IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Hybrid AO–GA Metaheuristic for Hyperparameter Optimization of Convolutional Neural Networks

[1]Sneha Padhy, [2]Akshada Pawar, [3]Sneha Urakade, [4]Dr. Rachana Dhannawat

Department of Computer Engineering,
Usha Mittal Institute of Technology, Mumbai, India

Abstract: Hyperparameter optimization (HPO) of deep neural networks is a computationally expensive black-box problem that significantly affects model performance. Manual tuning is impractical for modern architectures, motivating the use of automated optimization methods. This paper proposes a hybrid metaheuristic algorithm that integrates the Aquila Optimizer (AO) with Genetic Algorithm (GA) operators for automated HPO of a convolutional neural network (CNN). The proposed method incorporates three mechanisms: opposition-based learning (OBL) for population initialization, a cosine-annealed mutation schedule, and distance-aware archive-guided crossover to preserve population diversity during exploitation. The optimizer searches for four CNN hyperparameters—learning rate, batch size, dropout rate, and SGD momentum—by minimizing classification error on a held-out subset. Experiments are conducted on four image classification benchmarks: MNIST, Fashion-MNIST, EMNIST, and CIFAR-10, using three independent random seeds and a fixed evaluation budget of 610 fitness evaluations per run. The proposed hybrid achieves accuracies of 98.87%, 89.38%, 99.08%, and 62.38% on the respective datasets, outperforming the standalone AO and achieving competitive performance relative to the GA, with the largest improvement observed on Fashion-MNIST (+0.37%).

Index Terms **-** Hyperparameter Optimization, Convolutional Neural Networks, Aquila Optimizer, Genetic Algorithm, Metaheuristic Optimization, Opposition-Based Learning, Cosine Annealing.

I. INTRODUCTION

Deep learning models, particularly convolutional neural networks (CNNs), have demonstrated strong performance across a wide range of image classification tasks[1]. However, the predictive performance of a CNN is highly sensitive to its hyperparameters, including learning rate, batch size, dropout rate, and optimizer momentum. Selecting these values manually is time-consuming, error-prone, and typically relies on expert knowledge or extensive trial-and-error[2]. Hyperparameter optimization (HPO) has therefore been widely studied through automated search methods such as grid search, random search, Bayesian optimization, and population-based metaheuristics[3]. Among these approaches, metaheuristic algorithms provide a gradient-free and black-box-compatible search strategy, making them well suited to the non-convex and computationally expensive fitness landscapes encountered in neural network training. Evolutionary algorithms such as Genetic Algorithms (GA), as well as swarm-based optimizers, have been successfully applied to HPO problems with varying degrees of effectiveness. The Aquila Optimizer (AO), introduced by Abualigah et al.[4], is a recent nature-inspired optimization algorithm that models the hunting strategies of Aquila eagles through four distinct search phases. While AO has demonstrated promising results on several benchmark optimization problems, its fixed phase transitions and absence of recombination operators may reduce search diversity in mixed discrete–continuous optimization problems such as CNN hyperparameter tuning. To address these limitations, this paper proposes a hybrid AO–GA metaheuristic that combines AO's phase-based exploration with GA recombination mechanisms. The proposed algorithm incorporates three

enhancements: opposition-based learning (OBL) for population initialization[5], a cosine-annealed mutation schedule inspired by learning-rate annealing strategies[6], and distance-aware archive-guided crossover to balance exploration and exploitation during the hybrid search phase. Experiments are conducted on four image classification benchmarks—MNIST, Fashion-MNIST, EMNIST, and CIFAR-10—using a fixed evaluation budget of 610 fitness evaluations per run and three independent random seeds. The proposed hybrid approach is compared against standalone AO and GA baselines under identical experimental conditions. The main contributions of this paper are summarized as follows: A hybrid AO–GA metaheuristic algorithm designed for CNN hyperparameter optimization. Integration of three mechanisms—opposition-based learning initialization, cosine-annealed mutation scheduling, and distance-aware archive-guided crossover—within a unified optimization framework. A systematic empirical evaluation across four image classification datasets demonstrating that the proposed hybrid generally improves upon the standalone AO on the evaluated datasets and achieves competitive performance relative to the GA.

II. RELATED WORK

Hyperparameter optimization (HPO) plays a critical role in deep learning because the performance of neural networks is strongly influenced by their hyperparameter configurations. Early approaches relied on simple search strategies such as grid search and random search. Bergstra and Bengio demonstrated that random search can outperform grid search in high-dimensional hyperparameter spaces by allocating trials more efficiently across parameters [2]. More recent research has explored advanced optimization techniques such as Bayesian optimization and automated machine learning (AutoML) frameworks to improve search efficiency and reduce manual intervention [3]. In addition to model-based methods, population-based metaheuristic algorithms have gained attention for hyperparameter optimization due to their gradient-free nature and ability to operate on black-box objective functions. Evolutionary algorithms such as Genetic Algorithms (GA) have been widely used[7] to evolve candidate solutions through selection, crossover, and mutation operators. These algorithms are particularly suitable for mixed discrete–continuous search spaces that arise in neural network hyperparameter tuning, where traditional gradient-based optimization methods cannot be directly applied. Swarm-based optimization algorithms have also been explored for solving complex optimization problems. Among recent developments, the Aquila Optimizer (AO) proposed by Abualigah et al.[4] models the hunting behavior of Aquila eagles through four search phases that balance exploration and exploitation. AO has demonstrated competitive performance on a variety of benchmark optimization tasks. However, similar to many swarm-based algorithms, AO does not include recombination mechanisms that enable information exchange between candidate solutions, which may limit population diversity in complex optimization problems such as CNN hyperparameter optimization. Hybrid metaheuristic strategies have therefore been investigated to combine the complementary strengths of different optimization paradigms. Integrating evolutionary recombination mechanisms with swarm-based exploration may improve both population diversity and convergence behavior during optimization. Motivated by this idea, this work proposes a hybrid AO–GA framework that integrates AO's phase-based search mechanism with GA-style recombination operators. The proposed framework incorporates opposition-based learning initialization[5], cosine-annealed mutation scheduling [6], and a distance-aware archive-guided crossover mechanism to promote diversity and improve information exchange between candidate solutions under a fixed evaluation budget.

III. METHODOLOGY

3.1 Problem Formulation

Hyperparameter optimization (HPO) is formulated as a continuous minimization problem over a bounded search space. Let $x = [x_1, x_2, \dots, x_d] \in R^d$ denote a candidate CNN hyperparameter configuration where each dimension x_i is bounded by $[lb_i, ub_i]$, representing the lower and upper bounds of the i-th hyperparameter. The objective function minimizes classification error on a held-out subset:

$$f(x) = 1 - Acc(x)$$

where Acc(x) denotes the subset classification accuracy obtained after training a convolutional neural network (CNN)[1] using the hyperparameter configuration x. Minimizing $f(x)$ therefore corresponds to maximizing the classification accuracy.

The search space consists of four hyperparameters commonly used in CNN training: learning rate $lr \in [0.0001, 0.1]$, batch size $bs \in [16, 256]$, dropout rate $do \in [0, 0.5]$, and SGD momentum $mom \in [0.5, 0.99]$[8]–[10]. These ranges were selected based on commonly reported values in deep learning literature. All candidate solutions are clipped to their bounds before evaluation to ensure feasible hyperparameter configurations:

$$x_i = \min(\max(x_i, lb_i), ub_i)$$

where x_i represents the i-th hyperparameter value in the candidate solution and lb_i and ub_i denote its lower and upper bounds. This boundary handling operation ensures that every generated hyperparameter value remains within its valid search interval before the CNN model is trained.

3.2 Opposition-Based Initialization

The initial population is generated using Opposition-Based Learning (OBL)[5]. In standard random initialization, candidate solutions are generated uniformly within the search space. However, this approach may lead to limited coverage of the solution space and reduced diversity in the initial population.

OBL improves the diversity of the initial population by simultaneously considering a candidate solution and its opposite counterpart. For each randomly generated candidate x, an opposite solution x^{opp} is computed as

$$x_i^{\{opp\}} = lb_i + ub_i - x_i, i = 1,2, \dots, d$$

where x_i represents the i-th component of the candidate solution, and lb_i and ub_i denote the lower and upper bounds of that dimension. The opposite solution x_i^{opp} is therefore located symmetrically across the center of the search interval.

Both the original candidate and its opposite are evaluated, and the better solutions are retained in the population. This strategy may increase the probability that the initial population contains individuals located closer to promising regions of the search space.

3.3 Phase-Structured Search

The original Aquila Optimizer (AO) [4] consists of four phase-structured position update strategies but does not include recombination mechanisms or explicit diversity management. The standard Genetic Algorithm (GA) [7], in contrast, uses crossover and mutation operators but does not incorporate phase-structured exploration control.

The proposed AO–GA hybrid combines the strengths of both methods. Specifically, the algorithm retains the four-phase search structure of AO but replaces the original Phase 3 position update with a GA-based recombination operator. In addition, three mechanisms are introduced to improve population diversity and search stability: opposition-based learning (OBL) initialization to improve initial population coverage, a cosine-annealed mutation schedule to adapt mutation pressure during optimization, and a distance-aware archive to guide parent selection during crossover. The proposed optimizer follows the four-phase search strategy of the Aquila Optimizer (AO) [4]. The phase schedule controls how the search gradually moves from global exploration in early iterations to local exploitation in later iterations. Each phase applies a different update rule to balance exploration and exploitation during the search.

Table 1: Phase Schedule of the Hybrid Optimizer

Phase	Condition	Iterations	Strategy
1	$t < 0.20$	1–4	Levy-flight exploration
2	$0.20 \leq t < 0.45$	5–9	Narrowed AO search
3	$0.45 \leq t < 0.80$	10–16	AO–GA hybrid recombination
4	$t \geq 0.80$	17–20	Fine exploitation

The mathematical update rules for the four phases are described below.

Phase 1: Levy Exploration.

Candidate solutions are updated using a Levy-flight step:

$$x_i^{\{new\}} = x^{\{best\}} + (1-t)L(ub - lb) \cdot 0.1$$

where x^{best} denotes the best solution found so far, ub and lb represent the upper and lower bounds of the search space, and L is a Levy vector generated using Mantegna's ´ algorithm [4]. Levy-flight steps allow the optimizer to perform ´ large exploratory moves, helping the algorithm explore distant regions of the search space and avoid premature convergence during early iterations.

Phase 2: Narrowed Search.

$$x_{\{i,d\}}^{\{new\}} = x_d^{\{best\}} - r\left|x_d^{\{best\}} - x_{\{i,d\}}\right|$$

where $x_{i,d}$ represents the d-th dimension of the i-th candidate solution, x_d^{best} is the corresponding dimension of the best solution, and $r \sim U(0,1)$ is a uniformly distributed random number. This update moves candidate solutions toward the best solution, gradually reducing the search radius and focusing the search around promising regions.

Phase 3: AO–GA Hybrid.

The original Aquila Optimizer updates solutions using position update strategies but does not include recombination operators commonly used in evolutionary algorithms. To improve population diversity, the proposed method integrates genetic algorithm (GA) crossover operations with AO updates. First, an exploitation-oriented AO update is computed:

$$x_i^{\{AO\}} = x^{\{best\}} + N(0,1)\left(x^{\{best\}} - x_i\right)$$

where $N(0,1)$ denotes a Gaussian random variable. This step generates a new candidate solution around the current best solution while maintaining stochastic variation. The AO-generated candidate is then combined with an archive solution x^j using arithmetic crossover:

$$y_i = \alpha x_i^{AO} + (1 - \alpha)x^j$$

where $\alpha \sim \text{Beta}(2,1)$ is a crossover coefficient that controls the contribution of each parent solution. This hybrid recombination combines the exploitation capability of the Aquila Optimizer with the diversity generation mechanism of genetic algorithms, which may help generate more diverse candidate solutions.

Phase 4: Local Exploitation.

$$x_{\{i,d\}}^{\{new\}} = x_{\{d\}}^{\{best\}} - 0.05r\left|x_{\{d\}}^{\{best\}} - x_{\{i,d\}}\right|$$

where $r \sim U(0,1)$. This update performs small local adjustments around the best solution, enabling the optimizer to refine the final solution during the last iterations.

3.4 Archive-Guided Parent Selection

An archive is used to store the top K = 5 solutions discovered during the optimization process. Maintaining such an archive helps preserve high-quality solutions and prevents the loss of promising candidates during later iterations of the search, a strategy commonly used in evolutionary and population-based optimization algorithms [16]. When generating new solutions in the hybrid phase, a parent solution is selected from the archive. The selection process considers both the quality of the solution and its diversity relative to the current candidate solution.

The selection score is computed as

$$score(a) = w_f rank_f(a) + w_d \hat{d}(x^a, x_i)$$

where $rank_f(a)$ denotes the fitness rank of the archive solution a, with better solutions receiving higher ranks. The term $\hat{d}(x^a, x_i)$ represents the normalized Euclidean distance between the archive solution x^a and the current candidate solution x_i, which measures the diversity between the two solutions. The coefficients w_f and w_d control the relative importance of fitness and diversity in the selection process. In this work $w_f = w_d = 0.5$, meaning that both solution quality and diversity are given equal importance when selecting the archive parent.

3.5 Cosine-Annealed Mutation

Mutation probability follows a cosine annealing schedule:

$$\mu(t) = \mu_{min} + \frac{1}{2}(\mu_{max} - \mu_{min})\left(1 - cos(\pi t)\right)$$

where μ(t) denotes the mutation probability at iteration t, μ_{min} and μ_{max} represent the minimum and maximum mutation probabilities, and $t = k/T$ is the normalized iteration ratio with k being the current iteration and T the total number of iterations. This schedule gradually increases the mutation probability as the optimization progresses. In early iterations the mutation rate remains low, allowing the algorithm to exploit promising solutions. In later iterations the mutation probability increases, introducing additional diversity and helping the optimizer escape local optima.

3.6 Elite Preservation and Evaluation Budget

Elite preservation (elitism) ensures that the best solution discovered so far is retained in the population at every iteration. This strategy is commonly used in evolutionary algorithms to prevent the loss of high-quality solutions during the search process [7].

IV. Experimental Setup

4.1 Datasets

The proposed optimizer is evaluated on four widely used image classification benchmarks: MNIST [11], Fashion-MNIST [13], EMNIST [14], and CIFAR-10 [15]. MNIST contains grayscale images of handwritten digits, while Fashion-MNIST replaces these digits with clothing product categories. EMNIST extends MNIST with a larger set of handwritten digit samples, and CIFAR-10 contains color images from 10 natural object classes. These datasets represent tasks of increasing complexity, enabling a comprehensive evaluation of the optimization algorithm.

4.2 CNN Training Configuration

Each candidate hyperparameter configuration is evaluated by training a convolutional neural network (CNN). The CNN consists of two convolutional layers with 32 and 64 filters using 3 × 3 kernels. Each convolutional layer is followed by a max-pooling layer to reduce the spatial dimension of feature maps. The convolutional layers are followed by two fully connected layers with 128 hidden units and 10 output units.

Dropout regularization is applied before the output layer to reduce overfitting, and the network is trained using the stochastic gradient descent (SGD) optimizer [12]. The optimizer searches for the following hyperparameters: Learning rate, Batch size, Dropout rate, SGD momentum. Each candidate configuration is trained for 5 epochs on a stratified subset of 10,000 training samples and evaluated on 2,000 held-out samples. Stratified sampling is used to ensure that the class distribution of the subsets remains balanced.

The objective value returned to the optimizer is defined as

$$f(x) = 1 - Acc_{val}$$

Acc_{val} denotes the held-out subset classification accuracy obtained after training the CNN with configuration x. Minimizing this objective therefore corresponds to maximizing the held-out subset accuracy.

4.3 Optimization Budget

To ensure fair comparison across algorithms, all optimizers are evaluated under the same computational budget. Each optimization run is limited to a total of 610 fitness evaluations using a population size of 30 and 20 optimization iterations. The initialization stage evaluates 30 candidate solutions. During the optimization stage, 29 new candidate solutions are evaluated in each iteration while one elite solution is preserved from the previous iteration. This results in 20 × 29 = 580 evaluations during optimization, giving a total of 30 + 580 = 610 fitness evaluations per run. Using the same evaluation budget ensures that performance differences between algorithms are due to their search efficiency rather than differences in computational effort. Experiments are repeated using three independent random seeds (42, 123, and 456) to account for stochastic variation in both neural network training and the optimization process.

4.4 Implementation and Hardware

All experiments were implemented in Python. The optimization algorithms and CNN training pipeline were executed on a workstation equipped with an NVIDIA GeForce RTX 3050 Laptop GPU (6 GB VRAM) with CUDA support, an Intel CPU, and 16 GB of system memory. GPU acceleration was used during the fitness evaluation stage to speed up CNN training. The complete experimental suite of 36 runs (3 algorithms × 4 datasets × 3 seeds) required approximately 59.8 hours of wall-clock time, with an average runtime of 1.66 hours per run.

4.5 Baseline Algorithms

The proposed AO–GA hybrid optimizer is compared with two baseline metaheuristic algorithms: the Genetic Algorithm (GA) and the Aquila Optimizer (AO). All algorithms use the same fitness function, CNN architecture, evaluation budget, and dataset splits to ensure a fair comparison.

4.6 Evaluation Metrics

Performance is measured using the classification accuracy obtained with the best hyperparameter configuration discovered by each algorithm. For each dataset, results are reported as the mean accuracy across three independent runs.

V. Results And Discussion

5.1 Optimization Performance

Table II reports mean classification accuracy and average runtime across three independent seeds (42, 123, 456). The proposed AO–GA hybrid achieves the highest mean accuracy on Fashion-MNIST (0.8938 ± 0.0042), outperforming GA by 0.37 pp and AO by 1.33 pp, and attains the single highest observed accuracy on CIFAR-10 (0.6350 vs. GA's best of 0.6310). Across all twelve dataset–seed combinations, AO– GA and GA each win six runs while AO wins none. On MNIST and EMNIST, GA leads AO–GA by 0.03 pp and 0.13 pp respectively, which may reflect simpler optimization characteristics of digit-recognition tasks. AO–GA achieved larger improvements on Fashion-MNIST: it leads on Fashion-MNIST and achieves the best single run on CIFAR-10 despite a marginally lower mean (0.08 pp behind GA). The AO produced the lowest mean performance among the evaluated methods across the tested datasets, suggesting that OBL initialization, cosine mutation scheduling, and distance-aware archive crossover may improve exploration–exploitation balance relative to the standalone optimizer.

5.2 Convergence Behaviour

Fig. 1 plots the mean best-accuracy curve ± one standard deviation across seeds for all three algorithms on each dataset. AO–GA reached its best observed solution earlier in some evaluated runs on digit-recognition tasks, achieving peak performance by iteration 2–3 on MNIST versus iteration 16 for GA. This may be associated with OBL initialization and broader initial exploration during optimization. AO exhibits prolonged stagnation in some runs—holding at 0.8725 for fourteen consecutive iterations on Fashion-MNIST (seed 42) and improving only in the final one to three iterations on CIFAR-10—while AO–GA and GA generally continue improving throughout optimization. AO–GA exhibited competitive stability across seeds on several evaluated datasets, in contrast to AO, which exhibited larger variation on Fashion-MNIST (std 0.0060) and CIFAR-10 (std 0.0063).

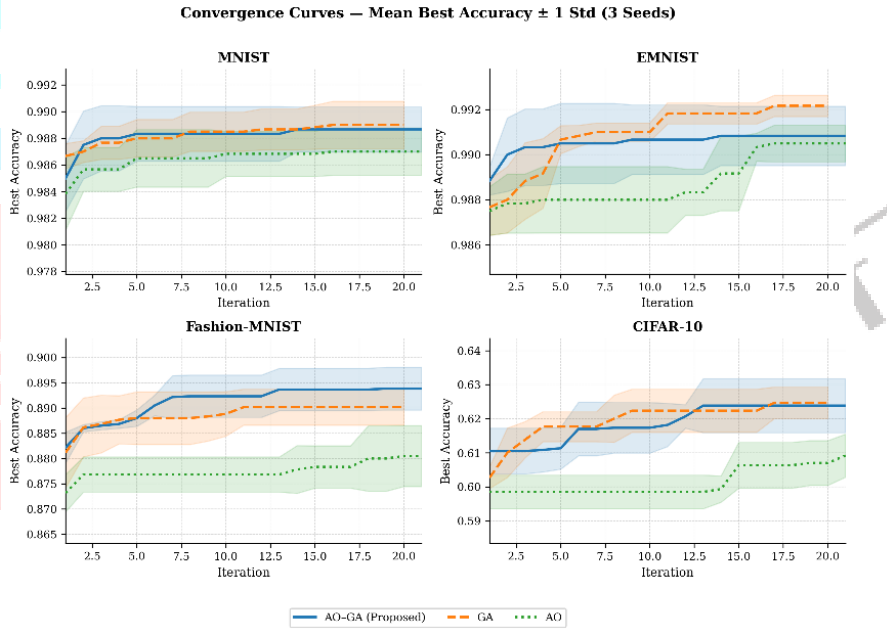

Figure 1: Convergence curves

5.3 Computational Cost

Runtime results are included in Table II. AO–GA exhibited lower runtime on several evaluated digit-recognition datasets, completing MNIST in 1.22 h versus 1.83 h for GA (−33%) and Fashion-MNIST in 1.70 h versus 2.10 h for GA (−19%). Runtime differences may be influenced by the interaction between discovered hyperparameter configurations, dataset characteristics, and optimization trajectories. On CIFAR-10, all three algorithms exhibited comparable runtime (1.44 h, 1.34 h, 1.51 h respectively). On EMNIST, AO requires 2.39 h nearly double AO–GA (1.26 h) and GA (1.17 h)—yet produces inferior accuracy, making it comparatively less time-efficient under the evaluated experimental setup.

5.4 Population Diversity Analysis

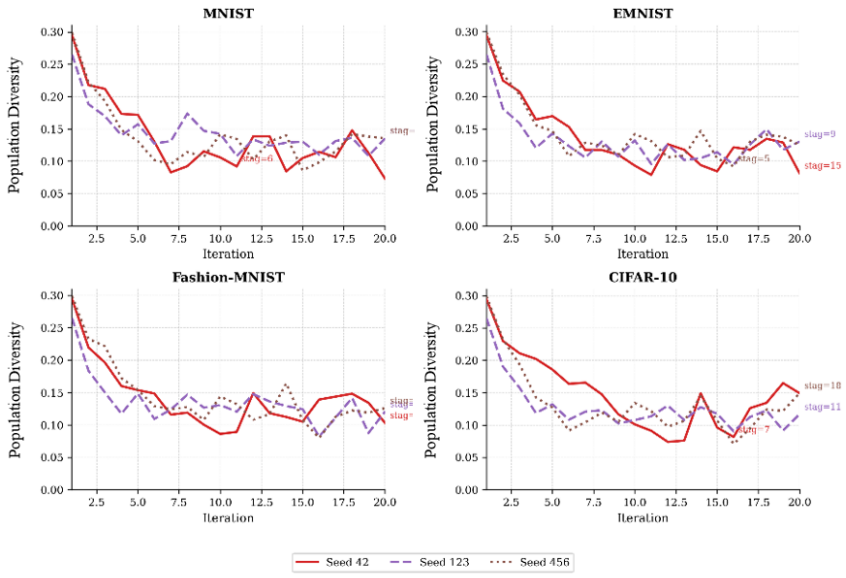

Figure 2: Population diversity during optimization

Fig. 2 illustrates changes in population diversity during optimization. GA diversity generally decreases over iterations, indicating convergence of candidate solutions as optimization progresses. AO–GA retains relatively higher diversity in later iterations, suggesting that archive-guided crossover and mutation mechanisms may contribute to maintaining broader exploration during the search process.

Table 2: CLASSIFICATION ACCURACY (MEAN ± STD) AND AVERAGE RUNTIME

Dataset	GA	AO	AO–GA	GA(h)	AO (h)	AO–GA (h)
MNIST	0.9890± 0.0018	0.9870 ±0.0018	0.9887± 0.0017	1.83	1.85	1.22
Fashion-MNIST	0.8902± 0.0036	0.8805 ±0.0060	0.8938± 0.0042	2.10	2.12	1.70
CIFAR-10	0.6247 ±0.0046	0.6092 ±0.0063	0.6238± 0.0079	1.44	1.34	1.51
EMNIST	0.9922 ±0.0005	0.9905 ±0.0008	0.9908 ±0.0013	1.17	2.39	1.26

5.5 Best Hyperparameter Configurations

Table III reports the best hyperparameter configuration per algorithm per dataset (best solutions returned by the optimizer; solutions are clipped to valid bounds before CNN evaluation, see Section III-A)

Table 3: BEST HYPERPARAMETER CONFIGURATIONS PER ALGORITHM

Dataset	Algo	Acc.	LR	Batch	Dropout	Momentum
MNIST	GA	0.9905	0.07751	139	0.315	0.848
MNIST	AO	0.9885	0.09491	168	0.447	0.863
MNIST	AO-GA	0.9910	0.10000	48	0.000	0.575
Fashion-MNIST	GA	0.8950	0.07004	53	0.110	0.778
Fashion-MNIST	AO	0.8890	0.06652	144	0.302	0.912
Fashion-MNIST	AO-GA	0.8980	0.05091	19	0.154	0.697
CIFAR-10	GA	0.6310	0.05202	82	0.149	0.788
CIFAR-10	AO	0.6180	0.03733	58	0.224	0.732
CIFAR-10	AO-GA	0.6350	0.03688	50	0.132	0.699
EMNIST	GA	0.9925	0.07185	97	0.341	0.766
EMNIST	AO	0.9915	0.10000	33	0.363	0.661
EMNIST	AO-GA	0.9920	0.07651	57	0.479	0.750

Across the evaluated datasets, AO–GA frequently produced smaller batch sizes (19–57) than GA (53–139) and AO (33–168), and lower momentum (0.575–0.750 vs. 0.778–0.912 for AO). Dropout values vary across datasets, ranging from 0.000 on MNIST to 0.132–0.479 on CIFAR-10 and EMNIST.

VI. Conclusion

This paper proposed a hybrid metaheuristic algorithm that integrates the Aquila Optimizer (AO) with Genetic Algorithm (GA) operators for automated hyperparameter optimization of convolutional neural networks. The proposed AO–GA hybrid incorporates opposition-based learning initialization, cosine-annealed mutation scheduling, and distance-aware archive-guided crossover to improve exploration–exploitation balance relative to standalone AO. Experiments on four image classification benchmarks (MNIST, Fashion-MNIST, EMNIST, and CIFAR-10) across three independent random seeds demonstrate that AO–GA improves performance relative to standalone AO while achieving competitive performance relative to GA. On Fashion-MNIST, AO–GA achieves the highest mean accuracy of 89.38%, outperforming GA by 0.37 percentage points and AO by 1.33 percentage points. On CIFAR-10, AO–GA achieves the single highest observed accuracy of 63.50% across all algorithms and seeds. On MNIST and EMNIST, GA achieves the highest mean accuracy, while AO–GA remains highly competitive, trailing by only 0.03 and 0.13 percentage points respectively and reducing runtime by 33% relative to GA on MNIST. Overall, AO–GA demonstrated competitive computational efficiency across the evaluated datasets under the experimental setup used in this study. One limitation of this work is that experiments were performed using a relatively small CNN architecture and fixed evaluation budget; results may differ for larger architectures and search spaces.

In summary, AO–GA provides a competitive optimization strategy that maintains AO exploration capabilities while leveraging GA-style recombination to promote solution diversity. Future work will investigate the scalability of the proposed hybrid to larger CNN architectures, higher-dimensional hyperparameter spaces, and other deep learning model families.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, vol. 25, 2012, pp. 1097–1105.

[2] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” Journal of Machine Learning Research, vol. 13, pp. 281–305, Feb. 2012.

[3] M. Feurer and F. Hutter, “Hyperparameter optimization,” in Automated Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Springer, 2019, ch. 1, pp. 3–33.

[4] L. Abualigah et al., “Aquila optimizer: A novel meta-heuristic optimization algorithm,” Computers & Industrial Engineering, vol. 157, p. 107250, Jul. 2021.

[5] H. R. Tizhoosh, “Opposition-based learning: A new scheme for machine intelligence,” in Proc. CIMCA, 2005, pp. 695–702.

[6] I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts,” in Proc. ICLR, 2017.

[7] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[8] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[9] N. Srivastava et al., “Dropout: A simple way to prevent neural networks from overfitting,” JMLR, vol. 15, pp. 1929–1958, 2014.

[10] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in Proc. ICML, 2013.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278– 2324, 1998.

[12] L. Bottou, “Stochastic gradient descent tricks,” in Neural Networks: Tricks of the Trade, Springer, 2012.

[13] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” arXiv:1708.07747, 2017.

[14] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “EMNIST: An extension of MNIST to handwritten letters,” in Proc. IJCNN, 2017.

[15] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Univ. of Toronto, Tech. Rep., 2009.

[16] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, 2001.