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, Autonomous driving, Deep learning**

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 on-board processors. 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 []. 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 [].

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 []. Limited data can lead to overfitting, where the model performs well on the training data but fails to generalize to unseen scenarios, ultimately compromising the safety and reliability of autonomous driving systems [].

A. Data Collection Equipment

The LTU ACTor autonomous driving platform, integrated with the Robot Operating System (ROS), was employed for data collection. A rosbag file, a standard ROS format for recording sensor data streams, captured images from the vehicle's forward-facing camera and the corresponding steering angles. This setup provided synchronized visual data and control signals essential for model training.

B. Data Collection Environment

Data was recorded along a fixed, predefined path: the red brick path surrounding the Ockham's Wedge at LTU. To introduce variability and ensure model robustness: Driving sessions included both clockwise and counterclockwise paths. Variations such as smooth, steady driving versus zigzag maneuvers were performed. Driving along the inner and outer edges of the path simulated diverse spatial alignments.

C. Data Extraction and Classification

Data Extraction and Classification: A custom script was developed to extract data from the rosbag files: Images were

saved at 200ms intervals, and file names included timestamps and steering angle labels. Each image was cropped to exclude irrelevant features (e.g., sky and surrounding buildings) The final dataset comprised of 2957 images (split into 70% Train: 2069 + 20

II. Research Goals

Use the (n+m) Evolutionary Strategy (ES) with the 1/5 success rule to automate hyperparameter tuning. Minimize the size of the CNN model (parameters and memory footprint) while maintaining satisfactory performance for steering angle prediction. Validate the applicability of ES-optimized models in real-world autonomous systems.

III. Methodology

Model Training and Establishing a Baseline Model Models are trained using the Keras API for Python on top of PyTorch backend to leverage GPU compute capacity. The models were trained on our dataset using the Mean Squared Error (MSE) loss function, which quantifies the average squared difference between predicted and actual steering angles. Lower MSE value and lower MAE value both indicate better performance. This serves as a benchmark for evaluating the improvements introduced by the evolution strategy (ES) with 1/5 success rule optimized models.

We began with a single-layer CNN as a simple baseline model. However, it quickly became evident that such a simplistic architecture lacked the capacity to produce results suitable for real-world applications. To address this, we adopted PilotNet—a pre-designed CNN developed by NVIDIA. PilotNet, an early milestone in autonomous steering research, demonstrated the potential of GPU-accelerated deep learning for this domain.

Additionally, early stopping was added to stop the training instances after 4 epochs with no improvement in mean squared error metric. This prevents hyperparameters with no impact towards model improvement from consuming time and processing power.

Dataset and Hyperparameter Management The dataset was randomly divided into 70% training, 20% validation, and 10% testing splits to ensure sufficient data for model evaluation. Initially, hyperparameters included individual layer units, batch sizes, learning rates, activation functions, and optimizer selection. The layer units ranged from 20% of the baseline to 300% as baseline to allow for a wide range of optimization in each layer.

However, after initial experiments, it was observed that reducing the optimization scope to focus solely on layer units significantly improved performance and reduced computation time. Consequently, batch size (32), learning rate (0.001), activation function (ReLU), and optimizer (Adam) were fixed to match the baseline model settings.

Optimization Framework The CNN hyperparameters—such as the number of filters in convolutional layers and the number of neurons in fully connected layers—were encoded as genes in an evolutionary strategy (ES) framework. Specifically, we employed the (N+M)-ES approach, which iteratively optimizes hyperparameters by combining the N best parent models and M offspring models generated through mutations.

A. ES Algorithm

The process proceeds as follows: Initialization: Generate and train N parent models with random hyperparameters within predefined ranges. Child Generation: Create M offspring by mutating parent hyperparameters with:

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

where h represents a hyperparameter and σ is the step size Training and Evaluation: Train all child models and evaluate their validation loss and accuracy. Selection: Retain the top N models (from parents and children) based on validation loss and accuracy to form the next generation's parents. This process iterates until the mutation step size (σ) is dynamically optimized using the 1/5 success rule:

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

where α increases the step size, β decreases it based on success rate—the fraction of offspring outperforming their parents. This ensures efficient exploration of the hyperparameter space while avoiding local minima.Computational Resources: Training was performed on Lawrence Technological University's NVIDIA A100 GPU server, enabling efficient parallel training of N parents and M offsprings model pools.

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.

IV. Experimental Results

V. 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 [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] 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" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. 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 [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114

[9] S. Liu, "Wi-Fi Energy Detection Testbed (12MTC)," 2023, gitHub repository. [Online]. Available: https://github.com/liustone99/Wi-Fi-Energy-Detection-Testbed-12MTC

[10] "Treatment episode data set: discharges (TEDS-D): concatenated, 2006 to 2009." U.S. Department of Health and Human Services, Substance Abuse and Mental Health Services Administration, Office of Applied Studies, August, 2013, DOI:10.3886/ICPSR30122.v2

[11] K. Eves and J. Valasek, "Adaptive control for singularly perturbed systems examples," Code Ocean, Aug. 2023. [Online]. Available: https://codeocean.com/capsule/4989235/tree