Evolutionary Hyperparameter Optimization to Find Lightweight CNN Models for Autonomous Steering

Devson Butani
Department of Math and Computer Science
Lawrence Technological University
Southfield, MI, USA
dbutani@ltu.edu

Ryan Kaddis
Department of Math and Computer Science
Lawrence Technological University
Southfield, MI, USA
rkaddis@ltu.edu

Chan-Jin Chung
Department of Math and Computer Science
Lawrence Technological University
Southfield, MI, USA
cchung@ltu.edu

***Abstract*—This research investigates the optimization of Convolutional Neural Networks (CNNs) for autonomous steering using the (N+M) Evolutionary Strategy (ES) with the 1/5th success rule. The primary objective is to develop a lightweight CNN model capable of real-time steering angle prediction, mimicking human driving behavior on predefined paths. The ES algorithm automates hyperparameter tuning, dynamically adjusting parameters such as filter sizes and layer configurations.**

Data collection encompasses driving scenarios recorded via the LTU ACTor autonomous driving platform, including variations in path direction and driving style. The dataset consists of timestamped images labeled with steering angles, pre-processed to focus on relevant visual information.

Initial experiments involve training a baseline CNN model, which is then refined using ES to significantly reduce model size while maintaining competitive predictive accuracy. The results highlight the viability of lightweight CNN architectures for real-time autonomous systems, striking a balance between computational efficiency and performance. This study not only advances research initiatives in using evolutionary strategies for autonomous driving applications but also lays the groundwork for deploying cost-effective, scalable solutions in self-driving technology.

***Index Terms*—Evolutionary strategy, Hyperparameter optimization, Convolutional neural networks, Lightweight CNN, Autonomous driving, Deep learning, Real-time performance, Autonomous vehicles, Cameras**

I. Introduction

Autonomous driving systems require robust models capable of making real-time decisions under varying conditions. A critical component of these systems is the prediction of steering angles based on camera input, which involves processing visual data effectively and efficiently on compute-limited onboard processors [2]. This research focuses on training and optimizing CNNs leveraging evolutionary strategies (ES) to automate the search for the best-performing and best size-reduced model configurations.

This research directly addresses the limitations that come with larger models and limited data. Larger models, while often achieving higher accuracy, come with increased computational cost and memory requirements, which can lead to slower inference times and higher energy consumption [5]. This poses a significant challenge for real-time applications like autonomous driving, where split-second decisions and energy efficiency are crucial. The size of a model directly impacts its deployment feasibility, particularly in resource-constrained on-board processors [8]. Minimizing model size while also training with limited data is the core design challenge in autonomous driving systems.

Furthermore, the performance of deep learning models is heavily reliant on the availability of large, diverse, and labeled datasets. In the context of autonomous driving, collecting and annotating such datasets for every conceivable scenario is impractical and cost-prohibitive [4] Even though limited data can lead to overfitting, where the model performs well on the training data but fails to generalize to unseen scenarios or conditions, this is often not a problem in real-world applications where the operational design domain (i.e., the area of interest) can be regulated. For example, deploying an autonomous vehicle specifically for a small city or a specific set of routes. With this method, we can leverage adding small datasets everyime the domain expands and utilize techniques like transfer learning and few shot learning to improve model performance over time [11]. While optimization techniques have been extensively explored in various domains, their application to creating lightweight and adaptable models for autonomous driving, specifically for real-time steering angle prediction, remains an active area of research.

Evolutionary Strategies (ES) offer a compelling advantage in this context. ES is a population-based optimization algorithm that leverages the collective intelligence of a population of candidate solutions to find the best-performing solution within a given search space [1]. Unlike gradient-based optimization methods, which can struggle in complex, non-differentiable, or noisy search spaces, ES algorithms are well-suited for exploring such landscapes [10]. This is particu-

larly relevant to hyperparameter optimization of CNNs for autonomous driving, where the relationship between model architecture, hyperparameters, and performance can be highly complex and non-linear.

A. Data Acquisition and Preprocessing

Equipment: The LTU ACTor autonomous driving platform, integrated with the Robot Operating System (ROS), was used for data collection. Sensor data, including images from the vehicle's forward-facing camera and corresponding steering angles, were recorded in rosbag files. This setup ensured synchronized visual and control data, essential for training and evaluating the models.

Environment: Data was collected along a predefined circular path: the red brick path surrounding Ockham's Wedge at LTU. To introduce variability and enhance model robustness, driving sessions included clockwise and counterclockwise directions, smooth and zigzag maneuvers, and driving along inner and outer path edges to simulate diverse spatial alignments. Figure X shows an overhead view of the data collection site.

Figure...

Extraction and Preprocessing: A custom script was developed to extract and preprocess data from rosbag files. Images were saved at 200 ms intervals, with filenames encoding timestamps and steering angles. Each image was cropped to exclude irrelevant regions, such as the sky and surrounding buildings. The resulting dataset consisted of 2,957 images, split into 70% training (2,069), 20% validation (592), and 10% testing (296). Even though the dataset is small, it is representative of real-world driving scenarios and provides a useful benchmark for evaluating model performance.

B. Research Goals

The primary goals of this research are:

1) Develop a framework to apply the (N+M) Evolutionary Strategy (ES) with the 1/5th success rule for automated hyperparameter tuning.
2) Minimize the size of the baseline CNN model while maintaining satisfactory steering angle prediction performance.
3) Validate the real-world applicability of ES-optimized models in autonomous vehicle systems.

II. Methodology

A. Model Training and Baseline Establishment

Models are trained using the Keras API with a PyTorch backend to leverage GPU compute capacity. The training process employs the Mean Squared Error (MSE) loss function, which quantifies the average squared difference between predicted and actual steering angles. The Mean Absolute Error (MAE) serves as a key performance metric, directly indicating the average deviation in steering angle predictions in degrees. This provides a straightforward and interpretable measure of model accuracy for autonomous steering applications. Minimizing both MSE and MAE is crucial for achieving precise control of the vehicle. These metrics serve as benchmarks for evaluating the efficacy of the Evolutionary Strategy (ES) with the 1/5 success rule in optimizing model architectures.

Initial experiments began with a single-layer CNN to establish a rudimentary baseline. However, the limited capacity of this architecture quickly became apparent, rendering it unsuitable for real-world driving scenarios. To address this, the PilotNet architecture, a CNN developed by NVIDIA [3] was adopted. PilotNet, an early milestone in autonomous steering research, demonstrated the potential of GPU-accelerated deep learning for this domain. Leveraging a proven architecture allowed for a more effective comparison of ES-optimized models against a recognized standard.

To enhance training efficiency and prevent over optimization, early stopping was implemented. Training instances are terminated if no improvement in the MSE metric is observed after four epochs. This strategy prevents the allocation of computational resources to hyperparameters that do not contribute to model performance, thereby accelerating the optimization process.

B. Hyperparameter Management

Initially, the hyperparameter space included individual layer units (number of filters in convolutional layers and neurons in fully connected layers), batch sizes, learning rates, activation functions, and optimizer selection. To facilitate a comprehensive exploration of potential architectures, layer units were allowed to vary from 20% to 300% of the PilotNet baseline.

However, preliminary experiments indicated that focusing the optimization efforts solely on layer units resulted in improved performance and reduced search time significantly. Consequently, batch size (32), learning rate (0.001), activation function (ReLU), and optimizer (Adam) were fixed to match the baseline model settings. This narrowed scope allowed the Evolutionary Strategy (ES) to concentrate on the most impactful architectural parameters, leading to more efficient exploration of the design space. The ranges of the layer units explored are provided in Table X.

Hyperparameter	**Search Space**
Layer Units	20% to 300% of PilotNet
Batch Size	1 - 32
Learning Rate	0.001 - 1.5
Activation Function	ReLU, eLU, Sigmoid, Tanh
Optimizer	Adam, SGD, RMSprop

C. Optimization Framework

CNN hyperparameters, specifically the number of filters in convolutional layers and the number of neurons in fully connected layers, are encoded as genes within an Evolutionary Strategy (ES) framework. The (N+M)-ES approach is employed, iteratively optimizing hyperparameters by combining the N best-performing parent models with M offspring models generated through mutation. This approach allows for both exploitation of promising regions of the hyperparameter space (through selection of top parents) and exploration of new possibilities (through mutation of offspring).

D. ES Algorithm

The optimization process unfolds as follows:

1) **Initialization:** Generate and train N parent models with random hyperparameters sampled within predefined ranges. The initial hyperparameters are drawn from a uniform distribution within the specified ranges. The number of parents, N, is a tunable parameter that controls the diversity of the population.
2) **Child Generation:** Create M offspring by mutating parent hyperparameters according to the following equation:

$$\hat{h} = h + \text{gauss}(\sigma * (\max(h) - \min(h))) \quad (1)$$

where $\hat{h}$ represents the mutated hyperparameter value. h is the original hyperparameter value from the parent model. gauss(μ, σ) is a random number drawn from a Gaussian distribution with mean μ and standard deviation σ. In this case, $\mu = 0$. σ is the mutation step size, controlling the magnitude of the mutation. $\max(h)$ and $\min(h)$ define the upper and lower bounds of the hyperparameter search space, respectively.
The term $(\max(h) - \min(h))$ normalizes the mutation step size to the range of the hyperparameter, ensuring that the mutation is proportional to the scale of the parameter.
3) **Training and Evaluation:** Train all offspring models and evaluate their performance based on validation loss (MSE) and accuracy (MAE).
4) **Selection:** Retain the top N models (from parents and children) based on validation performance to form the next generation's parent population. This selection process ensures that the most promising models are carried forward, driving the evolution towards improved performance.
The mutation step size (σ) is dynamically adjusted using the 1/5 success rule [12]:

$$\sigma_{t+1} = \begin{cases} \alpha * \sigma_t & \text{success rate} > \frac{1}{5} \\ \beta * \sigma_t & \text{success rate} \leq \frac{1}{5} \end{cases} \quad (2)$$

where σ_{t+1} is the updated step size for the next generation. σ_t is the current step size. $\alpha > 1$ is a scaling factor to increase the step size (e.g., 1.1). $\beta < 1$ is a scaling factor to decrease the step size (e.g., 0.9). The "success rate" is defined as the fraction of offspring that outperform their parents in terms of validation performance.
If the success rate is greater than 1/5, the step size is increased, encouraging greater exploration of the hyperparameter space. Conversely, if the success rate is less than or equal to 1/5, the step size is decreased, promoting more focused refinement of promising solutions. This adaptive mechanism ensures efficient exploration while avoiding premature convergence to local optima.

Computational Resources: Training is performed on Lawrence Technological University's NVIDIA A100 GPU server, enabling efficient parallel training of N parent and M offspring model pools. The large VRAM capacity of the A100 GPU allows for the use of larger N and M values, which increases the diversity of the population and enhances the exploration of the hyperparameter space. The specific values of N and M are determined based on the available computational resources and the complexity of the optimization problem.

Using a larger pool of parents and offsprings allows for the algorithm to explore the solution space nearby the current parameters. The larger N and M are, the faster the evolutionary search can find a viable solution. This enables an increase in N and M values as model size decreases. The 80GB VRAM of this GPU allows using higher N and M values to balance the tradeoff between exploration and exploitation.

E. Model Testing

Add paragraph detailing the real-time testing process using a 2D Sim. Also describe how the pipeline works (image -¿ model -¿ steering). References to use [9], [6], [7]. Since there are many papers using a similar approach and are able to improve the performance over the years, adding our ES based optimization should allow for a significant improvement in performance and ability to deploy in real-time.

III. Experimental Results

IV. Discussion

The results demonstrate the effectiveness of using evolutionary strategies (ES) with 1/5 success rule to optimize CNN architectures for steering angle prediction in autonomous driving systems and hence meets our goals. Key findings include: Baseline Performance: The PilotNET baseline achieved reasonable accuracy but required significant computational resources due to its large model size (245.40 million parameters, 936.44 MB). This highlights the need for architecture optimization to improve efficiency without sacrificing performance. Impact of ES Optimization: The ES-optimized PilotNET model achieved the lowest error (MSE: 0.49, MAE: 0.41 degrees) while significantly reducing the model size (50.97 million parameters, 194.45 MB). Models with reduced convolutional layers showed trade-offs: while they consumed fewer resources, their prediction accuracy relatively suffered, with the quarter-size model exhibiting the highest error (MSE: 1.88, MAE: 0.88 degrees). This study shows that using ES to optimize models for better performance or size reduction while does yield practical results that can improve application efficiency. Real-time Performance: Visual results confirm that the optimized models deliver accurate steering predictions even in complex driving scenarios, with low absolute errors between expected and predicted steering angles. In real-world applications a small difference in error (¡1.0 degree) with a major difference in model size (5-25% of baseline size) is a valid tradeoff.

In the future, we plan to: Expand the dataset by recording driving data on various roads across the LTU campus, including different surfaces, path geometries, and lighting conditions. Add Recurrent Neural Network layers to train models based

on the past steering inputs Deploy the optimized CNN models on the LTU ACTor autonomous driving platform. Evaluate optimized models on low-power compute platforms, such as NVIDIA Jetson or Raspberry Pi, to assess scalability for cost-effective systems.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets []. The sentence punctuation follows the bracket []. Refer simply to the reference number, as in []—do not use "Ref. []" or "reference []" except at the beginning of a sentence: "Reference [] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" []. Papers that have been accepted for publication should be cited as "in press" []. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [].

REFERENCES

[1] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing: An International Journal*, 1(1):3–52, 2002.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Truong-Dong Do, Minh-Thien Duong, Quoc-Vu Dang, and My-Ha Le. Real-time self-driving car navigation using deep neural network. In *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, pages 7–12, 2018.

[6] Fayez Faizi and Ahmed Alsulaifanie. Steering angle prediction via neural networks. *Indonesian Journal of Electrical Engineering and Computer Science*, 31:392–399, 07 2023.

[7] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[8] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[9] Junekyo Jhung, Il Bae, Jaeyoung Moon, Taewoo Kim, Jincheol Kim, and Shiho Kim. End-to-end steering controller with cnn-based closed-loop feedback for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 617–622, 2018.

[10] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

[11] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[12] W. Vent. Rechenberg, ingo, evolutionsstrategie — optimierung technischer systeme nach prinzipien der biologischen evolution. 170 s. mit 36 abb. frommann-holzboog-verlag. stuttgart 1973. broschiert. *Feddes Repertorium*, 86(5):337–337, 1975.