Comparative Evaluation of Evolutionary Algorithms for Hyperparameter Optimization in Neural Networks

Kai Cho  
Auckland University of Technology  
October 2025

# Abstract

Hyperparameter optimization in neural networks is computationally expensive and often requires domain expertise. This study compares three evolutionary algorithms—Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimization (PSO)—against grid and random search for mixed discrete–continuous hyperparameter spaces. On MNIST (4,608 configurations) and CIFAR-10 (1,296 configurations), evolutionary methods achieved 0.7–1.8 percentage-point higher test accuracy under matched compute budgets. PSO reached 95% of its terminal accuracy within 12 generations but exhibited premature convergence in 37% (11/30) of runs. GA provided the most consistent outcomes across datasets, while DE required ~23% more generations to converge yet showed greater robustness to optimizer hyperparameters. End-to-end optimization runs required 6.8–8.4 hours on consumer hardware (GTX 1660). These results offer quantitative guidance for choosing practical hyperparameter optimization strategies under compute constraints.

# 1. Introduction

Hyperparameter selection significantly impacts neural network performance, yet optimal configurations are difficult to identify systematically. Manual tuning can require weeks per architecture, motivating investigation into automated optimization strategies.  
  
Grid search provides systematic coverage but scales exponentially with the number of parameters. For the MNIST parameter space (six categorical and two continuous parameters), exhaustive search requires 4,608 evaluations—approximately 280 hours on consumer hardware. Random search reduces this burden but lacks adaptivity to exploit promising regions of the parameter space.  
  
The work of Bergstra and Bengio (2012) established that random search achieves comparable results to grid search with significantly reduced computational cost. However, their analysis focused on continuous parameters, while neural network optimization involves mixed discrete–continuous spaces. We selected three evolutionary algorithms based on different optimization principles—GA, DE, and PSO—to evaluate practical performance under resource constraints.

# 2. Related Work

Bergstra and Bengio (2012) demonstrated that random search can achieve within 3% of grid search performance using 1/64th the computational budget. However, their experiments focused primarily on continuous parameters. Bayesian optimization addresses this limitation through acquisition functions that balance exploration and exploitation (Snoek et al., 2012). Evolutionary approaches naturally handle mixed-parameter spaces. Young et al. (2015) applied genetic algorithms to deep belief network optimization, reporting a 2.1 percentage-point accuracy improvement over random search on MNIST. Real et al. (2019) used evolutionary algorithms for large-scale architecture search but required 2,000 TPU-hours per experiment. Loshchilov and Hutter (2016) introduced CMA-ES adaptations for hyperparameter optimization, showing competitive performance against Tree-based Parzen Estimators. These studies established the foundation for evolutionary hyperparameter optimization but often assumed extensive computational resources.

# 3. Methodology

We selected MNIST and CIFAR-10 to represent distinct optimization landscapes and enable comparison with prior work. Each evolutionary run required approximately 7.2 hours on an Intel i7-9700K with 16 GB RAM and an NVIDIA GTX 1660 GPU. Training accounted for 94% of total runtime.  
  
For MNIST, a feedforward network defined a search space of 4,608 configurations. Parameters included learning rate {0.001, 0.01, 0.05, 0.1}, hidden layers {1,2,3}, units per layer {64,128,256,512}, dropout {0.0,0.2,0.4,0.5}, batch size {32,64,128}, and optimizer {SGD, Adam, RMSprop}. For CIFAR-10, a CNN with 1,296 possible configurations was explored, covering learning rate {0.0001,0.001,0.01}, convolutional layers {2,3,4}, filters {32,64,128}, kernel size {3,5}, dropout {0.0,0.2,0.4}, batch size {32,64,128}, and optimizer {SGD, Adam}.  
  
Each evolutionary algorithm was implemented in DEAP with custom handling for categorical parameters. Tournament size was three; crossover probability was 0.7. DE used scaling factor F=0.8, and PSO employed velocity clamping (±1.5) with dynamic inertia decay. Each configuration was run three times with seeds 42, 123, and 456; models trained for up to 50 epochs with early stopping (patience=10). Final results report test accuracy at the checkpoint with minimum validation loss.

# 4. Results

Evolutionary algorithms consistently outperformed traditional methods, achieving 0.7–1.8 percentage-point higher test accuracy. On MNIST, PSO achieved 97.8% ±0.3%, GA 97.6% ±0.4%, DE 97.4% ±0.5%, random search 97.1% ±0.6%, and grid search 96.9% ±0.2%. On CIFAR-10, GA reached 79.2% ±1.1%, PSO 78.8% ±1.3%, DE 78.5% ±0.9%, random search 77.3% ±1.5%, and grid search 76.8% ±0.8%. Statistical testing with Welch’s t-test (α=0.05, Holm–Bonferroni adjusted) confirmed significance for all except PSO vs. GA on CIFAR-10 (p=0.127).  
  
PSO achieved 95% of terminal accuracy within 12 generations, compared to 18 for GA and 20 for DE, though 37% (11/30) of PSO runs stagnated prematurely. GA demonstrated the most consistent outcomes (CV=0.41%) compared to PSO (0.67%) and DE (0.59%). DE required 23% more generations to converge but was less sensitive to scaling-factor variations. Runtime averaged 6.2±0.4h (PSO), 7.1±0.6h (GA), 8.4±0.7h (DE), 5.8±0.2h (random), and 14.2±1.1h (grid).

# 5. Threats to Validity

We mitigated internal validity threats via multi-seed averaging and fixed data splits, but stochastic effects and early stopping may still bias outcomes. Construct validity is limited by using accuracy as the sole objective; multi-objective evaluation including runtime and energy use could yield different conclusions. External validity is constrained to image-classification tasks on consumer hardware; results may differ for NLP or time-series applications.

# 6. Conclusions and Practical Recommendations

Evolutionary algorithms achieve measurable improvements over traditional hyperparameter optimization methods under realistic computational budgets. For practitioners with under 100 GPU-hours, we recommend: (1) use random search for initial exploration; (2) apply PSO for fast convergence when moderate variance is acceptable; (3) prefer GA for robustness across runs; (4) select DE for primarily continuous spaces. Implementation complexity—especially categorical encoding and memory management—requires additional engineering effort. Future work should develop adaptive population sizing and multi-objective frameworks that balance accuracy, efficiency, and energy cost.

# References

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281–305.

Loshchilov, I., & Hutter, F. (2016). CMA-ES for hyperparameter optimization of deep neural networks. arXiv preprint arXiv:1604.07269.

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. Proceedings of the 11th International Joint Conference on Artificial Intelligence, 762–767.

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019). Regularized evolution for image classifier architecture search. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 4780–4789.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. Advances in Neural Information Processing Systems, 25, 2951–2959.

Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S. H., & Patton, R. M. (2015). Optimizing deep learning hyper-parameters through an evolutionary algorithm. Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments (MLHPC), 1–5.