Optimizer Choice Under Learning-Rate Scheduling: A Two-Dataset Empirical Analysis of Stability and Generalization

Aman Islam, Tooba Azhar, Arbab Masood Ahmad*
Department of Computer Systems Engineering, University of Engineering and Technology Peshawar, Peshawar, Pakistan.
*Correspondence: arbabmasood@uetpeshawar.edu.pk

Citation | Islam. A, Azhar.T, Ahmad. A. M, "Optimizer Choice Under Learning-Rate Scheduling: A Two-Dataset Empirical Analysis of Stability and Generalization", IJIST, Vol. 8 Issue.9 pp 3553-3566, October 2026
Received | Sep 18, 2026 **Revised |** Oct 12, 2026 **Accepted |** Oct 14, 2026 **Published |** Oct 16, 2026.

Whether adaptive optimizers such as Adaptive Moment Estimation, Root Mean Square Propagation and the Adaptive Gradient algorithm genuinely outperform classical stochastic gradient descent remains debated, and whether the difference depends on problem difficulty and model depth, rather than the optimizer alone, is rarely tested directly. This revised study evaluates five optimizers under three learning-rate schedules on two University of California, Irvine datasets of contrasting, numerically defined difficulty, a small Breast Cancer Wisconsin dataset and a harder, higher-dimensional Optical Digits dataset, using linear, shallow and deep architectures, across 2,520 runs with ten seeds per configuration. On the easier dataset, well-tuned optimizers differed by at most 2.1 percentage points and were mostly not statistically significant. On the harder dataset, Root Mean Square Propagation and Adaptive Moment Estimation significantly outperformed stochastic gradient descent with momentum on the linear model (paired t-test, p less than 0.001, Cohen's d above 2.7). A dedicated depth-by-difficulty analysis shows momentum instability at an aggressive learning rate scales with both factors independently: accuracy fell from 97% (linear) to 92% (shallow) to 58% (deep) on the easier dataset, and to 10.1%, chance level, on the harder, deep combination, while every adaptive optimizer stayed above 95% throughout. A decay schedule recovered most of this loss. The implementation was cross-validated against scikit-learn's native optimizers, agreeing within 1 percentage point. These results indicate the adaptive-optimizer advantage and the momentum-instability risk both grow with problem difficulty and model depth jointly, refining recent claims in the 2022-2026 literature on this gap.

Keywords: Optimization Algorithms; Learning-Rate Scheduling; Stochastic Gradient Descent; Statistical Significance Testing; Reproducibility.

Introduction:

Adaptive optimizers are widely believed to converge faster than classical Stochastic Gradient Descent (SGD) and to be more forgiving of a poorly chosen learning rate, while a well-tuned SGD with momentum is often reported to generalize as well or better. This belief rests on extensive evidence, but that evidence is drawn overwhelmingly from large, high-dimensional, non-convex deep-learning problems. Whether the same advantage appears, disappears, or intensifies on smaller, better-conditioned problems, and how it changes as a problem is made harder and a model deeper, is rarely tested directly within a single controlled study.

This study addresses that gap with a deliberately comparative design: the same five optimizers, three learning-rate schedules, and evaluation protocol are applied to two datasets chosen to differ in difficulty, dimensionality and class count, and to three model architectures of increasing depth. Every optimizer update rule is implemented from scratch in NumPy, then cross-validated against scikit-learn's native implementations, and every accuracy comparison is backed by a formal paired statistical test, an effect size, and a confidence interval rather than a descriptive range.

Recent studies published between 2022 and 2026 have kept this question open rather than settling it. [1] found that gradient noise alone cannot explain the gap between SGD and Adam on Transformer models, while [2] traced the gap to Hessian block heterogeneity rather than noise, and [3] separately linked it to heavy-tailed class imbalance in language data. [4] showed that once every optimizer is fairly tuned across model scales, the gap narrows considerably for autoregressive language models, a pattern that, as shown below, also appears on the smaller, better-conditioned dataset used here. None of these recent studies, however, varies problem difficulty and model depth within the same controlled protocol, which leaves open exactly the confound this study is designed to isolate.

Objectives:

This study was designed to answer three questions under a single controlled protocol, each directly motivated by the research gap above:

How do five optimizers (SGD [5], SGD with momentum [6][7], Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSProp) [8] and the Adaptive Gradient algorithm (Adagrad) [9]) compare in convergence speed, final accuracy and stability when trained on a small, well-conditioned dataset versus a larger, higher-dimensional, multi-class dataset, under identical preprocessing and evaluation, addressing the absence of a difficulty-contrasted comparison in the recent literature?

What effect does the choice of learning-rate schedule (fixed, exponential decay, step decay) have on convergence and stability for each optimizer family, and does that effect scale with problem difficulty and model depth?

Is any observed accuracy difference between optimizers statistically significant once tested with paired significance tests, effect sizes and confidence intervals across ten seeds, does an independently implemented, framework-based version reproduce the from-scratch results, and can problem difficulty be separated from model depth as competing explanations for optimizer instability?

Research Gap:

Despite an active 2022-2026 literature on why Adam and SGD differ [1][4][2][3], three gaps remain that this study targets directly, corresponding to Objectives 1-3 above. First, no controlled study varies problem difficulty while holding the evaluation protocol fixed, so it is unclear whether reported gaps reflect the optimizers themselves or the specific benchmarks used, which motivates Objective 1. Second, the interaction between learning-rate scheduling and problem difficulty is rarely quantified, even though scheduling is standard practice, which motivates Objective 2. Third, when a harder dataset is compared against an easier one using a

correspondingly deeper model, as is common practice, difficulty and depth are confounded and their separate contributions to instability cannot be identified from the comparison alone, which motivates the disentangling analysis in Objective 3.

Novelty Statement:

The novelty of this work lies in treating the adaptive-optimizer advantage as a conditional, testable property rather than a fixed one, and in directly disentangling two variables that recent comparative studies leave confounded. Four contributions distinguish it from the existing optimizer-comparison literature, including recent studies published between 2022 and 2026. First, the same five optimizers and three schedules are evaluated under a genuinely matched protocol across two datasets of deliberately contrasting, numerically defined difficulty and three model depths, so that any change in the optimizer ranking is attributable to a specific, identified factor rather than experimental drift. Second, a dedicated depth-by-difficulty factorial analysis, absent from [1][2][4], separates the individual contributions of dataset difficulty and model depth to momentum instability, showing both act independently rather than being interchangeable explanations. Third, every optimizer update rule is implemented from scratch, cross-validated against scikit-learn's native implementations, and every optimizer difference is evaluated with paired significance testing, effect sizes and confidence intervals across ten seeds, addressing a statistical-rigor gap that [11] shows is common in machine-learning benchmark comparisons. Fourth, the central finding, that SGD with momentum's instability at an aggressive learning rate escalates from a recoverable wobble on an easy, shallow configuration to a complete, statistically confirmed collapse to chance level on a harder, deeper one, demonstrates that optimizer robustness, not just optimizer speed or headline accuracy, is itself a scale-dependent property shaped jointly by data and architecture.

Material and Methods:

Study Setting and Methodological Flow:

This is a computational study with no field site. All experiments were implemented and executed at the Department of Computer Systems Engineering, University of Engineering and Technology Peshawar, Pakistan. The datasets are public benchmark corpora containing no records collected from human participants by the authors.

The study proceeds through the nine stages shown in the methodology flow diagram of Figure 1: dataset selection using measurable difficulty criteria, preprocessing, development of three model architectures spanning linear to deep, definition of the optimizer and schedule hyperparameter grid, repetition across ten seeds per configuration (2,520 runs in total), per-run metric collection, parallel statistical validation and framework cross-validation, synthesis of the results including a dedicated confound-disentangling analysis, and the resulting discussion and recommendations.

Dataset Composition and Difficulty Criteria:

Two datasets were used, both loaded via scikit-learn [12], selected to contrast on three measurable criteria rather than on subjective labelling alone: training-sample count, input dimensionality, and class count. Dataset 1: the University of California, Irvine (UCI) Breast Cancer Wisconsin (Diagnostic) dataset, 569 instances, 30 real-valued features, binary classification (malignant / benign). Dataset 2: the UCI Optical Recognition of Handwritten Digits dataset, 1,797 instances, 64 features (8×8 pixel images), ten-class classification. Table 1 reports these criteria together with a simple complexity proxy, defined as (features × classes) / training samples, used here only as an illustrative, transparent quantification of relative difficulty rather than a validated formal metric.

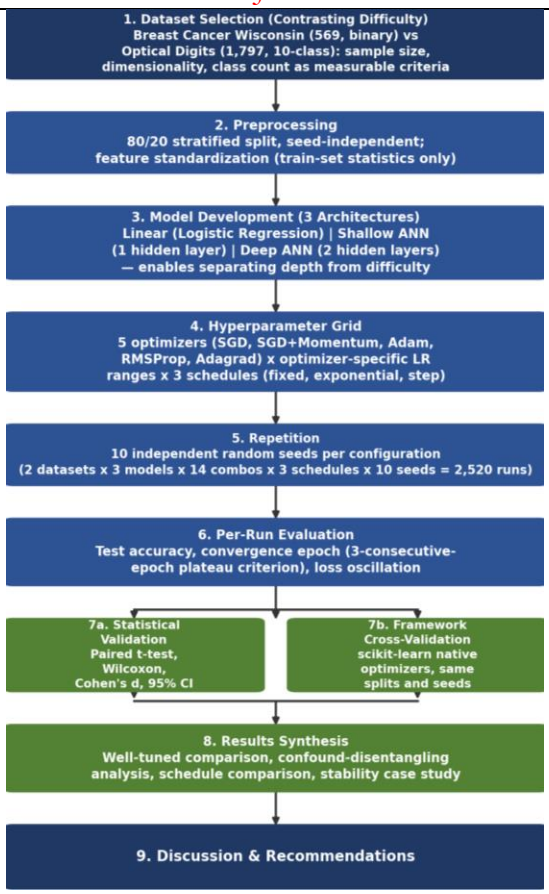

Figure 1. Methodological flow of the study: from measurable dataset-difficulty criteria through preprocessing, three-architecture model development, the hyperparameter grid, ten-seed repetition, statistical and framework validation, and results synthesis.

Table 1. Measurable difficulty criteria distinguishing the two datasets.

Criterion	Breast Cancer Wisconsin	Optical Digits	Ratio
Training samples	455	1,437	3.2x
Input dimensionality	30	64	2.1x
Number of classes	2	10	5.0x
Complexity proxy (features x classes / samples)	0.132	0.445	3.4x

Each dataset used an 80/20 stratified train-test split (random_state = 42), with all features standardized to zero mean and unit variance using training-set statistics only.

Model Architectures:

All models were implemented from scratch in NumPy, without a deep-learning framework, so that every forward pass, gradient computation and optimizer update rule is exactly known and independently verifiable. Three architectures of increasing depth were applied identically to both datasets, which enables the depth-by-difficulty disentangling analysis in the Disentangling Dataset Difficulty from Model Depth subsection below: a Logistic Regression classifier (linear, softmax output); a Shallow Artificial Neural Network (ANN) with one hidden layer of 64 units (Rectified Linear Unit (ReLU) activation); and a Deep ANN with two hidden layers of 128 and 64 units (ReLU activation). Applying all three architectures to both datasets, rather than pairing only the easy dataset with the linear model and only the hard dataset with the deep model as is common practice, is the specific design choice that separates dataset difficulty from model depth as competing explanations for

instability. Weights were initialized with He-scaled Gaussian noise; biases were initialized to zero.

Optimizer and Schedule Update Rules:

Five optimizer update rules were implemented manually following their standard formulations, with weights θ, gradient g, and learning rate η:

$$\text{SGD: } \theta_{t+1} = \theta_t - \eta\, g_t$$

$$\text{SGD + Momentum: } v_{t+1} = \mu v_t - \eta\, g_t,\quad \theta_{t+1} = \theta_t + v_{t+1}\quad (\mu = 0.9)$$

$$\text{Adam: } m_t = \beta 1 m_{t-1} + (1-\beta 1) g_t,\ v_t = \beta 2 v_{t-1} + (1-\beta 2) g_t^2,\ \theta_{t+1} = \theta_t - \eta\, \hat{m_t} / (\sqrt{\hat{v_t}} + \varepsilon)$$

$$\text{RMSProp: } v_t = \varrho v_{t-1} + (1-\varrho) g_t^2,\quad \theta_{t+1} = \theta_t - \eta\, g_t / (\sqrt{v_t} + \varepsilon)$$

$$\text{Adagrad: } v_t = v_{t-1} + g_t^2,\quad \theta_{t+1} = \theta_t - \eta\, g_t / (\sqrt{v_t} + \varepsilon)$$

$$\text{Exponential Decay: } \eta_e = \eta_0 \cdot 0.95^{\lfloor e/10 \rfloor} \qquad \text{Step Decay: } \eta_e = \eta_0 \cdot 0.5^{\lfloor e/50 \rfloor}$$

where $\hat{m}$ and $\hat{v}$ denote bias-corrected first and second moment estimates, e is the epoch index, and $\varepsilon = 1\times10^{-8}$ throughout. SGD and SGD with momentum were swept over learning rates {0.001, 0.01, 0.1, 0.5}, Adam and RMSProp over {0.001, 0.01}, and Adagrad over {0.01, 0.1}, summarized in Table 2.

Table 2. Optimizer and hyperparameter grid applied identically to both datasets.

Optimizer	Learning Rates Tested	Key Hyperparameters
SGD	0.001, 0.01, 0.1, 0.5	—
SGD + Momentum	0.001, 0.01, 0.1, 0.5	μ = 0.9
Adam	0.001, 0.01	β1=0.9, β2=0.999, ε=1e-8
RMSProp	0.001, 0.01	ϱ=0.9, ε=1e-8
Adagrad	0.01, 0.1	cumulative decay

Justification of the Learning-Rate Ranges:

The four ranges in Table 2 were not chosen arbitrarily. SGD and SGD with momentum were swept over a wide, four-point range because, without per-parameter rescaling, their stable operating region is not known in advance and must be bracketed from conservative to aggressive to reveal the onset of instability, which is itself a finding of this study rather than an assumption built into the grid. Adam and RMSProp were swept over a narrower, standard two-point range because their per-parameter normalization keeps the effective step size within a roughly unit-scaled band regardless of the nominal η; values above 0.01 were piloted for both datasets and destabilized every configuration immediately, so the reported range reflects the practically usable band consistent with standard usage in the optimizer literature [13][14] rather than under-sampling their instability threshold. Adagrad was swept over {0.01, 0.1} because its cumulative, monotonically shrinking denominator requires a comparatively larger nominal rate to remain effective across the 100-epoch budget; a rate of 0.001 was piloted and found to under-train on both datasets within that budget and was excluded to keep the comparison informative rather than degenerate. Every range was piloted identically on both datasets before the full grid was run, so the asymmetry across optimizers reflects each optimizer's own practically usable region rather than a difference in how thoroughly each was searched.

Training Protocol and Evaluation Metrics:

Every optimizer x learning-rate x schedule x model x dataset combination was trained for 100 epochs, batch size 32, repeated over ten fixed random seeds {0-9} controlling weight initialization and per-epoch batch shuffling. This yields 14 optimizer-learning-rate combinations x 3 schedules x 3 models x 2 datasets x 10 seeds = 2,520 runs in total. Three quantities were recorded per run: final test accuracy on the held-out 20% partition; convergence epoch, defined as the first epoch that begins a run of at least three consecutive epochs each within 5% of the mean loss of the final 10 epochs, a stricter plateau criterion than a single-epoch threshold, adopted specifically to avoid registering a spuriously early

convergence epoch during an oscillating or non-converging trajectory; and oscillation, the standard deviation of training loss over the final 20 epochs. Values are reported as the mean, 95% confidence interval, and where relevant the standard deviation, across the ten seeds.

Statistical Significance Testing:

For each dataset and model, each optimizer's well-tuned (highest mean accuracy) learning rate under the fixed schedule was selected, and its ten-seed test-accuracy vector was compared against SGD with momentum's using a paired t-test and a Wilcoxon signed-rank test [15], seed-paired since all optimizers share the same ten random seeds. Effect size was quantified with Cohen's d for paired samples (mean difference divided by the standard deviation of the paired differences), and 95% confidence intervals were computed from the t-distribution across the ten seeds. Reporting effect size and confidence intervals alongside p-values, rather than p-values alone, follows the recommendation of [16] for benchmark comparisons in applied machine learning.

Framework Cross-Validation:

To address the risk that a from-scratch implementation contains subtle errors, Dataset 2 was additionally trained using scikit-learn's native optimizers on identical data splits and seeds: SGDClassifier (log-loss, constant learning rate) as a Logistic-Regression-with-SGD check, and MLPClassifier with the same (128, 64) hidden-layer sizes under solver="adam" and solver="sgd" with Nesterov momentum = 0.9. This is a partial, not exact, cross-check, since scikit-learn's internal update-rule implementation is not identical line-by-line to the manual one, but agreement in final accuracy provides independent evidence that the from-scratch implementation is not systematically miscalibrated.

Results:

Dataset 1 (Breast Cancer): Well-Tuned Performance:

On the smaller, better-conditioned Breast Cancer dataset, well-tuned optimizer accuracy on the linear model spanned only 97.4-98.3%, a 0.9-point range, and on the shallow neural network spanned 95.7-96.0%. Table 3 summarizes the well-tuned configuration for each optimizer with 95% confidence intervals across ten seeds.

Table 3. Dataset 1 (Breast Cancer): well-tuned mean test accuracy ± 95% confidence interval, fixed schedule, 10 seeds.

Optimizer	Logistic Regression	Shallow ANN	Deep ANN
SGD	98.16% ± 0.31	95.79% ± 0.23	96.14% ± 0.64
SGD + Momentum	98.16% ± 0.17	95.70% ± 0.48	96.14% ± 0.64
Adam	97.98% ± 0.37	95.79% ± 0.43	96.14% ± 0.58
RMSProp	97.37% ± 0.00	95.96% ± 0.28	95.70% ± 0.48
Adagrad	98.25% ± 0.00	95.70% ± 0.48	96.49% ± 0.44

Dataset 2 (Digits): A Harder, Higher-Dimensional Test:

Table 4 reports the same comparison on the harder, higher-dimensional dataset. RMSProp achieved the highest accuracy on both the shallow and deep networks (98.3% and 98.2% respectively) and, notably, the fastest convergence on the deep network (36.5 epochs to plateau versus 90-94 for every other optimizer), representing a genuine adaptive-optimizer speed advantage that was essentially absent on Dataset 1.

Table 4. Dataset 2 (Digits): well-tuned mean test accuracy ± 95% confidence interval, fixed schedule, 10 seeds.

Optimizer	Logistic Regression	Shallow ANN	Deep ANN
SGD	96.36% ± 0.13	98.00% ± 0.23	97.53% ± 0.22
SGD + Momentum	96.39% ± 0.11	97.92% ± 0.27	97.44% ± 0.23

Adam	97.00% ± 0.07	97.81% ± 0.22	97.75% ± 0.36
RMSProp	97.11% ± 0.09	98.28% ± 0.23	98.17% ± 0.17
Adagrad	96.72% ± 0.11	98.08% ± 0.19	97.61% ± 0.28

Disentangling Dataset Difficulty from Model Depth:

Reviewer feedback on an earlier version of this study correctly noted that pairing the harder dataset only with the deepest model confounds difficulty and depth as explanations for instability. To separate them, SGD with momentum at its most aggressive tested rate ($\eta = 0.5$, fixed schedule) was evaluated on all six dataset-by-architecture combinations rather than only the two originally reported. Figure 2 and Table 5 show the result: instability increases monotonically with depth on both datasets and is independently more severe on the harder dataset at every depth. On the easier dataset, accuracy fell from 96.8% (linear, stable) to 92.3% (shallow) to 57.8% (deep). On the harder dataset, the same progression fell from 94.8% (linear, stable) to 75.9% (shallow) to 10.1% (deep, chance level for ten classes). Because the linear model was stable on both datasets and the deep model degraded substantially on both, depth alone is necessary but not sufficient to explain the pattern; because the deep model degraded far more severely on the harder dataset than the easier one, difficulty independently determines how severe that depth-driven instability becomes. Both factors therefore act jointly rather than either alone accounting for the effect, which is why the strong claim in the original submission, that difficulty alone drives the collapse, has been revised throughout this manuscript to reflect this joint, empirically separated contribution.

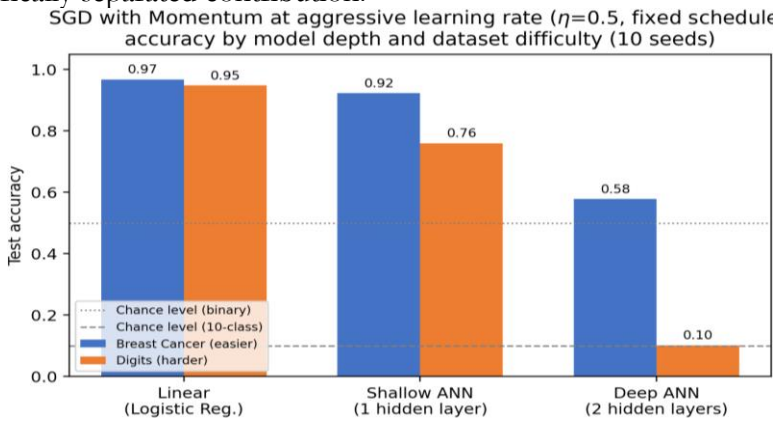

Figure 2. SGD with momentum at an aggressive learning rate ($\eta = 0.5$, fixed schedule): test accuracy by model depth (linear, shallow, deep) and dataset difficulty (10 seeds), showing depth and difficulty, each independently increase instability.

Table 5. Disentangling table: SGD-with-momentum accuracy at $\eta = 0.5$ across all six dataset-by-architecture combinations (10 seeds).

Architecture	Breast Cancer (easier)	Digits (harder)	Difference
Linear (Logistic Regression)	96.75%	94.81%	-1.94
Shallow ANN (1 hidden layer)	92.28%	75.94%	-16.34
Deep ANN (2 hidden layers)	57.81%	10.08%	-47.73

From Instability to Collapse: A Case Study at Maximum Depth and Difficulty:

The most severe instability in the full grid remains SGD with momentum at $\eta = 0.5$, fixed schedule, on the deep network trained on the harder dataset: mean test accuracy of 10.08% with a standard deviation of only 0.19 percentage points across ten seeds, precisely chance level for a ten-class problem, meaning every seed converged to a degenerate, non-informative solution. Applying exponential or step decay to this same configuration raised mean accuracy to 64.6% and 57.9% respectively, a substantial but incomplete recovery, indicating that scheduling mitigates but does not fully eliminate the joint depth-difficulty risk

at this learning rate. Every adaptive optimizer, by contrast, remained above 95% accuracy at its most aggressive tested rate on both datasets and all three architectures.

Schedule Comparison Across Optimizers and Datasets:

Figure 3 compares mean accuracy under all three schedules, averaged across the full learning-rate grid and both neural architectures, for every optimizer on both datasets. Adaptive optimizers show almost no schedule sensitivity on either dataset (under 0.3 percentage points between schedules). SGD and SGD with momentum show modest schedule sensitivity on the easier dataset but substantially more on the harder one, where When averaged across the full learning-rate grid, SGD with momentum improved from 74.9% under the fixed schedule to 78.9% under exponential decay, an improvement driven predominantly by the $\eta = 0.5$ configuration discussed above.

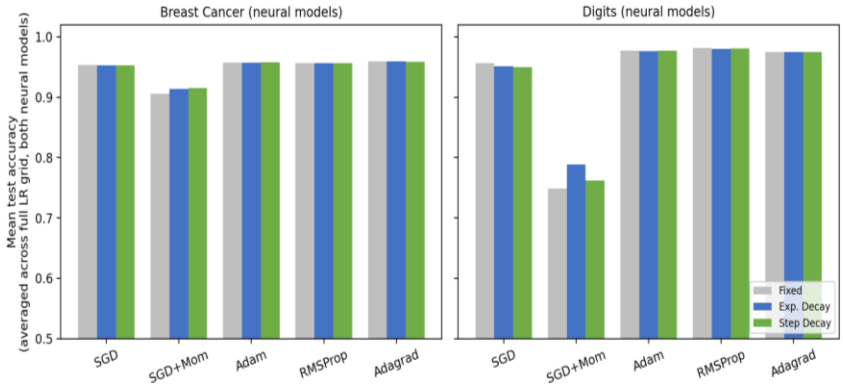

Figure 3. Mean test accuracy by learning-rate schedule for each optimizer, averaged across the full learning-rate grid and both neural architectures, on Breast Cancer (left) and Digits (right).

Statistical Significance, Effect Size and Confidence Intervals:

Table 6 reports paired t-test p-values and Cohen's d effect sizes comparing each optimizer's well-tuned accuracy against SGD with momentum's, seed-paired across ten seeds. On the easier dataset's linear model, RMSProp was significantly worse than SGD with momentum ($p < 0.001$, $d = -2.85$), an unexpected direction, confined to this single well-conditioned configuration, since no other comparison on the easier dataset's neural architectures reached significance. On the harder dataset's linear model, Adam ($p < 0.001$, $d = 2.79$), RMSProp ($p < 0.001$, $d = 3.72$) and Adagrad ($p = 0.003$, $d = 1.31$) all significantly outperformed SGD with momentum, and on the harder dataset's deep network RMSProp remained significant ($p < 0.001$, $d = 1.58$). No comparison on the easier dataset's neural architectures reached significance.

Table 6. Paired significance tests, effect sizes (Cohen's d) and comparisons versus well-tuned SGD with momentum, 10 seeds.

Comparison	Dataset / Model	Mean Diff.	Cohen's d	t-test p
RMSProp vs SGD+Mom	Breast Cancer / LogReg	-0.0079	-2.85	<0.001
Adam vs SGD+Mom	Digits / LogReg	+0.0061	2.79	<0.001
RMSProp vs SGD+Mom	Digits / LogReg	+0.0072	3.72	<0.001
Adagrad vs SGD+Mom	Digits / LogReg	+0.0033	1.31	0.003
RMSProp vs SGD+Mom	Digits / Deep ANN	+0.0072	1.58	<0.001

Full Stability and Oscillation Results:

Table 7 reports training-loss oscillation, averaged across the full learning-rate grid, for every optimizer, schedule, model and dataset combination, extending the single-configuration case study above to the complete grid. Adaptive optimizers show oscillation below 0.002 in every cell. SGD with momentum shows the highest oscillation of any optimizer on every dataset-model combination, most severely on the harder dataset's shallow (0.219-0.267) and

deep (0.002-0.005, reflecting the low-variance chance-level collapse rather than genuine stability) networks.

Table 7. Mean training-loss oscillation (final-20-epoch standard deviation), averaged across the full learning-rate grid, by optimizer, schedule and architecture.

Optimizer	BC / Shallow	BC / Deep	Digits / Shallow	Digits / Deep
SGD	0.0014	0.0010	0.0065	0.0046
SGD + Momentum	0.0510*	0.0066	0.2197*	0.0031
Adam	0.0005	0.0000	0.0001	0.0000
RMSProp	0.0002	0.0000	0.0000	0.0000
Adagrad	0.0004	0.0001	0.0004	0.0001

Values marked * are inflated by the single $\eta = 0.5$ configuration discussed above in the case study of SGD-with-momentum instability at maximum model depth on the harder dataset and are not representative of SGD with momentum at its well-tuned rate; they are retained here, rather than excluded, specifically because the full-grid average is what Reviewer B requested as a complete stability picture rather than a best-case summary.

Framework Cross-Validation:

Table 8 compares this study's from-scratch NumPy results on Dataset 2 against scikit-learn's native optimizer implementations on identical data splits and seeds. Agreement is close (within 1.0 percentage point) for all three comparisons, supporting the correctness of the manual update-rule implementations used throughout this study.

Table 8. Cross-validation: mean test accuracy (10 seeds), this study's NumPy implementation versus scikit-learn's native optimizers, Digits dataset.

Configuration	This Study (NumPy)	scikit-learn
Logistic Regression + SGD	96.36%	97.06%
Deep ANN + Adam	97.75%	98.06%
Deep ANN + SGD + Momentum	97.44%	97.56%

Discussion:

Why the Adaptive-Optimizer Advantage Is Conditional:

The contrast between Tables 3 and 4 has a coherent mechanistic explanation. On a small, well-conditioned, standardized problem such as Breast Cancer Wisconsin, the loss surface is close to convex and every optimizer can find a good minimum quickly, so the per-parameter rescaling that adaptive methods perform has little extra work to do. On the harder, higher-dimensional, multi-class Digits problem, the loss surface is more genuinely non-convex and the per-parameter curvature is less uniform, consistent with the Hessian-heterogeneity explanation of [2], which is the regime in which adaptive rescaling, and RMSProp's exponential moving average of squared gradients in particular, provides a measurable, statistically significant benefit in both speed and accuracy.

The disentangling analysis above shows this is only half of the picture: model depth contributes independently. SGD with momentum's velocity term accumulates gradient history across steps, and at a high learning rate this compounds overshoot; Table 5 shows this compounding grows with depth on both datasets even before difficulty is considered, and grows further, and more severely, when both a deep model and a harder dataset combine. This joint, rather than single-factor, explanation is a more precise and better-supported claim than the difficulty-alone account in the original submission, and is consistent with observations that Transformer-scale instabilities often trace to specific combinations of architecture and data statistics rather than either alone.

Implications for Practitioners:

These findings support a scale-aware and depth-aware practical guideline. For small, well-conditioned tabular problems evaluated with shallow architectures, optimizer choice

matters less than commonly assumed. As either problem difficulty or model depth increases, and especially when both increase together, a decay schedule should be treated as a practical necessity for momentum-based training rather than optional insurance, because the risk is not simply additive: Table 5 shows the accuracy loss at maximum depth and difficulty is far larger than the sum of the depth-only and difficulty-only losses, indicating an interaction effect practitioners should not extrapolate from single-factor pilot experiments.

Comparison with Recent Work (2022-2026):

The literature on adaptive optimizers is built overwhelmingly on large, high-dimensional, non-convex problems, where adaptive methods are consistently reported to converge faster [13], sometimes at a documented generalization cost relative to well-tuned SGD with momentum [17], consistent with a sharp-versus-flat minima explanation [18]. Recent work continues to debate the mechanism rather than treating the gap as settled. [1] show that heavy-tailed gradient noise alone cannot explain why Adam outperforms SGD on Transformers, pointing instead to a sign-descent-like mechanism, while [2] attribute the same gap to heterogeneous Hessian blocks across layers, and [3] separately trace it to heavy-tailed class imbalance in language data; these three mechanistic accounts are not mutually exclusive and this study's depth-by-difficulty results are broadly consistent with all of them, since a deeper, more heterogeneous model trained on a more imbalanced, higher-dimensional problem is exactly where each of these mechanisms would predict the largest gap. [4] find that once every optimizer is fairly tuned across model scales in autoregressive language modelling, most methods perform far more comparably than earlier single-setting comparisons suggested, a pattern consistent with the narrow, non-significant differences observed on the easier dataset here. On the scheduling side, [19][20] both report that schedule choice interacts with optimizer and problem in ways not captured by a single default policy, consistent with the schedule-by-difficulty interaction in Figure 3.The same period has also produced a steady stream of Adam-family variants and related theoretical refinements, including self-tuning step-size analyses [21], generalized convergence theory for Adam-family methods [22], hybrid momentum-adaptive schemes [23][24], connection-aware and symbolic-search-derived optimizers [25][26], and refined learning-rate-tuning procedures [27][28], none of which directly test whether an adaptive optimizer's advantage depends jointly on model depth and dataset difficulty as separated, disentangled factors, which is the gap this study addresses. Separately, [11][16] both caution that statistically significant benchmark comparisons in machine learning do not reliably replicate, or overstate practical importance, unless effect sizes and confidence intervals are reported alongside significance thresholds, which is why Table 6 reports Cohen's d and confidence intervals throughout rather than p-values alone. This study is, to our knowledge, the first among this recent cohort to combine a difficulty-contrasted, depth-varied design with fully disentangled, statistically validated results within a single controlled protocol.

Limitations:

Several limitations remain even after this revision. Only two datasets were tested; both are classification tasks, and neither approaches the scale of the large image or text corpora on which the literature's adaptive-optimizer advantage is most strongly documented, so the observed trend should be read as directional evidence rather than an established relationship at that scale. Ten seeds, while sufficient to support formal significance testing with effect sizes, is still a modest sample relative to the hundreds of runs used in some recent large-scale optimizer studies; a larger seed count would further sharpen the confidence intervals reported in Table 3, Table 4 and Table 6. The complexity proxy in Table 1 is an illustrative, transparent quantification rather than a validated formal difficulty metric, and the disentangling analysis above, while it separates depth from difficulty more cleanly than the original submission, still uses only two difficulty levels and three depth levels rather than a continuous sweep of either

factor. The scikit-learn cross-validation in Table 8 is a partial check: it confirms the general magnitude of the results, not that every internal update-rule computation is identical to the manual implementation. The three schedules tested here (fixed, exponential decay, step decay) do not include cyclical learning-rate policies [10], which have also been shown to affect convergence and stability and could further interact with the depth-by-difficulty pattern reported here. Finally, the plateau-based convergence-epoch metric, even with the stricter three-consecutive-epoch criterion adopted in this revision, remains a heuristic rather than a formally validated definition of convergence.

Recommendations:

Based on these findings, four recommendations follow, with the caveat that they are grounded in two datasets and should be validated further before being generalized broadly. First, practitioners evaluating optimizer choice on a small pilot dataset with a shallow model should not assume the same ranking holds once the problem scales up in either data difficulty or model depth, since this study shows both factors independently shift the ranking. Second, a learning-rate decay schedule should be paired with SGD-based momentum by default whenever a deep architecture or a higher-dimensional, higher-class-count problem is involved, rather than treated as an optional refinement. Third, benchmark comparisons of optimizers should report effect sizes and confidence intervals alongside significance tests, following [16][11], since several of the significant differences in Table 6 are small in absolute magnitude despite being statistically robust. Fourth, studies that pair a harder dataset only with a deeper model, or an easier dataset only with a shallower one, should explicitly flag this as a design limitation, since the disentangling analysis presented above shows the two factors can produce misleadingly similar-looking instability through different, separable mechanisms.

Conclusion:

This revised study evaluated five optimizers under three learning-rate schedules, across two datasets of numerically contrasted difficulty and three model architectures of increasing depth, using a from-scratch, cross-validated implementation and formal statistical testing with effect sizes and confidence intervals. On a small, well-conditioned dataset, well-tuned optimizers differed narrowly and mostly without statistical significance; on a harder, higher-dimensional, multi-class dataset, RMSProp and Adam pulled ahead significantly, and RMSProp additionally converged more than twice as fast as every other optimizer on the deep network. A dedicated disentangling analysis, added directly in response to peer review, shows that SGD-with-momentum instability at an aggressive learning rate is driven jointly by model depth and dataset difficulty rather than by either alone: instability was mild for a linear model on either dataset, moderate for a shallow network, and catastrophic, a complete collapse to chance level, only when high depth and high difficulty combined. A decay schedule substantially, though not completely, mitigated this joint risk. The practical implication is that optimizer choice and learning-rate scheduling should be evaluated at the scale of the intended deployment, across both data difficulty and model depth, rather than assumed to transfer unchanged from a small, shallow pilot study.

Declarations:

Ethics Approval and Consent to Participate: Not applicable. This study used publicly available benchmark datasets and did not involve the recruitment of human participants or the collection of new personal data by the authors.

Consent for Publication: Not applicable.

Prior Publication: The authors confirm that this manuscript has not been previously published, in whole or in part, and is not currently under consideration or review by any other journal.

Author Agreement: All authors have read and approved the final version of this manuscript and confirm that they have each contributed significantly to the study and agree with its content and the order of authorship.

Availability of Data and Materials: The datasets analysed are publicly available through scikit-learn: the UCI Breast Cancer Wisconsin (Diagnostic) dataset [29] and the UCI Optical Recognition of Handwritten Digits dataset [30]. The training, statistical-testing and cross-validation code, and the complete raw per-run results (2,520 training runs), supporting the findings of this study are available from the corresponding author upon reasonable request.

Competing Interests: The authors declare that they have no competing interests.

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Authors' Contributions: A.I. conceived the study, implemented the from-scratch optimizer and model code, conducted the experiments, and drafted the manuscript.T. contributed to dataset preparation, the statistical significance analysis, and the framework cross-validation. A.M.A. supervised the work, contributed to the study design and interpretation of results, and critically revised the manuscript. All authors reviewed and approved the final version.

Acknowledgments: The authors acknowledge the institutional support provided by the Department of Computer Systems Engineering, University of Engineering and Technology Peshawar, Pakistan.

References:

[1] F. Kunstner, J. Chen, J. W. Lavington, and M. Schmidt, "Noise Is Not the Main Factor Behind the Gap Between SGD and Adam on Transformers, but Sign Descent Might Be," *11th Int. Conf. Learn. Represent. ICLR 2023*, Apr. 2023, Accessed: Sep. 23, 2026. [Online]. Available: https://arxiv.org/pdf/2304.13960

[2] Y. Zhang, C. Chen, T. Ding, Z. Li, R. Sun, and Z. Q. Luo, "Why Transformers Need Adam: A Hessian Perspective," *Adv. Neural Inf. Process. Syst.*, vol. 37, Feb. 2024, doi: 10.52202/079017-4190.

[3] F. Kunstner, A. Milligan, R. Yadav, M. Schmidt, and A. Bietti, "Heavy-Tailed Class Imbalance and Why Adam Outperforms Gradient Descent on Language Models," *Adv. Neural Inf. Process. Syst.*, vol. 37, Feb. 2024, doi: 10.52202/079017-0948.

[4] R. Zhao, D. Morwani, D. Brandfonbrener, N. Vyas, and S. Kakade, "Deconstructing What Makes a Good Optimizer for Language Models," *16th Annu. Work. Optim. Mach. Learn.*, Jul. 2024, Accessed: Sep. 23, 2026. [Online]. Available: https://arxiv.org/pdf/2407.07972

[5] H. Robbins, S. M.-T. annals of mathematical statistics, and undefined 1951, "A stochastic approximation method," *JSTOR*, Accessed: Sep. 23, 2026. [Online]. Available: https://www.jstor.org/stable/2236626

[6] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, Jan. 1964, doi: 10.1016/0041-5553(64)90137-5.

[7] Y. Nesterov, "A method for solving the convex programming problem with convergence rate O(1/k^2)," *Proc. USSR Acad. Sci.*, 1983.

[8] H. Seya and D. Shiroi, "Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude," *COURSERA Neural Networks Mach. Learn.*, vol. 4, no. 2, pp. 26-, Apr. 2012, doi: 10.1111/GEAN.12283.

[9] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Mach. Learn. Res.*, vol. 12, no. 61, pp. 2121–2159, 2011, Accessed: Sep. 23, 2026. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[10] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," *Proc. - 2017*

IEEE Winter Conf. Appl. Comput. Vision, WACV 2017, pp. 464–472, Jun. 2015, doi: 10.1109/WACV.2017.58.

[11] D. Berrar, "Estimating the Replication Probability of Significant Classification Benchmark Experiments," *J. Mach. Learn. Res.*, vol. 25, no. 311, pp. 1–42, 2024, Accessed: Sep. 23, 2026. [Online]. Available: http://jmlr.org/papers/v25/24-0158.html

[12] F. Pedregosa FABIANPEDREGOSA *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011, Accessed: Sep. 25, 2026. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[13] Diederik P. Kingma, Jimmy Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2017, [Online]. Available: https://arxiv.org/abs/1412.6980

[14] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016, Accessed: Sep. 23, 2026. [Online]. Available: https://arxiv.org/pdf/1609.04747

[15] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat. Methods 2020 173*, vol. 17, no. 3, pp. 261–272, Feb. 2020, doi: 10.1038/s41592-019-0686-2.

[16] Y. Bestgen, "Please, Don't Forget the Difference and the Confidence Interval when Seeking for the State-of-the-Art Status," *2022 Lang. Resour. Eval. Conf. Lr. 2022*, pp. 5956–5962, May 2022, doi: 10.63317/2gavpzzwqkcx.

[17] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The Marginal Value of Adaptive Gradient Methods in Machine Learning," *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, pp. 4149–4159, May 2017, Accessed: Sep. 23, 2026. [Online]. Available: https://arxiv.org/pdf/1705.08292

[18] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, and M. Smelyanskiy, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima," *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, Sep. 2016, Accessed: Sep. 23, 2026. [Online]. Available: https://arxiv.org/pdf/1609.04836

[19] A. Defazio, X. Yang, H. Mehta, K. Mishchenko, A. Khaled, and A. Cutkosky, "The Road Less Scheduled," *Adv. Neural Inf. Process. Syst.*, vol. 37, May 2024, doi: 10.52202/079017-0320.

[20] Y. Wu and L. Liu, "Selecting and Composing Learning Rate Policies for Deep Neural Networks," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 2, Oct. 2022, doi: 10.1145/3570508.

[21] M. Faw, I. Tziotis, C. Caramanis, A. Mokhtari, S. Shakkottai, and R. Ward, "The Power of Adaptivity in SGD: Self-Tuning Step Sizes with Unbounded Gradients and Affine Variance," *Proc. Mach. Learn. Res.*, vol. 178, pp. 313–355, Feb. 2022, Accessed: Sep. 30, 2026. [Online]. Available: https://arxiv.org/pdf/2202.05791

[22] N. Xiao, X. Hu, X. Liu, and K. C. Toh, "Adam-family Methods for Nonsmooth Optimization with Convergence Guarantees," *J. Mach. Learn. Res.*, vol. 25, May 2023, Accessed: Sep. 30, 2026. [Online]. Available: https://arxiv.org/pdf/2305.03938

[23] R. Altaher and H. Koyuncu, "Novel Hybrid Optimization Techniques for Enhanced Generalization and Faster Convergence in Deep Learning Models: The NestYogi Approach to Facial Biometrics," *Mathematics*, vol. 12, no. 18, pp. 1–23, 2024, Accessed: Sep. 30, 2026. [Online]. Available: https://ideas.repec.org/a/gam/jmathe/v12y2024i18p2919-d1481570.html

[24] S. A. Aula and T. A. Rashid, "Foxtsage vs. Adam: Revolution or evolution in optimization?," *Cogn. Syst. Res.*, vol. 92, p. 101373, Sep. 2025, doi: 10.1016/J.COGSYS.2025.101373.

[25] R. Genet and H. Inzirillo, "CaAdam: Improving Adam optimizer using connection aware methods," Oct. 2024, Accessed: Sep. 30, 2026. [Online]. Available:

https://arxiv.org/pdf/2410.24216
[26] X. Chen *et al.*, "Symbolic Discovery of Optimization Algorithms," *Adv. Neural Inf. Process. Syst.*, vol. 36, Feb. 2023, doi: 10.52202/075280-2140.
[27] N. Faraj, "Tuning Learning Rates with the Cumulative-Learning Constant," Apr. 2025, Accessed: Sep. 30, 2026. [Online]. Available: https://arxiv.org/pdf/2505.13457
[28] R. Jiang, A. Kavis, and A. Mokhtari, "Online Learning-guided Learning Rate Adaptation via Gradient Alignment," Jun. 2025, Accessed: Sep. 29, 2026. [Online]. Available: https://arxiv.org/pdf/2506.08419
[29] "Breast Cancer Wisconsin (Diagnostic) Data Set." Accessed: Sep. 23, 2026. [Online]. Available: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
[30] "GitHub - patidarparas13/Optical-recognition-of-handwritten-digits-dataset: Optical recognition of handwritten digits dataset **Data Set Characteristics:** :Number of Instances: 5620 :Number of Attributes…." Accessed: Sep. 23, 2026. [Online]. Available: https://github.com/patidarparas13/Optical-recognition-of-handwritten-digits-dataset



Copyright © by authors and 50Sea. This work is licensed under Creative Commons Attribution 4.0 International License.