Deep Learning Classification of Heart Disease Risk Using Evolutionary Hyperparameter Optimization

Ryan Kaddis, Devson Butani, Siri Sri Churakanti, Batyr Kenzheakhmetov*, Bhavesh Krishnaram Bhavesh, Prerak Patel, CJ Chung
College of Arts and Sciences, Lawrence Technological University; *Astana IT University, Kazakhstan

ABSTRACT

Hyperparameters are an often overlooked aspect of training neural networks. There is, however, an advantage to finding a set of hyperparameters that produce optimal model results. Using a dataset that classifies the risk of heart disease based on a number of physical parameters, optimal hyperparameters are found for a classification model using various evolution-based optimization strategies. These methods include Evolutionary Strategy (1+1) with ⅕ rule, Evolutionary Strategy (N+M) with ⅕ rule, and Genetic Algorithm (GA) using the DEAP library. Models produced using these methods consistently achieved accuracies above 80%, with one model optimized using a GA-based approach reaching 96.6% accuracy.

INTRODUCTION

Hyperparameters detail the structure, functions, and methods from which a model is trained and evaluated. Hyperparameters encapsulate everything from the number of neurons in the model to the functions and parameters used to optimize model weights. The goal of this work is to investigate the impact of hyperparameters on model performance, and to develop algorithms for finding optimal hyperparameters that produce the best models. In essence, a model's performance is a function of the hyperparameters used in training. A number of hyperparameter optimization (HPO) techniques for different machine learning models are reviewed in [1], including the well known grid search and random search. Andoine expresses the benefits of finding alternative, less computationally expensive methods using auto-sklearn, AutoML [2]. Liao explains the strong relationship between hyperparameters and both model performance and inference time [3].

In this study, we leverage evolutionary computation, a robust and proven method for finding global minimums in complex and unknown functions. Three evolutionary algorithms are utilized to navigate the hyperparameter "function" to find the minimum model loss. Evolutionary strategies [5], like ES(1+1) and ES(N+M) with 1/5 rules, expand upon the concept of guided random searching by introducing a dynamic "step size", which specifies the range at which value are mutated. A Genetic Algorithm (GA) with DEAP (Distributed Evolutionary Algorithms in Python) [6] provides a flexible solution for HPO. GA mimics natural selection, evolving solutions through mutation and crossover to adapt to the optimization landscape.

DATASET

The dataset provided by Cleveland UCI [4] links heart disease risk factor to a number of different bodily metrics (13 attributes) including: age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting ECG, maximum heart rate, exercise-induced angina, ST depression, slope of peak ST segment, major vessels, and defects. The dataset contains only 297 samples.

This dataset is an excellent indicator of the performance capabilities of evolutionary HPO. Medical data is highly complex and abstract in terms of data analytics. In addition, there are less than 300 samples for a model to train on. The hyperparameter set is critical for finding working models to correctly interpret this dataset.

RESEARCH GOALS

The goal of this research is to develop Evolutionary Strategies and use them to find optimal hyperparameters for a best-fit model trained on the Heart Disease Risk dataset [4]. Model performance of tensorflow-based neural networks is evaluated on two metrics, model loss and accuracy. Model loss is calculated using binary cross-entropy (aka log-loss).

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i))$$

Model accuracy is calculated using the following formula.

$$\text{accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1 - |p(y_i) - y_i|)$$

One model is considered more fit than the other if the model loss is lower. This principle is used in evolution for identifying stronger offspring. Accuracy is used as the final human-readable evaluation metric. An accuracy above 80% indicates that a model has correctly identified patterns in the dataset.

METHODOLOGY

Dataset Management

The Cleveland UCI dataset is first split into an 80/20 split for training and testing data. The training set is then split again 80/20 for training and validation data. The dataset is stratified and scaled for consistency.

NN Model Architecture and Hyperparameters

NN Models with different hyperparameters are trained using the Keras API for Python. Models are composed of three Dense layers of neurons. Layers can have at most 1000 neurons. One instance of NN model architecture is shown in Figure 1. Neurons can use tanh, ReLU, ELU, and sigmoid as activation functions. The training batch size can be from 1 to 32. The optimizer can be SGD, RMSProp, or Adam. The learning rate can be from 0.01 to 0.1.

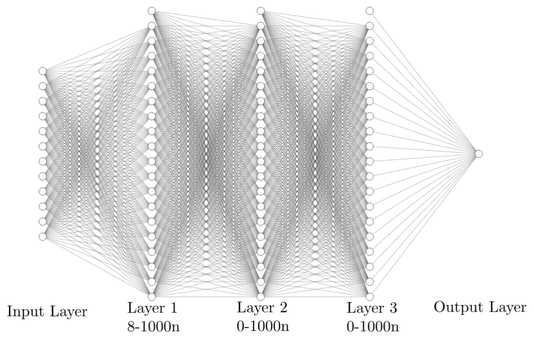

Fig. 1. DNN Layer Model Diagram. Three hidden layers in the model translate the 13 input variables into a single output neuron If output value is near 0, heart disease risk is low; if output value is close to 1, heart disease risk is high.

ES (N+M) with 1/5 success rule

The following mutation only method called Evolutionary Strategy (*N+M*) is used to optimize a set of hyperparameters to produce a model with the lowest possible loss [5]:

1. Based on the possible ranges for each of the hyperparameters, *N* parent models are generated with random hyperparameters as the baseline for the Evolutionary Strategy.
2. *M* children are created by selecting a hyperparameter used by a random parent.
3. The hyperparameters of the children are mutated using the following formula:

$$\hat{h} = h + \text{gauss}(\sigma * (\max(h) - \min(h)))$$

where *h* is the value of a hyperparameter and *σ* is the step size for the mutation.

4. The children are trained (using Keras) and evaluated, and the N best individuals by loss value among the parents and children are chosen as the parents of the next generation.
5. Steps 2-4 repeat, and the number of successful generations are tracked.
6. Whenever a large number of successes happen in a period, the step size *σ* is increased to potentially exit a local minima. Whenever there are few successes, the step size decreases.

$$\sigma_{t+1} = \begin{cases} \sigma_t \cdot \alpha & \text{if } S > \frac{1}{5} \\ \sigma_t \cdot \beta & \text{if } S \leq \frac{1}{5} \end{cases}$$

Where σ_t is current step size, α is the step size increase ratio, β is the step size decrease ratio, and *S* is the number of successes per period.

Genetic Algorithm (GA) using DEAP

A GA method utilizing DEAP (Distributed Evolutionary Algorithms in Python) library [6] were implemented and tested using the same dataset and the NN model architecture described in Fig 1. Unlike ES, this GA method uses both crossover and mutation operators. Crossover blends hyperparameters to create offspring. Mutation adds noise from N(0, σ^2). It did not use dynamic step sizes described in Step 6 above for ES. For the selection operator, tournament method is used with tournament size of 3.

EXPERIMENTAL RESULTS

Table1. Results using ES(1+1) with ⅕ Rule (N=1, M=1)

Contributor	Accuracy	Loss	Neurons*	Activation Function	Optimizer	Learning Rate	Batch Size
DB	80.0%	0.384	657	tanh	RMSProp	0.169	1
RK	85.0%	0.468	800	ELU	Adam	0.100	32
RK	85.0%	0.446	1310	ELU	Adam	0.001	32
RK	**86.6%**	**0.421**	**1069**	**ReLU**	**RMSProp**	**0.100**	**32**

(*) total number of neurons in the 3 layers

Table 2. Results using ES(N+M) with ⅕ Rule (N=10, M=80)

Contributor	Accuracy	Loss	Neurons*	Activation Function	Optimizer	Learning Rate	Batch Size
DB	85.0%	0.496	740	tanh	RMSProp	0.010	24
DB	83.3%	0.399	535	ReLU	Adam	0.042	1
DB	86.6%	0.426	794	ELU	RMSProp	0.001	32
DB	86.6%	0.447	1143	ELU	Adam	0.001	32
RK	**86.6%**	**0.403**	**794**	**ELU**	**RMSProp**	**0.005**	**32**

(*) total number of neurons in the 3 layers

Table 3. Results using GA-DEAP

Contributor	Accuracy	Loss	Neurons*	Activation Function	Optimizer	Learning Rate	Batch Size
BKB	**96.6%**	**N/A**	**125**	**ELU**	**Adam**	**0.040**	**1**
BKB	90.0%	N/A	125	ELU	RMSProp	0.042	1
BKB	86.6%	N/A	125	ELU	Adam	0.040	1
SIRI	89.9%	0.34	63	ELU	RMSProp	0.001	5
SIRI	83.3%	0.45	100	Sigmoid	RMSProp	0.001	3

(*) total number of neurons in the 3 layers

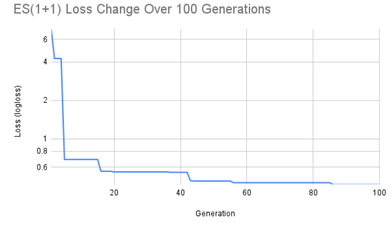

Fig. 2. ES(1+1) Loss Optimization. The loss value improves significantly in the first 20 generations, then continues to improve as the generations go on. Corresponds to row 4 in Table 1.

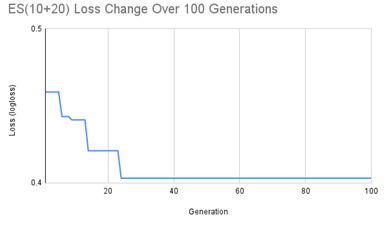

Fig. 3. ES(N+M) Loss Optimization. The loss value improves in fewer generations than in Fig. 2, since more children are created per generation. Corresponds to row 5 in Table 2.

DISCUSSION

Using evolution for hyperparameter optimization has been able to produce models consistently over the target accuracy of 80%, with some models reaching up to 96.6% accuracy. Certain hyperparameters, like optimizer and learning rate, correlate across trials (a lower learning rate and an optimizer (Adam or RMSProp) other than SGD produce the best results).

Across the three methods tested, ES(N+M) delivered the most consistent results and converged in fewer generations. The GA with DEAP found the best-performing model, reaching 96.6% accuracy, but it showed less consistent accuracies across trials. Our experiments show that Evolutionary HPO is a promising method for finding optimized hyperparameters, especially on smaller and more abstract datasets.

REFERENCES

[1] Monica and P. Agrawal, "A Survey on Hyperparameter Optimization of Machine Learning Models," *2024 2nd International Conference on Disruptive Technologies (ICDT)*, Greater Noida, India, 2024, pp. 11-15, doi: 10.1109/ICDT61202.2024.10489732.
[2] Andonie, R. (2019). Hyperparameter optimization in learning systems. Journal of Membrane Computing, 1(4), 279-291.
[3] Liao, L., Li, H., Shang, W., & Ma, L. (2022). An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks. ACM Transactions on Software Engineering and Methodology (TOSEM), 31(3), 1-40.
[4] Heart Disease Dataset, Cleveland UCI. https://www.kaggle.com/datasets/cherngs/heart-disease-cleveland-uci
[5] 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
[6] Fortin, F.-A., et al. (2012). DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research.