefore a small misalignment Δ. When the gradient of ℓ^{grp} points where the gradient of the label loss points over the fine-tuning region, the proxy induces the same class discrimination as the label loss, and the guarantee degrades smoothly as Δ grows.

A10 Error of the two-level online approximation

The ideal online set of equation 4 re-distills the whole history at every step, which a stream cannot afford. The cascaded merge equation 5 replaces it with two matching problems per step, both solved at $\boldsymbol{\theta}_0$. The first distills the new block $\mathcal{D}^t_{\text{real}}$ into $\mathcal{D}^t_{\text{syn}}$, and the second distills $\mathcal{D}^t_{\text{syn}} \cup \mathcal{M}^{t-1}$ back down to m examples. The past reaches the current step only through the memory $\mathcal{M}^{t-1}$. This section bounds the error of this approximation against the ideal and shows that it does not grow with the stream length T. We use Assumption 1 and Lemma 2 from Appendix A9.

Error of a single step. The only new source of error is that the memory summarizes the past imperfectly.

Definition 7. The *staleness* of the memory at step t is

$$\zeta^t = \big\|\nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{M}^{t-1}, \boldsymbol{\theta}_0) - \nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}^{1:t-1}_{\text{real}}, \boldsymbol{\theta}_0)\big\|.$$

It is zero when the memory reproduces the gradient of the whole past exactly, and grows as the fixed budget m has more history to summarize.

The merge combines two sets of the same size m, so its target is the average of their gradients. Write $w = \frac{1}{2}$ and take as reference the loss that weights the past and the new block in the same way,

$$\ell^{w,t}(\boldsymbol{\theta}) = w\,\ell(\mathcal{D}^{1:t-1}_{\text{real}}, \boldsymbol{\theta}) + (1-w)\,\ell(\mathcal{D}^t_{\text{real}}, \boldsymbol{\theta}).$$

Theorem 8. *Let the two solves at step t reach anchor errors ϵ_1^t and ϵ_2^t, and write $\epsilon^t = (1-w)\,\epsilon_1^t + \epsilon_2^t$. Under Assumption 1 with $\ell^{w,t}$ as the reference,*

$$\left\|\nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{M}^t, \boldsymbol{\theta}_0) - \nabla_{\boldsymbol{\theta}}\,\ell^{w,t}(\boldsymbol{\theta}_0)\right\| \le \epsilon^t + w\,\zeta^t,$$

and along fine-tuning this propagates to $\xi^t = 2\beta\delta + \epsilon^t + w\,\zeta^t$.

Proof. The merge target and the reference are both w-weighted combinations of a past term and a new term, so their difference splits in two. The past terms differ by at most ζ^t, scaled by w; the new terms differ by the first solve's error ϵ_1^t, scaled by $1-w$. The second solve matches the merge target within ϵ_2^t. Adding the pieces gives the bound, and Lemma 2 carries it through fine-tuning. □

Compared with the offline error ξ, the only extra term is $w\,\zeta^t$. This is the whole cost of not storing the history.

Remark 9. The reference $\ell^{w,t}$ weights the past and the new block equally, rather than by how much data each stands for. Let b^t be the distance between this balanced reference and the data-proportional one, so that $b^t = 0$ when w equals the data proportion. The bounds below hold for either weighting, with b^t accounting for the difference.

Errors do not accumulate. A small error at every step could still add up over a long stream. The next result shows that it does not, because the same contraction that makes fine-tuning converge also shrinks errors from earlier steps.

Theorem 10. *Suppose in addition that the cumulative loss is α-strongly convex and $\eta \le 1/\beta$. Let $\boldsymbol{\theta}^t$ be the parameters carried by fine-tuning on the memory and $\boldsymbol{\theta}_\star^t$ those obtained by fine-tuning on the whole history seen so far, and write $\bar{\xi} = \max_t \xi^t$ and $\bar{b} = \max_t b^t$. Then*

$$\|\boldsymbol{\theta}^t - \boldsymbol{\theta}_\star^t\| \le (1-\eta\alpha)\,\|\boldsymbol{\theta}^{t-1} - \boldsymbol{\theta}_\star^{t-1}\| + \eta\,(\xi^t + b^t), \qquad \|\boldsymbol{\theta}^t - \boldsymbol{\theta}_\star^t\| \le (1-\eta\alpha)^t\,\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}_\star^0\| + \frac{\bar{\xi} + \bar{b}}{\alpha}.$$

Proof. Both trajectories take one step from their previous iterates. Their difference splits into a descent step on the same strongly convex loss, which shrinks the distance by a factor $1 - \eta\alpha$, and the gap between the memory gradient and the cumulative gradient, which is at most $\xi^t + b^t$ by Theorem 8 and Remark 9. This gives the first inequality. Unrolling it sums a geometric series and gives the second. □

Corollary 11. *Once the first term has decayed, the per-step loss gap to fine-tuning on the whole history is at most $\frac{\beta}{2}\big((\bar{\xi} + \bar{b})/\alpha\big)^2$, and so is the average over the stream.*

Proof. The reference minimizes the cumulative loss, so its gradient is zero there, and β-smoothness bounds the loss gap by $\frac{\beta}{2}$ times the squared parameter distance. Insert Theorem 10 and average over the stream. □

Neither bound depends on T. The error of the cascaded merge settles at a fixed level, set by the per-step error $\bar{\xi}$ and the weighting bias $\bar{b}$, and stays there however long the stream runs.

A11 Cost of routing through the selection stage

The hybrid pipeline distills a selected pool rather than the full batch. This section shows that the result is still a controlled approximation to gradient matching on the full batch, and that the cost of distillation does not depend on the batch size B.

Write $\mathcal{D}_B$ for the incoming batch, $\mathcal{D}_K \subseteq \mathcal{D}_B$ for the pool the selector keeps, and $\mathcal{D}_{\text{syn}}$ for the m examples distilled from that pool. Matching is done against $\mathcal{D}_K$, but we want $\mathcal{D}_{\text{syn}}$ to represent $\mathcal{D}_B$. The difference is one quantity.

Definition 12. The *selection residual* is

$$\epsilon_{\text{sel}} = \left\|\nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_K, \boldsymbol{\theta}_0) - \nabla_{\boldsymbol{\theta}}\,\ell(\mathcal{D}_B, \boldsymbol{\theta}_0)\right\|,$$

the gradient distance between the selected pool and the full batch. A selector that covers the batch well makes it small.

Theorem 13. *Let the matching stage reach anchor error ϵ against $\mathcal{D}_K$, and let Assumption 1 hold with $\ell(\mathcal{D}_B, \cdot)$ as the reference. Then along fine-tuning*

$$\left\|\nabla_{\boldsymbol{\theta}}\, \ell(\mathcal{D}_B, \boldsymbol{\theta}_t) - \nabla_{\boldsymbol{\theta}}\, \ell(\mathcal{D}_{syn}, \boldsymbol{\theta}_t)\right\| \leq 2\beta\delta + \epsilon + \epsilon_{\mathrm{sel}} = \xi + \epsilon_{\mathrm{sel}},$$

with the corresponding loss gap $(\xi + \epsilon_{\mathrm{sel}})\big(2G - (\xi + \epsilon_{\mathrm{sel}})\big)/(2\mu)$. Moreover the matching stage reads only $\mathcal{D}_K$, so its cost is set by K and does not grow with B.

Proof. At $\boldsymbol{\theta}_0$, the triangle inequality puts the synthetic set within $\epsilon + \epsilon_{\mathrm{sel}}$ of the full batch. The synthetic set is within ϵ of the pool, and the pool is within ϵ_{sel} of the batch. This is an anchor error against $\mathcal{D}_B$, so Lemma 2 applies with $\ell(\mathcal{D}_B, \cdot)$ as the reference. Only one drift term appears, because the synthetic set and the batch are compared along the same fine-tuning trajectory. Theorem 3, with ξ replaced by $\xi + \epsilon_{\mathrm{sel}}$, gives the loss gap. For the cost, the solver optimizes the embeddings of the pool and never reads $\mathcal{D}_B$; the selector reads $\mathcal{D}_B$ once, in a single pass. □

Corollary 14. *Distilling the whole batch is the case $\mathcal{D}_K = \mathcal{D}_B$, where $\epsilon_{\mathrm{sel}} = 0$ and the bound returns to ξ.*

The hybrid pipeline therefore behaves like gradient matching on the full batch up to one explicit extra term, ϵ_{sel}, which a good selector keeps small, while its distillation cost depends on K rather than B.

The corollary compares the two routes at the same anchor error ϵ, but ϵ is the error the solver actually reaches, and it depends on the problem the solver is given. The matching problem is non-convex and the solver runs for a fixed number of iterations, and matching a pool of K anchors is a smaller problem than matching all B, so it can end at a lower anchor error on the pool than on the full batch. Whenever this reduction outweighs ϵ_{sel}, the bound for the hybrid pipeline is tighter than for distilling the whole batch, which is consistent with the hybrid pipeline outperforming the direct route in our experiments. The gain comes from giving the distillation stage a smaller problem, not from splitting the compression into more stages.