2026 Volume: 02 Issue: 05 www.ijrpa.com ISSN 2456-9995 Research Article

International Journal Research Publication Analysis

Page: 01-08

NEURAL ARCHITECTURE SEARCH USING EVOLUTIONARY ALGORITHMS FOR CIFAR-10 IMAGE CLASSIFICATION

Dr Ramya B. N.*[1], Subramanya R.[2], Shivatej S. Gowda[3]

[1]Associate Professor, Department of Computer Science and Engineering, Jyothy Institute of Technology, Bengaluru, India.

[2],[3]Department of Computer Science and Engineering, Jyothy Institute of Technology, Bengaluru, India.

Article Received: 21 March 2026
Article Revised: 11 April 2026
Published on: 01 May2026

***Corresponding Author: Dr Ramya B. N.**
Associate Professor, Department of Computer Science and Engineering, Jyothy Institute of Technology, Bengaluru, India.

DOI: https://doi-doi.org/101555/ijrpa.9421

ABSTRACT

The performance of the neural network is heavily reliant on the architecture of the neural network itself, which is usually constructed manually by means of experimenting with different designs. This research proposes a system capable of automatically creating architectures of neural networks using NAS technology based on the application of evolutionary algorithms. It is shown how a number of CNN architectures can be generated in the form of genomes, evaluated based on their performance on the CIFAR-10 dataset, and improved using the methods of selection and mutation of the best individuals. This process is repeated until certain criteria are fulfilled. A population-based algorithm is implemented whereby elite individuals serve as a basis for generating new generations.

KEYWORDS: Neural Architecture Search (NAS), Evolutionary Algorithms, Convolutional Neural Networks (CNN), CIFAR-10, Automated Machine Learning (AutoML), Deep Learning, Hyperparameter Optimization, Model Architecture Optimization, Genetic Algorithms, Image Classification

I. INTRODUCTION

Deep learning has achieved remarkable success in computer vision tasks, particularly with Convolutional Neural Networks (CNNs). However, designing optimal architectures remains a complex and time-consuming task requiring domain expertise.

Neural Architecture Search (NAS) aims to automate this process by exploring a predefined search space of architectures and identifying high-performing configurations. Traditional NAS approaches, although effective, are computationally expensive and require extensive resources.

This paper proposes a simplified and efficient NAS framework using evolutionary algorithms. The approach focuses on:

- Reducing computational overhead through partial training
- Using a compact search space
- Employing mutation-based evolution for architecture optimization

The CIFAR-10 dataset is used as a benchmark to evaluate the effectiveness of the proposed method.

In this paper, we present a systematic approach to automating neural network design using an evolutionary Neural Architecture Search (NAS) framework. Convolutional neural network architectures are dynamically generated, evaluated, and optimized on the CIFAR-10 dataset through a population-based search strategy. The proposed method incorporates genome-based encoding, fitness evaluation using validation accuracy, and mutation-driven exploration to iteratively improve model performance. The objective of this work is to reduce the dependency on manual architecture engineering while demonstrating that efficient and competitive models can be discovered through lightweight search mechanisms. This study contributes toward making automated machine learning more accessible, interpretable, and computationally feasible for real-world applications.

II. METHODOLOGY

III. Overview of the Proposed Approach

The proposed NAS system follows an evolutionary strategy consisting of:

1. Random architecture generation
2. Fitness Evaluation
3. Selection of elite architectures
4. Mutation-based offspring generation
5. Iterative evolution across generations

Dataset Preparation

The CIFAR-10 dataset is used, containing 60,000 32×32 color images across 10 classes.

- **Training set:** 50,000 images
- **Test set:** 10,000 images
- **Validation split:** 20% of training data

All images are normalized to the range [0,1] for stable training.

Baseline Model

A standard CNN architecture is implemented as a baseline consisting of:

Two convolutional layers

Extract hierarchical features from input images by learning spatial patterns through successive convolution operations.

Max-pooling layers

Reduce spatial dimensions by selecting the maximum value in each region, helping retain important features while lowering computation.

Fully connected layers

Combine extracted features to perform final classification by mapping them to output class probabilities.

This baseline serves as a reference to compare NAS-generated architectures.

Architecture Encoding (Genome Representation)

Each neural architecture is represented as a genome:

(kernel_size, filters, pooling_type)

Example:

[(3, 32, "max"), (5, 64, "none"), (3, 64, "max")]

This encoding defines:

- Convolution kernel size (3×3 or 5×5)
- Number of filters (16, 32, 64)
- Pooling operation (max or none)

Search Space Design

The search space includes:

- 2 to 5 convolutional blocks
- Variable kernel sizes and filter counts
- Optional pooling layers

This constrained search space ensures computational efficiency while maintaining diversity.

Fitness Evaluation

Each architecture is evaluated by:

- Training for 5 epochs (partial training)
- Measuring validation accuracy

This reduces computational cost while providing a reasonable estimate of model performance.

Selection Strategy

Top-performing architectures are selected using the following selection mechanism:

- Population is sorted by validation accuracy
- Top 40% are selected as elites

These elites act as parents for the next generation.

Mutation Operator

Mutation introduces variation into architectures:

The Types of mutations are as follows:

- Kernel size modification
- Filter size modification
- Pooling type change
- Addition of a new layer
- Removal of an existing layer

This ensures exploration of new architectural configurations.

Generation Update

A new population is created by:

- Retaining elite architectures
- Generating mutated offspring from elites

This process is repeated across multiple generations to improve performance.

III. SYSTEM ARCHITECTURE AND DATA FLOW

The section discusses the overall system architecture and the process of data flow inside the proposed NAS system. The processing of input data, generation and evaluation of candidate architectures, as well as the evolution strategy for model refinement will be discussed here in this section.

Data Preprocessing Module

- Normalizes input images and splits the dataset into **training and validation sets**.
- Ensures data consistency and prepares it for model training.
- Improves training stability and evaluation reliability.

Architecture Generator

- Generates random CNN architectures using predefined parameters.
- Introduces diversity in the search space for effective exploration.
- Initializes the population for the evolutionary process.

Model Builder

- Converts genome representations into TensorFlow models.
- Dynamically constructs CNN architectures based on encoded parameters.
- Compiles models for training and evaluation.

Evaluator

Regularization techniques are incorporated during the training phase as follows:

- Trains each architecture for a few epochs.
- Computes validation accuracy as a fitness score.
- Enables efficient comparison of different models.

Evolution Engine

After training, the model is evaluated using test data:

- Selects top-performing architectures as elites**.**
- Applies mutation to generate new candidate models.
- Iteratively improves architectures over generations.

Evolution Cycle Diagram (Conceptual)

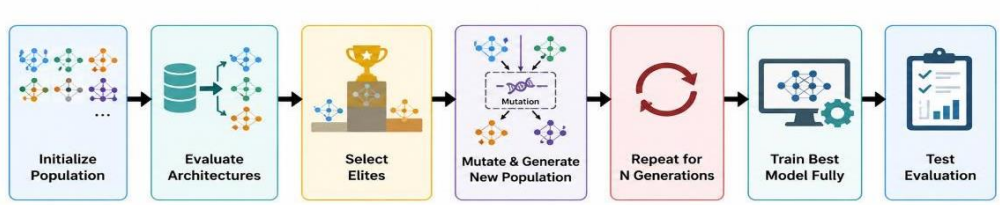

Fig 1 Evolution Cycle Flow.

RESULTS AND DISCUSSION

Experimental Setup

The entire process flow of the Neural Architecture Search algorithm involves the initialization of a population of architecture randomly generated. Evaluation of each candidate architecture according to their validation score is followed by selection of elites among the models with better performance. Mutation of the elites creates another population and allows exploration of better performing architectures. The process continues for a predetermined number of generations until the models become increasingly more efficient. Finally, the most efficient architecture among all will be trained rigorously on the test data set for final evaluation.

Performance Analysis

The NAS system successfully identified architectures with improved validation accuracy over generations. Key observations:

- Early generations show high variance in performance
- Later generations converge toward better architectures
- Mutation helps avoid local optima

```
Generation 0
Best validation accuracy: 0.7024000287055969

Generation 1
Best validation accuracy: 0.7107999920845032

Generation 2
Best validation accuracy: 0.6984999775886536

Generation 3
Best validation accuracy: 0.7182000279426575

Generation 4
Best validation accuracy: 0.7128000259399414
```

Fig.2 Validation Accuracy.

Best Architecture Find

```
Generation 0
Best validation accuracy: 0.7171000242233276

Generation 1
Best validation accuracy: 0.7196999788284302

Generation 2
Best validation accuracy: 0.724399983882904

Generation 3
Best validation accuracy: 0.7372999787330627

Generation 4
Best validation accuracy: 0.739300012588501

BEST ARCHITECTURE FOUND:
[(3, 32, 'none'), (3, 32, 'max'), (5, 64, 'max'), (3, 64, 'none')]
BEST VALIDATION ACCURACY: 0.739300012588501
```

Fig.3 Best Architecture.

This architecture balances:

- Feature extraction depth
- Computational efficiency
- Generalization capability

Test Performance

After full training:

- The final model achieves competitive accuracy on CIFAR-10
- Demonstrates effectiveness of evolutionary NAS even with limited resources

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.6622 - loss: 3.8765
Test accuracy: 0.6621999740600586
```

Fig.4 Test Performance.

DISCUSSION

The proposed system offers several advantages, including fully automated architecture design, which eliminates the need for manual model engineering, and significantly reduces human effort in the development process. It is also scalable and flexible, allowing the search process to adapt to different configurations and problem settings. However, the approach has certain limitations, such as high computational cost due to repeated training of multiple architectures. Additionally, the restricted search space may limit the exploration of more diverse and potentially better-performing models. Furthermore, the use of partial training for evaluation may not always accurately reflect the final performance of the architectures.

CONCLUSION

This paper presented an evolutionary Neural Architecture Search system for optimizing CNN architectures on the CIFAR-10 dataset. The proposed approach effectively automates the design of neural networks using a population-based search strategy.

The system demonstrates that even a simplified NAS framework can produce competitive architectures with reasonable computational resources. Future work can focus on expanding the search space, incorporating advanced search strategies such as reinforcement learning, and improving evaluation efficiency using surrogate models.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to Dr. Ramya B.N. for her valuable guidance and unwavering support throughout the course of this work. Her insightful feedback and expert advice played a crucial role in shaping the direction of this research. The encouragement she provided at every stage of this work was truly invaluable. The authors are deeply thankful for her time, dedication, and commitment to their academic growth.

REFERENCES

1. Zoph, B., & Le, Q. V. (2017). Neural Architecture Search with Reinforcement Learning.Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
2. Real, E., et al. (2019). Regularized Evolution for Image Classifier Architecture Search.
3. Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images (CIFAR-10).
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.