DRESS: DESCENT-ORIENTED DATA SELECTION FOR TOKEN-EFFICIENT LANGUAGE MODEL TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving token efficiency is increasingly important for training modern large language models. Training tokens differ in their contribution to learning, and their usefulness can change as training progresses, motivating online data selection that adapts to the current model. We propose Descent-oriented Reweighting for Efficient Sample Selection (DRESS), which scores training examples by their predicted contribution to reducing a held-out target loss. From first- and second-order Taylor expansions, we derive two variants, DRESS-FO and DRESS-SO, where DRESS-SO uses a generalized Gauss–Newton (GGN) approximation to capture interactions among candidate updates. The two variants offer a trade-off between scoring cost and the fidelity of the local target-loss approximation. We evaluate DRESS on GPT-2 pretraining across model sizes from 124M to 1.5B parameters and on Qwen3.5-9B-Base LoRA fine-tuning. On GPT-2 XL, both variants outperform the compared baselines in average zero-shot accuracy, and with only 30B update tokens, they even outperform random selection trained on 60B update tokens. When tested on Qwen3.5-9B-Base, both versions of DRESS improve over the standard fine-tuning baseline on evaluated benchmarks, while the second-order DRESS outperforms the first-order version.

1 INTRODUCTION

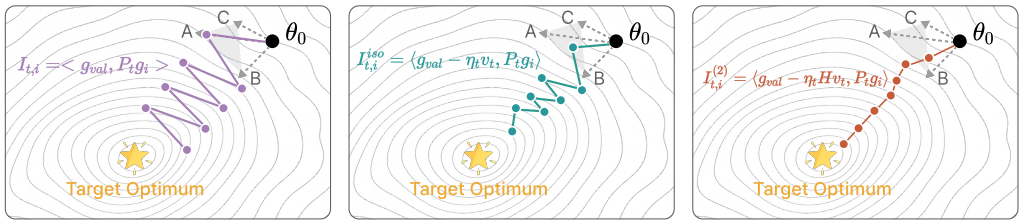
Large language models (LLMs) have demonstrated strong capabilities across a wide range of applications (Team et al., 2026; DeepSeek-AI, 2026). However, training these models requires substantial computational resources and financial investment. Pretraining corpora contain examples that differ in quality, relevance, and redundancy, so processing the same number of tokens from different examples can yield different learning gains. Improving token efficiency by making better use of each training token is therefore important for LLM pretraining, especially under limited compute budgets.

One approach to improving token efficiency is to train a proxy model and use it to guide data selection or reweighting (Wettig et al., 2024; Xie et al., 2023a). However, training the proxy incurs additional cost, and a fixed subset may become less suitable as the model evolves during training. An example that provides useful information at one step may contribute little to optimization at another. These limitations motivate online data selection that adapts to the current training state and prioritizes examples expected to improve language modeling performance.

A central challenge for online data selection is identifying useful examples without incurring the cost of full backpropagation for every candidate. Loss-based selection offers an inexpensive signal of how difficult an example is for the current model (Qin et al., 2024; Sow et al., 2025), but its loss alone does not directly estimate the improvement produced by an optimizer update. Gradient-based methods instead use gradient information to estimate which examples are likely to reduce the target loss most (Wang et al., 2024; 2026). However, computing per-example gradients is expensive, so practical methods rely on approximations whose quality depends on how well they preserve the relevant update directions. A selection rule must produce sufficiently better updates to justify the resources spent on scoring candidates.

One source of approximation error arises from interactions among samples selected for the same update. Although individual sample gradients may align with the target gradient, their combined effect depends on the local curvature of the target loss. GREATS (Wang et al., 2024) and OPUS (Wang et al., 2026) account for these interactions through second-order Taylor corrections, but their

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.



(a) DRESS (first order) (b) Isotropic curvature (c) DRESS (second order)

Figure 1: **Role of target curvature in data selection.** Contours show the held-out target loss, and A, B, and C denote candidate update directions at θ_0. (a) DRESS-FO (First order) uses alignment between preconditioned candidate gradients and the target gradient. (b) The identity approximation used in GREATS (Wang et al., 2024) and OPUS (Wang et al., 2026) treats target curvature as isotropic, discarding differences across update directions. (c) DRESS-SO (Second order) uses a generalized Gauss–Newton (GGN) approximation to capture directional curvature and interactions among candidate updates. See Section 5.4.2 for a comparison of target-loss prediction fidelity.

practical implementations approximate the target Hessian by an identity matrix. This approximation reduces computation but discards differences in curvature across update directions.

We introduce **Descent-oriented Reweighting for Efficient Sample Selection (DRESS)**, an online method that assigns candidate weights according to their predicted contribution to reducing a held-out target loss after a local optimizer update. The first-order variant, **DRESS-FO**, scores each example by the alignment between its preconditioned gradient and the target gradient. The second-order variant, **DRESS-SO**, incorporates interactions among candidate updates through a quadratic Taylor approximation, using a generalized Gauss–Newton (GGN) approximation (Martens & Grosse, 2015) to retain directional curvature information. To compute the second-order correction efficiently, we use a Jacobian–vector product (JVP) followed by a vector–Jacobian product (VJP) to evaluate a shared curvature-vector product without explicitly computing the curvature matrix. The two variants offer a trade-off between scoring cost and the fidelity of the local target-loss approximation.

DRESS converts influence scores into sampling weights through an exponentiated-gradient update derived from mirror descent (Beck & Teboulle, 2003), samples a subset according to these weights, and trains on the selected samples. To control scoring cost, we restrict influence estimation to the LM head during pretraining (Koh & Liang, 2017). We evaluate DRESS on GPT-2 pretraining across model sizes from Small (124M) to XL (1.5B) (Radford et al., 2019) and on Qwen3.5-9B-Base fine-tuning (Qwen Team, 2026), measuring downstream performance and selection overhead. On GPT-2 Large and XL, DRESS-SO achieves higher average zero-shot accuracy than the compared data selection methods at the same update-token budget. On Qwen3.5-9B-Base, both variants improve over the standard fine-tuning baseline across the evaluated reasoning benchmarks.

2 Related Work

Data Selection and Reweighting DSIR (Xie et al., 2023b) selects data through importance resampling to match a target distribution, while QuRating (Wettig et al., 2024) learns document-quality scores from language-model judgments. DCLM (Li et al., 2024a) systematically evaluates data curation strategies, and DoReMi (Xie et al., 2023a) learns domain weights using a small proxy model. During training, InfoBatch (Qin et al., 2024) combines loss-based pruning with gradient rescaling to correct selection bias, while Sow et al. (2025) dynamically reweight individual examples using their current losses. Gradient-based approaches are natural choices, as gradients can be used to estimate how much the loss will decrease. GREATS (Wang et al., 2024) derives online selection from Taylor approximations to validation-loss reduction, and OPUS (Wang et al., 2026) incorporates an optimizer preconditioner, projected scoring, and stochastic sampling. Their scalable interaction scores use an identity approximation to target curvature.

Data Selection in Frontier LLM Training Frontier LLM pipelines depend heavily on the quality of pretraining data and also perform data selection for their pretraining corpora. Kimi K3 (Team et al., 2026) uses heuristic and classifier-based filtering to build pretraining datasets. DeepSeek-V4

(DeepSeek-AI, 2026) filters automatically generated and templated web content. GLM-5 (Zeng et al., 2026) employs a DCLM classifier (Li et al., 2024a) for web data, quality-aware code sampling, and language-model scoring of mathematics and science documents. These practices establish the practical importance of data allocation at scale.

3 BACKGROUND

Training data and loss. Consider an LLM with parameters $\theta_t \in \mathbb{R}^d$ at optimizer step t, where d is the number of parameters. Let $\mathcal{D}_{\text{train}}$ denote the training set and $\mathcal{L}(x;\theta)$ the loss on an example x. For any dataset $\mathcal{D}$, we write its average loss as $\mathcal{L}(\mathcal{D};\theta) = |\mathcal{D}|^{-1}\sum_{x\in\mathcal{D}}\mathcal{L}(x;\theta)$. The gradient of a training example x_i at the current parameters is $g_i(\theta_t) = \nabla_\theta\mathcal{L}(x_i;\theta_t) \in \mathbb{R}^d$. We seek to prioritize training examples whose updates reduce the loss $\mathcal{L}(\mathcal{D}_{\text{val}};\theta)$ on a held-out validation set $\mathcal{D}_{\text{val}}$. In practice, we estimate validation-loss reduction using a separate selection-target set $\mathcal{D}_{\text{target}}$, drawn from the same distribution as $\mathcal{D}_{\text{val}}$.

Candidate weights and optimizer update. At step t, we receive a candidate batch $\mathcal{B}_t = \{x_i\}_{i=1}^n \subseteq \mathcal{D}_{\text{train}}$ and assign weights $w = (w_1,\dots,w_n)$ in the probability simplex $\Delta_n = \{w \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^n w_i = 1\}$. We model the resulting one-step parameter update as

$$\theta_{t+1}(w) = \theta_t - \eta_t P_t \sum_{i=1}^n w_i g_i(\theta_t), \tag{1}$$

where $\eta_t > 0$ is the learning rate and $P_t \in \mathbb{R}^{d\times d}$ is an optimizer-dependent preconditioner. When optimizing the weights, we hold the candidate gradients, P_t, and the optimizer state fixed. Thus, Equation (1) is a local model used to assess candidate updates. For SGD, P_t is the identity matrix; for optimizers such as Adam (Kingma & Ba, 2014), AdamW (Loshchilov & Hutter, 2017), and Muon (Liu et al., 2025), it represents a local preconditioning approximation. Our experiments follow the AdamW and Muon preconditioners used in OPUS (Wang et al., 2026).

The online weight-selection problem is therefore

$$\min_{w\in\Delta_n} \mathcal{L}(\mathcal{D}_{\text{val}};\theta_{t+1}(w)). \tag{2}$$

The goal is to allocate more weight to examples whose combined update yields a lower target loss. The following section develops first- and second-order approximations to guide this choice.

4 METHODS

We introduce Descent-oriented Reweighting for Efficient Sample Selection (DRESS), which assigns sample weights according to their predicted contribution to reducing the target loss in Equation (2). These weights serve as sampling probabilities for selecting a subset of the candidate batch. First- and second-order Taylor expansions of the target loss yield DRESS-FO and DRESS-SO, respectively, offering a trade-off between scoring cost and the fidelity of the local target-loss approximation.

4.1 FIRST-ORDER INFLUENCE SCORE

Let the gradient of the target loss at the current parameters be $g_{\text{val}}(\theta_t) = \nabla_\theta\mathcal{L}(\mathcal{D}_{\text{val}};\theta_t) \in \mathbb{R}^d$. A first-order Taylor expansion around θ_t, evaluated at the weighted update in Equation (1), gives

$$\mathcal{L}(\mathcal{D}_{\text{val}};\theta_{t+1}(w)) \approx \mathcal{L}(\mathcal{D}_{\text{val}};\theta_t) + \langle g_{\text{val}}(\theta_t), \theta_{t+1}(w) - \theta_t\rangle \tag{3}$$

$$= \mathcal{L}(\mathcal{D}_{\text{val}};\theta_t) - \eta_t \sum_{i=1}^n w_i \left\langle g_{\text{val}}(\theta_t), P_t g_i(\theta_t)\right\rangle. \tag{4}$$

The predicted reduction is linear in the sample weights, with each example contributing according to the alignment between its preconditioned gradient and the target gradient. We therefore define the first-order influence score as

$$I_{t,i} = \left\langle g_{\text{val}}(\theta_t), P_t g_i(\theta_t)\right\rangle. \tag{5}$$

When $I_{t,i} > 0$, the direction $-P_t g_i(\theta_t)$ is a local descent direction for the target loss. Under the simplex constraint, shifting weight from an example with a lower score to one with a higher score reduces the first-order approximation in Equation (4).

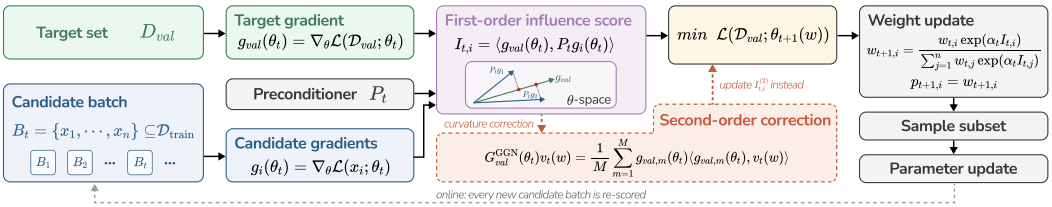


Figure 2: **Overview of DRESS.** Candidates are scored using alignment between their optimizer-preconditioned gradients and a direction estimated from held-out target data. DRESS-FO uses the target gradient; DRESS-SO corrects this direction using a shared generalized Gauss–Newton (GGN) curvature-vector product capturing interactions among candidate updates. A KL-based mirror-descent update converts scores into sampling probabilities, and the model is updated using selected examples.

4.2 Mirror-Descent Weight Update

We next convert the influence scores into feasible sample weights. For a candidate weight vector w, the weighted training loss is $\mathcal{L}^w(\mathcal{B}_t; \theta) = \sum_{i=1}^n w_i \mathcal{L}(x_i; \theta)$. For the local update in Equation (1), write $\theta(w) = \theta_{t+1}(w)$. The weight-selection objective in Equation (2) can then be expressed as

$$\min_w \ \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w)) \qquad \text{subject to} \qquad \sum_{i=1}^n w_i = 1, \quad w_i \geq 0. \tag{6}$$

Following Beck & Teboulle (2003), we take a mirror-descent step on this objective. Let w_t denote reference weights on the current candidate batch and w_{t+1} the weights after this step. Mirror descent minimizes a linear approximation to the objective while penalizing deviations from w_t:

$$\text{Prox}\big(\alpha_t \nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t)),\ w_t\big) := \arg\min_{w \in \Delta_n} \{\alpha_t \langle \nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t)),\ w \rangle + D(w \| w_t)\}. \tag{7}$$

Here, D is a divergence, $\alpha_t > 0$ is the mirror-descent step size, and $\nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t))$ is the gradient of the composite function $w \mapsto \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w))$ evaluated at w_t. Choosing the KL divergence yields the following closed-form solution to the proximal subproblem.

Lemma 1. *Let $D(x\|y) = \sum_i x_i \log(x_i/y_i)$ be the KL divergence. For any reference weights $w_t \in \Delta_n$ with full support, finite objective gradient, and step size $\alpha_t > 0$, the proximal subproblem in Equation (7) has the unique solution*

$$w_{t+1} \ \propto \ w_t \odot \exp\{-\alpha_t \nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t))\}. \tag{8}$$

The proof is given in Section A. Using the first-order approximation in Equation (4), the negative derivative of the target loss with respect to w_i is

$$-\frac{\partial}{\partial w_i} \mathcal{L}(\mathcal{D}_{\text{val}}; \theta_{t+1}(w)) \approx \eta_t I_{t,i}. \tag{9}$$

Absorbing the common factor η_t into the mirror-descent step size, and reusing α_t for the resulting score coefficient, gives the DRESS weight update

$$w_{t+1,i} = \frac{w_{t,i} \exp(\alpha_t I_{t,i})}{\sum_{j=1}^n w_{t,j} \exp(\alpha_t I_{t,j})}, \tag{10}$$

where $\alpha_t > 0$ controls the strength of reweighting. This closed-form update solves the KL-regularized linear subproblem. We then sample examples from the candidate batch using the updated weights as sampling probabilities, $p_{t+1,i} = w_{t+1,i}$, and perform backpropagation on the selected examples.

4.3 Second-Order Influence Score

The first-order score evaluates each candidate update independently. To account for interactions among candidates, we include the quadratic term in the Taylor expansion of the target loss. As in Section 3, the candidate gradients, preconditioner, and optimizer state remain fixed while we optimize the weights. Define the aggregate preconditioned update direction $v_t(w) := P_t \sum_{j=1}^n w_j g_j(\theta_t)$, so

that $\theta_{t+1}(w) = \theta_t - \eta_t v_t(w)$. Let $H_{\text{val}}(\theta_t) = \nabla^2_\theta \mathcal{L}(\mathcal{D}_{\text{val}}; \theta_t)$ denote the Hessian of the target loss. Substituting Equation (1) into a second-order Taylor expansion at θ_t gives

$$\mathcal{L}(\mathcal{D}_{\text{val}}; \theta_{t+1}(w)) \approx \mathcal{L}(\mathcal{D}_{\text{val}}; \theta_t) - \eta_t \sum_{i=1}^{n} w_i I_{t,i} + \frac{\eta_t^2}{2} v_t(w)^\top H_{\text{val}}(\theta_t) v_t(w). \tag{11}$$

The Hessian correction yields a third-order Taylor remainder, compared with the second-order remainder of the approximation used in DRESS-FO. The quadratic term captures how target curvature affects the combined update, including interactions between different samples.

Differentiating the local model in Equation (11) with respect to w_i, taking the negative derivative, and removing the common factor η_t yields the second-order influence score

$$I^{(2)}_{t,i}(w) := I_{t,i} - \eta_t \left\langle H_{\text{val}}(\theta_t) v_t(w), P_t g_i(\theta_t) \right\rangle = \left\langle g_{\text{val}}(\theta_t) - \eta_t H_{\text{val}}(\theta_t) v_t(w), P_t g_i(\theta_t) \right\rangle. \tag{12}$$

Define the effective target direction

$$g^{\text{eff}}_{\text{val}}(\theta_t; w) = g_{\text{val}}(\theta_t) - \eta_t H_{\text{val}}(\theta_t) v_t(w). \tag{13}$$

The second-order score retains the same inner-product structure as the first-order score:

$$I^{(2)}_{t,i}(w) = \left\langle g^{\text{eff}}_{\text{val}}(\theta_t; w), P_t g_i(\theta_t) \right\rangle. \tag{14}$$

Replacing $I_{t,i}$ by $I^{(2)}_{t,i}(w_t)$ in Equation (10) gives a mirror-descent step for the local quadratic model.

4.4 Efficient Computation of the Second-Order Correction

Computing the exact Hessian required by the second-order influence score can be expensive. We therefore approximate the Hessian by a positive-semidefinite generalized Gauss–Newton (GGN) matrix (Martens & Grosse, 2015; Liu et al., 2024; Grosse et al., 2023).

For softmax cross-entropy loss, the GGN can be expressed as an expectation of gradient outer products. Let $g_{\text{val},m}(\theta_t)$ denote the gradient of the target loss for sample m, with its label drawn from the model predictive distribution at θ_t (Martens & Grosse, 2015). The resulting Monte Carlo estimate $G^{\text{GGN}}_{\text{val}}(\theta_t)$ is given by

$$H_{\text{val}}(\theta_t) \approx G^{\text{GGN}}_{\text{val}}(\theta_t) = \frac{1}{M} \sum_{m=1}^{M} g_{\text{val},m}(\theta_t) g_{\text{val},m}(\theta_t)^\top. \tag{15}$$

This representation yields the curvature-vector product

$$G^{\text{GGN}}_{\text{val}}(\theta_t) v_t(w) = \frac{1}{M} \sum_{m=1}^{M} g_{\text{val},m}(\theta_t) \left\langle g_{\text{val},m}(\theta_t), v_t(w) \right\rangle. \tag{16}$$

This product can be computed in two steps, without forming the GGN matrix or storing individual sample gradients. For target samples x_m, a Jacobian–vector product (JVP) computes each loss's directional derivative: $\rho_{t,m} = \langle g_{\text{val},m}(\theta_t), v_t(w) \rangle$ for $m = 1, \ldots, M$. Next, a vector–Jacobian product (VJP) computes the gradient of the weighted sum of the same losses, treating $\rho_{t,m}$ as constants: $\frac{1}{M} \sum_{m=1}^{M} \rho_{t,m} g_{\text{val},m}(\theta_t) = G^{\text{GGN}}_{\text{val}}(\theta_t) v_t(w)$.

Under the GGN approximation, the effective target direction in Equation (13) becomes

$$g^{\text{eff}}_{\text{val}}(\theta_t; w) \approx g_{\text{val}}(\theta_t) - \eta_t G^{\text{GGN}}_{\text{val}}(\theta_t) v_t(w). \tag{17}$$

At fixed reference weights, this product is computed once and reused to correct every candidate score.

Restricted parameter space. Computing influence scores over all LLM parameters can be expensive. For pretraining, we restrict the candidate gradients, target gradient, and curvature computation to the final prediction layer, or LM head, while treating the Transformer backbone as fixed during scoring. This follows scalable approximations used in data-influence methods (Pruthi et al., 2020; Koh & Liang, 2017). Besides, Huang et al. (2021) showed that gradients from the last layers are sufficiently informative; Ran et al. (2026) report that LM-head gradients dominate the full-gradient norm and that their inner products align closely with full-gradient inner products.

Algorithm 1 Descent-oriented Reweighting for Efficient Sample Selection (DRESS)

Require: Initial parameters θ_0, training data $\mathcal{D}_{\text{train}}$, and disjoint target data $\mathcal{D}_{\text{target}}$
Require: Candidate count n, GGN sample count M, learning rates η_t, and reweighting strengths α_t
$t \leftarrow 0$
while training budget remains **do**
  Draw $\mathcal{B}_t = \{x_i\}_{i=1}^n$ from $\mathcal{D}_{\text{train}}$ and sample target data from $\mathcal{D}_{\text{target}}$
  Estimate $g_{\text{val}}(\theta_t) = \nabla_\theta \mathcal{L}(\mathcal{D}_{\text{target}}; \theta_t)$ from the sampled target data
  if using DRESS-SO **then**
    $v_t(w_t) \leftarrow P_t \sum_{j=1}^n w_{t,j} g_j(\theta_t) = P_t \nabla_\theta \mathcal{L}(\mathcal{B}_t; \theta_t)$
    Sample M target samples from $\mathcal{D}_{\text{target}}$ and compute gradient $g_{\text{val},m}(\theta_t)$
    Compute $G_{\text{val}}^{\text{GGN}}(\theta_t) v_t(w_t) = \frac{1}{M} \sum_{m=1}^M g_{\text{val},m}(\theta_t) \langle g_{\text{val},m}(\theta_t), v_t(w_t) \rangle$
    $g_{\text{val}}^{\text{eff}}(\theta_t; w_t) \leftarrow g_{\text{val}}(\theta_t) - \eta_t G_{\text{val}}^{\text{GGN}}(\theta_t) v_t(w_t)$
    $I_{t,i}^{(2)}(w_t) \leftarrow \langle g_{\text{val}}^{\text{eff}}(\theta_t; w_t), P_t g_i(\theta_t) \rangle$ for all i
    Use $I_{t,i}^{(2)}(w_t)$ in place of $I_{t,i}$ in the weighting steps below
  else
    $I_{t,i} \leftarrow \langle g_{\text{val}}(\theta_t), P_t g_i(\theta_t) \rangle$ for all i
  end if
  $w_{t+1,i} \propto w_{t,i} \exp(\alpha_t I_{t,i})$ for all i
  Set $p_{t+1,i} \leftarrow w_{t+1,i}$; sample a subset of $\mathcal{B}_t$
  for each optimizer minibatch formed from the selected subset, while budget remains **do**
    Update θ_t to θ_{t+1} and the optimizer state using the usual training loss and η_t
    $t \leftarrow t + 1$
  end for
end while
return θ_t

5 Experiments

We evaluate DRESS in language-model pretraining and supervised fine-tuning. The pretraining experiments assess different methods at a fixed update-token budget across GPT-2 Small, Large, and XL, while the fine-tuning experiments evaluate mathematical reasoning with Qwen3.5-9B-Base. We compare first- and second-order scores and report the additional training time required for selection.

5.1 Experimental Setup

Models and training budgets. We pretrain models at the GPT-2 Small (124M), Large (774M), and XL (1.5B) scales (Radford et al., 2019) on FineWeb (Penedo et al., 2024). Each run performs updates on approximately 30B tokens. DRESS selects 50% of the candidate sequences, drawing these update tokens from approximately 60B candidate tokens. Following standard practice in frontier LLM training (DeepSeek-AI, 2026), we use Muon (Liu et al., 2025) to update Transformer-block matrices and AdamW (Loshchilov & Hutter, 2017) to update the embeddings, LM head, and remaining parameters. For supervised fine-tuning, we train Qwen3.5-9B-Base (Qwen Team, 2026) on 50k samples from NuminaMath-CoT (Li et al., 2024b) with LoRA (Hu et al., 2021) and AdamW. Full details are given in Sections D and E.

Evaluation and baselines. For pretraining, we evaluate models in a zero-shot setting on MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy and ARC-Challenge (Clark et al., 2018), and CommonsenseQA (Talmor et al., 2019). For GPT-2 Large and XL, we compare DRESS with uniform random selection (Random), PPL (Ankner et al., 2025), GREATS (Wang et al., 2024), QuRating (Wettig et al., 2024), DSIR (Xie et al., 2023b), DCLM-FastText (Li et al., 2024a), FineWeb-Edu (Penedo et al., 2024), UltraFineweb (Wang et al., 2025), and OPUS (Wang et al., 2026), all using 30B update tokens. Random (60B) serves as an additional reference with twice the update-token budget. We assess downstream transfer by fine-tuning GPT-2 Small checkpoints on eight GLUE tasks (Wang et al., 2018). For supervised fine-tuning, we compare DRESS-FO and DRESS-SO with standard fine-tuning, Random, and GREATS using Qwen3.5-9B-Base. We evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), and the text-only English mathematics subset of OlympiadBench (He et al., 2024). Further evaluation details are in Section F.

Table 1: **Zero-shot accuracy (%) after pretraining on FineWeb.** DRESS and other methods use 30B update tokens with Muon, with Random (60B) using twice the update-token budget. W.G. denotes WinoGrande and C.QA denotes CommonsenseQA. Bold and underlining mark the best and second-best values, respectively, among the 30B rows in each column of each model block.

Method	MMLU	HellaSwag	PIQA	SIQA	W.G.	ARC-E	ARC-C	C.QA	Avg.
GPT-2 Large on 30B update tokens									
Random	28.46	42.71	69.70	40.07	49.17	37.57	28.14	31.94	40.97
PPL (Ankner et al., 2025)	28.40	42.69	70.13	40.17	48.38	36.16	23.05	31.86	40.11
GREATS (Wang et al., 2024)	28.49	42.22	70.18	39.46	49.41	36.86	24.41	33.25	40.54
QuRating (Wettig et al., 2024)	**31.53**	39.47	66.38	39.82	50.59	**40.92**	**30.51**	30.22	41.18
DSIR (Xie et al., 2023b)	28.50	43.04	69.70	40.53	49.64	37.39	24.41	32.27	40.69
DCLM-FastText (Li et al., 2024a)	29.36	44.26	**71.16**	39.82	49.96	37.92	24.75	32.02	41.16
FineWeb-Edu (Penedo et al., 2024)	28.83	43.09	70.02	40.28	47.75	39.15	24.75	33.66	40.94
UltraFineweb (Wang et al., 2025)	29.00	**44.38**	71.11	40.17	48.78	37.57	25.08	**33.91**	41.25
OPUS (Wang et al., 2026)	28.76	42.92	69.97	39.56	50.43	38.98	29.15	33.09	41.61
DRESS-FO	27.30	43.78	69.97	**40.74**	**52.72**	37.74	26.78	32.68	41.46
DRESS-SO	27.90	44.05	69.31	40.17	52.33	40.39	25.76	33.33	**41.66**
Random (60B)	28.70	45.20	71.16	40.79	49.41	39.68	25.42	31.12	41.44
GPT-2 XL on 30B update tokens									
Random	28.73	48.01	70.46	39.61	47.91	38.98	25.42	33.25	41.55
PPL (Ankner et al., 2025)	29.35	47.87	71.55	40.69	45.86	38.45	24.07	30.38	41.03
GREATS (Wang et al., 2024)	29.95	42.26	70.18	39.61	47.67	36.33	23.73	30.55	40.04
QuRating (Wettig et al., 2024)	**33.28**	48.62	70.95	41.20	48.70	37.04	26.78	30.88	42.18
DSIR (Xie et al., 2023b)	29.58	48.49	71.93	39.51	47.59	38.10	26.44	32.68	41.79
DCLM-FastText (Li et al., 2024a)	30.40	44.07	71.38	**41.97**	48.38	38.80	**29.49**	30.88	41.92
FineWeb-Edu (Penedo et al., 2024)	29.66	48.45	71.71	41.25	46.17	39.19	28.14	31.29	41.98
UltraFineweb (Wang et al., 2025)	29.95	43.11	70.57	40.79	47.51	36.51	26.44	31.70	40.82
OPUS (Wang et al., 2026)	29.89	48.39	71.27	41.10	47.99	39.68	26.44	31.37	42.02
DRESS-FO	28.01	48.51	**72.96**	41.76	**54.14**	40.56	23.05	32.76	42.72
DRESS-SO	28.34	**48.75**	71.98	40.79	52.88	**41.62**	24.75	**34.40**	**42.94**
Random (60B)	30.24	51.10	72.25	40.89	48.78	41.98	23.05	32.35	42.58

5.2 Language-Model Pretraining

Main results. Table 1 compares pretraining methods on GPT-2 Large and XL at 30B update tokens, with Random (60B) as an additional reference. DRESS-SO achieves the highest average accuracy among the 30B methods at both scales, reaching 41.66% on GPT-2 Large and 42.94% on GPT-2 XL. DRESS-FO ranks third on Large and second on XL, achieving 41.46% and 42.72%, respectively. The advantage over competing methods is more pronounced on the larger GPT-2 XL model, where DRESS-FO and DRESS-SO outperform QuRating, the strongest competing baseline, by 0.54 and 0.76 percentage points, respectively. Both DRESS variants also outperform Random (60B) in average accuracy at both scales while using half as many update tokens.

Validation loss. Figure 3 compares GPT-2 XL validation loss over 25–30B update tokens. Both DRESS variants achieve lower final validation loss than the baseline (2.656) and Random (2.638), reaching 2.631 for DRESS-FO and 2.628 for DRESS-SO. DRESS-FO performs comparably to PPL (2.630), while DRESS-SO achieves the lowest final loss among the compared methods.

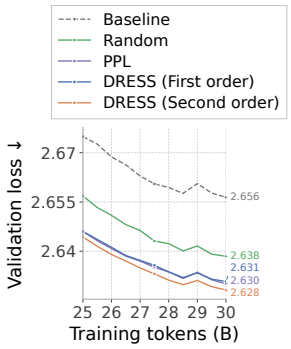


Figure 3: GPT-2 XL validation loss.

Full-gradient versus LM-head scoring. To evaluate the performance and computational cost of LM-head scoring, we compare DRESS-FO and DRESS-SO with a full-gradient first-order variant, DRESS-FO (FG), on GPT-2 Small (Table 2). The zero-shot average covers seven benchmarks, excluding ARC-C, on which all compared methods score below the random-guessing level. DRESS-FO (FG) achieves an average zero-shot accuracy of 39.40%, exceeding both the

Table 2: **Zero-shot accuracy (top) and GLUE scores (bottom) for GPT-2 Small (124M) pretrained on 30B update tokens of FineWeb.** DRESS-FO and DRESS-SO use LM-head gradients for scoring, while DRESS-FO (FG) uses full gradients. DRESS-SO achieves the highest average in both.

Zero-shot accuracy (%)

Method	MMLU	HellaSwag	PIQA	SIQA	W.G.	ARC-E	C.QA	Avg.
Baseline	25.23	30.38	63.17	36.69	50.75	35.10	29.16	38.64
DRESS-FO	**25.37**	31.25	63.76	38.18	**52.17**	34.92	28.58	39.18
DRESS-FO (FG)	25.29	31.38	64.58	**38.69**	51.14	35.80	28.91	39.40
DRESS-SO	25.33	**31.39**	**64.80**	38.02	52.01	**36.68**	**31.53**	**39.97**

GLUE after fine-tuning

Method	CoLA	SST-2	QNLI	QQP	MNLI-m	MRPC	RTE	STS-B	Avg.
Baseline	33.41	90.71	84.94	87.34	77.88	77.70	62.82	82.58	74.67
DRESS-FO	37.07	89.56	84.17	87.03	78.48	77.94	**70.04**	84.02	76.04
DRESS-FO (FG)	36.93	88.99	85.56	87.21	**79.40**	79.17	66.43	**85.95**	76.20
DRESS-SO	**38.20**	**90.94**	**86.27**	**87.59**	79.37	**80.39**	67.87	84.63	**76.91**

baseline (38.64%) and DRESS-FO (39.18%). DRESS-SO achieves the highest average accuracy of 39.97%, exceeding DRESS-FO (FG) by 0.57 percentage points despite using only LM-head gradients for scoring. The same ordering holds after fine-tuning on GLUE, where DRESS-SO achieves an average score of 76.91, exceeding DRESS-FO (FG) by 0.71 points.

For these GPT-2 Small runs, the training-time overheads are 8.3% for DRESS-FO, 12.8% for DRESS-SO, and 35.5% for DRESS-FO (FG). These results support second-order LM-head scoring as an effective alternative to first-order full-gradient scoring, achieving higher average zero-shot and GLUE scores with lower training-time overhead in this setting.

Computational cost. Table 3 summarizes the training-time overhead for the pretraining experiments in Tables 1 and 2, relative to the corresponding baseline at the same update-token budget. From GPT-2 Small to XL, the overhead decreases from 8.3% to 5.1% for DRESS-FO and from 12.8% to 12.1% for DRESS-SO. This trend is consistent with restricting gradient computations to the LM head, whose size grows more slowly than that of the Transformer backbone.

Table 3: **The training-time overhead (%) for GPT-2 pretraining.** Overhead is measured relative to the corresponding baseline at equally 30B update tokens.

Model	DRESS-FO	DRESS-SO
Small (124M)	8.3	12.8
Large (774M)	7.2	12.4
XL (1.5B)	5.1	12.1

5.3 Fine-Tuning

Table 4 reports mathematical reasoning accuracy at the evaluated checkpoint after fine-tuning on 50k NuminaMath-CoT samples. DRESS-SO achieves the highest accuracy on all three benchmarks and the highest average accuracy. DRESS-FO achieves the second-highest average accuracy. Compared with GREATS, DRESS-FO improves GSM8K and MATH-500 but scores 1.18 points lower on OlympiadBench.

5.4 Ablation Study

5.4.1 Effect of Reweighting Strength

We examine the reweighting strength α in DRESS-FO. Table 6 reports GPT-2 XL results on the same eight benchmarks as Table 1. As α increases from 0.5 to 1 and 2, average accuracy increases from 42.13% to 42.54% and 42.72%, respectively. Meanwhile, the effective sample size (ESS) decreases

Table 4: **Mathematical reasoning accuracy (%) for Qwen3.5-9B-Base after LoRA fine-tuning on 50k NuminaMath-CoT examples.** DRESS-SO achieves the highest accuracy on all three benchmarks.

Method	GSM8K	MATH-500	Olympiad Bench	Avg.
Qwen3.5-9B-Base	89.23	61.80	19.73	56.92
Baseline	90.90	68.20	27.30	62.13
Random	<u>91.21</u>	71.40	27.15	63.25
GREATS	91.05	69.80	<u>29.67</u>	63.51
DRESS-FO	91.13	<u>71.60</u>	28.49	<u>63.74</u>
DRESS-SO	**91.66**	**72.40**	**30.27**	**64.78**

Table 5: **Target-loss prediction fidelity on GPT-2 Small at the nominal update scale.** Isotropic curvature reduces RMSE relative to first order but worsens rank correlation, and remains substantially less accurate than GGN and the exact Hessian.

Predictor	RMSE median ↓	Spearman mean ↑
First order	8.44×10^{-7}	0.999318
Isotropic	2.53×10^{-7}	0.888548
GGN	1.60×10^{-9}	0.999969
Exact Hessian	$\mathbf{4.60 \times 10^{-10}}$	**1.000000**

Table 6: **GPT-2 XL zero-shot accuracy (%) and effective sample size for DRESS-FO at different reweighting strengths.** All checkpoints use 30B update tokens. The benchmarks match Table 1.

α	MMLU	HellaSwag	PIQA	SIQA	W.G.	ARC-E	ARC-C	C.QA	Avg.	ESS
0.5	**28.09**	**49.27**	71.55	40.23	53.20	39.15	23.39	32.19	42.13	12.18
1	27.87	48.85	72.52	40.23	53.04	39.15	**24.75**	**33.91**	42.54	6.84
2	28.01	48.51	**72.96**	**41.76**	**54.14**	**40.56**	23.05	32.76	**42.72**	3.08

from 12.18 to 6.84 and 3.08 out of 16 candidates, indicating that the sampling distribution is already highly concentrated at $\alpha = 2$. We therefore use $\alpha = 2$ as the default unless otherwise specified.

5.4.2 Is Second-Order Curvature Needed?

We compare predictors on identical LM-head updates in GPT-2 Small, with the backbone and input embeddings frozen. Each pool contains $K = 8$ random updates $\Delta\theta_k$ from the same checkpoint θ_t. The actual reduction is $\Delta_k = \mathcal{L}(\mathcal{D}; \theta_t) - \mathcal{L}(\mathcal{D}; \theta_t + \Delta\theta_k)$ and its prediction is $\widehat{\Delta}_k^{(C)} = -\langle g(\theta_t), \Delta\theta_k \rangle - \frac{1}{2}\Delta\theta_k^\top C \Delta\theta_k$, where $\mathcal{D}$ is the target dataset and $g(\theta_t)$ is the target gradient. Setting $C = 0$, cI, $G^{\mathrm{GGN}}(\theta_t)$, or $H(\theta_t)$ gives first order, *isotropic curvature*, analytic GGN, or exact Hessian, respectively. We fit $c = \arg\min_{u \geq 0} \sum_{k=1}^{K} (\widehat{\Delta}_k^{(uI)} - \Delta_k)^2$ once per checkpoint. Each pool's K predictions and actual reductions give $\mathrm{RMSE} = [K^{-1} \sum_{k=1}^{K} (\widehat{\Delta}_k^{(C)} - \Delta_k)^2]^{1/2}$ and Spearman correlation $\rho = \mathrm{Corr}(\mathrm{rank}(\widehat{\boldsymbol{\Delta}}^{(C)}), \mathrm{rank}(\boldsymbol{\Delta}))$. Table 5 shows that isotropic curvature reduces median RMSE from 8.44×10^{-7} to 2.53×10^{-7} but lowers mean Spearman from 0.999318 to 0.888548. GGN achieves a median RMSE of 1.60×10^{-9} and a mean Spearman correlation of 0.999969, supporting directional second-order curvature for more accurate local target-loss prediction.

6 Conclusion

We presented DRESS, an online data selection method that prioritizes training examples by their predicted contribution to reducing a held-out target loss. DRESS combines first- and second-order influence scores with mirror-descent reweighting. On GPT-2 Large and XL, both variants achieve higher average zero-shot accuracy than random training with twice the update-token budget. On Qwen3.5-9B-Base, second-order DRESS outperforms first-order DRESS, and both improve over the standard fine-tuning baseline on the evaluated reasoning benchmarks. On GPT-2 Small, second-order LM-head scoring achieves higher average zero-shot and GLUE performance than first-order full-gradient scoring with lower training-time overhead. Controlled LM-head experiments further show that GGN improves local target-loss prediction accuracy over isotropic curvature. These results support online selection guided by the evolving model and demonstrate the value of directional curvature for improving the fidelity of influence scores at a practical scoring cost.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, or interpret results. Generating synthetic datasets, assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, summarize or analyze existing literature, search for information, edit the paper to improve readability, and identify relevant literature. We have reviewed all AI-assisted work. Specifically, we checked the code for correctness and consistency with the methods described in the paper, verified the numerical values in the figures, read the identified papers to confirm their relevance to our work, and reviewed the revised text to ensure that it is accurate and faithfully conveys the authors' intended meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe the problem formulation, assumptions, and derivations of the first- and second-order influence scores in Sections 3 and 4, and summarize the DRESS procedure in Algorithm 1. The proof of the mirror-descent weight update and the autoregressive toy analysis, including its assumptions and proof, are provided in Sections A and B. Section 5.1 identifies the models, datasets, training budgets, and comparison methods. Sections D and E document optimization hyperparameters and implementation details for candidate scoring, target-data use, and subset selection. The zero-shot language-model and mathematical reasoning evaluation protocols, including evaluation tools and generation settings where applicable, are described in Section F.

REFERENCES

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max M Marion, Matthew Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models. In *International Conference on Learning Representations*, volume 2025, pp. 100615–100634, 2025.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. *Advances in Neural Information Processing Systems*, 34:677–689, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. Pmlr, 2017.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. *KR*, 2012(13th):3, 2012.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024a.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024b.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International conference on learning representations*, volume 2024, pp. 1621–1650, 2024.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4885–4901, 2020.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in neural information processing systems*, 33: 19920–19930, 2020.

Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, Zhaopan Xu, Zhou Daquan, Lei Shang, Baigui Sun, Xuansong Xie, et al. Infobatch: Lossless training speed up by unbiased dynamic data pruning. In *International Conference on Learning Representations*, volume 2024, pp. 39723–39744, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Yide Ran, Jianwen Xie, Minghui Wang, Wenjin Zheng, Denghui Zhang, Chuan Li, and Zhaozhuo Xu. Sketching the readout of large language models for scalable data attribution and valuation. *arXiv preprint arXiv:2604.16197*, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. 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. 4463–4473, 2019.

Daouda Sow, Herbert Woisetschläger, Saikiran Bulusu, Shiqiang Wang, Hans Arno Jacobsen, and Yingbin Liang. Dynamic loss-based sample reweighting for improved large language model pretraining. In *International Conference on Learning Representations*, volume 2025, pp. 89490–89520, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Jiachen T Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems*, 37:131197–131223, 2024.

Shaobo Wang, Xuan Ouyang, Tianyi Xu, Yuzheng Hu, Jialin Liu, Guo Chen, Tianyu Zhang, Junhao Zheng, Kexin Yang, Xingzhang Ren, et al. Opus: towards efficient and principled data selection in large language model pre-training in every iteration. *arXiv preprint arXiv:2602.05400*, 2026.

Yudong Wang, Zixuan Fu, Jie Cai, Peijun Tang, Hongya Lyu, Yewei Fang, Zhi Zheng, Jie Zhou, Guoyang Zeng, Chaojun Xiao, et al. Ultra-fineweb: Efficient data filtering and verification for high-quality llm training data. *arXiv preprint arXiv:2505.05427*, 2025.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. Qurating: Selecting high-quality data for training language models. *arXiv preprint arXiv:2402.09739*, 2024.

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

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

A MIRROR-DESCENT WEIGHT UPDATE

This appendix section gives a more detailed discussion of the constrained weight-update problem in Equation (6) and proves Lemma 1.

The *constrained* optimization problem in Equation (6) is

$$\min_{w} \ \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w)) \qquad \text{subject to} \qquad \sum_{i=1}^{n} w_i = 1, \quad w_i \geq 0, \tag{18}$$

i.e. minimizing the detached risk over $w \in \Delta_n$. Following Beck & Teboulle (2003), we solve Equation (6) with a *proximal step*, namely the mirror-descent step. At the current iterate w_t, we approximate the objective $\mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w))$ by its first-order Taylor expansion and choose the next iterate by trading off two objectives:

- moving as much as possible in the descent direction $-\nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t))$; and
- staying close to the current iterate w_t, so that the update remains inside the feasible set Δ_n and does not move too far in one step.

For a strongly convex distance-generating function (DGF) φ, the associated Bregman divergence is

$$D_\varphi(w \| w_t) = \varphi(w) - \varphi(w_t) - \langle \nabla\varphi(w_t), w - w_t \rangle. \tag{19}$$

The mirror-descent proximal step is therefore

$$\text{Prox}_\varphi\big(\alpha_t \nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t)),\, w_t\big) := \arg\min_{w \in \Delta_n} \left\{ \alpha_t \left\langle \nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t)),\, w \right\rangle + D_\varphi(w \| w_t) \right\}, \tag{20}$$

where the step size $\alpha_t > 0$ controls the trade-off between descent and proximity to the current weights. Since φ is strongly convex, $D_\varphi(\cdot \| w_t)$ is strongly convex. The proximal objective is therefore the sum of a linear term and a strongly convex term, which makes it strictly convex. Because the simplex Δ_n is compact and the objective is continuous on it, a minimizer exists by the Weierstrass extreme value theorem; by strict convexity, this minimizer is unique.

For the negative-entropy DGF $\varphi(w) = \sum_{i=1}^n w_i \log w_i$, the corresponding Bregman divergence is the forward KL divergence $D_\varphi(w \| w_t) = \sum_{i=1}^n w_i \log(w_i / w_{t,i})$. In this case, the proximal step over the simplex has the closed-form softmax update in Lemma 1, which we prove next.

Proof of Lemma 1. Write $G_t^{\text{toy}} = \nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t))$ for brevity, with $G_{t,i}^{\text{toy}} = [\nabla_w \mathcal{L}(\mathcal{D}_{\text{val}}; \theta(w_t))]_i$. With the negative-entropy DGF, the proximal objective in Equation (7) is

$$h(w) = \alpha_t \sum_{i=1}^{n} G_{t,i}^{\text{toy}} w_i + \sum_{i=1}^{n} w_i \log \frac{w_i}{w_{t,i}}, \qquad w \in \Delta_n. \tag{21}$$

Each summand $w_i \log(w_i / w_{t,i})$ is strictly convex, while the linear term $G_{t,i}^{\text{toy}} w_i$ does not affect convexity. Thus h is strictly convex on Δ_n and has a unique minimizer.

The minimizer is interior. As $w_i \to 0^+$, the partial derivative $\partial h / \partial w_i = \alpha_t G_{t,i}^{\text{toy}} + \log(w_i / w_{t,i}) + 1$ tends to $-\infty$. Hence, at any boundary point with $w_i = 0$, we can strictly decrease h by moving an infinitesimal amount of mass into coordinate i along a feasible edge. Therefore, no boundary point can be optimal, the constraints $w_i \geq 0$ are inactive at the optimum, and only the equality constraint $\sum_i w_i = 1$ is active.

Lagrangian stationarity. Let $\lambda \in \mathbb{R}$ be the Lagrange multiplier for $\sum_i w_i = 1$. The Lagrangian is

$$\mathcal{J}(w, \lambda) = \alpha_t \sum_{i=1}^{n} G_{t,i}^{\text{toy}} w_i + \sum_{i=1}^{n} w_i \log \frac{w_i}{w_{t,i}} + \lambda\Big(\sum_{i=1}^{n} w_i - 1\Big). \tag{22}$$

Setting $\partial \mathcal{J} / \partial w_i = 0$ gives, for every i,

$$\alpha_t G_{t,i}^{\text{toy}} + \log \frac{w_i}{w_{t,i}} + 1 + \lambda = 0 \quad \Longrightarrow \quad w_i = w_{t,i} \exp(-\alpha_t G_{t,i}^{\text{toy}}) \cdot \underbrace{\exp(-1 - \lambda)}_{\text{independent of } i}. \tag{23}$$

Imposing $\sum_i w_i = 1$ in Equation (23) gives $\exp(-1-\lambda) = \left(\sum_{\ell=1}^n w_{t,\ell} \exp(-\alpha_t G_{t,\ell}^{\text{toy}})\right)^{-1}$, and substituting this expression back yields

$$w_{t+1,i} = \frac{w_{t,i} \exp(-\alpha_t G_{t,i}^{\text{toy}})}{\sum_{\ell=1}^n w_{t,\ell} \exp(-\alpha_t G_{t,\ell}^{\text{toy}})} = \frac{w_{t,i} \exp(\alpha_t I_{t,i}^{\text{toy}})}{\sum_{\ell=1}^n w_{t,\ell} \exp(\alpha_t I_{t,\ell}^{\text{toy}})}. \tag{24}$$

This is the softmax update in Lemma 1, using $I_{t,i}^{\text{toy}} = -G_{t,i}^{\text{toy}}$. It shows that samples with the most positive influence $I_{t,i}^{\text{toy}}$ receive the largest upweighting relative to $w_{t,i}$. By strict convexity, this stationary point is the unique minimizer. □

B Autoregressive Toy Example

This section presents an autoregressive toy example to build intuition for the influence score. We proceed in three steps: formulate the *weighted* AR loss, derive the corresponding closed-form *weighted estimator*, and differentiate a *conditional-KL* target to obtain the influence score. The resulting score determines the mirror-descent weight update.

B.1 Gaussian Markov Next-Token Model

Assume scalar tokens in a sequence follow an AR(1) Gaussian Markov process. For sequence i and token position u, write the context and next token as

$$z_{i,u} = x_{i,u-1}, \qquad y_{i,u} = x_{i,u}, \tag{25}$$

and assume the real data distribution is

$$x_{i,u} = \phi_\star x_{i,u-1} + \epsilon_{i,u}, \qquad \epsilon_{i,u} \sim \mathcal{N}(0, \sigma_\star^2). \tag{26}$$

The model used to fit this distribution is

$$q_{\phi,\sigma^2}(y \mid z) = \mathcal{N}(y; \phi z, \sigma^2), \qquad \sigma^2 > 0. \tag{27}$$

The AR loss for each sample i is

$$\mathcal{L}^{\text{AR}}(x_i; \phi, \sigma^2) = \sum_u m_{i,u} \left[\frac{1}{2}\log(2\pi\sigma^2) + \frac{(y_{i,u} - \phi z_{i,u})^2}{2\sigma^2}\right], \tag{28}$$

where $m_{i,u} \in \{0, 1\}$ is a token mask.

B.2 Weighted AR Loss

Let $w \in \Delta_n$ be weights over a candidate batch $\mathcal{B}_t = \{x_i\}_{i=1}^n$ of n samples. Define the total weighted number of tokens and the normalized weighted transition statistics as

$$W(w) = \sum_{i=1}^n w_i \sum_u m_{i,u}, \qquad \bar{S}_{zz}(w) = \frac{1}{W(w)} \sum_{i=1}^n w_i \sum_u m_{i,u} z_{i,u}^2, \tag{29}$$

$$\bar{S}_{zy}(w) = \frac{1}{W(w)} \sum_{i=1}^n w_i \sum_u m_{i,u} z_{i,u} y_{i,u}, \qquad \bar{S}_{yy}(w) = \frac{1}{W(w)} \sum_{i=1}^n w_i \sum_u m_{i,u} y_{i,u}^2. \tag{30}$$

Here $\bar{S}_{ab}(w) = S_{ab}(w)/W(w)$ for $ab \in \{zz, zy, yy\}$, where $S_{ab}(w)$ denotes the unnormalized statistic. The weighted next-token objective is

$$\mathcal{L}^w(\mathcal{B}_t; \phi, \sigma^2) = \sum_{i=1}^n w_i \sum_u m_{i,u} \left[\frac{1}{2}\log(2\pi\sigma^2) + \frac{(y_{i,u} - \phi z_{i,u})^2}{2\sigma^2}\right], \tag{31}$$

and the weighted residual sum can be written as

$$\sum_{i=1}^n w_i \sum_u m_{i,u} (y_{i,u} - \phi z_{i,u})^2 = W(w) \left[\bar{S}_{yy}(w) - 2\phi \bar{S}_{zy}(w) + \phi^2 \bar{S}_{zz}(w)\right]. \tag{32}$$

Therefore, setting $\partial \mathcal{L}^w(\mathcal{B}_t;\phi,\sigma^2)/\partial\phi = 0$ and $\partial \mathcal{L}^w(\mathcal{B}_t;\phi,\sigma^2)/\partial\sigma^2 = 0$ gives the closed-form weighted estimator

$$\widehat{\phi}_t^w = \frac{\bar{S}_{zy}(w)}{\bar{S}_{zz}(w)}, \qquad \widehat{\sigma}_t^{2,w} = \bar{S}_{yy}(w) - \frac{\bar{S}_{zy}(w)^2}{\bar{S}_{zz}(w)}. \tag{33}$$

The slope is the weighted least-squares estimator, and the noise variance is the weighted residual variance. Equivalently, Equation (33) defines the plug-in Gaussian Markov model

$$q_t^w(x_{1:T}) = q_t^w(x_1) \prod_{u\geq 2} \mathcal{N}\big(x_u; \widehat{\phi}_t^w x_{u-1}, \widehat{\sigma}_t^{2,w}\big). \tag{34}$$

Thus, data reweighting affects the next model only through weighted sufficient statistics; here the sufficient statistics are the transition statistics $\big(\bar{S}_{zz}(w), \bar{S}_{zy}(w), \bar{S}_{yy}(w)\big)$.

B.3 Conditional-KL Target

We define the detached target transition distribution as

$$p_t^\dagger(y \mid z) = \mathcal{N}(\phi_t^\dagger z, \tau_t^2), \tag{35}$$

where $\phi_t^\dagger$ denotes the detached target slope. Define the moments over the context and next token as

$$T_{zz} = \mathbb{E}_\dagger[z^2], \quad T_{zy} = \mathbb{E}_\dagger[zy], \quad T_{yy} = \mathbb{E}_\dagger[y^2]. \tag{36}$$

The risk is the *conditional* forward KL between the detached target distribution and the plug-in conditional estimated distribution, averaged over the context distribution $\pi_t^\dagger$,

$$R_t(w) = \mathbb{E}_{z\sim\pi_t^\dagger}\left[D_{\mathrm{KL}}\Big(p_t^\dagger(\cdot \mid z) \,\Big\|\, q_t^w(\cdot \mid z)\Big)\right]. \tag{37}$$

For a fixed context z,

$$D_{\mathrm{KL}}\Big(\mathcal{N}(\phi_t^\dagger z, \tau_t^2) \,\Big\|\, \mathcal{N}(\widehat{\phi}_t^w z, \widehat{\sigma}_t^{2,w})\Big) = \log\frac{\widehat{\sigma}_t^w}{\tau_t} + \frac{\tau_t^2 + (\widehat{\phi}_t^w - \phi_t^\dagger)^2 z^2}{2\widehat{\sigma}_t^{2,w}} - \frac{1}{2}. \tag{38}$$

Taking the expectation over z gives

$$R_t(w) = \log\frac{\widehat{\sigma}_t^w}{\tau_t} + \frac{\tau_t^2 + (\widehat{\phi}_t^w - \phi_t^\dagger)^2 T_{zz}}{2\widehat{\sigma}_t^{2,w}} - \frac{1}{2}. \tag{39}$$

Minimizing $R_t(w)$ is therefore equivalent to maximizing the conditional likelihood under this Gaussian AR toy model.

B.4 Influence Score and Mirror-Descent Weight Update

We now compute the gradient of the risk and relate it to the influence score.

Theorem 1 (AR toy influence score)**.** *Fix the detached target moments T_{zz}, T_{zy}, T_{yy} and a current weight vector w_t. For the derivative below, index the weighted transition items in the batch by $i \in \{1,\dots,n\}$. Write the normalized weighted transition statistics as $\bar{S}_{zz}, \bar{S}_{zy}, \bar{S}_{yy}$. Define*

$$\widehat{\phi} = \frac{\bar{S}_{zy}}{\bar{S}_{zz}}, \qquad \widehat{\sigma}^2 = \bar{S}_{yy} - \frac{\bar{S}_{zy}^2}{\bar{S}_{zz}}, \qquad r_i = y_i - \widehat{\phi} z_i, \tag{40}$$

$$Q = T_{yy} - 2\widehat{\phi} T_{zy} + \widehat{\phi}^2 T_{zz}, \qquad A_\phi = \frac{\widehat{\phi} T_{zz} - T_{zy}}{\widehat{\sigma}^2}, \qquad A_\sigma = \frac{\widehat{\sigma}^2 - Q}{2\widehat{\sigma}^4}. \tag{41}$$

Then the derivative of $R_t(w)$ with respect to the weight on transition i is

$$G_{t,i}^{\mathrm{AR}} := \left.\frac{\partial R_t(w)}{\partial w_i}\right|_{w=w_t} = \frac{1}{W}\left[\frac{A_\phi}{\bar{S}_{zz}} z_i r_i + A_\sigma\big(r_i^2 - \widehat{\sigma}^2\big)\right], \tag{42}$$

where $W = \sum_{j=1}^n w_j$. Therefore, the first-order AR toy influence score, defined as the negative coefficient of transition i in the linearized detached risk, is

$$I_{t,i}^{\text{AR-toy}} = -\,G_{t,i}^{\mathrm{AR}}. \tag{43}$$

Proof. Because $R_t(w)$ depends on w only through the two estimators in Equation (33), we can compute its weight derivative by the chain rule. We now derive the local risk derivative $G_{t,i}^{\mathrm{AR}} := \partial R_t(w)/\partial w_i$ in detail.

Since R_t depends on w only through $\widehat{\phi} = \widehat{\phi}(w)$ and $\widehat{\sigma}^2 = \widehat{\sigma}^2(w)$,

$$\frac{\partial R_t}{\partial w_i} = \underbrace{\frac{\partial R_t}{\partial \widehat{\phi}}}_{=:A_\phi} \frac{\partial \widehat{\phi}}{\partial w_i} + \underbrace{\frac{\partial R_t}{\partial \widehat{\sigma}^2}}_{=:A_\sigma} \frac{\partial \widehat{\sigma}^2}{\partial w_i}. \tag{44}$$

We can write $R_t = \frac{1}{2}\log\widehat{\sigma}^2 + \frac{Q}{2\widehat{\sigma}^2} + \text{const}$ with $Q = T_{yy} - 2\widehat{\phi}T_{zy} + \widehat{\phi}^2 T_{zz}$, using $T_{zy} = \phi_t^\dagger T_{zz}$ and $T_{yy} = (\phi_t^\dagger)^2 T_{zz} + \tau_t^2$. Differentiating R_t with respect to $\widehat{\phi}$ and $\widehat{\sigma}^2$ gives

$$A_\phi = \frac{\widehat{\phi}T_{zz} - T_{zy}}{\widehat{\sigma}^2}, \qquad A_\sigma = \frac{1}{2\widehat{\sigma}^2} - \frac{Q}{2\widehat{\sigma}^4} = \frac{\widehat{\sigma}^2 - Q}{2\widehat{\sigma}^4}. \tag{45}$$

We write $\bar{S}_\bullet = \frac{1}{W}\sum_{j=1}^n w_j s_j$ to represent $\bar{S}_{zz}$, $\bar{S}_{zy}$, or $\bar{S}_{yy}$. Differentiating $\bar{S}_\bullet = \frac{1}{W}\sum_{j=1}^n w_j s_j$ with respect to w_i, and using $\partial W/\partial w_i = 1$, gives

$$\frac{\partial \bar{S}_\bullet}{\partial w_i} = \frac{s_i}{W} - \frac{1}{W^2}\sum_{j=1}^n w_j s_j = \frac{s_i - \bar{S}_\bullet}{W}, \tag{46}$$

so, with $s_i \in \{z_i^2,\, z_i y_i,\, y_i^2\}$,

$$\frac{\partial \bar{S}_{zz}}{\partial w_i} = \frac{z_i^2 - \bar{S}_{zz}}{W}, \qquad \frac{\partial \bar{S}_{zy}}{\partial w_i} = \frac{z_i y_i - \bar{S}_{zy}}{W}, \qquad \frac{\partial \bar{S}_{yy}}{\partial w_i} = \frac{y_i^2 - \bar{S}_{yy}}{W}. \tag{47}$$

For the slope $\widehat{\phi} = \bar{S}_{zy}/\bar{S}_{zz}$, the gradient is

$$\frac{\partial \widehat{\phi}}{\partial w_i} = \frac{1}{\bar{S}_{zz}}\frac{\partial \bar{S}_{zy}}{\partial w_i} - \frac{\bar{S}_{zy}}{\bar{S}_{zz}^2}\frac{\partial \bar{S}_{zz}}{\partial w_i} = \frac{1}{W\bar{S}_{zz}}\Big[(z_i y_i - \bar{S}_{zy}) - \widehat{\phi}\,(z_i^2 - \bar{S}_{zz})\Big] = \frac{z_i\, r_i}{W\bar{S}_{zz}}, \tag{48}$$

where the $\bar{S}_{zy}$ and $\widehat{\phi}\bar{S}_{zz}$ terms cancel. This leaves only $z_i y_i - \widehat{\phi} z_i^2 = z_i r_i$, with residual $r_i = y_i - \widehat{\phi} z_i$. For the variance $\widehat{\sigma}^2 = \bar{S}_{yy} - \bar{S}_{zy}^2/\bar{S}_{zz}$, the derivative is

$$\begin{aligned}\frac{\partial \widehat{\sigma}^2}{\partial w_i} &= \frac{\partial \bar{S}_{yy}}{\partial w_i} - \frac{2\bar{S}_{zy}}{\bar{S}_{zz}}\frac{\partial \bar{S}_{zy}}{\partial w_i} + \frac{\bar{S}_{zy}^2}{\bar{S}_{zz}^2}\frac{\partial \bar{S}_{zz}}{\partial w_i} \\ &= \frac{1}{W}\Big[(y_i^2 - \bar{S}_{yy}) - 2\widehat{\phi}(z_i y_i - \bar{S}_{zy}) + \widehat{\phi}^2(z_i^2 - \bar{S}_{zz})\Big] = \frac{r_i^2 - \widehat{\sigma}^2}{W},\end{aligned} \tag{49}$$

because $\widehat{\sigma}^2 = \bar{S}_{yy} - \widehat{\phi}^2\bar{S}_{zz} = \bar{S}_{yy} - 2\widehat{\phi}\bar{S}_{zy} + \widehat{\phi}^2\bar{S}_{zz}$. The slope derivative depends on transition i through $z_i r_i$, whereas the variance derivative depends on it through r_i^2.

Substituting Equation (48) and Equation (49) into Equation (44) gives the local risk derivative

$$G_{t,i}^{\mathrm{AR}} := \frac{\partial R_t(w)}{\partial w_i}\bigg|_{w=w_t} = \frac{1}{W}\left[\frac{A_\phi}{\bar{S}_{zz}}\, z_i r_i + A_\sigma\big(r_i^2 - \widehat{\sigma}^2\big)\right], \tag{50}$$

and the toy influence score is the negative derivative

$$I_{t,i}^{\text{AR-toy}} = -\,G_{t,i}^{\mathrm{AR}}. \tag{51}$$

Indeed, for any feasible first-order perturbation δw of the simplex, with $\sum_{i=1}^n \delta w_i = 0$,

$$R_t(w_t + \epsilon\delta w) = R_t(w_t) + \epsilon\sum_{i=1}^n \delta w_i G_{t,i}^{\mathrm{AR}} + o(\epsilon), \tag{52}$$

so transition i enters the linearized detached conditional-KL risk with coefficient $G_{t,i}^{\mathrm{AR}}$. Hence the corresponding local influence coefficient is $I_{t,i}^{\text{AR-toy}} = -G_{t,i}^{\mathrm{AR}}$. □

We obtain a mirror-descent update for the constrained risk-minimization problem by solving the following KL-regularized linearized subproblem

$$w_{t+1} = \arg\min_{\bar{w}\in\Delta_n} \left\{ \sum_{i=1}^{n} \bar{w}_i \, G_{t,i}^{\mathrm{AR}} + \frac{1}{\alpha_t} D_{\mathrm{KL}}(\bar{w} \,\|\, w_t) \right\}. \tag{53}$$

Following Lemma 1, we can find the solution in closed form. Using $I_{t,i}^{\text{AR-toy}} = -G_{t,i}^{\mathrm{AR}}$, the exponentiated-gradient update is

$$w_{t+1,i} = \frac{w_{t,i} \exp\big(\alpha_t I_{t,i}^{\text{AR-toy}}\big)}{\sum_{\ell=1}^{n} w_{t,\ell} \exp\big(\alpha_t I_{t,\ell}^{\text{AR-toy}}\big)}. \tag{54}$$

C Optimizer Preconditioners Used for Scoring

This section derives the scoring maps from the AdamW and Muon updates, following OPUS (Wang et al., 2026). At parameters θ_t, let $g_i(\theta_t) = \nabla_\theta \mathcal{L}(x_i; \theta_t)$ and $g_t(S) = |S|^{-1} \sum_{i\in S} g_i(\theta_t)$ be the mean gradient of a candidate subset S; a subscript L denotes restriction to parameter block L. We write $\Delta\theta_t(S) = \theta_{t+1}(S) - \theta_t$ for the parameter change, and similarly $\Delta W_{t,L}(S)$ for a matrix block. Optimizer hyperparameters and stored history are fixed while comparing candidates. We write $\mathcal{P}_t$ for the resulting linear map including the learning rate, and use the local scoring update

$$\Delta\theta_t^{\mathrm{score}}(w) = -\mathcal{P}_t \sum_i w_i g_i(\theta_t) = -\eta_t P_t \sum_i w_i g_i(\theta_t), \qquad w_i \geq 0, \quad \sum_i w_i = 1. \tag{55}$$

Here $\eta_t > 0$ is the common learning-rate factor written separately in the main text, so $P_t = \mathcal{P}_t/\eta_t$.

C.1 AdamW

Optimizer update. Let $m_{t-1,L}$ and $v_{t-1,L}$ be the first- and second-moment buffers before the candidate update. AdamW uses decay coefficients $\beta_1, \beta_2 \in [0, 1)$, a numerical stabilizer $\epsilon > 0$, weight decay λ, and effective learning rate $\eta_{t,L}^{\mathrm{eff}}$. Its update is

$$\begin{aligned}
m_{t,L}(S) &= \beta_1 m_{t-1,L} + (1-\beta_1) g_{t,L}(S), \\
v_{t,L}(S) &= \beta_2 v_{t-1,L} + (1-\beta_2) g_{t,L}(S)^{\odot 2}, \\
\widehat{m}_{t,L}(S) &= \frac{m_{t,L}(S)}{1-\beta_1^t}, \qquad \widehat{v}_{t,L}(S) = \frac{v_{t,L}(S)}{1-\beta_2^t}, \\
\Delta\theta_{t,L}(S) &= -\eta_{t,L}^{\mathrm{eff}} \frac{\widehat{m}_{t,L}(S)}{\sqrt{\widehat{v}_{t,L}(S)} + \epsilon} - \eta_{t,L}^{\mathrm{eff}} \lambda \theta_{t,L}.
\end{aligned} \tag{56}$$

Here AdamW blocks are represented in vectorized coordinates, and $\odot 2$ are elementwise. The effective learning rate includes the parameter-group rate and any parameter-specific multiplier.

Frozen denominator. For scoring, OPUS freezes the RMS denominator across candidates. Define

$$D_{t,L} = \mathrm{diag}\left[\frac{1}{\sqrt{\widehat{v}_{t-1,L}} + \epsilon} \right], \qquad C_{t,L} = \eta_{t,L}^{\mathrm{eff}} \frac{1-\beta_1}{1-\beta_1^t}. \tag{57}$$

Substituting the first-moment update gives the affine approximation

$$\begin{aligned}
\Delta\theta_{t,L}(S) &\approx r_{t,L}^{\mathrm{AdamW}} - C_{t,L} D_{t,L} g_{t,L}(S), \\
r_{t,L}^{\mathrm{AdamW}} &= -\eta_{t,L}^{\mathrm{eff}} \frac{\beta_1}{1-\beta_1^t} D_{t,L} m_{t-1,L} - \eta_{t,L}^{\mathrm{eff}} \lambda \theta_{t,L}.
\end{aligned} \tag{58}$$

The offset $r_{t,L}^{\mathrm{AdamW}}$ is independent of S.

C.2 MUON

Optimizer update. For a matrix parameter $W_L \in \mathbb{R}^{o_L \times i_L}$, let $m_{t-1,L}$ be the stored momentum and $\mu_t \in [0, 1)$ its coefficient. With indices aligned to the AdamW update above,

$$\begin{aligned} m_{t,L}(S) &= \mu_t m_{t-1,L} + (1-\mu_t) g_{t,L}(S), \\ q_{t,L}(S) &= (1-\mu_t) g_{t,L}(S) + \mu_t m_{t,L}(S) \\ &= \mu_t^2 m_{t-1,L} + (1-\mu_t^2) g_{t,L}(S), \\ \Delta W_{t,L}(S) &= -\eta_{t,L}^{\mathrm{eff}} \mathcal{O}(q_{t,L}(S)) \,. \end{aligned} \tag{59}$$

The nonlinear map $\mathcal{O}$ applies five Newton–Schulz iterations.

Frozen Newton–Schulz map. Following OPUS, scoring replaces $\mathcal{O}$ by a fixed left-multiplication map. The implementation uses

$$\begin{aligned} \widetilde{q}_{t,L} &= \frac{q_{t-1,L}}{\|q_{t-1,L}\|_F}, \qquad A_{t,L} = \widetilde{q}_{t,L} \widetilde{q}_{t,L}^{\top}, \\ S_{t,L} &= aI + bA_{t,L} + cA_{t,L}^2, \qquad \mathcal{O}(Z) \approx S_{t,L} Z. \end{aligned} \tag{60}$$

Substituting the expression for $q_{t,L}(S)$ gives

$$\begin{aligned} \Delta W_{t,L}(S) &\approx r_{t,L}^{\mathrm{Muon}} - \kappa_{t,L} S_{t,L} g_{t,L}(S), \\ r_{t,L}^{\mathrm{Muon}} &= -\eta_{t,L}^{\mathrm{eff}} \mu_t^2 S_{t,L} m_{t-1,L}, \qquad \kappa_{t,L} = \eta_{t,L}^{\mathrm{eff}} (1-\mu_t^2). \end{aligned} \tag{61}$$

Again, the offset is independent of S. The resulting scoring map acts on a matrix gradient G as

$$\mathcal{P}_{t,L}^{\mathrm{Muon}}[G] = \kappa_{t,L} S_{t,L} G. \tag{62}$$

D PRETRAINING IMPLEMENTATION AND HYPERPARAMETERS

Optimization. Table 7 gives the model-specific settings. AdamW uses $(\beta_1, \beta_2) = (0.8, 0.95)$ and $\epsilon = 10^{-8}$. Weight decay is zero for both optimizers. Learning rates follow cosine decay to 10% of their peak values, with no learning-rate warmup. Muon momentum increases linearly from 0.85 to 0.95 over the first 300 updates. We clip the global gradient norm at 1.0. Cross-entropy computation is chunked in blocks of 4,096 positions to reduce memory use.

Table 7: **Pretraining hyperparameters.**

Setting	Small	Large	XL
Transformer layers	12	36	48
Hidden dimension	768	1,280	1,600
Attention heads	12	20	25
Training-stream length per rank	49,152	24,576	6,144
Validation-stream length per rank	65,536	32,768	8,192
Number of GPUs	8	8	8
Global update tokens per step	393,216	196,608	49,152
AdamW learning rate	8×10^{-3}	4×10^{-3}	2×10^{-3}
Muon learning rate	5×10^{-2}	1.5×10^{-2}	10^{-2}

Target-set construction and domain composition. We construct $\mathcal{D}_{\mathrm{target}}$ once from the same FineWeb English web corpus used for evaluation. The target set contains about 30M tokens, which we partition into 4,096-token sequences, discarding the incomplete final sequence. The target therefore contains general web text without explicit topic balancing or domain reweighting. Target documents are excluded from both the candidate training stream and the language-model validation stream, and downstream benchmark examples are not used to construct the target. The target set remains fixed throughout training. Each selection call draws the next batch of 8 sequences.

Table 8: **Shared pretraining selection hyperparameters for GPT-2 Small, Large, and XL.**

Setting	Value
Selection fraction	0.50
Candidate scoring windows	1
Tokens per candidate window	512
Target set size	30M
Global selection-target batch	8
Loaded target sequence length	4,096
Scored target sequence length	2,048
Target-batch refresh interval	1

Candidate pools and score computation. Each rank draws 16 candidate sequences and retains 8, giving a 50% selection ratio and 64 retained sequences from 128 candidates across eight GPUs. A candidate score is computed on one randomly sampled 512-token window from its sequence. Target batches contain 8 sequences globally, distributed across ranks; each target sequence is loaded at length 4,096 and scored on its first 2,048 tokens. Target statistics and the scoring direction are refreshed for each new candidate pool.

Preconditioning and curvature. Candidate gradients, target gradients, and curvature products are restricted to the LM head; the backbone is held fixed for scoring and updated during ordinary training. The scoring preconditioner uses the current AdamW state as a frozen local approximation. For second-order pretraining, the GGN-vector product is computed analytically in LM-head coordinates, evaluating the expectation underlying Equation (15). The correction uses the aggregate candidate direction under uniform reference weights and is shared across the candidate scores. Neither a full curvature matrix nor all pairwise candidate interactions are materialized.

Evaluation and baselines. The full zero-shot evaluation suite comprises the eight benchmarks reported in Table 1, together with ANLI (Nie et al., 2020) and WSC (Levesque et al., 2012). Under this protocol, all compared methods score below the 50% random-guessing baseline on WSC, which has two answer choices. Most ANLI results also fall below its 33.33% random-guessing baseline. We therefore exclude these two benchmarks from the GPT-2 Large/XL table and its reported average.

For the GPT-2 Large and XL comparisons, we use the same training hyperparameters and evaluation protocol as Wang et al. (2026). We therefore use the baseline benchmark results reported by Wang et al. (2026) for the competing baseline methods listed in Table 1.

E Fine-Tuning Implementation and Hyperparameters

Data and exposure budget. We use MS-SWIFT to fine-tune Qwen3.5-9B-Base with LoRA on a shuffled sample of 50k NuminaMath-CoT examples. We reserve 2048 examples for language-model validation. Random selection and both DRESS variants additionally reserve 1K target examples. This target is distinct from the validation split. The selection variants draw 16 candidates per rank and retain four, giving 64 global candidates per update.

Training and selection. Table 9 lists the optimization settings. We use the Qwen template with the non-thinking prefix and the `ignore_empty_think` loss scale. Scores use all trainable LoRA A/B coordinates and the full supervised sequence up to the training length limit, with dropout disabled during scoring. Selection is active throughout training, without an initial annealing phase. The random control executes the first-order scoring path and then draws a uniform subset, controlling for the candidate stream, target split, and scoring work when compared with DRESS-FO. The ordinary baseline uses the standard training loader without candidate scoring.

Table 9: **Qwen3.5-9B-Base LoRA hyperparameters.**

Setting	Value
Optimizer	AdamW
Peak learning rate	10^{-4}
AdamW (β_1,β_2)	(0.9,0.95)
AdamW ϵ	10^{-8}
Weight decay	0.1
Gradient clipping	1.0
Learning-rate schedule	Cosine
Warmup ratio	10%
LoRA rank	8
LoRA scaling parameter	32
LoRA dropout	0.05
Training sequence length	2,048 tokens
Global training batch	16
Global candidate batch	64
Global selected batch	16
Global selection-target batch	16
Selection fraction	0.25
Target-set size	1,000
Curvature samples M	16
Target-batch refresh interval	1
Candidate examples per rank	16

F Evaluation Protocols

F.1 Zero-Shot Language-Model Evaluation

MMLU uses lighteval, and the other nine benchmarks use OpenCompass. Evaluation is zero-shot. MMLU and WinoGrande rank choices by log-likelihood; the remaining benchmarks use multiple-choice perplexity scoring.

F.2 Mathematical Reasoning Evaluation

We evaluate with EvalScope 1.11.0 on the full GSM8K test set (1,319 questions), all 500 MATH-500 questions, and all 674 questions in OlympiadBench's `OE_TO_maths_en_COMP` subset. This OlympiadBench subset contains text-only English mathematics problems; multimodal and theorem-proving subsets are outside this evaluation. GSM8K uses zero-shot prompting, and the other tasks use their EvalScope benchmark templates. Generation is deterministic, with temperature 0, sampling disabled, and at most 768 generated tokens. We use the benchmark-specific answer extraction and correctness checks supplied by EvalScope.

G Directional Computation

The influence score has an equivalent directional-derivative form. Expanding Equation (5) gives

$$I_{t,i} = g_i(\theta_t)^\top P_t^\top g_{\mathrm{val}}(\theta_t) = \left[\nabla_\theta \mathcal{L}(x_i;\theta_t)\right]^\top P_t^\top g_{\mathrm{val}}(\theta_t). \tag{63}$$

Equivalently,

$$I_{t,i} = \frac{d}{d\epsilon}\mathcal{L}(x_i;\theta_t + \epsilon P_t^\top g_{\mathrm{val}}(\theta_t))\Big|_{\epsilon=0}. \tag{64}$$

This formulation allows each candidate's score to be computed as a directional derivative along the shared direction $P_t^\top g_{\mathrm{val}}(\theta_t)$, avoiding the need to explicitly form individual candidate gradients.