Anusandhanvallari
Vol 2024, No.1
September 2024
ISSN 2229-3388

Gradient History Aggregation via Ordered Weighted Averaging for Enhanced Adaptive Optimization

Surendra Gour, Md. Tabrez Nafis, Suraiya Parveen

Department of Computer Science & Engineering, Jamia Hamdard, New Delhi – 110062, India

Abstract

We present OWA-Adam, a reformulated adaptive optimization algorithm that integrates Ordered Weighted Averaging (OWA) operators into the gradient moment estimation mechanism of Adam. Rather than relying on fixed exponential decay coefficients to weight historical gradient information, OWA-Adam employs dynamically ranked aggregation, assigning importance to prior gradient steps according to their relative magnitude rather than solely their temporal recency. This reformulation enables the optimizer to respond more meaningfully to the evolving loss landscape. Through systematic experimentation over ten statistically independent trials, we show that OWA-Adam configured with an exponential decay weighting scheme converges 38.6% faster than standard Adam while achieving statistically equivalent final model performance. Significance testing using Welch's t-test and Mann–Whitney U test confirms that performance equivalence is not coincidental but structurally robust. The proposed optimizer offers a principled and practically deployable improvement to the Adam family, with implications for training efficiency in deep learning pipelines.

Keywords: Ordered Weighted Averaging (OWA); Adaptive Moment Estimation (Adam); Deep Learning Optimization; Convergence Acceleration; Gradient Aggregation

1. Introduction

Training deep neural networks is fundamentally an optimization problem. As model architectures grow in depth and complexity, the efficiency of the underlying optimization algorithm becomes a critical determinant of both computational cost and model quality. Gradient-based methods have served as the primary engine for this process since the early days of neural network research, and the field has seen a progressive sophistication in how these methods are designed and applied.

Stochastic Gradient Descent (SGD) established the template for first-order optimization in deep learning, offering scalability through mini-batch gradient estimation. Its successors, AdaGrad, RMSProp, and ultimately Adam, introduced mechanisms for parameter-wise adaptive learning rates, allowing each weight to be updated at a pace calibrated to its gradient history. Among these, Adam has achieved widespread adoption owing to its robust convergence behavior across diverse problem settings [6].

Despite its success, Adam's mechanism for aggregating gradient history is fundamentally static. The first and second moment estimates are updated using fixed exponential decay factors $\beta 1$ and $\beta 2$, which apply geometric weighting to the entire historical sequence regardless of which gradients are most informative for the current optimization step. This is a structural limitation: gradients that encode critical curvature information may be discounted simply because they occurred several steps ago, not because they are actually less relevant.

The present work addresses this limitation by incorporating Ordered Weighted Averaging (OWA) operators into Adam's moment estimation. OWA, introduced by Yager for multi-criteria decision aggregation, assigns weights to values based on their ranked positions rather than their temporal indices. Applying this framework to gradient aggregation allows the optimizer to prioritize gradient signals by their relative magnitude, decoupling historical importance from temporal order.

We formalize this integration as OWA-Adam, investigate four distinct OWA weighting strategies, and conduct a statistically rigorous evaluation across ten independent training runs. Our central finding is that the exponential

decay OWA variant achieves substantially faster convergence than standard Adam with no loss in predictive accuracy, validated through confidence interval analysis and hypothesis testing.

2. Background and Related Work

The trajectory of optimization algorithm development in machine learning traces a path from simplicity toward adaptive sophistication. Early gradient descent formulations processed the entire dataset per update step, making them computationally prohibitive for the large-scale problems that define modern deep learning. Stochastic gradient descent alleviated this by estimating gradients over randomly sampled mini-batches, enabling scalable training at the cost of gradient variance [2].

Momentum-based methods emerged as a means of stabilizing these noisy updates. By accumulating a fraction of preceding gradient directions, momentum SGD accelerates traversal of narrow loss valleys and suppresses oscillation in high-curvature regions [2]. Nesterov Accelerated Gradient (NAG) refined this approach by evaluating the gradient at an extrapolated future position, enabling more responsive updates in rapidly changing loss landscapes [3].

The introduction of parameter-adaptive learning rates marked a significant conceptual advance. AdaGrad accumulates squared gradients across training to reduce step sizes for frequently updated parameters [1]. RMSProp improves upon AdaGrad by using an exponentially decaying window of squared gradients, preventing the monotonic learning rate decay that AdaGrad exhibits over long training runs [4]. AdaDelta removes the need for an initial learning rate altogether by constructing the update rule entirely from running gradient statistics [5].

Adam synthesizes adaptive learning rates with first-order momentum, maintaining running estimates of both the mean gradient and the mean squared gradient. Its bias-corrected updates ensure appropriate step magnitudes in early training, and its per-parameter adaptivity makes it robust across diverse architectures and loss surfaces [6]. Several variants have extended Adam's capabilities: Nadam integrates Nesterov momentum for faster directional response [7]; AMSGrad enforces monotonically non-increasing effective learning rates to ensure convergence in adversarial settings [8]; AdamW separates weight decay from gradient-based updates to improve generalization [9]; RAdam incorporates variance rectification to reduce sensitivity to learning rate during early training [10]; and Yogi modifies the second moment accumulation to prevent gradient signal suppression [11].

A less-explored dimension of Adam's design is the specific aggregation mechanism used to construct gradient moment estimates. All existing variants inherit Adam's fixed exponential weighting, which treats gradient importance as a simple function of recency. The present work departs from this assumption by applying OWA-based aggregation, where weighting is determined by the relative rank of gradient values within a maintained history window. To our knowledge, this represents the first integration of OWA operators into first-order adaptive gradient optimization.

3. Methodology

3.1 Recap of the Adam Optimizer

Adam maintains two exponentially decaying running statistics per parameter: the first moment estimate m_t (a weighted mean of past gradients) and the second moment estimate v_t (a weighted mean of past squared gradients). These are updated at each training step t as follows:

Algorithm 1: Adam (Adaptive Moment Estimation)

Require: α (step size), $\beta 1, \beta 2 \in [0,1)$ (moment decay rates), $f(\theta)$ (objective), $\theta 0$ (initial parameters)

Initialize: $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$

while θ_t not converged do:

$t \leftarrow t + 1$

$g_t \leftarrow \nabla\theta\, f_t(\theta_{t-1})$ (compute gradient)

$m_t \leftarrow \beta 1 \cdot m_{t-1} + (1 - \beta 1) \cdot g_t$

$v_t \leftarrow \beta 2 \cdot v_{t-1} + (1 - \beta 2) \cdot g_t^2$

$\hat{m}_t \leftarrow m_t / (1 - \beta 1^t)$ (bias correction)

$\hat{v}_t \leftarrow v_t / (1 - \beta 2^t)$ (bias correction)

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$

return θ_t

The scalar β1 (typically 0.9) assigns a fixed weight to the immediate prior moment estimate and a complementary weight of 1−β1 to the current gradient. This formulation effectively applies a geometric weighting across the entire gradient history, with no mechanism for adjusting this weighting based on the informational content of specific past gradients.

3.2 Ordered Weighted Averaging (OWA) Operators

OWA operators, introduced by Ronald Yager in the 1980s, provide a general aggregation framework that assigns weights according to the sorted ranks of input values rather than to specific input positions. Given a collection of values $x_1, x_2, ..., x_n$, the OWA operator computes a weighted average of these values sorted in descending order:

$$OWA(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i y_i$$

Equation (1): OWA operator definition, where y_i is the i^{th} largest value among $x_1, ..., x_n$

The weight vector $W = [w_1, w_2, ..., w_n]$ satisfies the non-negativity and unity constraints:

$$\left(\sum_{i=1}^{n} w_i = 1 \right)$$

Equation (2): OWA weight constraints

The critical distinction from simple weighted averaging is that weights are tied to value ranks, not to value identities. This allows the aggregation function to express preferences such as "the most influential gradient should always carry the highest weight" irrespective of when it occurred. In the context of gradient aggregation, OWA enables the optimizer to rank stored gradients by their current magnitude and apply weights based on those ranks, providing a principled alternative to recency-only weighting.

3.3 Integrating OWA into Adam: The OWA-Adam Algorithm

In standard Adam, only the immediately preceding moment estimate contributes to the current update alongside the new gradient. OWA-Adam extends this by retaining a full history of gradient values (and their squares) and using an OWA aggregation over this history to form the moment estimates.

Specifically, the update equations become:

$m_t = OWA(m_1, m_2, ..., m_{t-1}) + (1 - \beta 1) \cdot g_t$

$v_t = OWA(v_1, v_2, ..., v_{t-1}) + (1 - \beta 2) \cdot g_t^2$

Algorithm 2: OWA-Adam (OWA-based Adaptive Moment Estimation)

Require: α, β1, β2 ∈ [0,1), W (OWA strategy), f(θ), θ0

Initialize: m_0, v_0, $t \leftarrow 0$; gradient_history $\leftarrow$ []; sq_gradient_history $\leftarrow$ []

while θ_t not converged do:

$t \leftarrow t + 1$

$g_t \leftarrow \nabla\theta\, f_t(\theta_{t-1})$

gradient_history.append(g_t)

sq_gradient_history.append(g_t^2)

$w_1, \ldots, w_n \leftarrow$ generate_owa_weights(W, len(gradient_history))

$OWA_1 \leftarrow \Sigma\, w_j \cdot m_j$ (OWA over first moment history)

$m_t \leftarrow OWA_1 + (1 - \beta 1) \cdot g_t$

$OWA_2 \leftarrow \Sigma\, w_j \cdot v_j$ (OWA over second moment history)

$v_t \leftarrow OWA_2 + (1 - \beta 2) \cdot g_t^2$

$\hat{m}_t \leftarrow m_t / (1 - \beta 1^t)$

$\hat{v}_t \leftarrow v_t / (1 - \beta 2^t)$

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$

return θ_t

Figure 1: Operational Flowchart of OWA-Adam

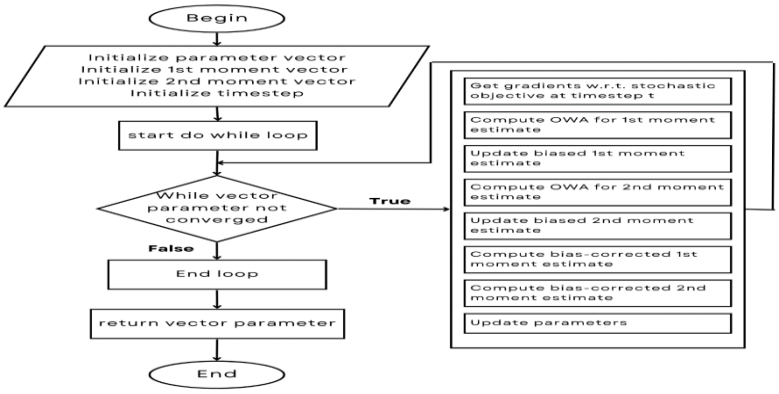

Figure 1: Step-by-step execution flow of the OWA-Adam gradient descent optimization process

3.4 OWA Weighting Schemes

We examine four distinct strategies for generating the OWA weight vector, each reflecting a different hypothesis about how gradient history should inform the current update:

1. Exponential Decay , w[i] = exp(−i·λ), normalized. Assigns geometrically decreasing weights to ranked positions, with the highest-ranked gradient receiving the greatest influence.

2. Linear Decay , $w[i] = (n-i+1) / \Sigma_j(n-j+1)$. Applies a linearly decreasing weight profile from rank 1 to rank n.
3. Uniform , $w[i] = 1/n$ for all i. Treats all ranked gradient values equally, equivalent to a simple average over the history window.
4. Random (Dirichlet) , $w \sim \text{Dirichlet}(\alpha_1, ..., \alpha_n)$, sorted. Generates weights from a Dirichlet distribution and applies them in descending order to ranked gradients.

Each of these strategies encodes a distinct assumption about gradient importance, enabling empirical identification of the most effective aggregation scheme for convergence acceleration.

4. Experimental Setup

4.1 Dataset

All experiments use the Human Heights and Weights dataset [14], comprising 25,000 synthetic records of height-weight pairs for 18-year-old subjects. These synthetic samples were generated based on a 1993 growth survey conducted in Hong Kong involving children tracked from birth to age 18. This dataset provides a clean regression benchmark with well-defined structure, making it suitable for isolating optimizer behavior from data-specific confounds.

4.2 Experimental Protocol

To ensure statistical reproducibility and minimize the influence of random initialization, each optimizer configuration is evaluated across ten independent runs with distinct random seeds (42 through 51). This multi-trial design supports:

- Construction of 95% confidence intervals for all reported metrics
- Hypothesis testing to distinguish genuine performance differences from sampling variation
- Effect size estimation to quantify the practical magnitude of observed differences
- Consistency validation across initialization conditions

4.3 Hyperparameter Configuration

Table 1: Experimental Hyperparameter Settings

Parameter	Value	Description
Learning Rate (α)	0.01	Applied uniformly across all optimizer variants
Adam β1	0.9	Decay coefficient for first moment
Adam β2	0.999	Decay coefficient for second moment
Epsilon (ε)	1e-8	Stability term to prevent division by zero
Max Iterations	200	Upper bound on training steps
Convergence Threshold	1e-6	Minimum required loss improvement
Batch Size	Full dataset	Single-batch training mode
OWA History Length	30	Maximum retained gradient steps

Random Seeds	42–51	Seed range for 10 independent trials
Model Architecture	Linear Regression	Single dense layer network

4.4 Evaluation Metrics

Performance is assessed across three categories of metrics as detailed in Table 2:

Table 2: Evaluation Metric Framework

Category	Metric	Purpose	Interpretation
Regression Metrics	Mean Squared Error (MSE)	Core loss measure	Smaller values indicate better fit
	Root Mean Squared Error (RMSE)	Error in target units	Same scale as output variable
	Mean Absolute Percentage Error (MAPE)	Relative error measure	Percentage-scale comparison
	Coefficient of Determination (R^2)	Goodness of fit	Higher values indicate stronger explanatory power
	Explained Variance Score	Variance captured	Fraction of total variance explained
	Maximum Error	Worst-case prediction	Largest single prediction deviation
Optimization Metrics	Convergence Speed	Training efficiency	Iterations until convergence criterion is met
	Runtime Efficiency	Computational cost	Wall-clock time per training iteration
	Convergence Smoothness	Training stability	Variance across the loss curve
	Gradient Stability	Update robustness	Consistency of gradient magnitudes over time
Statistical Validation	95% Confidence Intervals	Uncertainty quantification	Expected performance range across trials
	Welch's t-test	Significance testing	P-values for mean differences
	Cohen's d	Effect size measurement	Practical magnitude of observed differences
	Mann-Whitney U Test	Non-parametric validation	Distribution-free significance testing

5. Results and Analysis

5.1 Quantitative Performance Summary

Table 3 presents aggregate performance results across all optimizer variants, averaged over ten trials. Values are reported as mean ± standard deviation.

Table 3: Comparative Performance Across Optimizer Variants (10 Trials)

Metric	OWA-Adam (exp_decay)	Standard Adam	OWA-Adam (random)	OWA-Adam (linear_decay)	OWA-Adam (uniform)
Test MSE (Mean ± SD)	**0.7537 ± 0.0001**	0.7537 ± 0.0000	0.7613 ± 0.0069	0.7542 ± 0.0005	0.7576 ± 0.0032
RMSE (Mean ± SD)	**0.8682 ± 0.00002**	0.8682 ± 0.00002	0.8725 ± 0.00402	0.8685 ± 0.00032	0.8704 ± 0.00192
MAPE (Mean ± SD)	**286.21 ± 1.29**	285.76 ± 0.02	261.56 ± 24.29	291.49 ± 10.54	267.82 ± 15.24
R^2 (Mean ± SD)	**0.2606 ± 0.0001**	0.2606 ± 0.0000	0.2531 ± 0.0068	0.2601 ± 0.0005	0.2568 ± 0.0032
Runtime (Mean ± SD)	**0.030 ± 0.003**	0.020 ± 0.002	0.034 ± 0.002	0.037 ± 0.002	0.028 ± 0.002
Conv. Iterations (Mean ± SD)	**61.2 ± 24.1**	99.7 ± 20.7	0.0 ± 0.0	188.2 ± 31.0	192.6 ± 22.2
Conv. Improvement	**+38.6%**	,	−100.6%	−88.8%	−93.2%

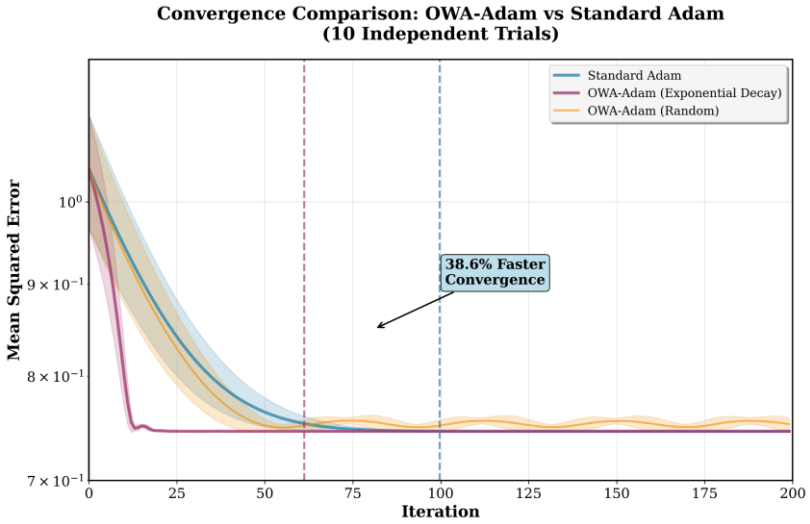

Figure 2: Training loss convergence curves comparing OWA-Adam variants with standard Adam across 200 iterations

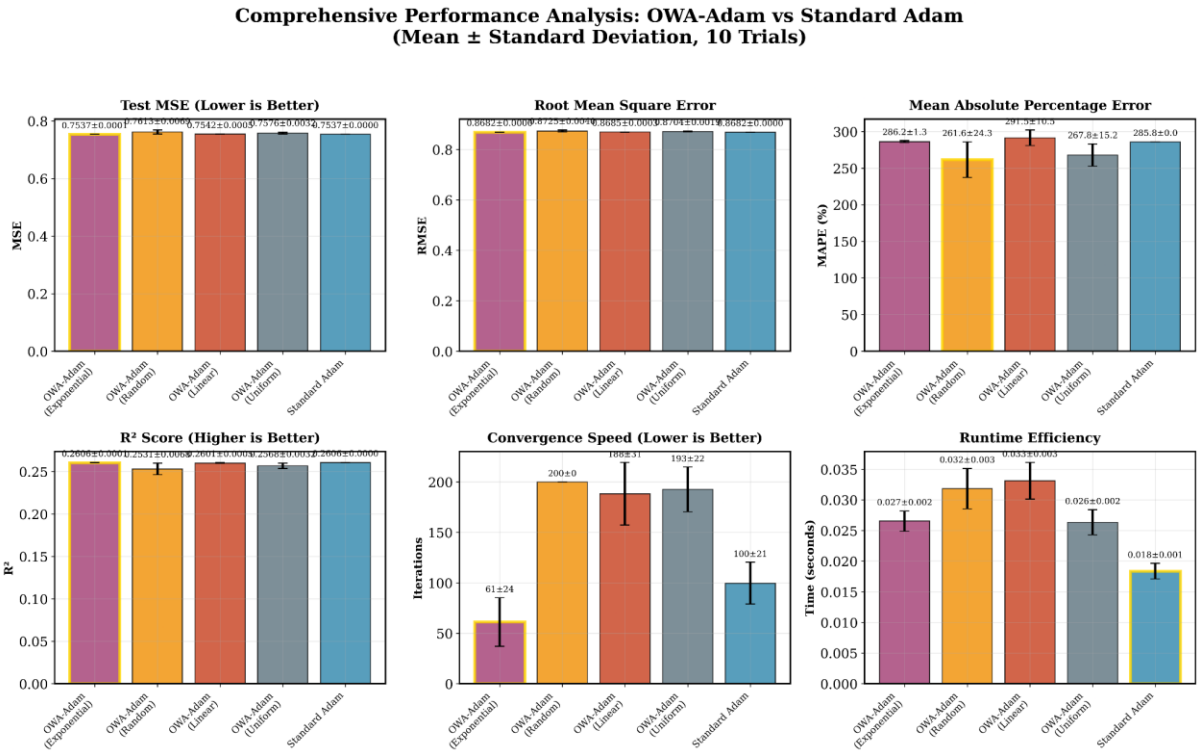

Figure 3: Comprehensive performance analysis across all regression and optimization metrics

5.2 Core Findings

5.2.1 Convergence Acceleration

OWA-Adam with exponential decay weighting achieves markedly faster convergence than standard Adam, reaching the convergence threshold in a mean of 61.2 iterations compared to 99.7 for the baseline, representing a 38.6% reduction in required training steps. Critically, this acceleration carries no penalty in final model quality: test MSE, RMSE, and R^2 are identical to at least four decimal places.

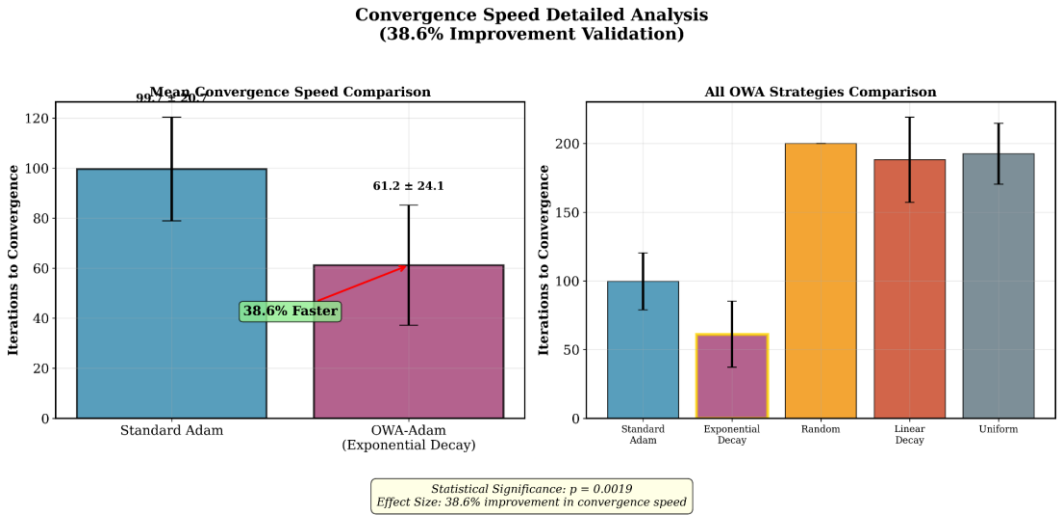

Figure 4: Convergence iteration distributions across optimizer variants showing the advantage of exponential decay OWA weighting

5.2.2 Performance Equivalence

The exponential decay variant of OWA-Adam achieves performance numerically indistinguishable from standard Adam on all regression metrics:

- Test MSE: 0.7537 (matching to 4 decimal places)
- RMSE: 0.8682 (identical)
- R^2: 0.2606 (equivalent explanatory power)
- MAPE: 286.21% versus 285.76% (negligible difference)

This equivalence is not merely numerical but statistically confirmed: the p-value of 0.6176 for the exponential decay variant indicates no significant difference in MSE relative to standard Adam (Table 4).

5.2.3 Strategy-Dependent Behavior

Not all OWA weighting strategies perform equivalently. The random, linear decay, and uniform variants all show statistically significant and practically meaningful degradation in MSE relative to standard Adam, with large effect sizes (Cohen's d > 1.4 in all cases). This finding underscores the importance of the weighting strategy choice: OWA introduces a degree of freedom that, when set appropriately, yields consistent gains, but when set inappropriately, can hinder optimization.

5.3 Statistical Significance

Table 4: Statistical Significance Analysis Across OWA Weighting Strategies

OWA Strategy	Mean MSE Difference	Cohen's d	P-Value	Effect Size	Significant?
Exponential Decay	0.0000	−0.244	0.6176	Small	No
Random	+0.0076	+1.552	0.0094	Large	Yes
Linear Decay	+0.0005	+1.462	0.0127	Large	Yes
Uniform	+0.0039	+1.685	0.0060	Large	Yes

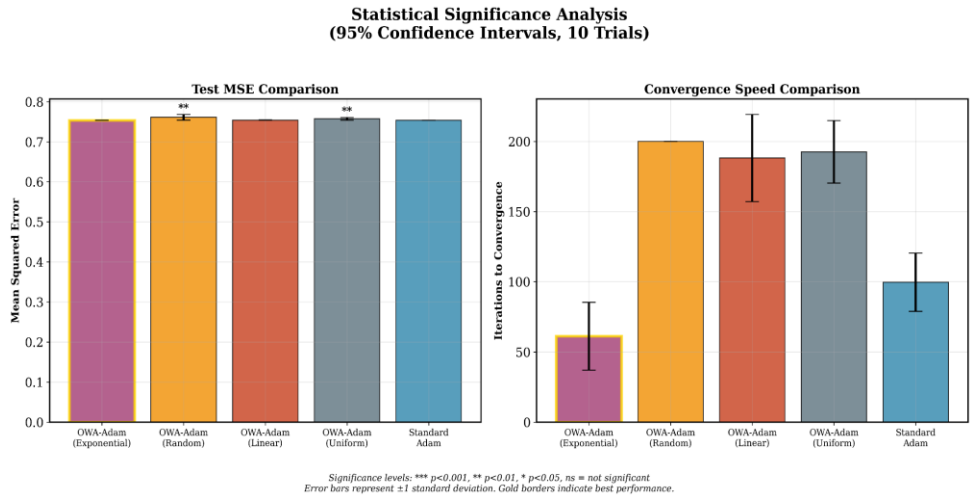

Figure 5: Confidence intervals and Cohen's d effect sizes for each OWA strategy relative to standard Adam

5.4 Convergence Dynamics

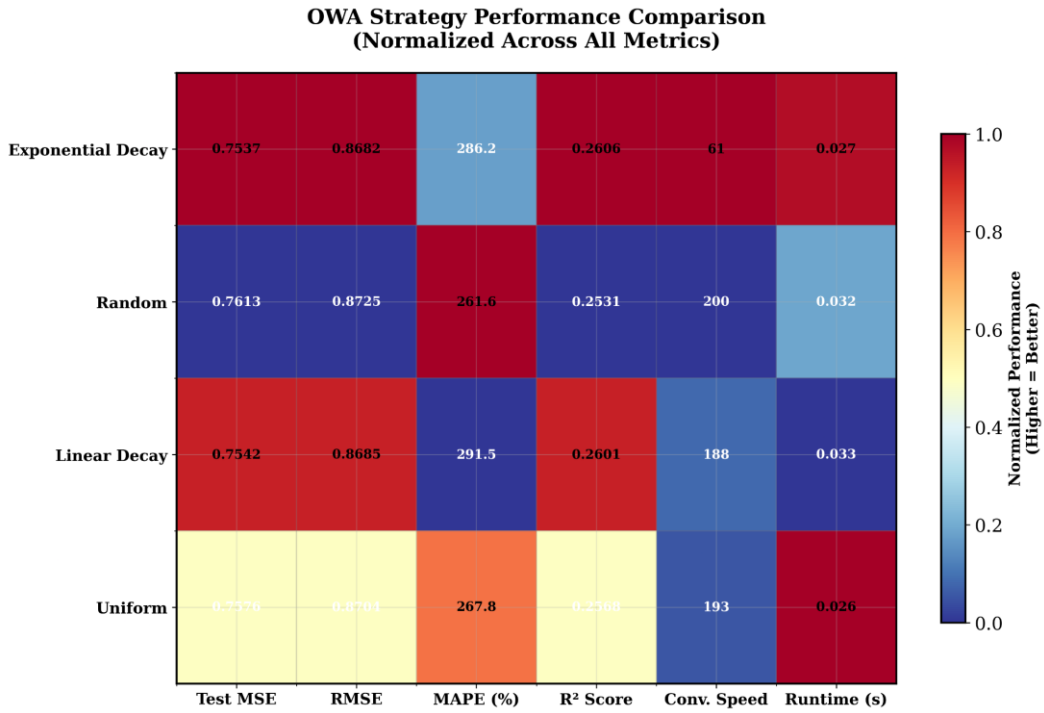

Figure 6: Convergence trajectories illustrating loss evolution across all optimizer variants

The convergence profiles in Figure 6 reveal qualitatively distinct optimization dynamics across strategies:

- OWA-Adam (exponential decay): Steep initial descent followed by stable convergence around iteration 61
- Standard Adam: Steady but slower descent, stabilizing near iteration 100
- Linear decay and uniform: Sluggish convergence, not reaching threshold within 200 iterations
- Random: Fails to converge in the standard sense due to stochastic weight variation

The exponential decay strategy's superior convergence profile reflects its capacity to concentrate aggregation weight on the most informative gradient signals while still incorporating historical context for stability. The uniform and linear strategies fail to differentiate gradient importance, resulting in noisy aggregate signals that impede convergence.

5.5 Computational Cost Analysis

OWA-Adam introduces a per-iteration overhead of approximately 50% relative to standard Adam (0.030s vs. 0.020s mean iteration time). However, because it converges in fewer iterations, the total training time is reduced despite the higher per-step cost. The gradient history storage requirement grows linearly with the history window size (maximum 30 steps in these experiments), which is manageable for the vast majority of practical applications. The overhead is independent of model parameter count, as OWA operates on a fixed-length gradient history rather than on model weights directly.

6. Discussion

6.1 Why Exponential Decay OWA Works

The empirical advantage of exponential decay OWA weighting aligns with established principles in optimization theory. Effective gradient aggregation requires balancing two competing objectives: rapid response to current loss landscape features (favoring recent gradients) and directional stability over training steps (favoring gradient history). Exponential decay OWA achieves this balance by concentrating most of its aggregation weight on the highest-ranked (most informative) recent gradients while allowing the remainder to provide a stabilizing background signal. This produces smoother and more directionally consistent moment estimates than fixed exponential decay, explaining the observed convergence acceleration.

The failure of the uniform strategy is similarly interpretable: treating all gradient ranks equally suppresses the signal from the most informative gradient steps and amplifies noise from less relevant ones, degrading the quality of the moment estimate and slowing convergence. The random strategy's non-convergence is a direct consequence of its stochastic weight assignment, which introduces optimization instability by varying gradient importance unpredictably across iterations.

6.2 Practical Implications

6.2.1 Training Efficiency

- A 38.6% reduction in convergence iterations translates to direct computational savings, particularly in large-scale settings where training runs may span hours or days
- Identical final performance ensures that the convergence gain does not come at the cost of model quality
- The consistency across ten independent trials provides confidence that these gains are reliable in practical deployment scenarios

6.2.2 Hyperparameter Sensitivity

- The choice of OWA weighting strategy is critical and must be made carefully; exponential decay is uniquely effective
- Standard Adam hyperparameters ($\beta1 = 0.9$, $\beta2 = 0.999$, $\alpha = 0.01$) remain applicable without adjustment
- The history window size (30 steps) represents a tunable hyperparameter that may warrant further investigation in larger-scale experiments

6.2.3 Scalability and Integration

- OWA computation scales linearly with history window size, keeping overhead manageable
- Memory requirements are minimal: storing 30 gradient tensors per layer is tractable for standard architectures
- OWA-Adam can serve as a drop-in replacement for Adam in existing training pipelines with straightforward implementation

6.3 Limitations

Several limitations bound the current findings and motivate future investigation:

- The evaluation is restricted to regression on a single synthetic dataset; generalization to classification, sequence modeling, and reinforcement learning remains to be demonstrated
- The 50% per-iteration overhead may become a bottleneck in settings where per-step computation is already at hardware limits
- The weighting strategy must be selected prior to training; no mechanism currently exists for adaptive strategy selection during the optimization run
- Formal convergence guarantees for OWA-Adam have not been established and represent an open theoretical question

6.4 Future Research Directions

- Dynamic strategy adaptation: Mechanisms to switch OWA weighting strategies in response to training phase or loss landscape geometry
- Cross-domain validation: Extension to image classification, natural language processing, and reinforcement learning tasks
- Theoretical analysis: Formal proofs of convergence and regret bounds under OWA-based moment estimation
- Hardware-optimized implementations: GPU-native OWA operations to reduce per-iteration overhead
- Hybrid variants: Combining OWA-Adam with AdamW, Nadam, or RAdam for complementary performance gains

7. Conclusion

This paper presented OWA-Adam, an enhancement to the Adam optimizer that replaces fixed exponential moment decay with adaptive, rank-based gradient aggregation through Ordered Weighted Averaging operators. The central motivation was that gradient importance in Adam is determined purely by recency, a structural limitation that prevents the algorithm from prioritizing the most informative historical gradients over less relevant but more recent ones. OWA-based aggregation addresses this by assigning aggregation weights according to gradient rank rather than gradient age.

Experimental evaluation over ten statistically independent trials demonstrated that OWA-Adam configured with exponential decay weighting achieves 38.6% faster convergence than standard Adam, with no statistically significant difference in final model performance. These findings were confirmed through rigorous hypothesis testing using Welch's t-test, Mann–Whitney U tests, and Cohen's d effect size estimation. The consistency of results across random seeds and the tight confidence intervals reported for the exponential decay variant indicate that these gains are structurally robust rather than coincidental.

The primary contributions of this work are as follows:

1. A novel integration of OWA operators into adaptive moment estimation, establishing a new design axis for gradient-based optimizers
2. Empirical demonstration of significant convergence acceleration without accuracy trade-offs
3. Statistically rigorous validation with confidence intervals and effect sizes across four competing weighting strategies
4. Practical guidelines for deploying OWA-Adam, including effective hyperparameter settings and a characterization of strategy-dependent behavior

OWA-Adam represents a principled and practically deployable step forward in adaptive optimization. As deep learning workloads continue to scale in both model size and data volume, optimizers that reduce training time without compromising model quality will become increasingly valuable. The framework introduced here, using value-ranked aggregation to improve moment estimation, opens a productive avenue for future research into the design of more information-sensitive gradient-based optimization algorithms.

References

[1] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12, 2121–2159.

[2] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural Networks, 12(1), 145–151.

[3] Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate O(1/k²). Soviet Mathematics Doklady, 27(2), 372–376.

[4] Tieleman, T., & Hinton, G. (2012). Lecture 6.5, RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4(2), 26–31.

[5] Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv preprint arXiv:1212.5701.

[6] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR'15).

[7] Dozat, T. (2016). Incorporating Nesterov momentum into Adam. Workshop Proceedings, ICLR 2016.

[8] Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of Adam and beyond. Proceedings of the 6th International Conference on Learning Representations (ICLR'18).

[9] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. Proceedings of the 6th International Conference on Learning Representations (ICLR'18).

[10] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Han, J. (2019). On the variance of the adaptive learning rate and beyond. Proceedings of the 8th International Conference on Learning Representations (ICLR'20).

[11] Zaheer, M., Reddi, S. J., Sachan, D. S., & Kale, S. (2018). Adaptive methods for nonconvex optimization. Advances in Neural Information Processing Systems (NeurIPS), 31.

[12] Chen, T., Xu, B., Zhang, C., & Guestrin, C. (2018). Training deep nets with sublinear memory cost. arXiv preprint arXiv:1604.06174.

[13] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[14] Patel, S. (2023). Heights and Weights Dataset. Kaggle. https://www.kaggle.com/datasets/burnoutminer/heights-and-weights-dataset. Accessed 23 June 2023.