AMH: AdaGrad-Momentum Hybrid for Robust Training on Small Coresets

Saleh Abdillahi Saleh **Sri Advaita Varakavi** **Umniya Bakari Maulid**

Abstract

Training deep neural networks under extreme data scarcity is challenging due to high gradient variance and unstable optimization dynamics. Standard methods such as SGD suffer from noisy updates, while adaptive optimizers like AdamW can overfit to spurious gradient statistics when data is limited. These challenges are particularly acute in low-resource settings common in many Muslim-majority regions, where large datasets are often unavailable.

We propose AMH (AdaGrad-Momentum Hybrid), a simple yet effective optimization method designed for robust training on small coresets. AMH combines adaptive per-parameter scaling, momentum-based smoothing, and a leaky squared-gradient accumulator that prevents the rapid learning-rate decay of classical AdaGrad. This results in stable updates while preserving responsiveness to new gradient information.

We evaluate AMH on EuroSAT, CIFAR-10, and BloodMNIST using ResNet-18 and EfficientNet-B0 trained from scratch across coreset fractions of 100

1. Introduction

Deep neural networks have achieved strong performance across a wide range of visual recognition tasks, but their success often depends on the availability of large-scale labeled datasets. In many practical scenarios, however, only a limited subset of data is available due to labeling costs, domain constraints, or data collection limitations. Training under such conditions can be framed as a coreset-based learning problem, where only a small fraction of the full dataset is used for optimization.

Small coresets introduce a fundamental challenge: the stochastic gradient becomes a poor approximation of the true population gradient. This leads to high-variance updates, unstable convergence, and reduced generalization. Standard optimization methods are not always well-suited to this regime. Vanilla gradient descent is often impractical due to computational cost, while mini-batch methods can amplify stochastic noise when data is scarce. SGD with momentum improves directional consistency but lacks adaptive scaling, and Adam-based methods can overfit noisy gradients due to aggressive adaptation.

To address these challenges, we propose AMH (AdaGrad-Momentum Hybrid), a gradient descent method specifically designed for data-scarce settings. AMH integrates three key components: adaptive per-parameter scaling (inspired by AdaGrad) to normalize high-variance gradient coordinates; momentum-based smoothing to preserve consistent descent directions across mini-batches; and a leaky squared-gradient accumulator, which prevents the rapid decay of effective learning rates observed in standard AdaGrad. The leaky accumulator introduces a controlled forgetting mechanism, allowing the optimizer to retain useful historical gradient information while remaining responsive to new updates. This is particularly important under small coresets, where gradient statistics evolve rapidly and may otherwise lead to premature convergence.

We evaluate AMH on multiple datasets (EuroSAT, CIFAR-10, and BloodMNIST) using two architectures, ResNet-18 and EfficientNet-B0, trained from scratch. Experiments are conducted across coreset fractions of 100

2. Proposed Method

2.1. AMH Optimizer

We propose AMH (AdaGrad–Momentum Hybrid), a gradient descent variant designed for training under small coreset settings, where gradient estimates are noisy. The method combines adaptive gradient scaling with a leaky accumulation mechanism and momentum to stabilize updates.

Let $g_t = \nabla_\theta L(\theta_t)$. AMH maintains a squared-gradient accumulator v_t and a momentum buffer m_t:

$$v_t = \rho v_{t-1} + (1-\rho) g_t^2, \tag{1}$$

$$\tilde{g}_t = \frac{g_t}{\sqrt{v_t} + \epsilon}, \tag{2}$$

$$m_t = \beta m_{t-1} + \tilde{g}_t, \tag{3}$$

$$\theta_{t+1} = \theta_t - \alpha m_t. \tag{4}$$

Here, ρ controls the decay of past gradient information, β is the momentum coefficient, and ϵ ensures numerical stabil-

ity. Unlike standard AdaGrad, which uses a non-decaying sum of squared gradients (leading to rapid decrease of the effective learning rate), the leaky accumulator (with $\rho < 1$) gives more weight to recent gradients. This is essential for small coresets because gradient statistics change quickly as the model sees the same limited samples repeatedly; the optimizer must remain responsive to new information while still benefiting from long-term variance reduction.

2.2. Relation to Existing Adaptive Optimizers

AMH is closely related to adaptive optimization methods such as RMSProp and Adam. Like RMSProp, AMH uses an exponentially weighted accumulation of squared gradients to construct a per-parameter scaling factor. However, the key distinction lies in the ordering of operations. RMSProp and Adam accumulate momentum on the raw gradient and subsequently apply adaptive scaling, whereas AMH first normalizes the gradient using the leaky accumulator and then applies momentum to the normalized update direction. This design smooths already preconditioned gradients, which we hypothesize is advantageous in small-coreset settings where gradient statistics are highly unstable.

The leaky accumulator can also be interpreted as a form of adaptive preconditioning with controlled forgetting. Unlike classical AdaGrad, whose effective learning rates monotonically decrease, AMH retains responsiveness to recent gradients through exponential decay. A direct empirical comparison against RMSProp remains an important direction for future work.

2.3. Why Leakage May Help

Classical AdaGrad accumulates squared gradients indefinitely, causing effective learning rates to shrink throughout training. AMH instead employs a leaky accumulator,

$$v_t = \rho v_{t-1} + (1-\rho) g_t^2,$$

which assigns exponentially decreasing weight to older gradient information. This mechanism prevents early gradients from dominating the learning-rate schedule and allows the optimizer to adapt to changing optimization dynamics. The resulting update can be viewed as adaptive preconditioning followed by momentum smoothing. While this work does not provide formal convergence guarantees, the design is motivated by the need to balance variance reduction and adaptability under extreme data scarcity.

2.4. Hyperparameters

We use $\rho = 0.995$, $\beta = 0.9$, $\epsilon = 10^{-8}$, and weight decay 10^{-4} across all experiments. The learning rate α for AMH was tuned over $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ for the EuroSAT dataset; the best value was 10^{-3} for most configurations. All other optimizers had their learning rates tuned independently via a validation search (see Section 3).

Algorithm 1 AMH Optimizer

Initialize parameters θ, accumulator $v \leftarrow 0$, momentum $m \leftarrow 0$
for each training step t **do**
  Compute gradient $g_t = \nabla_\theta L(\theta_t)$
  $v_t \leftarrow \rho v_{t-1} + (1-\rho) g_t^2$
  $\tilde{g}_t \leftarrow g_t / (\sqrt{v_t} + \epsilon)$
  $m_t \leftarrow \beta m_{t-1} + \tilde{g}_t$
  $\theta_{t+1} \leftarrow \theta_t - \alpha m_t$
end for

3. Experimental Setup

3.1. Datasets

We evaluate our method on three datasets that represent different types of visual data. EuroSAT consists of satellite imagery with 10 land-cover classes; this dataset has structured, low-variance intra-class patterns. CIFAR-10 is a standard benchmark of natural object images with 10 classes; it has high intra-class variability and is commonly used for optimisation studies. BloodMNIST is a medical microscopy dataset with 8 cell types; it is more structured but has challenges due to class imbalance and medical relevance. All images are resized to 32×32 pixels using bilinear interpolation. Grayscale images (BloodMNIST) are converted to 3 channels by replication.

3.2. Data Splits and Coresets

Each dataset is split into 80% training, 10% validation, and 10% test. To analyse data efficiency, we construct three coreset fractions from the training split: the full 100% training set, a 25% random sample (with fixed seed 42), and a 5% random sample (same seed). The same coreset subsets are used across all optimizers at each fraction to ensure fair comparison. The validation set is used only for learning rate tuning; all final results are reported on the held-out test set.

3.3. Models and Training Configuration

We use two architectures trained from scratch without pretrained weights. Pretraining was intentionally excluded in order to isolate optimizer behavior and avoid confounding improvements arising from transferred representations. ResNet-18, a standard residual network with 18 layers, and EfficientNet-B0, a compound-scaled network that is more parameter-efficient but also more sensitive to hyperparameters. All experiments follow a fixed configuration: 30 epochs (no early stopping), random seed 42 (used for

Variant	Accuracy (%)
Full AMH	65.07
w/o Momentum	61.2
w/o Adaptive Scaling	58.7
w/o Leak ($\rho = 1$)	60.3

Table 1. Ablation study on EuroSAT (ResNet-18, 5% coreset). Removing any single component reduces accuracy, confirming that all three contribute to AMH's performance.

data splits, coreset sampling, and weight initialisation), cross-entropy loss, and Top-1 accuracy as the evaluation metric. For AMH, we use a batch size of 32. For mini-batch baselines, we sweep batch sizes 8, 16, 32, 64, and 128.

3.4. Baselines and Hyperparameter Tuning

We compare AMH against several standard optimizers: vanilla Gradient Descent (full-batch, evaluated only for 100% coreset due to memory constraints); SGD with momentum ($\beta = 0.9$, weight decay 10^{-4}); mini-batch Gradient Descent (no momentum, five batch sizes); and AdamW (Adam with decoupled weight decay). For each baseline, the learning rate is tuned over the set $\{10^{-3}, 10^{-2}, 10^{-1}\}$ for SGD, mini-batch GD, and AdamW, and over $\{10^{-2}, 10^{-1}, 1\}$ for vanilla GD. The best configuration is selected based on validation accuracy after five epochs of a separate tuning run. AMH uses a fixed learning rate from its own tuning grid $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$.

3.5. Statistical Reliability

For EuroSAT and CIFAR-10, experiments were run with a single random seed (42) due to computational constraints. For BloodMNIST, we performed multiple runs and report mean ± standard deviation (Table 3).All results are reported as Top-1 test accuracy. Consequently, small differences between methods on EuroSAT and CIFAR-10 should be interpreted cautiously.

4. Results and Discussion

4.1. Main Results

Table 2 reports Top-1 accuracy of AMH and all baselines across datasets, coreset fractions, and architectures. Several observations stand out.

On EuroSAT, a structured satellite image dataset, AMH is highly competitive. At 100% coreset, AMH achieves 88.00% (ResNet-18) and 90.04% (EfficientNet-B0), slightly above Adam in the first case and within 0.8% in the second. At 25%, AMH (77.74%) matches Adam (77.85%) on ResNet-18 and outperforms Adam (76.37% vs. 74.00%) on EfficientNet-B0. At the extreme 5% coreset, AMH (65.07%) is essentially identical to Adam (65.19%) on ResNet-18, and even exceeds Adam (57.30% vs. 54.11%) on EfficientNet-B0. This shows that AMH is robust even when only 5% of the data is available.

On BloodMNIST, a medical dataset with some class imbalance, the situation is more varied. At 100%, AMH (85.06% on ResNet-18) is lower than the best mini-batch methods (over 93%), but still respectable. At 5%, AMH (76.35%) substantially outperforms Adam (42.79%) and achieves performance comparable to the strongest methods in this setting. However, Table 3 shows that SGD+Momentum and some mini-batch variants achieve slightly higher mean accuracy. Therefore, AMH should be viewed as a competitive alternative rather than a universally superior optimizer in low-resource medical settings.

On CIFAR-10, a high-variance natural image dataset, AMH struggles. On ResNet-18 at 100%, AMH (66.10%) is far below Adam (82.95%) and even below vanilla mini-batch methods. At 5%, AMH (46.96%) is similar to Adam (52.01%) but worse than SGD (51.36%) and mini-batch methods. The most severe failure is on EfficientNet-B0 at 5%, where AMH achieves only 10.09% (near chance) while Adam reaches 42.18%. We analyse this failure in Section 4.3.

4.2. Ablation Study

To isolate the contribution of each component of AMH, we performed an ablation study on EuroSAT with ResNet-18 at the 5% coreset (the most challenging setting). Table 1 shows the accuracy when each component is removed. The full AMH achieves 65.07%. Removing momentum reduces accuracy by 3.87 percentage points to 61.2%, showing that momentum helps smooth noisy gradient directions. Removing adaptive scaling (using only momentum and the accumulator without dividing by $\sqrt{v_t}$) causes the largest drop, 6.37 percentage points to 58.7%, confirming that normalising by the gradient magnitude is essential for handling high-variance updates. Removing the leak (setting $\rho = 1$, which makes the accumulator non-decaying, reverting to classical AdaGrad) reduces accuracy by 4.77 percentage points to 60.3%. This demonstrates that the controlled forgetting mechanism is critical: with a non-decaying sum, the learning rate becomes too small too quickly, preventing the model from adapting to new gradient information after the first few epochs. All three components are therefore necessary for optimal performance under extreme data scarcity.

4.3. Why AMH Fails on CIFAR-10 with EfficientNet-B0 at 5%

As shown in Table 2, AMH on EfficientNet-B0 at 5% coreset achieves only 10.09% accuracy on CIFAR-10, which is close to random chance (10 classes). By contrast, Adam

reaches 42.18%. We attribute this failure to a learning rate sensitivity mismatch combined with the high intrinsic variance of CIFAR-10. EfficientNet-B0's compound-scaled architecture (depth, width, and resolution balanced together) creates a loss landscape that is much more sensitive to the learning rate than ResNet-18's plain residual blocks. At 5% coreset, only about 39 gradient steps are taken per epoch with batch size 32. AMH's adaptive scaling, which works well on structured datasets like EuroSAT, cannot compensate for the combination of a learning rate that is too high for this architecture and extreme gradient noise from the tiny coreset. These observations suggest that AMH is more naturally suited to structured and lower-variance datasets than highly heterogeneous natural-image datasets. Future work will investigate adaptive leakage schedules, learning-rate warmup strategies, and alternative second-moment estimators to improve robustness across a wider range of optimization landscapes.

5. Discussion and Limitations

AMH performs best on structured datasets such as EuroSAT and BloodMNIST, where it remains competitive with strong baselines and substantially outperforms Adam under extreme data scarcity. However, performance degrades on high-variance natural-image datasets such as CIFAR-10, particularly with EfficientNet-B0, suggesting that the current adaptive scaling mechanism is less effective for highly heterogeneous gradient distributions.

The main limitations of this study are the lack of multi-seed evaluation for EuroSAT and CIFAR-10, the absence of comparisons with closely related adaptive optimizers such as RMSProp, AdaBelief, Lion, LARS, and LAMB, and the lack of a sensitivity analysis for the leakage parameter ρ. While $\rho = 0.995$ performed reasonably across datasets, the CIFAR-10 results suggest that dataset-specific forgetting rates may improve robustness.

6. Conclusion

We introduced AMH, a lightweight optimizer that combines adaptive scaling, momentum smoothing, and a leaky squared-gradient accumulator for training under data scarcity. Across three datasets and two architectures, AMH showed competitive performance on structured low-resource tasks such as EuroSAT and BloodMNIST, while exhibiting limitations on more challenging natural-image datasets such as CIFAR-10. An ablation study confirmed the contribution of all three components.

Overall, the results suggest that variance-aware optimization can be beneficial in extreme low-data settings, although AMH is not universally superior and may require additional tuning for highly heterogeneous datasets. Future work includes comparisons with additional optimizers, broader multi-seed evaluation, and deeper theoretical analysis.

References

[1] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

[2] Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *ICLR*.

[3] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

[4] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

[5] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

[6] Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*.

[7] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

[8] Yang, J. et al. (2023). MedMNIST v2: A large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Scientific Data*, 10(1):41.

[9] Helber, P., Bischke, B., Dengel, A., and Borth, D. (2019). EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 12(7):2217–2226.

[10] Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. Neural Networks for Machine Learning, Coursera.

Optimizer	Coreset	EuroSAT	Blood	C10
ResNet18				
AMH	100%	88.00	85.06	66.10
Adam	100%	87.59	93.10	82.95
SGD+Momentum	100%	83.93	93.54	80.75
MiniBatch-bs8	100%	83.37	93.22	80.83
MiniBatch-bs16	100%	83.63	88.25	80.29
MiniBatch-bs32	100%	84.74	91.87	77.82
MiniBatch-bs64	100%	84.04	93.01	71.99
MiniBatch-bs128	100%	80.96	91.52	72.25
VanillaGD	100%	53.78	68.69	11.26
ResNet18				
AMH	25%	77.74	67.93	61.12
Adam	25%	77.85	70.42	72.92
SGD+Momentum	25%	75.48	86.79	69.65
MiniBatch-bs8	25%	71.04	83.13	22.52
MiniBatch-bs16	25%	73.11	89.54	18.53
MiniBatch-bs32	25%	74.93	85.38	60.77
MiniBatch-bs64	25%	73.22	85.59	61.59
MiniBatch-bs128	25%	66.52	86.61	56.16
ResNet18				
AMH	5%	65.07	76.35	46.96
Adam	5%	65.19	42.79	52.01
SGD+Momentum	5%	61.59	75.56	51.36
MiniBatch-bs8	5%	61.33	72.05	23.10
MiniBatch-bs16	5%	63.11	73.69	51.09
MiniBatch-bs32	5%	63.66	71.76	47.42
MiniBatch-bs64	5%	64.44	69.01	42.86
MiniBatch-bs128	5%	62.93	76.41	44.00
ENet-B0				
AMH	100%	90.04	88.98	44.69
Adam	100%	90.85	40.43	71.39
SGD+Momentum	100%	86.85	93.66	75.19
MiniBatch-bs8	100%	85.67	93.42	68.49
MiniBatch-bs16	100%	88.96	91.23	72.24
MiniBatch-bs32	100%	87.33	93.92	73.86
MiniBatch-bs64	100%	89.59	91.79	68.03
MiniBatch-bs128	100%	87.30	91.79	66.16
ENet-B0				
AMH	25%	76.37	80.80	41.48
Adam	25%	74.00	63.29	62.98
SGD+Momentum	25%	71.74	87.72	63.68
MiniBatch-bs8	25%	73.26	87.05	16.63
MiniBatch-bs16	25%	71.00	87.31	12.28
MiniBatch-bs32	25%	72.81	79.13	55.99
MiniBatch-bs64	25%	71.26	86.44	52.58
MiniBatch-bs128	25%	67.19	82.46	45.20
ENet-B0				
AMH	5%	57.30	50.83	10.09
Adam	5%	54.11	36.89	42.18
SGD+Momentum	5%	56.74	73.25	40.49
MiniBatch-bs8	5%	44.89	64.60	12.10
MiniBatch-bs16	5%	51.48	74.13	39.75
MiniBatch-bs32	5%	60.74	74.22	38.98
MiniBatch-bs64	5%	60.30	72.87	31.70
MiniBatch-bs128	5%	50.63	67.23	29.29

Table 2. Comparison of AMH against all baselines across datasets and coreset fractions. Values are Top-1 accuracy (%) (single seed for EuroSAT and CIFAR-10).

Optimizer	Coreset	Accuracy (Mean ± Std)
ResNet18		
AMH	100%	85.94 ± 1.30
Adam	100%	89.13 ± 4.13
SGD+Momentum	100%	92.17 ± 1.85
MiniBatch-bs8	100%	93.65 ± 0.57
MiniBatch-bs16	100%	91.19 ± 2.26
MiniBatch-bs32	100%	92.53 ± 0.57
MiniBatch-bs64	100%	91.51 ± 1.12
MiniBatch-bs128	100%	91.38 ± 0.67
VanillaGD	100%	70.23 ± 1.76
AMH	25%	78.45 ± 7.44
Adam	25%	77.86 ± 5.52
SGD+Momentum	25%	86.70 ± 0.33
MiniBatch-bs8	25%	87.06 ± 2.78
MiniBatch-bs16	25%	89.33 ± 0.25
MiniBatch-bs32	25%	84.95 ± 3.87
MiniBatch-bs64	25%	86.85 ± 0.89
MiniBatch-bs128	25%	86.89 ± 0.20
AMH	5%	70.60 ± 6.07
Adam	5%	48.67 ± 4.32
SGD+Momentum	5%	75.27 ± 0.65
MiniBatch-bs8	5%	77.95 ± 4.18
MiniBatch-bs16	5%	78.91 ± 3.70
MiniBatch-bs32	5%	77.37 ± 4.03
MiniBatch-bs64	5%	76.37 ± 5.22
MiniBatch-bs128	5%	77.49 ± 1.04
EfficientNet-B0		
AMH	100%	90.50 ± 2.36
Adam	100%	56.88 ± 16.09
SGD+Momentum	100%	93.73 ± 0.39
MiniBatch-bs8	100%	93.76 ± 0.28
MiniBatch-bs16	100%	92.42 ± 0.89
MiniBatch-bs32	100%	93.46 ± 0.46
MiniBatch-bs64	100%	92.18 ± 1.20
MiniBatch-bs128	100%	91.25 ± 0.48
VanillaGD	100%	90.87 ± 0.48
AMH	25%	83.79 ± 2.89
Adam	25%	48.30 ± 22.38
SGD+Momentum	25%	88.36 ± 1.67
MiniBatch-bs8	25%	87.77 ± 0.51
MiniBatch-bs16	25%	86.25 ± 2.27
MiniBatch-bs32	25%	83.70 ± 3.24
MiniBatch-bs64	25%	85.93 ± 0.49
MiniBatch-bs128	25%	81.96 ± 1.08
VanillaGD	25%	82.93 ± 1.23
AMH	5%	59.06 ± 9.62
Adam	5%	51.08 ± 11.56
SGD+Momentum	5%	74.47 ± 1.08
MiniBatch-bs8	5%	70.68 ± 4.31
MiniBatch-bs16	5%	66.81 ± 6.67
MiniBatch-bs32	5%	75.13 ± 0.85
MiniBatch-bs64	5%	63.94 ± 9.00
MiniBatch-bs128	5%	68.07 ± 1.25
VanillaGD	5%	26.10 ± 4.44

Table 3. BloodMNIST results (mean ± standard deviation across multiple seeds). These results provide a statistically more reliable comparison than the single-seed experiments and highlight both the strengths and limitations of AMH under data scarcity.