Journal of Soft Computing Paradigm (ISSN: 2582-2640)
https://irojournals.com/jscp/

Adaptive Multi-Objective Deep Reinforcement Learning for Policy-Driven Neural Architecture Search of Resource-Efficient CNNs

Ram Teja T.[1], Madhav G.[2], Sowmya Latha P.[3], Santhosh Kumar CH.V.V.[4], Vamsi G.[5], Yogendra B.[6]

[1,2,3,5,6]Student, Department of Electronics and Communication Engineering, QISCET, Ongole, India.

[4]Trainer, Bytexl, India.

E-mail: [1]ramtejatalluri@gmail.com, [2]madhavgarapati206@gmail.com, [3]24491a04g5@qiscet.edu.in,

[4]chevurisanthosh@gmail.com, [5]golladharivamsi@gmail.com, [6]yogendrabandaru876@gmail.com

Abstract

Neural Architecture Search (NAS) automates the process of designing neural networks but the focus is mostly on prediction accuracy without considering the model complexity and computational cost. This research study proposes a new framework called Adaptive Multi-Objective Policy-Driven Neural Architecture Search (AMO-PDNAS) based on Deep Reinforcement Learning (DRL). The CNN architecture design is formulated using Markov Decision Process. A parameterized policy network generates CNN architecture parameters, which include kernel size, number of filters, activation functions, and dropout rate. Reward weighting is adapted through each search cycle in an attempt to shift from an explorative approach driven by accuracy towards an approach driven by both accuracy and efficiency considerations. Candidate architectures are learned and evaluated in cycles and feedback is provided to the policy based on the REINFORCE algorithm. Results on the MNIST problem show robust behavior of searching for high-accuracy architectures along with efficient trade-offs of accuracy and complexity.

Keywords: Neural Architecture Search, Deep Reinforcement Learning, Multi-Objective

DOI: https://doi.org/10.36548/jscp.2026.4.001
Extended version of article received from **6th International Conference on Artificial Intelligence, 6G Communications and Network Technologies 2026**
Received: 03.08.2026, received in revised form: 07.09.2026, accepted: 01.10.2026, published: 12.10.2026
© 2026 Inventive Research Organization. This is an open access article under the Creative Commons Attribution-NonCommercial International (CC BY-NC 4.0) License

Optimization, Policy-Driven Architecture Generation, Adaptive Reward Optimization, Accuracy-Complexity Trade-off, Resource-Aware CNN Design.

1. Introduction

The performance of Convolutional Neural Networks (CNN) highly depends on the configurations, such as kernel size, channel capacity, activation function, regularization, and number of layers. These are the factors that not only affect the representational ability of CNN but also the number of parameters and training cost. Selection of these configurations manually requires repetitive experimentation and does not offer any systematic way to find out architecture under various conflicting conditions. Neural Architecture Search formulates the task of architecture search as an automated optimization process [1], [4].

Integration of Reinforcement Learning (RL) with NAS enables the possibility to represent architecture construction in terms of a sequence of discrete decisions. RL controller performs architectural actions, obtains feedback on its performance from the built CNN, and adapts its policy in the direction of architectures with higher rewards [3]. Nevertheless, if such rewards are defined only through validation accuracy, architecture search may lead to unnecessary complexity of the obtained models in terms of the number of parameters and computational resources.

The architecture search problem is further made difficult by the simultaneous optimization of accuracy, model complexity, and computational time. Recent works on NAS have focused on the development of architecture evaluation techniques, resource-aware optimization, scalable RL-based search, and multi-objective search [5]- [10]. All of these techniques eliminate the problem of relying on accuracy as the only optimization objective; however, the problem of assigning a certain weight to competing objectives delays the search process since the importance of the competing objectives may vary during the exploratory and convergence phases of architecture optimization.

To address this limitation, the proposed research introduces AMO-PDNAS, a RL-based approach where CNN architecture creation can be considered as a Markov Decision Process. The policy network selects the different configurations of architectures within a restricted search space, and each architecture is then trained and scored according to the validation accuracy,

number of parameters, and training time. The obtained metrics are then combined by using an adaptive multi-objective reward function in which the objectives' weights are gradually tuned during the search process. Hence, the learned policy is initially oriented towards efficient prediction accuracy and then towards more computationally efficient architectures.

2. Literature Review

The efficiency of structural advancements in deep CNN architectures is evident with the introduction of deep residual networks [1], while reinforcement learning NAS proved that the sequence decision making can be used to automate CNN architecture generation [3]. The hardware-aware NAS included computational considerations when selecting CNN architectures [2].

Recent researches on NAS have concentrated on enhancing efficiency and scaling. Multi-agent automatic machine learning added several agents to conduct model optimization automatically [5], whereas proxy-based NAS made the evaluation process more efficient [6]. Hardware and computation costs were considered by resource-aware NAS [7].

RL-based NAS techniques have also continuously been tackling issues with scalability and improved search strategies [8]. On the other hand, multi-objective methods have been incorporating several architectural attributes instead of focusing solely on prediction accuracy. Search space division can be seen as a way to incorporate multiple objectives when searching for an optimal architecture [9], whereas training-based estimation further optimizes multi-objective NAS techniques [10].

The existing researches creates a good starting point for automated, efficient, and multi-objective optimization of architectures [2], [5]-[10]. On the other hand, adjusting the priority level of various objectives throughout the search process is a direction worth considering. The proposed AMO-PDNAS framework takes care of this problem by incorporating the adaptive reward system within the reinforcement-learning approach for generating architectures.

3. Proposed Methodology

The proposed AMO-PDNAS framework automatically discovers the suitable CNN architectures by using a reinforcement learning-based controller. As shown in Figure 1,

the entire process starts with the MNIST dataset that serves as the source of input data for evaluating the proposed architecture. The Policy Network (RL Controller) identifies architectural configurations that are fed to the Architecture Generator to create the candidate CNN. The candidate CNN is trained and further evaluated with respect to its performance through the Performance Evaluation module, which evaluates the CNN according to its predictive and computational properties. The results from the evaluation are sent to the adaptive reward computation module that generates the multi-objective reward considering classification accuracy, model complexity, and training time. The resulting reward is used to update the Policy through the REINFORCE stage.

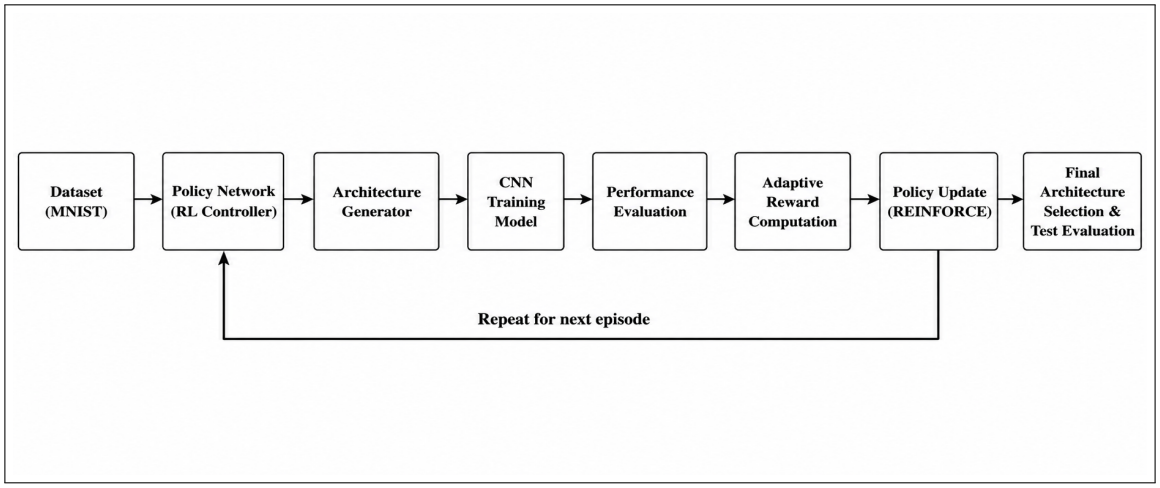

Figure 1. Block diagram of the proposed AMO-PDNAS framework

3.1 Dataset Description and Preprocessing

The MNIST handwritten digits dataset is employed to perform the evaluation of the proposed AMO-PDNAS framework. This data set includes grayscale images of ten handwritten digits ranging from zero to nine and is of size 28 × 28 pixels per image. The entire dataset has 70,000 images, where 60,000 images are for training and 10,000 images are for testing [11]. The former part of the dataset is used during the architecture search phase, whereas the latter part is used for the evaluation of the architecture.

For the search approach being developed, the set of 60,000 training images is split up into 54,000 training examples and 6,000 validation examples. The former is used to train each generated CNN at the episode, while the latter supplies performance information necessary for computing adaptive rewards. The set of 10,000 test images is not used in optimizing policies at all.

Prior to the training of CNNs, the gray-scale intensities of pixels get normalized to the new range of [0, 1] from the initial range of [0, 255]. This pre-processing offers a uniform scale for the optimization of CNNs as well as the evaluation of potential architectures.

3.2 AMO-PDNAS Framework and Search Space

As mentioned above, the AMO-PDNAS scheme relies on a reinforcement learning controller that creates CNN models based on a sequence of architectural decisions. The controller, architecture generator, CNN model training, performance assessment, reward adaptation, and policy updating via the REINFORCE algorithm make up a closed-loop scheme for architecture search, as illustrated in Figure 1. In each episode of the search, an architectural model is chosen by the controller, translated into a candidate CNN model, and then assessed.

The architecture search is represented as a Markov Decision Process (MDP). The state is the current architecture configuration that the controller has access to, whereas the action is the selection of one of the architectures. The policy network, parameterized by θ, determines the probability of selecting an action according to $\pi_\theta(a_t \mid s_t)$. The selected actions are combined together to generate the candidate architecture. This way of formulation makes the controller capable of learning its architecture selection behavior based on performance feedbacks gained through consecutive searches.

The search space is deliberately restricted in order to have architectural diversity while keeping the computation unnecessary high. Four main parameters defining the CNN architecture are determined by the controller: convolutional kernel size, number of filters, activation function, and dropout rate. The chosen architecture is then provided to the architecture generator that will transform the decision made by the controller into a valid CNN architecture.

3.3 CNN Training and Performance Evaluation

The candidate CNN architecture produced by CNN is then trained using the training subset consisting of 54,000 images from MNIST dataset [11]. Training each candidate architecture is done under the same set of experimental conditions such that the performance of the candidate architecture can be properly compared to those of the other candidates produced by CNN during the search process.

After training, the candidate model is evaluated using the 6,000-image validation subset. Prediction accuracy measures the predictive power of the model while parameter count determines the complexity of the model and the training time is the training cost.

These evaluation results are not utilized independently; rather, they are sent to the Adaptive Reward Computation step, where the three factors are considered in order to evaluate the selected architecture.

3.4 Adaptive Multi-Objective Reward and Policy Optimization

Reward for each candidate CNN is transformed into a single value that incorporates both the prediction accuracy and computation cost. The proposed reward scheme involves use of validation accuracy, trainable parameters and training time as the three objectives for optimization. While accuracy positively impacts the reward score, parameter count and training time affect negatively. The reward at episode t is defined as:

$$R_t = \omega_{1,t} A_t - \omega_{2,t} P_t - \omega_{3,t} T_t \tag{1}$$

where A_t denotes the validation accuracy, P_t represents the number of trainable parameters, T_t denotes training time, and $\omega_{1,t}$,$\omega_{2,t}$, and $\omega_{3,t}$ represent the corresponding objective weights. The weights are adjusted according to the search progression. During the initial episodes, there is an increased importance on precision to promote the exploration of high performing structures. As the process proceeds, the terms that relate to efficiency become increasingly significant, allowing for the discovery of small and efficient architectures.

The reward value is then passed to the Policy Update (REINFORCE). The controller's parameters are updated according to the following policy gradient algorithm:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log \pi_\theta (a_t \mid s_t) R_t \tag{2}$$

where θ_t denotes the current policy parameters, α is the policy learning rate, a_t is the selected architectural action, s_t is the current search state, and R_t is the obtained reward. Higher rewards from the architectures imply a higher probability of selecting such architectures during future episodes.

3.5 Iterative Architecture Search and Final Model Selection

The architecture search process is conducted iteratively for a certain number of episodes. At the start of every episode, the new policy network generates a new configuration of architectural parameters, which is turned into the proposed CNN architecture by means of the architecture generator. After that, the model is trained and tested according to the same protocol, and its validation accuracy, the number of parameters, and training duration are fed into the adaptive reward system. This reward becomes an input for the policy update based on REINFORCE algorithm.

After completion of the search, the architectural candidates will be compared in terms of the reward and validation performance that have been achieved. The architecture exhibit the best trade-off between classification performance, model complexity, and training cost will be chosen as the RL-NAS architecture. The selected architecture will be validated using the MNIST independent test data that was not involved in policy training.

4. Results and Discussion

4.1 Episode-wise Validation Accuracy

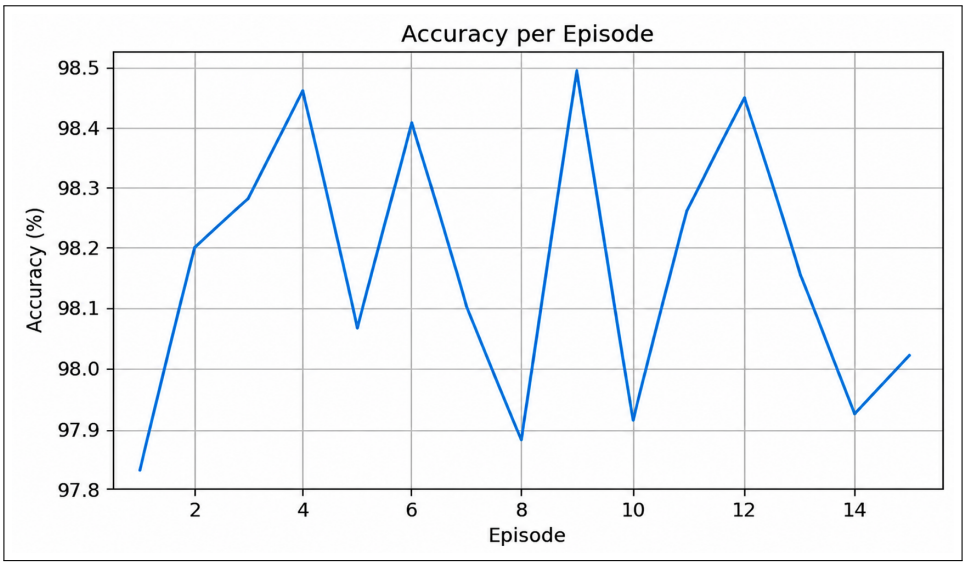

Figure 2. Validation accuracy across architecture-search episodes

Evaluation of the proposed AMO-PDNAS architecture has been done using 15 architecture search episodes. Figure 2 depicts the validation accuracy obtained from each architecture candidate during the process. Throughout the search process, a consistently high predictive accuracy has been achieved, where the maximum validation accuracy recorded is

98.49%, which was achieved at Episode 9.

These variations are anticipated in a policy-driven search process since every trial might pick a different set of architectural parameters. Hence, the goal of the search process is not only to ensure monotonic increases in accuracy in every trial but also to perform an exploration of the architecture space in order to find a good combination of predictive capabilities and computational properties.

4.2 Adaptive Reward Progression

Figure 3 illustrates the rewards collected during the 15 iterations of search. It is evident that the rewards fall from about 94 in the first iteration to roughly 40 in the last iteration. Such an observation should not be viewed in isolation of the formulation of multiple objectives for rewards.

The reward includes the validation accuracy along with the complexity of the model and training costs. Moreover, the weights of these objectives are varying during the search process. Therefore, the decrease in the scalar reward may happen even if the validation accuracy is very close to 98%. Thus, the evolution of the reward reflects the changing influence of the efficiency objectives.

Figure 3. Adaptive reward progression across architecture-search episodes

The reward curve clearly shows that AMO-PDNAS utilizes various criteria to test architecture candidates, instead of depending solely on classification accuracy.

4.3 Accuracy-Parameter Trade-off

To analyze the architectural search behavior further, both the validation accuracy and the number of trainable parameters of the proposed candidate architectures are considered together.The episode-wise validation accuracy and number of parameters are shown in Table 1. The searched architectures are able to keep their validation accuracies ranging from 97.82% to 98.49% with varying numbers of parameters from 21,120 to 52,480. This shows that the controller is able to explore architectures with diverse levels of complexity without any considerable drop in terms of predictive accuracy. It is shown in Table 1 that the maximum validation accuracy of 98.49% is achieved by Episode 9 having 26,880 trainable parameters, followed by Episode 12 and 15 having 98.46% and 98.02% accuracy and 23,920 and 21,120 parameters, respectively.

Table 1. Episode-wise validation accuracy and trainable parameters

Episode	Accuracy (%)	Trainable Parameters
1	97.82	52,480
2	98.20	48,960
3	98.28	44,320
4	98.47	39,680
5	98.07	36,240
6	98.41	33,120
7	98.08	30,560
8	97.88	28,960
9	98.49	26,880
10	97.91	25,760
11	98.26	24,640
12	98.46	23,920

13	98.12	22,880
14	97.92	21,760
15	98.02	21,120

4.4 Overall Search Behavior

The aggregated findings reveal three distinctive features of the architecture search procedure in question. First, the accuracy curve illustrates the fact that the controller always identifies CNN architectures with around 98% validation accuracy. Second, the reward curve indicates the application of several objectives when evaluating architectures, instead of pursuing only the accuracy goal. Finally, the accuracy parameters chart reveals that CNN architectures with significantly different complexities can be equally accurate.

This result proposed that the search process AMO-PDNAS has the ability to explore CNN architecture taking into consideration the classification capability as well as the architecture compactness. The most successful architecture has achieved around 98.49% accuracy on the validation set, while other search processes show the possibility of achieving 98% accuracy with less parameters.

5. Conclusion and Future Work

The Adaptive Multi-Objective Deep Reinforcement Learning-based Policy-Driven Neural Architecture Search (AMO-PDNAS) approach shows that automatic discovery of CNN architectures via policy learning driven by rewards is possible. Considering such factors as validation accuracy, number of parameters, and training time during the process helps achieve not only accuracy-based architecture selection but also beneficial accuracy-complexity trade-offs. The findings show the stability of predictive accuracy while searching for architectures with different model complexity. Future works could involve expanding the search space by incorporating deeper architectural operations, normalizations, and hardware constraints. Testing on larger and more varied image datasets would further allow evaluation of scalability and generalization of the proposed method. Benchmarking using multiple devices, efficient cost evaluation for candidate architectures, and policy optimization could also be considered for further improving the search process and deployment on edge devices.

References

[1] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. ”Deep Residual Learning for Image Recognition.” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 770-778.

[2] Cai, Han, Ligeng Zhu, and Song Han. ”Proxylessnas: Direct Neural Architecture Search on Target Task and Hardware.” arXiv preprint, 2018, arXiv:1812.00332.

[3] Baker, Bowen, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. ”Designing Neural Network Architectures Using Reinforcement Learning.” arXiv preprint, 2016,arXiv: 1611.02167.

[4] Poyser, Matt, and Toby P. Breckon. ”Neural Architecture Search: A Contemporary Literature Review for Computer Vision Applications.” Pattern Recognition, 2024, vol. 147: 110052.

[5] Wang, Zhaozhi, Kefan Su, Jian Zhang, Huizhu Jia, Qixiang Ye, Xiaodong Xie, and Zongqing Lu. ”Multi-agent Automated Machine Learning.” In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2023, 11960-11969.

[6] Li, Yuhong, Jiajie Li, Cong Hao, Pan Li, Jinjun Xiong, and Deming Chen. ”Extensible and Efficient Proxy for Neural Architecture Search.” In IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, 2023, 6176-6187.

[7] Bellodi, Elena, Davide Bertozzi, Alice Bizzarri, Michele Favalli, Michele Fraccaroli, and Riccardo Zese. ”Efficient Resource-Aware Neural Architecture Search with a Neuro-Symbolic Approach.” In IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), IEEE, 2023, 171-178.

[8] Cassimon, Amber, Siegfried Mercelis, and Kevin Mets. ”Scalable reinforcement Learning-Based Neural Architecture Search.” Neural Computing and Applications, 2025, vol. 37, no. 1: 231-261.

[9] Zhao, Yiyang, Linnan Wang, and Tian Guo. ”Multi-objective Neural Architecture Search by Learning Search Space Partitions.” Journal of Machine Learning Research, 2024, vol. 25, no. 177: 1-41.

[10] Zhang, Yu-Ming, Jun-Wei Hsieh, Xin Li, Ming-Ching Chang, Chun-Chieh Lee, and Kuo-Chin Fan. ”MOTE-NAS: Multi-objective Training-Based Estimate for Efficient Neural Architecture Search.” Advances in Neural Information Processing Systems, 2024, vol. 37: 100845-100869.

[11] LeCun, Y., Cortes, C. and Burges, C.J.C. (1998) The MNIST Database of Handwritten Digits. New York, USA. http://yann.lecun.com/exdb/mnist/