Optimizing Hyperparameters for Deep Learning Models Using Evolutionary Algorithms: Solving the Four-Class Intertwined Spiral Classification Problem

Siri Sri Churakanti*, Batyr Kenzheakhmetov*+, Bhavesh Krishnaram Bhavesh, CJ Chung*,
College of Arts and Sciences, Lawrence Technological University; +Astana IT University, Kazakhstan

*This project began in Fall 2024 as part of the MCS 5993 Topics in Computer Science: Evolutionary Computation and Deep Learning course. Authors marked with * continued to make additional contributions to the project during Spring 2025.*

ABSTRACT

This paper presents novel approaches to optimizing Deep Neural Networks (DNN) for the 4-class intertwined spiral classification problem using an Evolution strategy algorithm with 1/5 success rule and a Genetic Algorithm (GA) implemented via the DEAP framework. Unlike the simpler two-class spiral, this 4-class variant remains largely unexplored in machine learning fields. Evolutionary algorithms can provide a dynamic and adaptive mechanism that optimizes hyperparameters of DNNs for non-linear classification tasks and this study leverages ES(1+1) with the 1/5 success rule and DEAP based Genetic Algorithms. Without hyperparameter optimization, the baseline model achieved an average accuracy of 34.1% with an average loss of 1.32. GA with DEAP optimizing hyperparameters achieved the highest classification accuracy of up to 96.2%. ES(1+1) with the 1/5 success rule found a model that delivers 94.1% accuracy. These findings establish evolutionary algorithms as powerful tools for enhancing DNN performance and provide valuable insights for future advancements in deep learning optimization.

DATA COLLECTION

The dataset is constructed as four intertwined spirals, each represented in 2D polar coordinates. Although synthetic, the four-class spiral dataset serves as a well-established benchmark for evaluating machine learning models on nonlinear separability [1]. Its controlled complexity allows for a clear understanding of how evolutionary algorithms enhance deep neural network (DNN) performance in challenging classification scenarios. For a class, the spiral points are generated using the equations shown below:

$$x(t) = a.t.\cos(t + b),\ y(t) = a.t.\sin(t + b)$$

where, a = scaling factor, t = angle in radians (range: [0, 10]), and b = angular offset for each class (b = 0, $\pi/2$, π, $3\pi/2$). These parameters ensure proper generation of the spiral classes, with t ∈ [0, 10] representing the angle. Gaussian noise $N(\mu, \sigma^2)$ was added to simulate real-world imperfections. The Gaussian noise with a mean (μ) of 0.0 and a variance (σ^2) of 0.03, which introduces variability to the dataset, making it more representative of real-world data. The input data X ∈ $\mathbb{R}^2$ is standardized to zero mean and unit variance using:

$$x' = \frac{(x-\mu_x)}{\sigma_x}$$

where μ_x and σ_x are the mean and standard deviation of the input data, respectively. Standardization ensures that all features are on the same scale, which is crucial for aiding convergence during training. Here in Fig. 1 shows the visualization of training dataset with 4 classes in red, blue, green and purple.

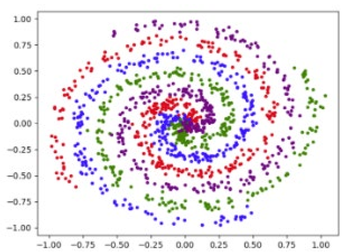

Fig. 1. A sample training dataset visualization

REFERENCES

1. Lang, K. J., Witbrock, M. J., 1988. Learning to tell two spirals apart. In: Touretzky, D.,Hinton, G., Sejnowski, T. (Eds.), Proceedings 1988 Connectionist Models SummerSchool. Morgan Kaufmann, Los Altos, CA, pp. 52–59.
2. Chan-Jin Chung and Robert G. Reynolds, "Knowledge-Based Self-Adaptation in Evolutionary Search", International Journal of Pattern Recognition and Artificial Intelligence, Vol. 14 No. 1 (2000), pp. 19-33
3. F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," Journal of Machine Learning Research, vol. 13, pp. 2171–2175, 2012.

METHODOLOGY

Baseline model: As a baseline model, a deep neural network (DNN) with three hidden layers was implemented and tested using Keras with TensorFlow as the backend. The network architecture consists of

- Input Layer: 2 neurons (x, y coordinates).
- Hidden Layers: 64 → 32 → 16 neurons, ReLU activation for hierarchical feature extraction.
- Output Layer: 4 neurons, SoftMax activation for classification.
- Training Parameters: RMSprop optimizer (learning rate: 0.01), batch size = 16, and 50 epochs with early stopping

ES (1+1) with 1/5 Success Rule Algorithm: Unlike conventional evolutionary strategies or genetic algorithms, ES(1+1) generates one offspring per iteration from a single parent using the 1/5 success rule [2]. This rule dynamically adjusts the mutation step size: if more than 20% of previous mutations improve fitness, the step size increases to promote exploration; otherwise, it decreases to exploit the search space. Hyperparameter structure of a candidate solution includes:

- Number of neurons: Between 8 and 512 per hidden layer
- Activation function: randomly selected from ReLU, ELU, Sigmoid, and Tanh
- Optimization: Between SGD, RMSprop, and Adam,
- Learning rate: Between 0.01 and 0.1, and
- Training batch size: between 16 and 64

The ES(1+1) optimization begins with a single parent solution. To generate a new offspring solution, Gaussian random number is added to the parent solution's parameters using the formula:

$$\hat{h} = h + gauss\left(\sigma.\left(max(h) - min(h)\right)\right)$$

where h = hyperparameter, σ = mutation step size, and gauss adds Gaussian noise. The mutation step size (σ) is adapted using the 1/5 success rule, which adjusts success rate of offspring over the past 10 generations:

$$\sigma_{t+1} = \begin{cases} \sigma_t \cdot \alpha, & if\ sucess\ rate > \frac{1}{5} \\ \sigma_t \cdot \beta & otherwise \end{cases}$$

where $\alpha > 1$ and $\beta < 1$ are scaling factors, and the success rate is defined as the fraction of offspring outperforming their parent.

Genetic Algorithm (GA) using DEAP: Genetic Algorithms (GAs) are powerful optimization techniques inspired by natural evolutionary processes, including selection, crossover, and mutation, to solve complex problems. Distributed Evolutionary Algorithms in Python (DEAP) framework [3] is utilized to implement the following GA. Each individual represents a unique configuration of hyperparameters, defined in the search space as $x \in R^4$:

$$x = [m, \eta, b, d]$$

where m, η, b, and d represent the number of neurons, learning rate, batch size, and dropout rate, respectively.

Selection: A tournament selection method selects individuals for reproduction. For a tournament size T, a subset of T individuals is randomly sampled, and the individual with the lowest validation loss is chosen:

$$x_{best} = arg\min_{x \in T} f(x)$$

where $f(x)$ denotes the fitness value (validation loss). Repeat this process until the desired number of individuals are selected for the next generation.

Crossover: To create offspring, the Blend Crossover (BLX - α) operator generates offspring by linearly interpolating between two parent solutions :

$$x_{offspring} = x_1 + a(x_2 - x_1)$$

where x_2 and x_1 are the parent individuals, α controls the interpolation extent.

Mutation: Gaussian mutation introduces diversity by perturbing parameters with noise from a Gaussian distribution.

$$x_i' = x_i + N(\mu = 0, \sigma)$$

where $N(\mu,\sigma)$ has mean μ=0 and standard deviation σ, where each parameter is adjusted by adding noise sampled from a Gaussian distribution.

EXPERIMENTAL RESULTS

Algorithm	Contributor	Run No.	Accuracy	Loss	Activation Function	Num neurons for each layer	Learning Rate	Batch Size	Optimizers
Baseline	Siri	40	0.341	1.31	ReLU	64, 32, 16	0.01	16	RMSprop
ES(1+1) 1/5 rule	Siri	4	**0.941**	**0.15**	ReLU	157,153,160	0.01	53	Adam
	Siri	5	0.937	0.19	ReLU	8, 512, 512	0.01	62	Adam
	Siri	3	0.934	0.17	ELU	272, 86, 214	0.01	44	SDG
GA-DEAP	Batyr	0	**0.962**	0.22	ReLU	68, 95	0.003	22	Adam
	Batyr	5	0.956	**0.17**	ReLU	187, 35	0.009	32	Adam
	Batyr	6	0.943	0.19	ReLU	190, 95	0.007	31	Adam

Table 1. Result comparisons between Baseline model, ES(1+1) with 1/5 success rule, and GA-DEAP

- Baseline model: Without evolutionary optimization, the baseline model achieved average accuracy of 34.1% with 1.31 as loss value over 50 runs. This serves as a reference point, underscoring the significant improvements gained through evolutionary algorithms.
- ES(1+1) with 1/5 rule: Models demonstrated robust performance across multiple runs. Run no. 4 achieved the highest accuracy among ES(1+1) configurations at 94.1% with a loss of 0.15. These results showcase the adaptability of ES(1+1) in finding effective architectures across varying network sizes and its robustness for complex 4 class spiral datasets with noises. A DNN architecture found at run no. 4 is drawn in Fig. 4.

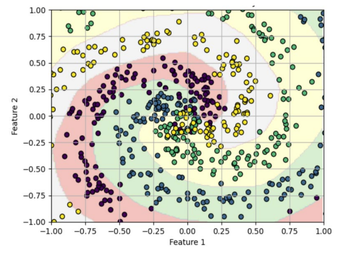

Fig. 2. Decision boundary of the best-performing model (ES(1+1) with 1/5 success rule of 94% , illustrating the separation of the four spiral classes

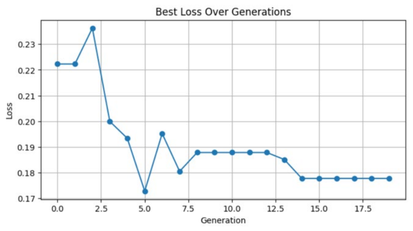

Fig. 3. Loss over Generations graph from Run No. 5 to illustrate the ES(1+1) with 1/5 succes rule algorithm's effectiveness in finding models with optimized hyperparameters.

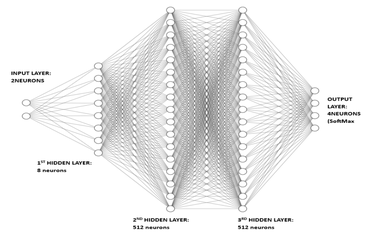

Fig. 4. Optimized DNN for ES(1+1), showcasing from Run No. 4 model with 8, 512, 512 hidden neurons.

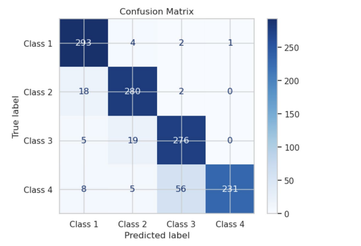

Fig. 5. Confusion matrix from the ES(1+1) optimized model (Run 5), showing true labels vs. predicted labels. Diagonal elements indicate correct classifications, while off-diagonal elements highlight misclassifications

- **GA using DEAP:** The best accuracy on Run no. 5 is 95.6% with loss of 0.17 in Fig. 7, but the highest accuracy is 96.2% on Run no. 0 with 0.22 loss, these both are balancing complexity and performance effectively. This demonstrates its efficiency in resource utilization. The slightly lower learning rate (0.009 vs. 0.01) likely contributed to finer optimization adjustments, resulting in a low loss.

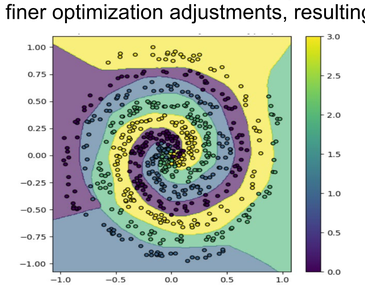

Fig. 6. Decision Boundary of a model found by Genetic Algorithm using DEAP which achieved 95.6% (Run 5)

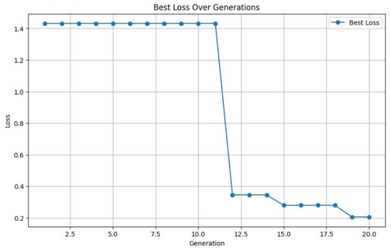

Fig. 7. Loss over generation graph from Run No. 5 to ilústrate the GA-DEAP algoritmos effectiveness in finding models with optimized hyperparameters