LEARNING TO SHUFFLE: BLOCK RESHUFFLING AND REVERSAL SCHEMES FOR STOCHASTIC OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Shuffling strategies for stochastic gradient descent (SGD), including incremental gradient, shuffle-once, and random reshuffling, are supported by rigorous convergence analyses for arbitrary within-epoch permutations. In particular, random reshuffling is known to improve optimization constants relative to cyclic and shuffle-once schemes. However, existing theory offers limited guidance on how to design new data-ordering schemes that further improve optimization constants or stability beyond random reshuffling. In this paper, we design a pipeline using a large language model (LLM)-guided program evolution framework to discover an effective shuffling rule for without-replacement SGD. Abstracting from this instance, we identify two fundamental structural components: block reshuffling and paired reversal. We analyze these components separately and show that block reshuffling strictly reduces prefix-gradient variance constants within the unified shuffling framework, yielding provable improvements over random reshuffling under mild conditions. Separately, we show that paired reversal symmetrizes the epoch map and cancels the leading order-dependent second-order term, reducing order sensitivity from quadratic to cubic in the step size. Numerical experiments with the discovered algorithm validate the theory and demonstrate consistent gains over standard shuffling schemes across convex and nonconvex benchmarks.

1 INTRODUCTION

Stochastic gradient descent (SGD) remains a fundamental optimization method for large-scale machine learning (Nemirovski et al., 2009; Ghadimi & Lan, 2013; Bottou et al., 2018). In practice, SGD is almost always implemented in an epoch-based, without replacement manner, where the training data are processed according to a permutation at the beginning of each epoch. Such shuffling strategies are standard in modern machine learning frameworks, including TensorFlow, PyTorch, and Keras (Abadi et al., 2015; Paszke et al., 2019; Chollet et al., 2015). Classical approaches include incremental gradient, shuffle-once, and random reshuffling, whose convergence properties are now well understood under standard smoothness assumptions (Nedic & Bertsekas, 2001). Incremental gradient (IG) uses a fixed data order, shuffle-once (SO) samples one random order and reuses it across epochs, and random reshuffling (RR) draws a new random permutation at each epoch. In particular, recent work shows that RR often enjoys improved optimization constants compared to IG and SO (Mishchenko et al., 2020), and unified shuffling analyses establish convergence guarantees for arbitrary within-epoch permutations (Nguyen et al., 2021).

Despite these advances, existing theory largely treats permutations as unstructured objects. As a result, it provides limited guidance on how to design new data-ordering schemes that further improve optimization constants or enhance stability beyond RR. In this work, we adopt a different perspective. Rather than proposing a shuffling rule solely from analytical intuition, we construct a pipeline based on large language model (LLM)-guided program evolution to explore the space of valid without replacement shuffling rules. This process yields a concrete adaptive reshuffling algorithm, which we formalize as *Adaptive Block Reshuffling with Periodic Transforms* (APR) in Algorithm 1. APR consistently outperforms standard shuffling schemes across a range of benchmarks. The LLM is used only offline as a discovery tool and does not participate in training or inference.

Motivated by the structure of the discovered algorithm, we abstract away from its specific parameters and isolate two recurring components: *block reshuffling* and *paired reversal*. We analyze these

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.

components separately. Within the unified shuffling framework, we show that block reshuffling strictly reduces prefix-gradient variance constants under mild conditions, yielding provable improvements over the existing shuffling schemes. Separately, we show that paired reversal symmetrizes the epoch map and cancels the leading order-dependent second-order term, reducing order sensitivity from quadratic to cubic in the learning rate. Our experiments, conducted using the LLM-discovered APR algorithm, validate these theoretical predictions and demonstrate consistent improvements over IG, SO, and RR on both convex and non-convex problems. Together, these results suggest that structured data-ordering schemes beyond random reshuffling can offer meaningful optimization benefits, and that automated discovery can help reveal such structure.

Contributions. Our main contributions are:

- We introduce an LLM-guided program evolution pipeline to discover effective shuffling rules for without-replacement SGD, resulting in the APR algorithm.
- We abstract the discovered algorithm into two core components, *block reshuffling* and *paired reversal*, and study them independently. To the best of our knowledge, these components have not been previously formalized as independent structures within a unified theory for without replacement SGD.
- We show that block reshuffling reduces prefix-gradient variance constants within the unified shuffling framework, yielding provable improvements over existing schemes.
- We prove that paired reversal cancels the leading order-dependent second-order term in the epoch map, reducing order sensitivity from $\Theta(\gamma^2)$ to $O(\gamma^3)$ (in learning rate γ).
- We demonstrate empirically that APR improves performance over standard shuffling schemes across convex and nonconvex benchmarks.

1.1 Related Work

Without-replacement SGD. The convergence properties of classical strategies (IG, SO, RR) and arbitrary permutations have been extensively analyzed in finite-sum optimization (Gürbüzbalaban et al., 2019; Haochen & Sra, 2019; Safran & Shamir, 2020; Nagaraj et al., 2019; Rajput et al., 2020; Nguyen et al., 2021; Mishchenko et al., 2020; Ahn et al., 2020). However, these analyses generally treat permutations as unstructured and do not provide constructive principles for designing improved schemes.

Variance and order sensitivity reductions. Classical variance reduction methods (e.g., SAG (Le Roux et al., 2012), SAGA (Defazio et al., 2014), SVRG (Johnson & Zhang, 2013), SARAH (Nguyen et al., 2017)) require storing gradient information or computing periodic full gradients. In contrast, our block reshuffling mechanism improves variance constants purely through data ordering, without extra memory or computational overhead. Additionally, our paired reversal technique addresses a complementary issue, permutation-induced variability, by symmetrizing the epoch update for greater stability.

LLM-guided algorithm discovery. Recent work shows LLMs can assist in discovering algorithms via program synthesis (Romera-Paredes et al., 2024; Novikov et al., 2025). Unlike approaches that deploy LLMs during training or inference, we use the LLM purely as an offline tool. The resulting APR algorithm is deterministic, and our theoretical guarantees apply independently of the LLM.

2 Problem Setup and Background

We consider the finite-sum optimization problem

$$\min_{w\in\mathbb{R}^d} F(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w). \tag{1}$$

We study epoch-based without replacement stochastic gradient methods. At the beginning of each epoch e, a permutation π_e of $\{1,\dots,n\}$ is generated, and the iterates are updated sequentially

as $w_t = w_{t-1} - \gamma \nabla f_{\pi_e(t)}(w_{t-1})$, $t = 1, \ldots, n$. Let T_π denote the epoch map corresponding to permutation π, i.e., $T_\pi(w)$ is the iterate after one full epoch starting from w. Recent analyses characterize convergence in terms of permutation sensitive quantities such as prefix-gradient sums and associated variance constants. These quantities will play a central role in our analysis of block reshuffling (Mishchenko et al., 2020). The unified analyses show that convergence guarantees hold for any permutation selected at each epoch (Nguyen et al., 2021). Consequently, the relevant algorithmic question is no longer whether a permutation converges, but rather how the structure of the permutation affects optimization efficiency.

2.1 LLM-Guided Discovery of Shuffling Rules

We use OpenEvolve (Sharma, 2025), an LLM-guided program evolution framework, to discover effective reshuffling rules for without-replacement SGD. The search space was restricted to deterministic Python functions producing permutations of $\{1, \ldots, n\}$ (see Appendix D for more detail). Guided by empirical training feedback such as loss statistics, the LLM proposes concrete shuffling programs with fixed parameter choices and was prohibited from modifying the model, optimizer, or learning rate. One such discovered instance consistently outperformed standard shuffling schemes in preliminary experiments. The detailed settings are in Appendix E.

Motivated by the LLM-discovered instance, we generalize its behavior into Algorithm 1, which generates a single permutation $\pi_e \in \mathcal{S}_n$ at each epoch e, where $\mathcal{S}_n$ denotes the symmetric group on n elements. The permutation is selected based on the epoch-level training loss ℓ_e through the loss ratio $\rho_e = \ell_e/(\ell_{e-1} + \varepsilon)$, which determines the reshuffling regime. Block sizes are specified by fractions α_{strong} and α_{mild}, yielding $b_{\text{strong}} = \max(1, \lfloor \alpha_{\text{strong}} n \rfloor)$ and $b_{\text{mild}} = \max(1, \lfloor \alpha_{\text{mild}} n \rfloor)$. The algorithm optionally applies simple sequence transforms, including reversal $\text{Rev}(x_1, \ldots, x_n) = (x_n, \ldots, x_1)$ and even-odd interleaving $\text{EO}(x) = (x_1, x_3, \ldots; x_2, x_4, \ldots)$. All randomness is governed by a deterministic epoch dependent seed u_e. Importantly, Algorithm 1 always outputs a valid permutation and therefore remains within the standard without-replacement SGD framework, making it compatible with unified shuffling convergence analyses. A detailed description of this procedure is provided in Appendix E.

Inspection of the LLM-generated code and its generalized form reveals two recurring structural patterns: reshuffling data in contiguous blocks and reversing the data order to mitigate systematic ordering effects. Importantly, Algorithm 1 generates a single valid permutation at each epoch, and therefore remains fully within the standard without replacement SGD framework. As a result, its convergence behavior can be analyzed using existing unified shuffling theory, which applies to arbitrary (possibly data dependent) permutations. Rather than analyzing the full adaptive logic of Algorithm 1, we isolate these two components for theoretical study. Block reshuffling partitions the dataset into consecutive blocks and randomly permutes the blocks while preserving the order within each block. This operation lies within the unified shuffling framework and is shown to strictly reduce prefix-gradient variance constants relative to standard schemes, including random reshuffling, under mild conditions. While the discovered instance applies one sided reversal heuristically, we analyze a paired reversal operator that averages the effects of a permutation and its reverse. This symmetrization cancels the leading order-dependent second-order term in the epoch map expansion, reducing order sensitivity from quadratic to cubic in the learning rate.

2.2 Block Reshuffling: Variance Reduction

Block reshuffling partitions the index set $\{1, \ldots, n\}$ into consecutive blocks of size b, randomly permutes the blocks, and processes the blocks sequentially while preserving the order within each block. Let $I = (1, 2, \ldots, n)$ and partition I into blocks $\mathcal{B}(I; b) = (B_1, \ldots, B_K)$ and $B_k = \big((k-1)b+1, \ldots, \min(kb, n)\big)$, where $K = \lceil n/b \rceil$. Given a random permutation σ of $\{1, \ldots, K\}$, the block-reshuffled order is defined as $\text{BlockShuffle}(I; b, \sigma) = (B_{\sigma(1)}, \ldots, B_{\sigma(K)})$. Block reshuffling interpolates between classical schemes: (1) $b = 1$ corresponds to random reshuffling (RR); and (2) $b = n$ corresponds to incremental gradient (IG) methods.

From a statistical perspective, block reshuffling can be viewed as sampling block-averaged gradients without replacement rather than individual gradients. Let $G_k(w)$ denote the average gradient within block k. A standard variance decomposition yields $\sigma^2_{\text{ind}}(w) = \sigma^2_{\text{within}}(w) + \sigma^2_{\text{blk}}(w)$, where $\sigma^2_{\text{blk}}(w)$ is the variance of block-averaged gradients. Whenever gradients within blocks are non-identical,

Algorithm 1 Adaptive Block Reshuffling with Periodic Transforms (APR)

Require: Dataset size n, epoch index e, current loss ℓ_e

Parameters: $0 < \tau_{\text{strong}} < \tau_{\text{mild}}$, $\alpha_{\text{strong}}, \alpha_{\text{mild}} \in (0, 1]$, $(p_{\text{rev}}, r_{\text{rev}})$, $(p_{\text{eo}}, r_{\text{eo}})$, $\varepsilon > 0$
$u_e \leftarrow \text{Seed}(e)$; $I \leftarrow (1, 2, \ldots, n)$; $b_{\text{strong}} \leftarrow \max\big(1, \lfloor \alpha_{\text{strong}} n \rfloor\big)$; $b_{\text{mild}} \leftarrow \max\big(1, \lfloor \alpha_{\text{mild}} n \rfloor\big)$
if $e = 0$ **then**
  Sample $\pi_0 \sim \text{Unif}(\mathcal{S}_n)$ using seed u_0
  output π_0 and stop
end if
$\rho_e \leftarrow \ell_e / (\ell_{e-1} + \varepsilon)$
if $\rho_e < \tau_{\text{strong}}$ **then**
  Partition I into consecutive blocks of size b_{strong}
  Randomly permute blocks using seed u_e; concatenate to form π_e
  if $e \equiv r_{\text{rev}} \pmod{p_{\text{rev}}}$ **then**
    $\pi_e \leftarrow \text{Rev}(\pi_e)$
  end if
else
  if $\rho_e < \tau_{\text{mild}}$ **then**
    Partition I into consecutive blocks of size b_{mild}
    Randomly permute blocks using seed u_e; concatenate to form π_e
  else
    Sample $\pi_e \sim \text{Unif}(\mathcal{S}_n)$ using seed u_e
    if $e \equiv r_{\text{eo}} \pmod{p_{\text{eo}}}$ **then**
      $\pi_e \leftarrow \text{EO}(\pi_e)$
    end if
  end if
end if
output π_e

$\sigma^2_{\text{blk}}(w) < \sigma^2_{\text{ind}}(w)$, implying that block reshuffling strictly reduces the variance constants that govern without-replacement SGD. Importantly, this improvement is achieved without altering the convergence rate guarantees provided by unified shuffling theory.

Within the unified shuffling framework, convergence bounds depend on prefix-gradient variance constants that measure the deviation between partial sums of gradients along a permutation and the full gradient. We show that block reshuffling strictly reduces these constants under mild coherence assumptions on the gradients within each block.

2.3 Paired Reversal: Symmetrization of Order Effects

Sequential gradient updates within an epoch introduce order-dependent drift relative to full gradient descent. Under standard smoothness assumptions, this drift appears at second order in the stepsize and depends explicitly on the permutation. The reversal operator $\text{Rev}(\cdot)$ maps a permutation $\pi = (\pi(1), \ldots, \pi(n)) \quad \mapsto \text{Rev}(\pi) = (\pi(n), \ldots, \pi(1))$.

Instead of using a single permutation π, we consider the symmetrized epoch map obtained by averaging the updates induced by π and $\text{Rev}(\pi)$, that is, $\bar{T}_\pi = \frac{1}{2}(T_\pi + T_{\text{rev}(\pi)})$. This construction preserves the without replacement structure while introducing a fundamental symmetry in the update.

Under standard smoothness assumptions, the epoch map admits a second-order expansion in the stepsize. Averaging a permutation with its reversal cancels the entire order-dependent second-order term, reducing permutation sensitivity from $\Theta(\gamma^2)$ to $O(\gamma^3)$. As a consequence, paired reversal provably stabilizes without replacement SGD beyond what is achievable by random reshuffling alone. Importantly, this symmetrization does not introduce additional randomness. It purely exploits structural symmetry in the permutation.

3 ADVANTAGES OF STRUCTURED SHUFFLING

This section presents two complementary, provable mechanisms that improve the practical behavior of without-replacement SGD: (i) variance-constant reduction via block reshuffling, and (ii) order-sensitivity reduction via paired reversal. All proofs are deferred to Appendix B.

3.1 VARIANCE REDUCTION BY BLOCK RESHUFFLING

Fix an iterate $w \in \mathbb{R}^d$. Define component gradients and the full gradient

$$g_i(w) := \nabla f_i(w), \qquad \nabla F(w) := \frac{1}{n}\sum_{i=1}^{n} g_i(w). \tag{2}$$

Define the population variance of individual gradients

$$\sigma^2_{\mathrm{ind}}(w) := \frac{1}{n}\sum_{i=1}^{n} \|g_i(w) - \nabla F(w)\|^2. \tag{3}$$

Partition $\{1, \dots, n\}$ into K disjoint blocks $B_1, \dots, B_K$ of equal size $b = n/K$ (assume $K \mid n$ for simplicity). Define the block-averaged gradients

$$G_r(w) := \frac{1}{b}\sum_{i \in B_r} g_i(w), \qquad r = 1, \dots, K, \tag{4}$$

the block-level variance

$$\sigma^2_{\mathrm{blk}}(w) := \frac{1}{K}\sum_{r=1}^{K} \|G_r(w) - \nabla F(w)\|^2, \tag{5}$$

and the within-block variance

$$\sigma^2_{\mathrm{within}}(w) := \frac{1}{K}\sum_{r=1}^{K} \frac{1}{b}\sum_{i \in B_r} \|g_i(w) - G_r(w)\|^2. \tag{6}$$

Lemma 3.1. *With equal block sizes $b = n/K$, $\sigma^2_{\mathrm{ind}}(w) = \sigma^2_{\mathrm{within}}(w) + \sigma^2_{\mathrm{blk}}(w)$. Consequently, $\sigma^2_{\mathrm{blk}}(w) \le \sigma^2_{\mathrm{ind}}(w)$, with strict inequality whenever $\sigma^2_{\mathrm{within}}(w) > 0$.*

Proof. Deferred to Appendix B.1. □

3.2 ORDER-SENSITIVITY REDUCTION BY PAIRED REVERSAL

We now formalize the benefit of paired reversal. Consider one epoch of SGD with stepsize $\gamma > 0$ under a permutation $\pi \in \mathcal{S}_n$, starting from $w \in \mathbb{R}^d$:

$$w_0 = w\ ,\ w_t = w_{t-1} - \gamma\, \nabla f_{\pi(t)}(w_{t-1}), \quad t = 1, \dots, n\ ,\ T_\pi(w) := w_n. \tag{7}$$

Let $\mathrm{Rev}(\pi)$ denote the reversed permutation $\mathrm{Rev}(\pi)(t) = \pi(n + 1 - t)$.

Assumptions. Assume each f_i, $i = 1, \dots, n$, is twice continuously differentiable and satisfies:

$$\|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x - y\|, \tag{8}$$

$$\|\nabla f_i(x)\| \le G, \tag{9}$$

$$\|\nabla^2 f_i(x) - \nabla^2 f_i(y)\| \le \rho\|x - y\|. \tag{10}$$

Define $g_i := \nabla f_i(w)$ and $H_i := \nabla^2 f_i(w)$.

Lemma 3.2. *Assume equation 8-equation 10 hold. For any $\pi \in \mathcal{S}_n$,*

$$T_\pi(w) = w - \gamma\sum_{i=1}^{n} g_i + \gamma^2 \sum_{1 \le s < t \le n} H_{\pi(t)}\, g_{\pi(s)} + R_\pi(w), \tag{11}$$

where the remainder satisfies $\|R_\pi(w)\| \le C_{\mathrm{rem}}\, \gamma^3 n^3$, where $C_{\mathrm{rem}} := \frac{\rho G^2}{2} + \frac{L^2 G}{2}$.

Proof. Deferred to Appendix B.4. □

Theorem 3.3. *Define the paired-reversal averaged epoch map*

$$\bar{T}_\pi(w) := \frac{1}{2}\Big(T_\pi(w) + T_{\mathrm{Rev}(\pi)}(w)\Big). \tag{12}$$

Under equation 8-equation 10 assumptions, $\bar{T}_\pi(w)$ admits the expansion

$$\bar{T}_\pi(w) = w - \gamma \sum_{i=1}^{n} g_i + \frac{\gamma^2}{2} \sum_{i \neq j} H_i\, g_j + \bar{R}_\pi(w), \tag{13}$$

where the second-order term is independent of π and $\|\bar{R}_\pi(w)\| \le C_{\mathrm{rem}}\gamma^3 n^3$. Consequently, for any $\pi, \pi' \in \mathcal{S}_n$,

$$\|\bar{T}_\pi(w) - \bar{T}_{\pi'}(w)\| \le 2C_{\mathrm{rem}}\,\gamma^3 n^3, \tag{14}$$

i.e., order sensitivity is $O(\gamma^3)$ rather than $\Theta(\gamma^2)$ in general.

Proof. Deferred to Appendix B.5. □

Theorem 3.3 shows that averaging a permutation with its reversal removes the entire order-dependent γ^2 term in the epoch map expansion. Thus, when paired reversal is used, the leading contribution of permutation induced variability is pushed to higher order, providing a rigorous stability mechanism complementary to block variance reduction.

3.3 Order Sensitivity: A Rigorous $O(\gamma^2)$ Upper Bound

A standard second-order expansion immediately implies that, without any symmetrization, the dependence of the epoch map on the within-epoch order is *at most quadratic* in the stepsize.

Theorem 3.4. *Assume equation 8-equation 10 hold. Let T_π be the epoch map defined in equation 7. Then for any $w \in \mathbb{R}^d$ and any two permutations $\pi, \pi' \in \mathcal{S}_n$,*

$$\|T_\pi(w) - T_{\pi'}(w)\| \;\le\; LG\,\gamma^2\, n(n-1) \;+\; 2C_{\mathrm{rem}}\gamma^3 n^3, \tag{15}$$

where $C_{\mathrm{rem}} = \frac{\rho G^2}{2} + \frac{L^2 G}{2}$ is the constant from Lemma 3.2. In particular, for fixed n the order sensitivity is $O(\gamma^2)$ as $\gamma \to 0$.

Proof. Deferred to Appendix B.6. □

Proposition 3.5. *There exist smooth finite-sum problems and two permutations π, π' such that for all $\gamma > 0$, $\|T_\pi(w) - T_{\pi'}(w)\| \ge c\,\gamma^2$ for a constant $c > 0$.*

Proof. Deferred to Appendix B.7. □

Theorem 3.4 shows that permutation effects enter at second order through the term $B_\pi(w) = \sum_{1 \le s < t \le n} H_{\pi(t)} g_{\pi(s)}$ (Lemma 3.2), so one cannot expect order dependence larger than $O(\gamma^2)$ for sufficiently small learning rate. Our explicit $n = 2$ construction (Proposition 3.5) shows that $\Theta(\gamma^2)$ order dependence can occur, hence the quadratic scaling is unavoidable in general. Moreover, paired reversal (Theorem 3.3) cancels the entire order-dependent γ^2 term exactly, improving the bound from $O(\gamma^2)$ to $O(\gamma^3)$, which is a strict stability gain.

3.4 Permutation-Variance Reduction and Sharpness

Paired reversal also reduces the permutation-induced variance of epoch maps.

Permutation variance. Let π be random (typically $\pi \sim \mathrm{Unif}(\mathcal{S}_n)$). Define

$$\mathrm{Var}_\pi\big(T_\pi(w)\big) := \mathbb{E}_\pi \big\|T_\pi(w) - \mathbb{E}_\pi[T_\pi(w)]\big\|^2,\ \mathrm{Var}_\pi\big(\bar{T}_\pi(w)\big) := \mathbb{E}_\pi \big\|\bar{T}_\pi(w) - \mathbb{E}_\pi[\bar{T}_\pi(w)]\big\|^2. \tag{16}$$

Second-order decomposition. From Lemma 3.2,

$$T_\pi(w) = A(w) + \gamma^2 B_\pi(w) + R_\pi(w)\ ,\ A(w) := w - \gamma \sum_{i=1}^{n} g_i\ ,\ B_\pi(w) := \sum_{1\le s<t\le n} H_{\pi(t)}\, g_{\pi(s)}. \tag{17}$$

From Theorem 3.3,

$$\bar{T}_\pi(w) = \bar{A}(w) + \bar{R}_\pi(w)\ ,\ \bar{A}(w) := w - \gamma \sum_{i=1}^{n} g_i + \frac{\gamma^2}{2} \sum_{i\neq j} H_i\, g_j, \tag{18}$$

where $\bar{A}(w)$ is deterministic (independent of π) and $\|\bar{R}_\pi(w)\| \le C_{\mathrm{rem}}\gamma^3 n^3$.

Theorem 3.6. *Assume equation 8-equation 10 hold. Then for any fixed w,*

$$\mathrm{Var}_\pi\big(T_\pi(w)\big) \le 2\gamma^4\, \mathrm{Var}_\pi\big(B_\pi(w)\big) + 2C_{\mathrm{rem}}^2\, \gamma^6 n^6, \tag{19}$$

$$\mathrm{Var}_\pi\big(\bar{T}_\pi(w)\big) \le C_{\mathrm{rem}}^2\, \gamma^6 n^6. \tag{20}$$

Proof. Deferred to Appendix B.8. □

The following explicit constructions show that $\Theta(\gamma^2)$ order dependence of T_π is unavoidable in general, and $\mathrm{Var}_\pi(T_\pi(w)) = \Theta(\gamma^4)$ in general.

Proposition 3.7. *There exist smooth finite-sum problems and an iterate w such that for $\pi \sim \mathrm{Unif}(\mathcal{S}_2)$,*

$$\mathrm{Var}_\pi\big(T_\pi(w)\big) = \Theta(\gamma^4) \qquad (\textit{for fixed } n = 2).$$

Proof. Deferred to Appendix B.9. □

We summarize the key properties of the two components as follows.

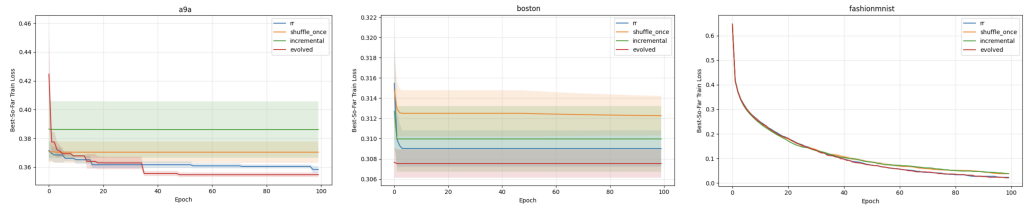
- **Block reshuffling.** Lemma 3.1 shows that $\sigma^2_{\mathrm{blk}}(w) \le \sigma^2_{\mathrm{ind}}(w)$, with strict improvement when within-block gradients are non-identical. Propositions A.2-A.3 further show that block reshuffling obeys the same without-replacement prefix-variance formula with the reduced constant $\sigma^2_{\mathrm{blk}}(w)$, improving constants in unified bounds involving $\mathbb{E}\|\Delta^{(k)}_\pi(w)\|^2$.
- **Paired reversal.** Theorem 3.3 shows that averaging a permutation and its reverse cancels the order-dependent γ^2 term, reducing order sensitivity from $\Theta(\gamma^2)$ to $O(\gamma^3)$.

Together, these results show that block reshuffling and paired reversal capture complementary mechanisms in without replacement SGD. Block reshuffling reduces variance constants through structured aggregation, while paired reversal reduces order sensitivity through symmetry. These principles suggest broader classes of structured permutations and open directions for designing new algorithms and convergence theory.

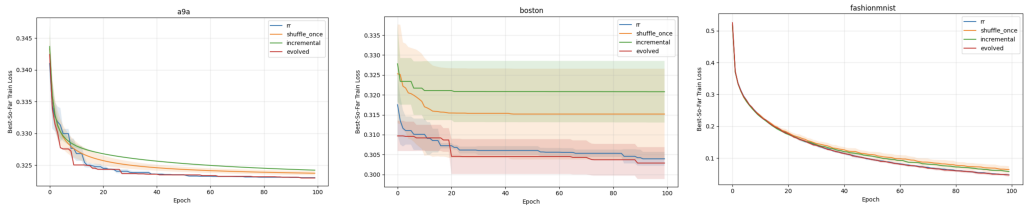
4 Numerical Experiments

We evaluate the empirical performance of the shuffling rule discovered by the LLM-guided program evolution pipeline (APR) and compare it against standard shuffling schemes on classification and regression tasks. For the `a9a`, `breast_cancer`, and `digits` datasets (Chang & Lin, 2011; Kaynak, 1995), we use logistic regression models for classification, resulting in convex objectives. For the `california`, `boston`, and `diabetes` datasets (Pace & Barry, 1997; Buitinck et al., 2013), we use linear regression with mean squared error loss, yielding convex regression problems. For the `MNIST` and `FashionMNIST` datasets (LeCun et al., 1998; Xiao et al., 2017), we use fully connected neural networks for classification, leading to non-convex optimization problems. Additional details on datasets, models, and training configurations are provided in Appendix C.

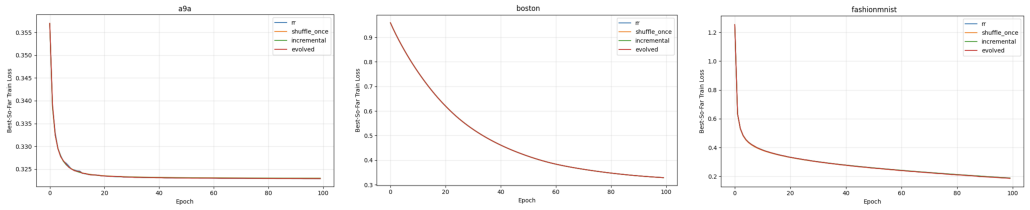
We compare the proposed shuffling strategy, the LLM-discovered APR scheme (Evolved), against standard baselines including incremental gradient (IG), shuffle-once (SO), and random reshuffling (RR) on a range of convex and nonconvex problems. All experiments focus on training performance, consistent with our theoretical analysis. For each dataset, model, learning-rate schedule, and shuffling



(a) Constant learning rates



(b) Diminishing learning rates



(c) Small learning rates

Figure 1: Performance comparison under different learning-rate schedules for `a9a`, `boston`, and `FashionMNIST` datasets.

Table 1: Comparisons using different shuffling schemes to SGD. RR: Random Reshuffle; SO: Shuffle Once; IG: Incremental Gradient

Dataset	**Task**	**LR**	**Evolved (APR)**	**RR**	**SO**	**IG**
a9a	Classification (Convex)	Fixed	0.354733 ±0.001576	0.358406 ±0.002402	0.370421 ±0.007436	0.386135 ±0.019584
	Classification (Convex)	Dimishing	0.322985 ±0.000009	0.323040 ±0.000062	0.323738 ±0.000323	0.324219 ±0.000041
breast_cancer	Classification (Convex)	Fixed	0.043627 ±0.001694	0.045695 ±0.003468	0.074279 ±0.021865	0.056839 ±0.012038
	Classification (Convex)	Dimishing	0.038790 ±0.000249	0.038632 ±0.000112	0.040084 ±0.001086	0.039313 ±0.000288
digits	Classification (Convex)	Fixed	0.199566 ±0.000046	0.199883 ±0.000162	0.201699 ±0.000904	0.204401 ±0.000331
	Classification (Convex)	Dimishing	0.199568 ±0.000048	0.199974 ±0.000230	0.201680 ±0.000910	0.204488 ±0.000327
california	Regression (Convex)	Fixed	0.371610 ±0.002141	0.375588 ±0.004742	0.460118 ±0.029614	0.568954 ±0.027146
	Regression (Convex)	Dimishing	0.373325 ±0.001430	0.374319 ±0.002249	0.414826 ±0.019466	0.543131 ±0.020387
boston	Regression (Convex)	Fixed	0.307530 ±0.001394	0.309014 ±0.001794	0.312260 ±0.001948	0.309964 ±0.003237
	Regression (Convex)	Dimishing	0.308947 ±0.003023	0.309462 ±0.001816	0.313370 ±0.000803	0.313111 ±0.001262
diabetes	Regression (Convex)	Fixed	0.494990 ±0.000777	0.495689 ±0.000378	0.508570 ±0.008987	0.497465 ±0.002258
	Regression (Convex)	Dimishing	0.494849 ±0.000732	0.494986 ±0.000737	0.500723 ±0.000952	0.498138 ±0.002611

scheme, we repeat experiments using 5 random initializations and 5 independent runs, and report the mean and standard deviation of the best-so-far training loss after 100 epochs. Complete experimental settings are provided in Appendix E.

Table 2: Comparisons using different shuffling schemes to Adam. RR: Random Reshuffle; SO: Shuffle Once; IG: Incremental Gradient

Dataset	Task	LR	Evolved (APR)	RR	SO	IG
MNIST	Classification (Non-Convex)	Fixed	0.000001 ±0.000001	0.000001 ±0.000001	0.000001 ±0.000001	0.000001 ±0.000001
	Classification (Non-Convex)	Dimishing	0.000016 ±0.000001	0.000024 ±0.000001	0.000003 ±0.000001	0.000005 ±0.000001
FashionMNIST	Classification (Non-Convex)	Fixed	0.021357 ±0.002395	0.023210 ±0.000929	0.038008 ±0.000675	0.038755 ±0.001205
	Classification (Non-Convex)	Dimishing	0.022238 ±0.003201	0.022286 ±0.002430	0.040572 ±0.004566	0.046295 ±0.004722

Table 1 reports results for SGD under constant and diminishing learning rates. Across all convex classification and regression tasks, the evolved shuffling strategy consistently achieves lower best-so-far training loss than IG, SO, and RR. The gains are most pronounced under constant learning rates, where order-dependent effects are strongest. In addition to improved mean performance, the evolved scheme typically exhibits smaller standard deviations, indicating more stable optimization trajectories. This behavior aligns with our theory: block reshuffling reduces prefix-gradient variance constants, leading to smoother within-epoch updates. Under diminishing learning rates, performance differences become smaller, but the evolved scheme remains competitive and often retains a slight advantage. Table 2 summarizes results for Adam (Kingma & Ba, 2014) on `MNIST` and `FashionMNIST`. Although Adam introduces additional adaptivity beyond data ordering, the evolved shuffling strategy remains competitive and often improves best-so-far training loss relative to IG, SO, and RR. These results suggest that structured shuffling can be beneficial beyond plain SGD.

Figure 1 illustrates training dynamics on representative datasets (`a9a`, `boston`, and `FashionMNIST`) under different learning rate schedules. Under constant learning rates (Figure 1a), the evolved scheme converges faster and to lower training loss, with visibly tighter confidence bands, reflecting reduced variance across runs. Under diminishing learning rates (Figure 1b), convergence curves become more similar, though the evolved scheme remains competitive throughout training. In the small-variance regime induced by very small learning rates (Figure 1c), all shuffling schemes exhibit nearly indistinguishable behavior. This empirical observation matches our theoretical results, which show that order sensitivity and permutation-induced variance vanish rapidly as the learning rate decreases. Consequently, differences between shuffling strategies become negligible in this regime.

Overall, the experiments support the theoretical insights of this work. APR improves both training loss and stability, particularly when learning rates are not overly small. These benefits are achieved without additional computation or memory overhead, highlighting the practical value of structured data ordering.

5 Conclusion and Limitations

We study structured data-ordering strategies for without-replacement stochastic optimization by combining LLM-guided algorithm discovery with theoretical analysis. Starting from an LLM-discovered reshuffling rule, we isolate two core components, block reshuffling and paired reversal, and show that they address complementary sources of inefficiency in epoch-based SGD. Our results establish that block reshuffling reduces prefix-gradient variance constants within the unified shuffling framework, while paired reversal cancels the leading order-dependent second-order term and reduces order sensitivity. Empirical results on convex and nonconvex benchmarks validate these insights and show consistent improvements over standard shuffling schemes, particularly outside the small learning rate regime. Overall, this work takes a first step toward understanding how structured data ordering can improve optimization beyond random reshuffling. Our findings indicate that block reshuffling and paired reversal reflect underlying variance and symmetry principles in without-replacement SGD. These principles suggest a broader design space of structured permutations that remain unexplored, and further investigation may lead to new algorithms and sharper convergence theory.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Kwangjun Ahn, Chulhee Yun, and Suvrit Sra. Sgd with shuffling: optimal rates without component convexity and large epoch requirements. *arXiv preprint arXiv:2006.06946*, 2020.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.*, 60(2):223–311, 2018.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Francois Chollet et al. Keras, 2015. URL `https://github.com/fchollet/keras`.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.*, 23(4):2341–2368, 2013.

M. Gürbüzbalaban, A. Ozdaglar, and P. A. Parrilo. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, Oct 2019. ISSN 1436-4646. doi: 10.1007/s10107-019-01440-w. URL `http://dx.doi.org/10.1007/s10107-019-01440-w`.

Jeff Haochen and Suvrit Sra. Random shuffling beats sgd after finite epochs. In *International Conference on Machine Learning*, pp. 2624–2633. PMLR, 2019.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pp. 315–323, 2013.

C Kaynak. Methods of combining multiple classifiers and their applications to handwritten digit recognition. *Unpublished master's thesis, Bogazici University*, 1995.

D. P. Kingma and J. Ba. ADAM: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, abs/1412.6980, 2014.

Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, pp. 2663–2671, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Konstantin Mishchenko, Ahmed Khaled Ragab Bayoumi, and Peter Richtárik. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33, 2020.

Dheeraj Nagaraj, Prateek Jain, and Praneeth Netrapalli. Sgd without replacement: Sharper rates for general smooth convex functions. In *International Conference on Machine Learning*, pp. 4703–4711, 2019.

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optim.*, 12(1):109–138, 2001.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4):1574–1609, 2009.

L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2613–2621, 2017.

Lam M Nguyen, Quoc Tran-Dinh, Dzung T Phan, Phuong Ha Nguyen, and Marten Van Dijk. A unified convergence analysis for shuffling-type gradient methods. *Journal of Machine Learning Research*, 22(207):1–44, 2021.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33 (3):291–297, 1997.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Shashank Rajput, Anant Gupta, and Dimitris Papailiopoulos. Closing the convergence gap of sgd without replacement. In *International Conference on Machine Learning*, pp. 7964–7973. PMLR, 2020.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Itay Safran and Ohad Shamir. How good is sgd with random shuffling? In *Conference on Learning Theory*, pp. 3250–3284. PMLR, 2020.

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent, 2025. URL `https://github.com/algorithmicsuperintelligence/openevolve`.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Learning to Shuffle: Block Reshuffling and Reversal Schemes for Stochastic Optimization Supplementary Material, ICLR 2027

A Useful Lemmas

Unified analyses for without-replacement SGD often control the deviation of prefix averages

$$\Delta_\pi^{(k)}(w) := \frac{1}{k}\sum_{t=1}^{k} g_{\pi(t)}(w) - \nabla F(w), \tag{21}$$

where π denotes the within-epoch permutation.

The next lemma gives the exact second moment of the prefix mean under uniform sampling without replacement, a key ingredient in RR-type bounds.

Lemma A.1 ((Mishchenko et al., 2020)). *Let $X_1, \dots, X_m \in \mathbb{R}^d$ be fixed with $\bar{X} = \frac{1}{m}\sum_{i=1}^m X_i$ and $\sigma^2 = \frac{1}{m}\sum_{i=1}^m \|X_i - \bar{X}\|^2$. Let $\pi \sim \mathrm{Unif}(\mathcal{S}_m)$ and define $\bar{X}_\pi^{(k)} = \frac{1}{k}\sum_{t=1}^k X_{\pi(t)}$. Then*

$$\mathbb{E}\big[\bar{X}_\pi^{(k)}\big] = \bar{X}, \qquad \mathbb{E}\big\|\bar{X}_\pi^{(k)} - \bar{X}\big\|^2 = \frac{m-k}{k(m-1)}\,\sigma^2. \tag{22}$$

Proposition A.2 (RR prefix variance (sample-level)). *Let $\pi \sim \mathrm{Unif}(\mathcal{S}_n)$ and $\widehat{g}_m(w) = \frac{1}{m}\sum_{t=1}^m g_{\pi(t)}(w)$. Then*

$$\mathbb{E}\big\|\widehat{g}_m(w) - \nabla F(w)\big\|^2 = \frac{n-m}{m(n-1)}\,\sigma_{\mathrm{ind}}^2(w). \tag{23}$$

Proof. Deferred to Appendix B.2. □

Proposition A.3 (Block reshuffling prefix variance (block-level)). *Let $\sigma \sim \mathrm{Unif}(\mathcal{S}_K)$ and $\widehat{G}_k(w) = \frac{1}{k}\sum_{j=1}^k G_{\sigma(j)}(w)$. Then*

$$\mathbb{E}\big\|\widehat{G}_k(w) - \nabla F(w)\big\|^2 = \frac{K-k}{k(K-1)}\,\sigma_{\mathrm{blk}}^2(w). \tag{24}$$

Proof. Deferred to Appendix B.3. □

Equations equation 23-equation 24 show that the same without replacement prefix-variance formula holds at the block level, but with variance constant $\sigma_{\mathrm{blk}}^2(w)$ in place of $\sigma_{\mathrm{ind}}^2(w)$. By Lemma 3.1, this constant is strictly smaller whenever within-block gradients are not identical. Thus, in regimes where gradients are locally coherent, block reshuffling reduces the magnitude of prefix-gradient noise and improves the constants appearing in convergence bounds driven by $\mathbb{E}\|\Delta_\pi^{(k)}(w)\|^2$.

B Proofs for Section 3

B.1 Proof of Lemma 3.1

Proof. Fix $w \in \mathbb{R}^d$ and a block B_r. For any $i \in B_r$,

$$g_i(w) - \nabla F(w) = \big(g_i(w) - G_r(w)\big) + \big(G_r(w) - \nabla F(w)\big).$$

Taking squared norms and summing over $i \in B_r$ yields

$$\sum_{i\in B_r} \|g_i(w) - \nabla F(w)\|^2 = \sum_{i\in B_r} \|g_i(w) - G_r(w)\|^2 + 2\Big\langle \sum_{i\in B_r} (g_i(w) - G_r(w)),\, G_r(w) - \nabla F(w)\Big\rangle + \sum_{i\in B_r} \|G_r(w) - \nabla F(w)\|^2.$$

By the definition of $G_r(w)$, we have $\sum_{i\in B_r}(g_i(w) - G_r(w)) = 0$, so the cross term vanishes. Hence

$$\sum_{i\in B_r} \|g_i(w) - \nabla F(w)\|^2 = \sum_{i\in B_r} \|g_i(w) - G_r(w)\|^2 + b\,\|G_r(w) - \nabla F(w)\|^2.$$

Divide by b, then average over $r = 1, \dots, K$ and use $n = Kb$:

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} \|g_i(w) - \nabla F(w)\|^2 &= \frac{1}{K}\sum_{r=1}^{K}\frac{1}{b}\sum_{i\in B_r} \|g_i(w) - G_r(w)\|^2 + \frac{1}{K}\sum_{r=1}^{K} \|G_r(w) - \nabla F(w)\|^2 \\ &= \sigma^2_{\text{within}}(w) + \sigma^2_{\text{blk}}(w),\end{aligned}$$

which is the desired result. The inequality $\sigma^2_{\text{blk}}(w) \le \sigma^2_{\text{ind}}(w)$ follows immediately since $\sigma^2_{\text{within}}(w) \ge 0$, and it is strict whenever $\sigma^2_{\text{within}}(w) > 0$. □

B.2 Proof of Proposition A.2

Proof. Apply Lemma A.1 with $m = n$ and $X_i = g_i(w)$. Then $\bar{X} = \nabla F(w)$ and $\sigma^2 = \sigma^2_{\text{ind}}(w)$ by equation 3. For $\widehat{g}_m(w) = \frac{1}{m}\sum_{t=1}^{m} g_{\pi(t)}(w)$, Lemma A.1 yields equation 23. □

B.3 Proof of Proposition A.3

Proof. Apply Lemma A.1 with $m = K$ and $X_r = G_r(w)$. Then $\bar{X} = \nabla F(w)$ and $\sigma^2 = \sigma^2_{\text{blk}}(w)$ by equation 5. For $\widehat{G}_k(w) = \frac{1}{k}\sum_{j=1}^{k} G_{\sigma(j)}(w)$, Lemma A.1 yields equation 24. □

B.4 Proof of Lemma 3.2

Proof. Fix $w \in \mathbb{R}^d$ and $\pi \in \mathcal{S}_n$. Define the within-epoch iterates

$$w_0 := w, \qquad w_t := w_{t-1} - \gamma \nabla f_{\pi(t)}(w_{t-1}), \quad t = 1, \dots, n, \qquad T_\pi(w) := w_n.$$

Step 1: Telescoping representation. From the recursion,

$$T_\pi(w) - w = w_n - w_0 = -\gamma \sum_{t=1}^{n} \nabla f_{\pi(t)}(w_{t-1}). \tag{25}$$

Step 2: Second-order expansion of each gradient around w. Fix $t \in \{1, \dots, n\}$. By the fundamental theorem of calculus applied to the gradient map,

$$\nabla f_{\pi(t)}(w_{t-1}) = \nabla f_{\pi(t)}(w) + \int_0^1 \nabla^2 f_{\pi(t)}\big(w + \tau(w_{t-1} - w)\big)\,(w_{t-1} - w)\,d\tau. \tag{26}$$

Note that $g_i := \nabla f_i(w)$ and $H_i := \nabla^2 f_i(w)$, $i = 1, \dots, n$. Add and subtract $H_{\pi(t)}(w)(w_{t-1} - w)$ inside the integral to obtain

$$\nabla f_{\pi(t)}(w_{t-1}) = g_{\pi(t)} + H_{\pi(t)}(w)(w_{t-1} - w) + r_t, \tag{27}$$

where

$$r_t := \int_0^1 \Big(\nabla^2 f_{\pi(t)}\big(w + \tau(w_{t-1} - w)\big) - \nabla^2 f_{\pi(t)}(w)\Big)\,(w_{t-1} - w)\,d\tau. \tag{28}$$

Using the Hessian-Lipschitz condition equation 10,

$$\Big\|\nabla^2 f_{\pi(t)}\big(w + \tau(w_{t-1} - w)\big) - \nabla^2 f_{\pi(t)}(w)\Big\| \le \rho\,\tau \|w_{t-1} - w\|,$$

hence

$$\|r_t\| \le \int_0^1 \Big\|\nabla^2 f_{\pi(t)}\big(w+\tau(w_{t-1}-w)\big) - \nabla^2 f_{\pi(t)}(w)\Big\| \, \|w_{t-1}-w\| \, d\tau$$
$$\le \int_0^1 \rho\,\tau \|w_{t-1}-w\|^2 \, d\tau = \frac{\rho}{2}\|w_{t-1}-w\|^2. \tag{29}$$

Substituting equation 27 into equation 25 gives

$$T_\pi(w) - w = -\gamma \sum_{t=1}^n g_{\pi(t)} - \gamma \sum_{t=1}^n H_{\pi(t)}(w)(w_{t-1}-w) - \gamma \sum_{t=1}^n r_t. \tag{30}$$

Since $\sum_{t=1}^n g_{\pi(t)} = \sum_{i=1}^n g_i$ for any permutation, the first term is the desired first-order term.

Step 3: First-order expansion of $w_{t-1} - w$ and collection of second-order terms. From the recursion,

$$w_{t-1} - w = -\gamma \sum_{s=1}^{t-1} \nabla f_{\pi(s)}(w_{s-1}). \tag{31}$$

We have

$$\nabla f_{\pi(s)}(w_{s-1}) = g_{\pi(s)} + \big(\nabla f_{\pi(s)}(w_{s-1}) - \nabla f_{\pi(s)}(w)\big).$$

Define the accumulated error

$$\delta_{t-1} := -\gamma \sum_{s=1}^{t-1} \big(\nabla f_{\pi(s)}(w_{s-1}) - \nabla f_{\pi(s)}(w)\big), \tag{32}$$

so that

$$w_{t-1} - w = -\gamma \sum_{s=1}^{t-1} g_{\pi(s)} + \delta_{t-1}. \tag{33}$$

Plugging equation 33 into the Hessian term in equation 30 yields

$$-\gamma \sum_{t=1}^n H_{\pi(t)}(w)(w_{t-1}-w) = -\gamma \sum_{t=1}^n H_{\pi(t)}(w)\Big(-\gamma \sum_{s=1}^{t-1} g_{\pi(s)} + \delta_{t-1}\Big)$$
$$= \gamma^2 \sum_{t=1}^n \sum_{s=1}^{t-1} H_{\pi(t)}(w)\, g_{\pi(s)} - \gamma \sum_{t=1}^n H_{\pi(t)}(w)\, \delta_{t-1}$$
$$= \gamma^2 \sum_{1\le s<t\le n} H_{\pi(t)}\, g_{\pi(s)} - \gamma \sum_{t=1}^n H_{\pi(t)}(w)\, \delta_{t-1}. \tag{34}$$

Step 4: Define and bound the remainder. Combine equation 30 and equation 34 and define

$$R_\pi(w) := -\gamma \sum_{t=1}^n r_t - \gamma \sum_{t=1}^n H_{\pi(t)}(w)\, \delta_{t-1}. \tag{35}$$

Then equation 11 holds.

We now bound $\|R_\pi(w)\|$. First, using equation 29,

$$\gamma \sum_{t=1}^n \|r_t\| \le \frac{\rho\gamma}{2} \sum_{t=1}^n \|w_{t-1}-w\|^2. \tag{36}$$

By equation 9, $\|w_t - w_{t-1}\| = \gamma\|\nabla f_{\pi(t)}(w_{t-1})\| \le \gamma G$, hence

$$\|w_{t-1}-w\| \le \sum_{j=1}^{t-1} \|w_j - w_{j-1}\| \le (t-1)\gamma G. \tag{37}$$

Therefore,

$$\sum_{t=1}^{n} \|w_{t-1} - w\|^2 \le \gamma^2 G^2 \sum_{t=1}^{n} (t-1)^2 \le \gamma^2 G^2 \, n^3,$$

and equation 36 gives

$$\gamma \sum_{t=1}^{n} \|r_t\| \le \frac{\rho}{2} \gamma^3 G^2 n^3. \tag{38}$$

Second, bound the term involving δ_{t-1}. By equation 32 and equation 8,

$$\|\delta_{t-1}\| \le \gamma \sum_{s=1}^{t-1} \|\nabla f_{\pi(s)}(w_{s-1}) - \nabla f_{\pi(s)}(w)\| \le \gamma \sum_{s=1}^{t-1} L \|w_{s-1} - w\|.$$

Using equation 37,

$$\|\delta_{t-1}\| \le \gamma \sum_{s=1}^{t-1} L(s-1)\gamma G = LG\gamma^2 \frac{(t-1)(t-2)}{2} \le \frac{LG}{2} \gamma^2 (t-1)^2.$$

Moreover, since ∇f_i is L-Lipschitz and f_i is twice differentiable, we have $\|H_i(x)\| \le L$ for all x, hence $\|H_{\pi(t)}(w)\| \le L$. Thus

$$\begin{aligned} \gamma \sum_{t=1}^{n} \|H_{\pi(t)}(w)\delta_{t-1}\| &\le \gamma \sum_{t=1}^{n} L \|\delta_{t-1}\| \\ &\le \gamma \sum_{t=1}^{n} L \cdot \frac{LG}{2} \gamma^2 (t-1)^2 = \frac{L^2 G}{2} \gamma^3 \sum_{t=1}^{n} (t-1)^2 \le \frac{L^2 G}{2} \gamma^3 n^3. \end{aligned} \tag{39}$$

Finally, combining equation 35, equation 38, and equation 39,

$$\|R_\pi(w)\| \le \left(\frac{\rho G^2}{2} + \frac{L^2 G}{2} \right) \gamma^3 n^3 = C_{\mathrm{rem}} \, \gamma^3 n^3,$$

which matches the desired result. □

B.5 Proof of Theorem 3.3

Proof. Start from the expansions in Lemma 3.2 for π and for $\mathrm{Rev}(\pi)$:

$$T_\pi(w) = w - \gamma \sum_{i=1}^{n} g_i + \gamma^2 B_\pi + R_\pi(w), \qquad T_{\mathrm{Rev}(\pi)}(w) = w - \gamma \sum_{i=1}^{n} g_i + \gamma^2 B_{\mathrm{Rev}(\pi)} + R_{\mathrm{Rev}(\pi)}(w),$$

where

$$B_\pi := \sum_{1 \le s < t \le n} H_{\pi(t)} g_{\pi(s)}.$$

Averaging gives

$$\bar{T}_\pi(w) = \frac{1}{2}\big(T_\pi(w) + T_{\mathrm{Rev}(\pi)}(w)\big) = w - \gamma \sum_{i=1}^{n} g_i + \frac{\gamma^2}{2}\big(B_\pi + B_{\mathrm{Rev}(\pi)}\big) + \bar{R}_\pi(w),$$

where $\bar{R}_\pi(w) := \frac{1}{2}(R_\pi(w) + R_{\mathrm{Rev}(\pi)}(w))$ so $\|\bar{R}_\pi(w)\| \le C_{\mathrm{rem}} \gamma^3 n^3$.

It remains to show $B_\pi + B_{\mathrm{Rev}(\pi)} = \sum_{i \ne j} H_i g_j$, which is independent of π. Using the index substitution $t' = n + 1 - t$, $s' = n + 1 - s$ (so that $s < t \Leftrightarrow t' < s'$),

$$B_{\mathrm{Rev}(\pi)} = \sum_{1 \le s < t \le n} H_{\mathrm{Rev}(\pi)(t)} \, g_{\mathrm{Rev}(\pi)(s)} = \sum_{1 \le s < t \le n} H_{\pi(n+1-t)} \, g_{\pi(n+1-s)} = \sum_{1 \le t' < s' \le n} H_{\pi(t')} \, g_{\pi(s')}.$$

Therefore,

$$B_\pi + B_{\mathrm{Rev}(\pi)} = \sum_{1 \le s < t \le n} H_{\pi(t)} g_{\pi(s)} + \sum_{1 \le t < s \le n} H_{\pi(t)} g_{\pi(s)} = \sum_{\substack{1 \le s,t \le n \\ s \ne t}} H_{\pi(t)} g_{\pi(s)} = \sum_{i \ne j} H_i g_j,$$

independent of π. This proves the expansion equation 13.

Finally, for any π, π',

$$\bar{T}_\pi(w) - \bar{T}_{\pi'}(w) = \bar{R}_\pi(w) - \bar{R}_{\pi'}(w),$$

so by the triangle inequality and $\|\bar{R}_\pi(w)\| \le C_{\mathrm{rem}}\gamma^3 n^3$,

$$\|\bar{T}_\pi(w) - \bar{T}_{\pi'}(w)\| \le \|\bar{R}_\pi(w)\| + \|\bar{R}_{\pi'}(w)\| \le 2C_{\mathrm{rem}}\gamma^3 n^3,$$

which is equation 14. □

B.6 Proof of Theorem 3.4

Proof. Fix $w \in \mathbb{R}^d$ and two permutations $\pi, \pi' \in \mathcal{S}_n$. By Lemma 3.2, we have the second-order expansions

$$T_\pi(w) = w - \gamma \sum_{i=1}^{n} g_i + \gamma^2 B_\pi(w) + R_\pi(w),$$

$$T_{\pi'}(w) = w - \gamma \sum_{i=1}^{n} g_i + \gamma^2 B_{\pi'}(w) + R_{\pi'}(w),$$

where

$$B_\pi(w) := \sum_{1 \le s < t \le n} H_{\pi(t)}\, g_{\pi(s)}, \qquad \|R_\pi(w)\| \le C_{\mathrm{rem}}\gamma^3 n^3, \qquad \|R_{\pi'}(w)\| \le C_{\mathrm{rem}}\gamma^3 n^3.$$

Subtracting yields

$$T_\pi(w) - T_{\pi'}(w) = \gamma^2\big(B_\pi(w) - B_{\pi'}(w)\big) + \big(R_\pi(w) - R_{\pi'}(w)\big).$$

Taking norms and using the triangle inequality gives

$$\|T_\pi(w) - T_{\pi'}(w)\| \le \gamma^2 \|B_\pi(w) - B_{\pi'}(w)\| + \|R_\pi(w)\| + \|R_{\pi'}(w)\|. \tag{40}$$

It remains to bound $\|B_\pi(w)\|$ uniformly over π. Since each f_i is twice differentiable and ∇f_i is L-Lipschitz by equation 8, we have $\|H_i(x)\| \le L$ for all x (in particular at $x = w$). Moreover, equation 9 implies $\|g_i\| = \|\nabla f_i(w)\| \le G$. Therefore,

$$\|B_\pi(w)\| = \left\| \sum_{1 \le s < t \le n} H_{\pi(t)}\, g_{\pi(s)} \right\| \le \sum_{1 \le s < t \le n} \|H_{\pi(t)}\|\, \|g_{\pi(s)}\| \le \sum_{1 \le s < t \le n} L\, G = LG \cdot \frac{n(n-1)}{2}.$$

By the same bound for π', we get

$$\|B_\pi(w) - B_{\pi'}(w)\| \le \|B_\pi(w)\| + \|B_{\pi'}(w)\| \le LG\, n(n-1). \tag{41}$$

Finally, combining equation 40–equation 41 with $\|R_\pi(w)\|, \|R_{\pi'}(w)\| \le C_{\mathrm{rem}}\gamma^3 n^3$,

$$\|T_\pi(w) - T_{\pi'}(w)\| \le LG\,\gamma^2\, n(n-1) + 2C_{\mathrm{rem}}\gamma^3 n^3,$$

which is exactly equation 15. □

B.7 Proof of Proposition 3.5

Proof. Let $n = 2$ and consider the one-dimensional functions

$$f_1(x) = \frac{a}{2}x^2, \qquad f_2(x) = bx,$$

with $a \neq 0$ and $b \neq 0$. Fix any $w \neq 0$ and stepsize $\gamma > 0$.

For $\pi = (1, 2)$:

$$x_1 = w - \gamma \nabla f_1(w) = w - \gamma a w = (1 - \gamma a)w, \qquad x_2 = x_1 - \gamma \nabla f_2(x_1) = (1 - \gamma a)w - \gamma b,$$

so $T_{(1,2)}(w) = (1 - \gamma a)w - \gamma b$.

For $\pi' = (2, 1)$:

$x_1' = w - \gamma \nabla f_2(w) = w - \gamma b, \qquad x_2' = x_1' - \gamma \nabla f_1(x_1') = (w - \gamma b) - \gamma a(w - \gamma b) = (1 - \gamma a)w - \gamma b + \gamma^2 ab,$

so $T_{(2,1)}(w) = (1 - \gamma a)w - \gamma b + \gamma^2 ab$.

Thus,

$$T_{(2,1)}(w) - T_{(1,2)}(w) = \gamma^2 ab,$$

and therefore

$$\left|T_{(2,1)}(w) - T_{(1,2)}(w)\right| = |ab|\,\gamma^2.$$

This establishes the claim with $c = |ab|$. □

B.8 Proof of Theorem 3.6

Proof. From Lemma 3.2, write

$$T_\pi(w) = A(w) + \gamma^2 B_\pi(w) + R_\pi(w), \qquad A(w) := w - \gamma \sum_{i=1}^{n} g_i, \qquad B_\pi(w) := \sum_{1 \le s < t \le n} H_{\pi(t)} g_{\pi(s)}.$$

Since $A(w)$ is deterministic, subtracting expectations yields

$$T_\pi(w) - \mathbb{E}_\pi[T_\pi(w)] = \gamma^2 \big(B_\pi(w) - \mathbb{E}B_\pi(w)\big) + \big(R_\pi(w) - \mathbb{E}R_\pi(w)\big).$$

Using $\|x + y\|^2 \le 2\|x\|^2 + 2\|y\|^2$ and $\mathbb{E}\|R_\pi - \mathbb{E}R_\pi\|^2 \le \mathbb{E}\|R_\pi\|^2$,

$$\begin{aligned}
\mathrm{Var}_\pi\big(T_\pi(w)\big) &= \mathbb{E}_\pi \big\|T_\pi(w) - \mathbb{E}_\pi[T_\pi(w)]\big\|^2 \\
&\le 2\gamma^4\, \mathbb{E}_\pi \big\|B_\pi(w) - \mathbb{E}_\pi[B_\pi(w)]\big\|^2 + 2\,\mathbb{E}_\pi \|R_\pi(w)\|^2 \\
&= 2\gamma^4\, \mathrm{Var}_\pi\big(B_\pi(w)\big) + 2\,\mathbb{E}_\pi \|R_\pi(w)\|^2 \;\le\; 2\gamma^4\, \mathrm{Var}_\pi\big(B_\pi(w)\big) + 2C_{\mathrm{rem}}^2 \gamma^6 n^6,
\end{aligned}$$

which gives equation 19.

For the paired-reversal map, from Theorem 3.3 we have

$$\bar{T}_\pi(w) = \bar{A}(w) + \bar{R}_\pi(w),$$

where $\bar{A}(w)$ is deterministic (independent of π). Hence

$$\mathrm{Var}_\pi\big(\bar{T}_\pi(w)\big) = \mathbb{E}_\pi \big\|\bar{R}_\pi(w) - \mathbb{E}_\pi[\bar{R}_\pi(w)]\big\|^2 \le \mathbb{E}_\pi \|\bar{R}_\pi(w)\|^2 \le C_{\mathrm{rem}}^2 \gamma^6 n^6,$$

which is equation 20. □

B.9 Proof of Proposition 3.7

Proof. We use a fully explicit $n = 2$, one-dimensional instance. Let

$$f_1(x) = \frac{a}{2} x^2, \qquad f_2(x) = bx,$$

with $a \neq 0$ and $b \neq 0$. Fix $w \neq 0$ and stepsize $\gamma > 0$. Let π be uniform over $\mathcal{S}_2$, i.e., $\mathbb{P}(\pi = (1, 2)) = \mathbb{P}(\pi = (2, 1)) = 1/2$.

Bernoulli second-order term. For $n = 2$, the second-order term is

$$B_\pi(w) = H_{\pi(2)}(w)\, g_{\pi(1)}(w),$$

where $g_1(w) = aw$, $H_1(w) = a$, $g_2(w) = b$, $H_2(w) = 0$. Thus,

$$B_{(1,2)}(w) = H_2(w) g_1(w) = 0, \qquad B_{(2,1)}(w) = H_1(w) g_2(w) = ab,$$

so $B_\pi(w)$ is Bernoulli:

$$B_\pi(w) = \begin{cases} 0, & \text{w.p. } 1/2, \\ ab, & \text{w.p. } 1/2. \end{cases}$$

Hence

$$\mathbb{E}B_\pi(w) = \frac{ab}{2}, \qquad \mathrm{Var}_\pi(B_\pi(w)) = \frac{(ab)^2}{4}.$$

A variance perturbation inequality. We use the following inequality: for any square-integrable random vectors X, Y,

$$\mathrm{Var}(X+Y) \geq \frac{1}{2}\mathrm{Var}(X) - \mathbb{E}\|Y\|^2. \tag{42}$$

To see this, let $\mu_X = \mathbb{E}X$, $\mu_Y = \mathbb{E}Y$ and write $\mathrm{Var}(X+Y) = \mathbb{E}\|(X-\mu_X)+(Y-\mu_Y)\|^2$. Using $\|u+v\|^2 \geq \frac{1}{2}\|u\|^2 - \|v\|^2$ yields equation 42.

Lower bound for $\mathrm{Var}_\pi(T_\pi(w))$. From Lemma 3.2, for $n=2$ we have the expansion

$$T_\pi(w) = A(w) + \gamma^2 B_\pi(w) + R_\pi(w),$$

where $A(w)$ is deterministic and $\|R_\pi(w)\| \leq C_{\mathrm{rem}}\gamma^3 n^3 = 8C_{\mathrm{rem}}\gamma^3$. Subtracting expectations removes $A(w)$, so

$$T_\pi(w) - \mathbb{E}T_\pi(w) = \gamma^2(B_\pi(w) - \mathbb{E}B_\pi(w)) + (R_\pi(w) - \mathbb{E}R_\pi(w)).$$

Apply equation 42 with $X = \gamma^2(B_\pi - \mathbb{E}B_\pi)$ and $Y = R_\pi - \mathbb{E}R_\pi$. Then

$$\begin{aligned}
\mathrm{Var}_\pi(T_\pi(w)) &= \mathrm{Var}_\pi\big(\gamma^2 B_\pi(w) + R_\pi(w)\big) \\
&\geq \frac{1}{2}\mathrm{Var}_\pi\big(\gamma^2 B_\pi(w)\big) - \mathbb{E}_\pi\|R_\pi(w) - \mathbb{E}R_\pi(w)\|^2 \\
&\geq \frac{1}{2}\gamma^4\,\mathrm{Var}_\pi\big(B_\pi(w)\big) - \mathbb{E}_\pi\|R_\pi(w)\|^2 \\
&\geq \frac{1}{2}\gamma^4 \cdot \frac{(ab)^2}{4} - (8C_{\mathrm{rem}}\gamma^3)^2 \\
&= \frac{(ab)^2}{8}\gamma^4 - 64C_{\mathrm{rem}}^2\gamma^6.
\end{aligned}$$

In particular, for sufficiently small γ (e.g., $64C_{\mathrm{rem}}^2\gamma^2 \leq (ab)^2/16$),

$$\mathrm{Var}_\pi(T_\pi(w)) \geq \frac{(ab)^2}{16}\gamma^4,$$

which proves $\mathrm{Var}_\pi(T_\pi(w)) = \Omega(\gamma^4)$ for this instance. Together with the general upper bound $\mathrm{Var}_\pi(T_\pi(w)) = O(\gamma^4) + O(\gamma^6)$ from the second-order representation, this establishes $\mathrm{Var}_\pi(T_\pi(w)) = \Theta(\gamma^4)$ for fixed $n=2$. □

C More Experimental Results

C.1 Datasets and Models

We evaluate shuffling strategies across a diverse collection of benchmark datasets, covering both classification and regression tasks for convex and non-convex objectives. All models are intentionally lightweight, so that performance differences primarily reflect the effect of data ordering rather than architectural complexity.

Classification datasets. We consider the following classification benchmarks:

- **a9a** and **breast_cancer** from `LIBSVM` (Chang & Lin, 2011). These datasets are treated as binary classification problems and are trained using linear logistic regression models. Features are standardized, and the objective is the regularized log-loss. Optimization is performed using stochastic gradient descent (SGD) with constant and diminishing learning rates. These tasks correspond to convex finite-sum optimization problems and are well suited for validating theoretical predictions on variance and order sensitivity.
- **digits** (Kaynak, 1995). The Digits dataset consists of 8×8 grayscale images. We binarize the labels (digits > 5 vs. ≤ 5) and use a logistic regression model, yielding a convex objective. This dataset serves as a low-dimensional image-based benchmark that remains analytically tractable.

- **MNIST** (LeCun et al., 1998) and **FashionMNIST** (Xiao et al., 2017). These datasets contain 28×28 grayscale images with 10 classes both with $60,000$ samples. We use a simple multilayer perceptron (MLP) with two hidden layers: $784 \to 256 \to 128 \to 10$, with ReLU activations. The networks are trained using cross-entropy loss and Adam. No convolutional layers, batch normalization, dropout, or data augmentation are used. This results in non-convex objectives while keeping the architecture minimal, allowing us to isolate the impact of shuffling on optimization dynamics.

Regression datasets. We also evaluate on standard regression benchmarks:

- **California Housing**, **Boston Housing**, and **Diabetes** from `sklearn.datasets` (Pace & Barry, 1997; Buitinck et al., 2013). These datasets involve predicting continuous targets from tabular features. We use linear regression models trained with mean squared error (MSE) loss and SGD. All features are standardized. These problems are convex and allow us to test shuffling strategies in regression settings complementary to classification.

Training process. For all experiments, models are trained for 100 epochs. To account for randomness arising from both initialization and data ordering, each configuration is evaluated using 5 different random initializations and 5 independent runs, resulting in 25 trials in total. We report the mean and standard deviation of the best-so-far training loss across these trials.

For SGD-based experiments, we evaluate a range of constant learning rates

$$\{0.5,\ 0.1,\ 0.05,\ 0.01,\ 0.005,\ 0.001,\ 0.0005,\ 0.0001\},$$

as well as a diminishing learning-rate schedule of the form $\gamma_t = \gamma_0/t^\alpha$ with $\alpha = 0.5$ with the mini-batch sizes of $\{64, 128, 256\}$. This range allows us to study shuffling behavior across both moderate and small learning rate regimes, including the small variance regime predicted by theory. For each dataset and shuffling scheme, we report results corresponding to the best performing constant learning rate selected from this grid, as well as the diminishing schedule. This process reflects standard practice and ensures that comparisons are not biased by suboptimal step-size choices.

For MNIST and FashionMNIST, we use the Adam optimizer with learning rate 0.001, following standard practice. Unless otherwise specified, all other hyperparameters are kept fixed across shuffling schemes, ensuring that differences in performance are attributable solely to the data ordering strategy.

C.2 Large Learning Rates

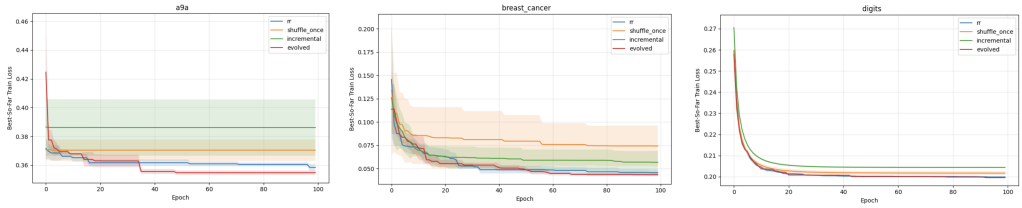
This appendix provides additional experimental results that complement and support the findings reported in the main text.

Across these additional experiments, we observe trends consistent with the conclusions drawn in the main text. In particular, the LLM-discovered shuffling strategy continues to outperform or match standard shuffling schemes in terms of best-so-far training loss, while often exhibiting reduced variability across runs. These results further support our claims that structured data ordering can improve optimization performance beyond random reshuffling, especially outside the small learning rate regime.

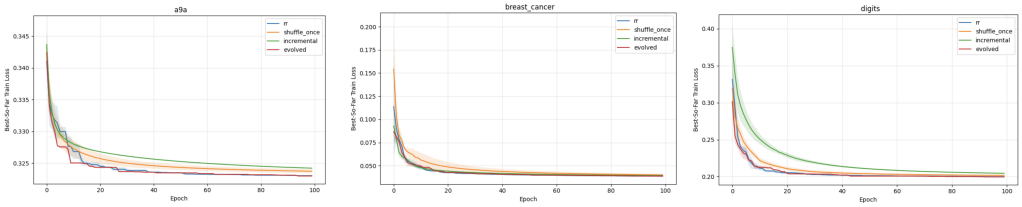
We emphasize that no new algorithmic components are introduced in this appendix. Rather, these results serve to demonstrate the robustness and generality of the empirical behavior discussed in the main paper. Taken together with the main results, the experiments in this appendix strengthen the evidence that block reshuffling and reversal-based structure are key contributors to the observed performance gains.

C.3 Small Learning Rates

In the small-variance regime induced by very small learning rates, all shuffling schemes exhibit nearly indistinguishable behavior.

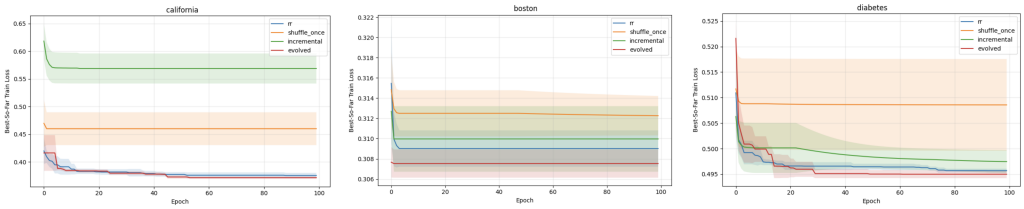


(a) Classification Datasets - Constant Learning Rates

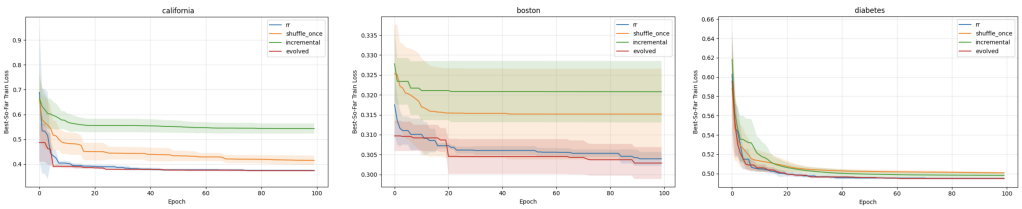


(b) Classification Datasets - Diminishing Learning Rates

Figure 2: Classification Datasets

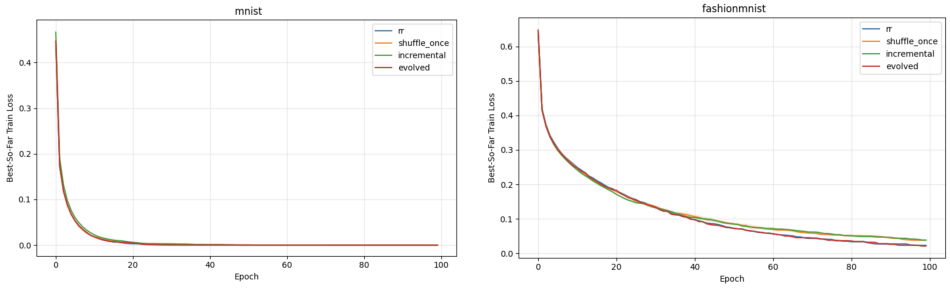


(a) Regression Datasets - Constant Learning Rates

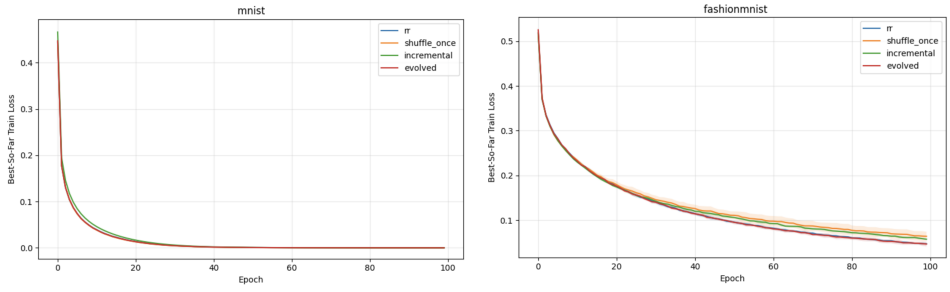


(b) Regression Datasets - Diminishing Learning Rates

Figure 3: Regression Datasets

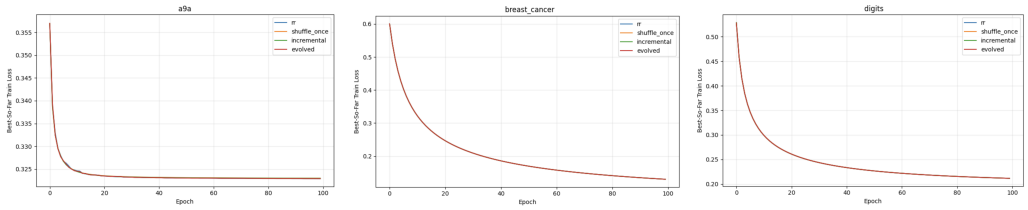


(a) Classification Datasets (NN) - Constant Learning Rates

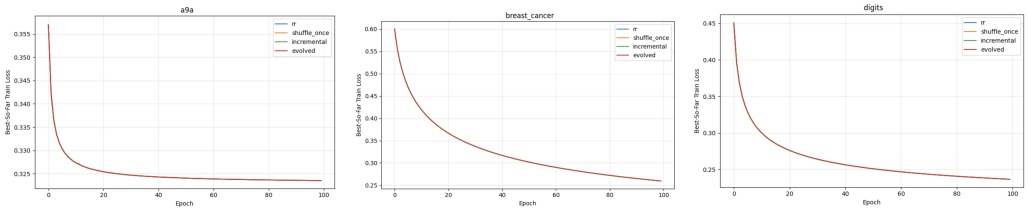


(b) Classification Datasets (NN) - Diminishing Learning Rates

Figure 4: Classification Datasets (NN)

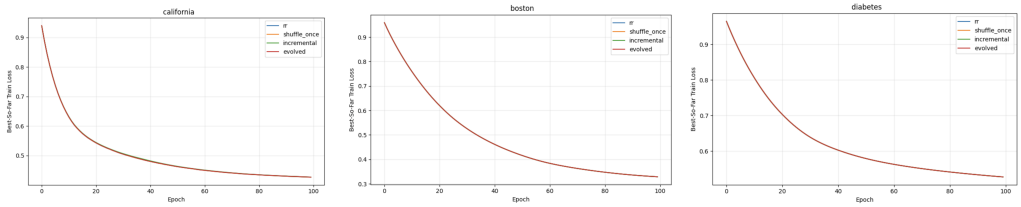


(a) Classification Datasets - Constant Learning Rates

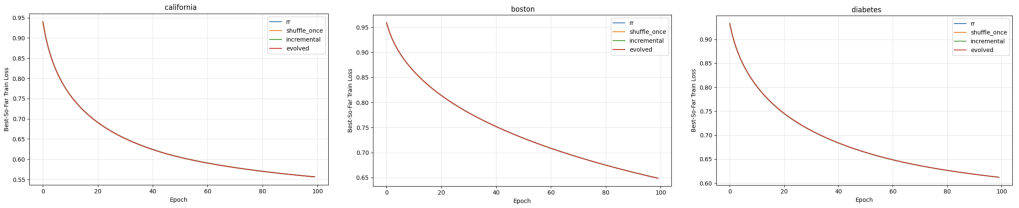


(b) Classification Datasets - Diminishing Learning Rates

Figure 5: Classification Datasets

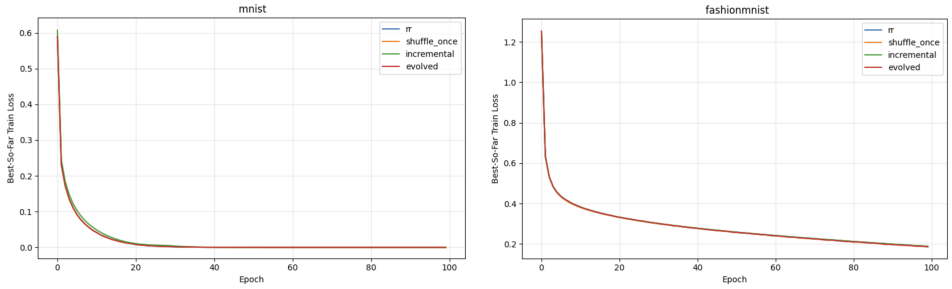


(a) Regression Datasets - Constant Learning Rates

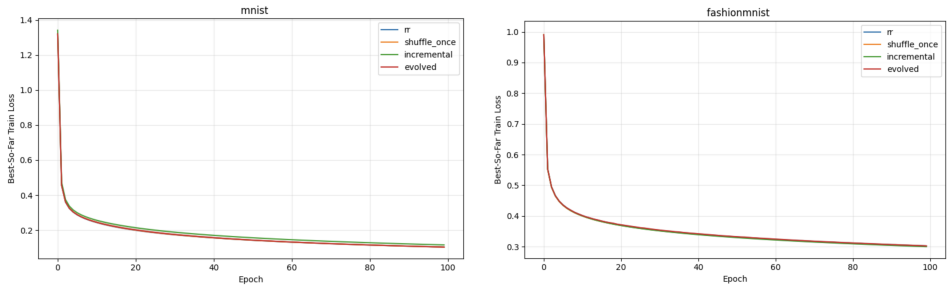


(b) Regression Datasets - Diminishing Learning Rates

Figure 6: Regression Datasets



(a) Classification Datasets (NN) - Constant Learning Rates



(b) Classification Datasets (NN) - Diminishing Learning Rates

Figure 7: Classification Datasets (NN)

D LLM-GUIDED DISCOVERY OF SHUFFLING RULES

We employ OpenEvolve (Sharma, 2025), a large language model (LLM)-guided program evolution pipeline based on the Microsoft Phi-4 model (Abdin et al., 2024), to explore a space of deterministic shuffling rules. Each candidate rule maps epoch-level training statistics, such as loss values or gradient norms from previous epochs, to a permutation of the dataset indices. To ensure practical feasibility, the search space is restricted to permutation generators with $O(n)$ computational complexity.

The pipeline iteratively generates candidate programs via LLM-based mutations, evaluates them on a suite of benchmark tasks, and retains high-performing rules for further refinement. This process yields a concrete shuffling algorithm, which we refer to as Adaptive Block Reshuffling with Periodic Transforms (APR). APR adaptively switches between structured blockwise permutations and reversal-based transforms depending on observed training progress.

Evolution setup. We used OpenEvolve, an LLM-guided program evolution framework, to explore adaptive data shuffling strategies for SGD. The search space was restricted to deterministic Python functions producing permutations of $\{0, \ldots, n-1\}$ in $O(n)$ time. The evolved function could access only coarse epoch-level statistics (training loss and gradient norm) and was prohibited from modifying the model, optimizer, or learning rate.

Prompt and constraints. The LLM was instructed to design shuffling rules that reduce training loss and gradient norm as rapidly as possible.

Emergent structure. The evolved programs exhibited two components:

- Grouping indices into contiguous blocks and reshuffling at the block level.
- Applying deterministic reversal or palindromic traversal of blocks, often periodically across epochs.

Role of the LLM. We emphasize that the LLM does not participate in training, inference, or optimization. It is used only offline to explore a constrained design space and suggest candidate shuffling rules. All theoretical guarantees and experimental results apply to the final APR algorithm, which is fully specified and deterministic.

E SETTINGS FOR ADAPTIVE BLOCK RESHUFFLING WITH PERIODIC TRANSFORMS (APR)

E.1 EXPLANATION OF ALGORITHM 1

Adaptive reshuffling with loss-ratio gating. Algorithm 1 generates a permutation $\pi_e \in \mathcal{S}_n$ at each epoch e based on the observed training progress. The algorithm uses an epoch-level loss ratio

$$\rho_e = \frac{\ell_e}{\ell_{e-1} + \varepsilon},$$

which measures relative improvement between consecutive epochs and serves as a coarse indicator of the optimization regime.

When ρ_e indicates strong improvement ($\rho_e < \tau_{\text{strong}}$), the algorithm applies *block reshuffling* with small blocks of size $b_{\text{strong}} = \max(1, \lfloor \alpha_{\text{strong}} n \rfloor)$. The dataset indices are partitioned into consecutive blocks, the blocks are randomly permuted, and the indices are processed block by block while preserving the order within each block. At specified epochs, a reversal transform is additionally applied to mitigate order-dependent effects.

When improvement is mild ($\tau_{\text{strong}} \leq \rho_e < \tau_{\text{mild}}$), the same block reshuffling procedure is used with larger blocks of size $b_{\text{mild}} = \max(1, \lfloor \alpha_{\text{mild}} n \rfloor)$, introducing more randomness while retaining partial structure. When no improvement is detected ($\rho_e \geq \tau_{\text{mild}}$), the algorithm falls back to full random reshuffling, optionally followed by an even-odd interleaving transform to reduce local correlations.

All randomization is controlled by a deterministic epoch-dependent seed, ensuring reproducibility. Importantly, Algorithm 1 always outputs a single valid permutation per epoch and therefore remains within the standard without replacement SGD framework. This allows the method to be analyzed using unified shuffling theory while enabling structured modifications of the data order in response to training dynamics.

Sequence transforms. Algorithm 1 optionally applies simple deterministic transforms to the generated permutation. The reversal operator $\mathrm{Rev}(\cdot)$ reverses the order of indices within an epoch, i.e., $\mathrm{Rev}(x_1, \ldots, x_n) = (x_n, \ldots, x_1)$. This transform is used to counteract systematic order effects arising from sequential data processing.

The even-odd interleaving operator $\mathrm{EO}(\cdot)$ reorders a sequence by listing all odd-indexed elements first, followed by all even-indexed elements, $\mathrm{EO}(x) = (x_1, x_3, \ldots; x_2, x_4, \ldots)$. This operation increases the typical distance between originally adjacent indices and helps reduce correlations induced by data locality. Both operators preserve the without replacement structure and introduce no additional randomness or computational overhead.

E.2 Experimental Setup for APR

All experiments are conducted using a specific instance of Adaptive Block Reshuffling with Periodic Transforms (APR) discovered by an LLM-guided program evolution framework. The LLM proposes concrete reshuffling rules with fixed parameter choices based on training feedback, and the instance reported here was selected due to its consistent empirical performance across datasets. Importantly, this instance always produces a single permutation per epoch and therefore remains within the standard without-replacement SGD framework.

APR instance and parameter settings. The LLM-discovered reshuffling rule corresponds to Algorithm 1 with the following fixed parameters:

- **Loss-ratio thresholds:** $\tau_{\mathrm{strong}} = 0.9$ and $\tau_{\mathrm{mild}} = 1.0$.
- **Block fractions:** $\alpha_{\mathrm{strong}} = 0.1$ (small blocks) and $\alpha_{\mathrm{mild}} = 0.2$ (medium blocks).
- **Reversal schedule:** reversal is applied every $p_{\mathrm{rev}} = 3$ epochs with phase $r_{\mathrm{rev}} = 0$.
- **Even–odd interleaving:** applied every $p_{\mathrm{eo}} = 3$ epochs with phase $r_{\mathrm{eo}} = 1$.
- **Initialization:** the first epoch uses a fully random permutation.

At each epoch e, the reshuffling rule computes the loss ratio

$$\rho_e = \frac{\ell_e}{\ell_{e-1} + \varepsilon},$$

with $\varepsilon = 10^{-10}$ for numerical stability. When $\rho_e < \tau_{\mathrm{strong}}$, the algorithm applies block reshuffling with small blocks, followed by periodic reversal. When $\tau_{\mathrm{strong}} \le \rho_e < \tau_{\mathrm{mild}}$, block reshuffling with larger blocks is used. When $\rho_e \ge \tau_{\mathrm{mild}}$, a full random reshuffling is performed, optionally followed by even-odd interleaving.