ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Robust Evolutionary Multi-Objective Neural Architecture Search for Reinforcement Learning (EMNAS-RL)

Nihal Acharya Adde[1,2], Alexandra Gianzina[1], Hanno Gottschalk[2], Andreas Ebert[1]

1- Volkswagen Group Innovation, Volkswagen AG, Wolfsburg, Germany
2- Institute of Mathematics, TU Berlin, Berlin, Germany

Abstract. This paper introduces Evolutionary Multi-Objective Neural Architecture Search (EMNAS) for the first time to optimize neural network architectures in large-scale Reinforcement Learning (RL) for Autonomous Driving (AD). EMNAS uses genetic algorithms to automate network design, tailored to enhance rewards and reduce model size without compromising performance. Additionally, parallelization techniques are employed to accelerate the search, and teacher-student methodologies are implemented to ensure scalable optimization. Experimental results demonstrate that tailored EMNAS outperforms manually designed models, achieving higher rewards with fewer parameters.

1 Introduction

Neural Architecture Search (NAS) automates the traditionally manual and time-intensive process of designing neural network architectures. NAS algorithms use different methods like Reinforcement Learning (RL), Evolutionary Algorithms (EA), and Gradient-based Optimization (GO), to explore potential architectures to identify the most effective designs for specific tasks. EAs are particularly effective, consistently achieving state-of-the-art performance [1], with superior anytime performance [1] that enables the discovery of smaller models compared to RL and addresses multi-objective problems in NAS [2, 3]. This approach seeks high-performing models, reflected in RL rewards, while minimizing constraints such as parameter size for memory efficiency and Floating Point Operations Per Second (FLOPS) to meet power consumption or latency requirements.

This paper explores the use of EAs to optimize network architectures for an RL task in Autonomous Driving (AD), aiming to outperform human-designed baselines. A teacher-student framework is employed to transfer learned knowledge across generations, where student networks are trained via behavior cloning [4] using the top-performing teacher network from the previous generation.

Autonomous Driving Task: The AD task is trained in the Unity 3D simulator [5] using the Proximal Policy Optimization (PPO) algorithm [6]. In this setup, multiple agents navigate a three-lane oval track, aiming for smooth, collision-free driving with minimal abrupt movements. Agents use semantic segmentation from a forward-facing camera to predict trajectory points, which are processed by an external vehicle controller for maneuvering. Reward structures incentivize

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

precise trajectory placement and smooth, collision-free navigation. This study focuses on optimizing convolutional network architectures for this RL task.

2 Background

Evolutionary Algorithms (EAs) are effective in NAS due to their ability to explore complex search spaces. Early methods like NeuroEvolution of Augmenting Topologies (NEAT) [7] demonstrated the potential of genetic algorithms in evolving neural architectures. A comparison by [1] highlighted EA's superior performance over RL and random search, leading to models like AmoebaNet [1]. Recently, Multi-Objective Evolutionary Algorithms (MOEAs) have been applied to NAS, optimizing multiple objectives in parallel. Notably, NSGA-Net [3] leverages MOEA to produce a Pareto front of architectures balancing accuracy with computational efficiency for image classification tasks. Early Exit Evolutionary Neural Architecture Search (EEEANet) [2] combines MOEA with early-exit population initialization, yielding compact architectures for resource-constrained devices. EEEANet uses tournament selection and non-dominated sorting with the NSGA-II algorithm [8]. While NAS has advanced broadly, its application to RL remains limited. Our research addresses this gap by applying EA-based NAS to optimize network architectures for RL in AD, aiming to improve efficiency and scalability.

3 EMNAS-RL Methodology

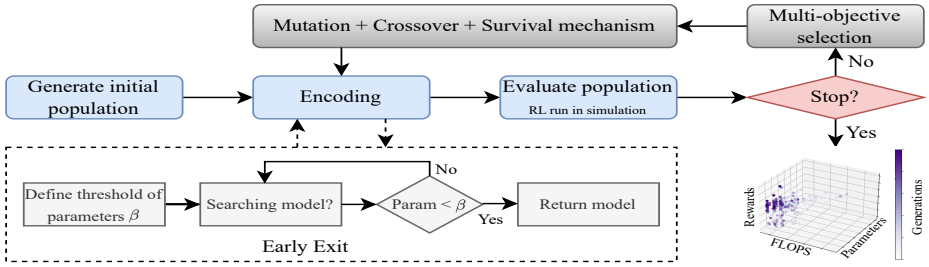


Fig. 1: Multi-objective Evolutionary Algorithm with EEPI.

In this work, we build upon EEEANet [2], using MOEA-based NAS as the foundation. EA, based on Genetic Algorithms (GA), iteratively evaluates a randomly initialized population of neural architectures using a fitness function to assess performance [2]. The population evolves over generations through crossover and mutation, where a population represents candidate architectures, and a generation is a single evolutionary iteration. Figure 1 illustrates the algorithm framework.

In MOEA, the fitness function combines multiple objectives: reward (model effectiveness), computational cost (FLOPS), and model complexity (parame-

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

ters), minimizing negative reward while balancing efficiency, as shown in Eq. (1).

$$f(x) = \min \{\textbf{-Reward}(x), \textbf{FLOPS}(x), \textbf{Params}(x)\}, x \in X \quad (1)$$

where x is an individual architecture in the set X of a given generation. These objectives are normalized and equally weighted to optimize for limited computational resources. The NSGA-II algorithm ranks the population based on these objectives, evolving a diverse set of non-dominated solutions guided by Pareto optimality principles [9]. We use a Pareto front-based tournament selection, dividing the population into subsets to compare fitness values, with winners reproducing through crossover and mutation to enhance diversity [2]. Crossover, mutation, and survival mechanisms drive the creation of new generations, retaining promising architectures while exploring the search space [2]. Mutation alters genetic information to explore new search space regions, while crossover combines genetic material from two parent solutions to create offspring. Mutation and crossover probabilities, set as hyperparameters, determine the likelihood of these operations. The search space includes various convolutional operations (3×3, 5×5 depth-wise separable, dilated and inverted convolutions, and 7×7 convolution), pooling operations (max and average), and skip connections. These operators, encoded within normal and reduction cells (chromosomes), alternate to form the network architecture. A normal cell extracts features with unchanged dimensions, while a reduction cell downsamples features, halving spatial dimensions to manage complexity. The encoding scheme is defined as: chromosome$(x) = LA_1LA_2, LB_1LB_2, LC_1LC_2, LD_1LD_2$, where L represents operators, and A, B, C, D are indices determining connections. GA initialization uses Early Exit Population Initialization (EEPI) [2], ensuring initial population parameters remain below a threshold β (in millions). NAS executes for user-defined generations and population sizes, progressively discovering high-performing architectures.

Retaining the architecture search properties of EEEANet [2], including EA with tournament selection, multiple objectives, and an early exit strategy, we incorporated driving data from the simulation. The focus is on optimizing the convolutional network component of the PPO algorithm while keeping the fully connected layers constant. Based on our previous experiments, we observed that improvements in RL reward performance become evident after approximately 20 epochs. Hence, to accelerate training, we use lower fidelity estimates and learning curve extrapolation, ranking architectures after 20 epochs instead of the typical 300 full RL training epochs. We also emphasize relative ranking to mitigate estimation bias. Additional lower fidelity adjustments include reducing input resolution to $84 \times 84 \times 3$, stacking 4 cells instead of 20, reducing blocks per cell from 5 to 4, and halving initial channels to 16, where blocks define combinations of convolutional operations, and initial channels set the first layer's output.

In this paper we enhance EMNAS-RL methodology by two novel modules:

- **Optimized Transfer Learning (OTL):** From the second generation, a teacher-student framework transfers the policy of the best-performing

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

network to student networks through behavior cloning[4], leveraging prior knowledge to accelerate training.

- **Parallel Training:** EMNAS automates the parallel training of population individuals using four NVIDIA V100 32 GB GPUs, running four trainings simultaneously to optimize resource utilization.

4 Experimental Results

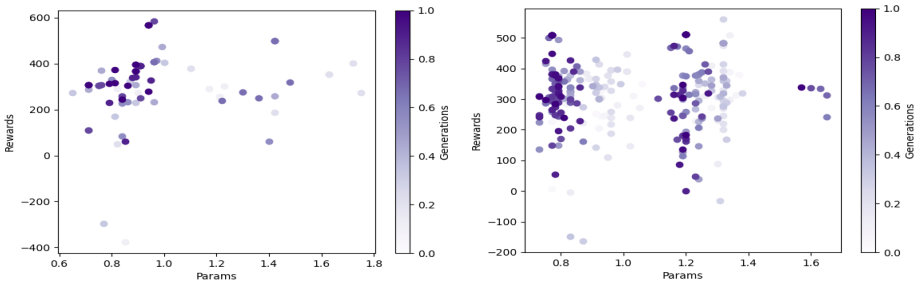
In this section, we describe the experimental setup and the impact of various hyperparameters on the performance of the NAS process. To ensure consistency across experiments, hyperparameters are kept constant: mutation probability is 0.1 and crossover probability is randomized between 0.5-0.9. The survival mechanism retains 1 to 4 individuals per generation, depending on population size. The number of generations and population size impact the search performance and duration, with larger values improving exploration but also increasing time. In this study, small to mid-range values are used to balance search quality and time efficiency. Additionally, the threshold parameter (β) for EEPI is varied to assess its impact. Table 1 shows the effect of β on performance, with each run repeated twice for better generalization. For fixed generation (Gen) and population (Pop) values, increasing β led to better rewards, improved parameters, and lower FLOPs. The last column shows the GPU days required to complete the NAS experiment on an NVIDIA Tesla V100-32GB, with minimal difference in resource usage despite higher values of β. Based on these results, a threshold β of 5 is chosen for subsequent experiments.

Table 1: Experiment details to compare the impact of threshold parameter (β).

Exp	**Gen**	**Pop**	$\boldsymbol{\beta}$	**Reward**	**Param(M)**	**FLOPS(G)**	**GPU**
1.1	6	6	3	452	1.13	1.13	1.0
1.2	6	6	5	**482**	1.02	1.01	1.4
2.1	10	4	3	412	0.99	0.99	1.2
2.2	10	4	5	**622**	0.98	0.93	1.5

Figure 2 depicts the evolution of rewards and parameters for EMNAS, color-coded by generation. To improve clarity, a 2D plot displaying only rewards and parameters is presented, with FLOPS omitted due to the similar trends observed between FLOPS and parameters. This figure indicates successful evolution, with rewards maximized and parameters and FLOPS minimized over time. Improved models consistently emerge in later generations across all experiments. These findings demonstrate the effectiveness of MOEA for NAS in identifying optimal architectures for AD. Despite variations in experiment size, no consistent advantage is observed between larger generations or population sizes, highlighting the stochastic nature of the search process. While larger experiments tend to yield better models on average across all objectives, exceptional models can

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.



(a) Generation 10, Population 4 (b) Generation 20, Population 15

Fig. 2: Evolution of rewards and number of model parameters during the architecture search over different generations. Here, the generation bar is normalized.

still emerge from smaller searches. The increased exploration enabled by larger experiments is evident in the distinct clusters observed in Figure 2(b).

Although EMNAS yielded satisfactory results, each generation learns from scratch. To address this, we experiment with applying the OTL teacher-student method to transfer the policy of the winning network to the next generation and evaluate whether the model performs better. Table 2 compares EMNAS and EMNAS with Optimized Transfer Learning (OTL) across six experimental setups, using evaluation metrics such as the 25th percentile, median, 75th percentile, and highest reward. The "P" under the method column indicates that parallelization was employed during the training process. The results clearly show that parallelization reduces runtime (in days (d)) by a factor of 2-3, with greater speed improvements observed in larger training setups. As shown in

Table 2: Reward Comparison for 6 Experiments

Experiment	25th Percentile	Median Reward	75th Percentile	Max Reward	Method	Run Time(d)
4 populations over 10 generations	320	353	**535**	**622**	EMNAS	1.5
2 survivors per generation	**335**	**362**	384	585	OTL(P)	0.7
6 populations over 6 generations	154	280	376	482	EMNAS	1.4
3 survivors per generation	**318**	**340**	**385**	**538**	OTL(P)	0.8
10 populations over 30 generations	269	328	390	560	EMNAS	15.9
3 survivors per generation	**351**	**388**	**426**	**579**	OTL(P)	6.8
15 populations over 10 generations	275	**334**	**403**	593	EMNAS	7.7
4 survivors per generation	**296**	321	342	**753**	OTL(P)	3
15 populations over 15 generations	219	315	**395**	**683**	EMNAS	13.4
4 survivors per generation	**325**	**355**	391	584	OTL(P)	4.4
20 populations over 15 generations	189	291	351	509	EMNAS	18.6
3 survivors per generation	**280**	**323**	**367**	**604**	OTL(P)	6.3

the Table 2, OTL generally outperforms the basic EMNAS method 60% of the time when comparing maximum rewards. However, OTL consistently surpasses EMNAS in median reward, suggesting that while EMNAS can achieve high rewards, it tends to be less stable. Using OTL for knowledge transfer stabilizes network performance in subsequent generations, ensuring consistent and accurate results. Since rewards are the primary focus, FLOPs and parameter counts

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

are not included in the table, though they exhibit a similar performance trend.

Lastly, as seen in Table 2, the best-performing model was found in the OTL method with a population of 15 over 10 generations, achieving a maximum reward of 753. We fully retrained the winning model over 300 PPO iterations on simulation data to assess how the architecture performs over full training. The model achieved a peak total cumulative reward of 1190, a 4% increase over the manually set architecture (reward: 1140). The stochastic nature of evolution limits correlations between generation and population size with performance, though larger sizes improve search space exploration and optimization potential.

5 Conclusion

This study marks the first application of EMNAS, a multi-objective NAS framework, to large-scale RL in AD. Building on EMNAS's foundation of automating network design with evolutionary algorithms, we tailored the framework for large-scale RL and introduced parallelization to accelerate the search process by a significant factor. These advancements enabled EMNAS to consistently outperform manually designed architectures while reducing parameter counts. Additionally, to avoid learning each architecture from scratch, the OTL teacher-student method was employed to transfer policies from one generation to the next, improving stability and efficiency with significant gains in both median and maximum rewards. Further improvements are anticipated through concurrent hyperparameter and NAS optimization. Future work will focus on refining these aspects to further enhance performance.

References

[1] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[2] Chakkrit Termritthikun, Yeshi Jamtsho, Jirarat Ieamsaard, Paisarn Muneesawang, and Ivan Lee. Eeea-net: An early exit evolutionary neural architecture search. *Engineering Applications of Artificial Intelligence*, 104:104397, 2021.

[3] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: NAS using multi-objective genetic algorithm. In *Proceedings of the genetic and evolutionary computation conference*, 2019.

[4] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. pages 4950–4957, 07 2018.

[5] Unity3D. `https://unity.com/`. Accessed: 2023-03-04.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[7] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[9] David A Van Veldhuizen, Gary B Lamont, et al. Evolutionary computation and convergence to a pareto front. In *genetic programming conference*, pages 221–228, 1998.