Multicriteria optimization of hybrid convolutional neural network structural synthesis using evolutionary algorithms

Illia Boryndo[1], Victor Siveglazov[2] and Michael Z. Zgurovsky[3]

[1] National Aviation University, Kyiv, Ukraine
[2] National Aviation University, Kyiv, Ukraine
[3] National Technical University of Ukraine, Kyiv, Ukraine

Abstract

This paper defines and describes the promising architectural solutions for convolutional neural networks and considers their key parameters for further structural and parametric synthesis. It's been proven that for better qualitive results these networks should include functional blocks such as SRU (Spatial Reconstruction Unit), CRU (Channel Reconstruction Unit), dense residual attention unit, etc., to the traditional supportive layers (batch normalization layer, 1x1 convolutional layer, dropout layer, etc). Described the advantages and disadvantages of different blocks as well as reasoning of their usage during structural synthesis. It is proposed to use a genetic evolutionary algorithm for structural-parametric synthesis and reviewed modern approaches. It is shown and described the process of configuring evolutionary algorithm. Based on optimization criteria the fitness function, selection, mutation and crossover approaches were defined. The results of experimental evolutionary process were shown and analyzed. It is considered an example of model generated by evolutionary algorithms that is based on using the functional blocks aggregated from different CNN architectural approaches. The performance criteria for each model during synthesis process is calculated, including average training time shortening, their advantages and architectural integration details. Based on the experimental results it is proven that utilizing complex structural blocks instead of traditional layers with flexible configuration of fitness function for both qualitive and performant criteria shows significant improvement for resulting model.

Keywords

structural-parametric synthesis; convolutional neural networks; genetic algorithm.

1. Introduction

Nowadays, even with significant progress in computer vision and the use of advanced convolutional networks, especially visual transformers [1], many image processing challenges remain unresolved and need tailored solutions. This is largely because of the unique characteristics of training datasets and the high computational demands of complex neural network topologies, like transformers, which face hardware constraints. Most CNN architectures suffer from low performance, slow learning rates, and simultaneously demand high-quality, well-balanced training datasets.

The natural way out of this situation is to create hybrid convolutional neural networks [2, 3]. However, many problems arise in this process: optimal choice of the basic topology of the convolutional neural network, optimal choice of various structural blocks to be used in the process of synthesizing the structure of the hybrid convolutional neural network, optimal choice of their locations. From the point of view of machine learning, there are problems of selecting learning criteria when we have a single-criteria or multi-criteria optimization problem, hitting a local extremum, overtraining, gradient drop, pre-training, pre-training, pre-training, creating a hybrid learning algorithm.

This paper will outline and discuss the research findings on the application and architectural characteristics of hybrid convolutional neural networks (HCCN) and various building blocks essential for their synthesis. It involves studying the performance of individual components, examining contemporary CNN architectures, and applying various training data to address

ICST-2024: Information Control Systems & Technologies, September 23-25, 2023, Odesa, Ukraine.
ibo.mistle@gmail.com (I. Boryndo); svm@nau.edu.ua (V. Sineglazov); zgurovsm@hotmail.com (M. Zgurovsky);
0000-0001-5375-6272 (I. Boryndo); 0000-0002-3297-9060 (V. Sineglazov); 0000-0001-5896-7466 (M. Zgurovsky);

performance challenges. The primary focus of this study is to identify the best kinds and arrangements of current CNNs, extract operational components, and incorporate them in the HCNN synthesis using evolutionary algorithms to attain favorable performance, accuracy outcomes and other optimization criteria. The main goal of this paper is to develop the evolutionary mechanism that will utilize structural components of different CNN architectures to create model that will satisfy predefined optimization criteria.

2. Related works and existing approaches

Convolutional Neural Networks have revolutionized the field of computer vision since their inception, offering unparalleled performance in image classification, object detection, and segmentation. Despite their success, the quest for improved accuracy and versatility has led researchers to explore hybrid models that combine CNNs with other neural network architectures. These hybrid models leverage the unique strengths of different networks, addressing the limitations of pure CNNs and enhancing their applicability across a wider range of tasks.

For structural synthesis of HCNNs there two main components – evolutionary algorithm (EA) itself and building components to be utilized by this algorithm. There're multiple existing solutions of evolutionary algorithms in integration with CNNs, but most of them have their own limitations.

Modern evolutionary algorithms are inspired by the process of natural selection and genetic evolution. They operate on a population of potential solutions, iteratively selecting, recombining, and mutating individuals to produce new generations of solutions. Key modern EAs include:

Genetic Algorithms (GAs):

- Operate with a population of candidate solutions.
- Use crossover and mutation to explore new architectures.
- Selection mechanisms favor better-performing individuals.

Genetic Programming (GP) [4]:

- Similar to GAs but operates on tree-like structures.
- Can evolve entire programs or architectures.

NeuroEvolution of Augmenting Topologies (NEAT) [5]:

- Evolves both the weights and the architecture of neural networks.
- Introduces innovations by tracking genes and structures through generations.

Evolutionary Strategies (ES) [6]:

- Focus on the optimization of continuous parameters.
- Use strategies like covariance matrix adaptation to guide the search.

The architecture of Hybrid Convolutional Neural Networks (HCNNs) [3] integrates CNNs with various other neural network models, capitalizing on their respective strengths to create more powerful and versatile models. Here, we delve into the specific architectures and mechanisms of three prominent types of HCNNs: CNN-RNN hybrids, CNN-GNN hybrids, and CNN-Transformer hybrids.

The architecture of Hybrid Convolutional Neural Networks exemplifies the synergy achieved by combining the complementary strengths of different neural network models. CNNs excel at spatial feature extraction, making them a crucial component in these hybrid architectures. When paired with RNNs, they can model temporal sequences effectively; when combined with GNNs, they can handle relational data adeptly; and when integrated with transformers, they can capture both local and global dependencies in data. These hybrid architectures have demonstrated superior performance across various domains, from video classification and image captioning to social network analysis and molecular graph prediction, showcasing their versatility and efficacy.

The goal of this paper is to define main criteria for CNN synthesis such as accuracy, computational cost, model robustness, etc., analyze and extract structural blocks of modern CNN architectures, modify the existing solution of evolutionary algorithms and apply it to synthesize optimal HCNN architecture.

3. Problem Statement

Hybrid Convolutional Neural Networks have demonstrated significant improvements in performance across various complex tasks by integrating the strengths of Convolutional Neural Networks with other neural network architectures. Despite their potential, designing optimal HCNN architectures remains a challenging task due to the vast search space of possible configurations and the intricate balance required between different components.

To address this, multicriteria optimization [2, 7] provides a framework for systematically evaluating and optimizing multiple performance metrics simultaneously. Evolutionary algorithms (EAs) are particularly well-suited for this task due to their ability to explore large, complex search spaces and their flexibility in handling multiple objectives.

The goal of this research is to develop and configure evolutionary algorithm for the multicriteria optimization of CNN structural synthesis using evolutionary algorithms. The key objectives include:

Defining the Optimization Criteria: Establishing a set of relevant performance metrics that reflect the quality and efficiency of HCNN architectures. These criteria typically include [8, 9]:

- Accuracy: The ability of the HCNN to correctly classify or predict data.
- Computational Efficiency: Metrics such as inference time and memory usage.
- Robustness: The model's resilience to variations in input data and adversarial attacks.
- Scalability: The model's ability to maintain performance when scaled to larger datasets or more complex tasks.

CNN Structural Design Space: Identifying and parameterizing the components and configurations of HCNNs, including:

- Type and Depth of CNN Layers: The number of convolutional, pooling, and fully connected layers.
- Type and Integration Method of Hybrid Components: The choice between structural blocks and components used for structural synthesis, types of integration components (e.g., SRU, CRU, LSTM, GRU, GCN, GAT, etc.).
- Connectivity Patterns: How different components are connected and the flow of data between them.
- Hyperparameters: Parameters such as learning rates, batch sizes, and dropout rates.

Develop Evolutionary Algorithm: Design and implement an evolutionary algorithm tailored for CNN structural synthesis. This involves:

- Encoding CNNs: Developing an encoding scheme for representing HCNN architectures in a manner suitable for evolutionary operations.
- Fitness Function: Formulating a multi-objective fitness function that balances accuracy, efficiency, robustness, and scalability.
- Selection Mechanism: Implementing a selection process to maintain diversity and guide the search towards optimal solutions.
- Crossover and Mutation Operators: Creating operators to generate new HCNN architectures by recombining and modifying existing ones based on preselected structural blocks.

By addressing these objectives, this research aims to significantly advance the state-of-the-art in hybrid neural network design and optimization, providing a powerful tool for the development of more efficient, robust, and scalable deep learning models.

4. Decomposition of modern CNN architectures and evaluation of their structural blocks

Currently, there is continuous advancement in the topology analysis of modern convolutional neural networks, with new architectural solutions and applications being continuously proposed. Due to increased applied tasks instead of iteratively increasing the complexity most modern convolutional neural network architectures started to implement functional structural components. These functional structural components incorporate the set of basic layers enhanced with unique connectivity approach, combinations and functional postprocessing. Such components could greatly increase qualitive parameters of networks and could be both used independently or as part of more complex structure.

For structural synthesis process of CNNs the main components are the structural blocks that will be utilized to form the generated architecture. To define the list of suitable blocks and to label them based on their qualitive parameters and functional means it is necessary to analyze and define such blocks within modern CNN architectures. Most of them are well known so we will take a look on modern corner cases instead.

Each of these blocks possesses its own distinct conceptual framework and attributes, necessitating their evaluation both as individual entities and as paired combinations within a single neural network. Subsequently, it is essential to conduct performance evaluations to analyze their intrinsic properties, effects on the overall system performance, and variations in accuracy [10].

In our experimental study, we will examine these individual blocks as well as pairs of blocks while employing genetic algorithms for HCNN synthesis. The designated CNN should be extremely uncomplicated and easy to understand. Simplicity reduces outside influences and randomness, making it easier to focus on each block's internal impact. The testing and training will be conducted using a sample known as "CIFAR-100".

Following the implementation of genetic algorithms to produce a fundamental CNN lacking distinct blocks, the system achieves an initial accuracy of 86.3% with a learning duration of 5.3 hours. These values will act as the baseline for future performance test comparisons. It's important to recognize that the overall training time depends on the hardware utilized, so only the differences in time should be taken into account.

The fundamental test-driven CNN structure is created using a series of predefined blocks, as per the algorithm. Next, the generated result model undergoes performance analysis [11]. By adding structural blocks to generation process the result values are listed in Table 1.

Table 1

Result parameter comparison table of single-used blocks

Block type	Top-1 Error(%)	Top-5 Error(%)	Time (H)	GFLOPS	Diff (~)
Densely connected layer	22.80	7.8	5.88	3.8	3.2
SCConv block	22.96	7.1	5.1	3.91	2.2
SCConc-A block	22.1	6.67	5.5	3.94	2.6
SE-BN-Inception module	22.68	6.94	4.92	2.87	1.8
Convolutional block attention module	24.66	8.34	5.42	3.71	5.9
DenRes-Att module	23.21	8.04	8.81	4.41	1.7
Inception-ResNet-V2	19.91	6.88	12	11.2	4.12
PolyInception module	24.48	8.25	8	3.98	2.7
Non-local Block	23.11	7.94	8.45	4.17	3.1

When facing issues with low learning performance, several solutions can be implemented. One effective approach is to enhance the current system's architecture by incorporating supportive blocks. The main ones are: batch normalization layer [15]; 1x1 convolution layer; dropout layer [12]; residual block.

5. Proposed structural synthesis of HCNN using evolutionary algorithm with functional modules and optimization-based construction blocks

5.1. Optimal combination and placement of structural blocks

The structural synthesis of Hybrid Convolutional Neural Networks involves the integration of various advanced building blocks to enhance network performance, efficiency, and robustness. This process requires a careful selection of building blocks such as Channel Boosting-Based CNNs (CB-CNN), Squeeze-and-Excitation (SE) blocks [13], Split-Combine-Convolutions (SCConv), and attention-based blocks. Each of these components brings unique strengths to the architecture, and their optimal combination can significantly improve the overall capabilities of HCNNs.

Comprehensive Integration: The optimal synthesis of HCNNs involves a strategic combination and placement of the aforementioned building blocks to leverage their strengths synergistically. For instance, an HCNN might start with CB-CNN layers to enhance initial feature extraction, followed by SCConv layers to capture diverse patterns. SE blocks can be interspersed throughout the network to recalibrate channel-wise features, while attention-based blocks can be integrated into deeper layers to focus on important contextual information.

Task-Specific Configuration: The choice and arrangement of these building blocks should be tailored to the specific requirements of the task at hand. For instance, in image classification, a configuration emphasizing CB-CNN and SE blocks might be optimal, whereas in object detection, a combination of SCConv and attention-based blocks could provide the best performance [1, 16].

Computational resources and efficiency should also guide the integration strategy. SCConv and attention-based blocks can be computationally intensive, so their use should be balanced with the overall resource budget.

This involves:

- Scaling Depth: Adding more layers to increase the network's capacity to learn complex features.
- Scaling Width: Increasing the number of channels in each layer to improve feature representation.
- Scaling Resolution: Using higher resolution input images to capture more detail.

5.2. Evolutionary algorithm for structural synthesis of Hybrid Convolutional Neural Networks

Genetic algorithms are part of evolutionary computing, a field of artificial intelligence. They are inspired by evolution and natural selection, where the strongest traits are passed down from generation to generation. The multicriteria genetic algorithm (MCGA) is an

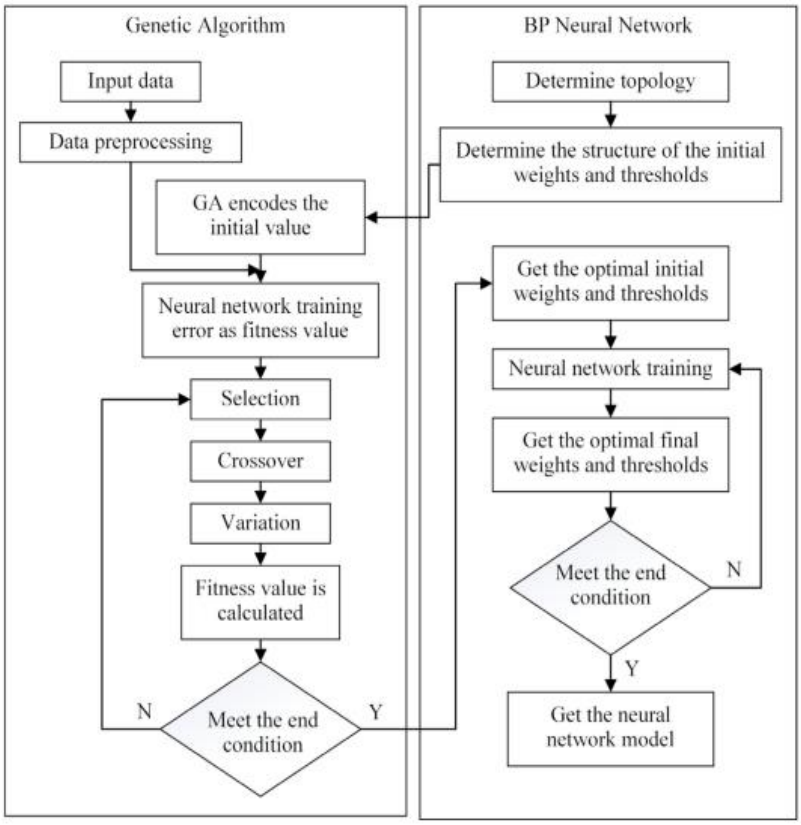

Figure 1: Algorithmic scheme of applying a multicriteria evolutionary algorithm to obtain an optimal topology of CNN

extension of this process. It focuses on optimizing multiple objectives simultaneously. Each solution provided by the algorithm is associated with a set of objective function values. The BCGA optimizes these values and provides a set of Pareto-optimal solutions.

In multi-objective optimization problems, there are several conflicting objectives that need to be optimized. This results in a set of possible solutions, known as Pareto solutions, where no other solution can improve all objectives simultaneously. Therefore, the goal is not to find a single optimal solution, but to generate a set of Pareto-optimal solutions that provide a trade-off between the conflicting objectives [2].

Multi-criteria genetic algorithms, such as NSGA-II1, SPEA3, have demonstrated strong performance in various engineering optimization problems. By utilizing selection, mutation, and crossover operators in iteration, competitive individuals can be generated, drawing inspiration from the evolutionary theory of "survival of the fittest." These individuals, who cannot surpass each other in every aspect, form a group known as the non-dominance front. From the point of view of physical optimization problems, in which evaluations are always computationally complex, the population size in MOGA is usually small due to limited computing resources. On the Figure 1 presented the logical flow for application of evolutionary algorithm to structurally synthesize optimal CNN structure. This is a high-level block-diagram and it does not reflect low level logic.

For structural synthesis of target CNNs we consider to user *modified* SPEA-3[8] algorithm to overcome the aforementioned problems.

5.2.1. Defining an Individual

In context of this paper, an individual is representing a specific CNN architecture. Based on the criteria defined in problem statement section, the target individual considered to be encoded directly into the string that explicitly describes the architecture. Genome structure is considered to encapsulate the following:

- number of layers;

- types of layers/blocks (SCConv, SE-BE-Inc, Dense block, standard convolutional, pooling, 1x1, batch normalization, etc.);
- kernel sizes;
- number of filters;
- stride and padding;
- activation functions;
- block-related specific parameters;
- learning rate, batch size, etc.

We will use a mixed encoding scheme where each individual (genome) consists of a series of structural blocks and hyperparameters. Each gene in the genome represents either a layer or a block, with specific parameters encoded within it.

The qualitive criteria on which the final individual is selected are defined and described in details in the next section.

Simplified example of the genome could be the following and represented in the unified JSON format:

[{"type": "Conv", "filters": 32, "kernel_size": 3, "stride": 1, "padding": "same", "activation": "relu"},
{"type": "DenseBlock", "num_layers": 4, "growth_rate": 12, "bottleneck_size": 4},
{"type": "SEBlock", "reduction_ratio": 16},
{"type": "SCConv", "filters": 64, "kernel_size": 3, "stride": 1, "padding": "same"},
{"type": "Pooling", "pool_size": 2, "stride": 2, "pool_type": "MaxPooling"},
{"type": "FC", "units": 10, "activation": "softmax"}].

5.2.2. Formulating a Multi-Objective Fitness function and evaluation criteria

To quantify the performance of Hybrid Convolutional Neural Networks based on multiple objectives, we need to construct a multi-objective fitness function that integrates various performance metrics. These metrics typically include accuracy, computational efficiency, robustness, and scalability. The proposed fitness function balances these aspects to provide a comprehensive evaluation of HCNN architectures.

Criteria that is considered for formulating fitness functions are following:

- Accuracy (A): The primary measure of how well the network performs on the task, such as classification accuracy on a validation dataset.
- Computational Efficiency (E): Metrics like inference time and memory usage indicate the efficiency of the network.
- Robustness (R): The network's resilience to adversarial attacks or noisy data, often measured by accuracy under adversarial conditions or performance degradation.
- Scalability (S): The ability to maintain performance when scaled to larger datasets or more complex tasks, often evaluated by the change in accuracy and efficiency when scaling the network.

Let $w_A, w_E, w_R,$ and w_S be the weights assigned to each performance metric to reflect their relative importance. The fitness function FFF is formulated as a weighted sum of the normalized performance metrics:

$$F = w_A \cdot \frac{A - A_{min}}{A_{max} - A_{min}} + w_E \cdot \left(\alpha \cdot \frac{T_{max} - T_{\text{infer}}}{T_{max} - T_{min}} + \beta \cdot \frac{M_{max} - M_{\text{usage}}}{M_{max} - M_{min}} \right) +$$
$$+ w_R \cdot \frac{A_{adv} - A_{adv,min}}{A_{adv,max} - A_{adv,min}} + w_S \cdot \left(\gamma \cdot \frac{\Delta A_{\text{scale}} - \Delta A_{\text{scale, min}}}{\Delta A_{\text{scale, max}} - \Delta A_{\text{scale, min}}} + \delta \cdot \frac{\Delta E_{\text{scale}} - \Delta E_{\text{scale,min}}}{\Delta E_{\text{scal, max}} - \Delta E_{\text{scale, min}}} \right) \quad (1)$$

Accuracy Normalization: $\frac{A - A_{min}}{A_{max} - A_{min}}$ normalizes the accuracy A between its minimum A_{min} and maximum A_{max} possible values.

Inference Time Normalization: $\frac{T_{max}-T_{\text{infer}}}{T_{max}-T_{min}}$ normalizes the inference time T_{infer}, where T_{min} and T_{max} are the minimum and maximum inference times.

Memory Usage Normalization: $\frac{M_{max}-M_{\text{usage}}}{M_{max}-M_{min}}$ normalizes memory usage M_{usage}, where M_{min} and M_{max} are the minimum and maximum memory usages.

The α and β are weights to balance the contributions of inference time and memory usage.

Robustness Normalization: $\frac{A_{adv}-A_{adv,min}}{A_{adv,max}-A_{adv,min}}$ normalizes the robustness metric A_{adv}, which measures accuracy under adversarial conditions.

Accuracy Scaling Normalization: $\frac{\Delta A_{\text{scale}}-\Delta A_{\text{scale, min}}}{\Delta A_{\text{scale, max}}-\Delta A_{\text{scale, min}}}$ normalizes the change in accuracy (ΔA_{scale}) when scaling the network.

Efficiency Scaling Normalization: $\frac{\Delta E_{\text{scale}}-\Delta E_{\text{scale,min}}}{\Delta E_{\text{scal, max}}-\Delta E_{\text{scale, min}}}$ normalizes the change in efficiency (ΔE_{scale}) when scaling the network. The γ and δ are weights to balance the contributions of accuracy and efficiency scaling. The weights w_A, w_E, w_R, and w_S, as well as the sub-weights α, β, γ, and δ, are chosen based on the specific requirements and priorities of the task. For example, if accuracy is paramount, w_A should be set higher relative to the other weights. The values of these weights could vary from 0 to 1.

5.2.3. Selection of Individual

Selection is the process of choosing individuals from the current population to serve as parents for the next generation. The goal is to favor individuals with higher fitness scores, ensuring that better-performing CNN architectures are more likely to propagate their genetic material. Fitness Evaluation: Evaluate each individual's fitness using a predefined fitness function described in previous section. Regarding the selection method, the are several ones, such as:

- Roulette Wheel Selection: Assign a selection probability to each individual proportional to its fitness. Randomly select individuals based on these probabilities.
- Tournament Selection: Randomly select a subset of individuals (tournament) and choose the best-performing one as a parent. Repeat to select the required number of parents.
- Rank-Based Selection: Rank individuals by fitness and assign selection probabilities based on rank, giving higher-ranked individuals a better chance of being selected.

In our approach we choose the rank-based selection due to its implementation simplicity and straight forward evaluation-based approach.

5.2.4. Defining crossover approach

Crossover combines the genetic material of two parent solutions to produce one or more offspring. This process creates new architectures by mixing and matching structural blocks and hyperparameters from the parents. For our implementation we decided to utilize modified multi-point crossover. This method enhances genetic diversity and allows for a more extensive exploration of the solution space compared to single-point crossover. In the context of CNN structural synthesis, multi-point crossover involves combining different structural blocks and hyperparameters from two parent CNN architectures to create new offspring architectures. Proposed crossover process is the following [1, 17, 18]:

1. Selection of Crossover Points:

- Choose multiple crossover points randomly within the genomes of the parent CNN architectures.
- The number of crossover points is typically predefined and can vary depending on the complexity of the problem.

2. Splitting and Combining:

- Split the parent genomes at the chosen crossover points.
- Alternate the segments from each parent to form the offspring genomes.

3. Reconstruction of Offspring:

- Reconstruct the offspring genomes by merging the alternating segments.
- Ensure that the resulting offspring are valid CNN architectures.

Example:
Parent 1: [Conv, 32, 3x3, relu]-[DenseBlock, 4, 12, 4]-[SEBlock, 16]-[SCConv, 64, 3x3]
Parent 2: [Conv, 64, 5x5, relu]-[SCConv, 32, 3x3]-[SEBlock, 8]-[DenseBlock, 6, 16, 4]
Offspring 1: [Conv, 32, 3x3, relu]-[SCConv, 32, 3x3]-[SEBlock, 16]-[DenseBlock, 6, 16, 4]
Offspring 2: [Conv, 64, 5x5, relu]-[DenseBlock, 4, 12, 4]-[SEBlock, 8]-[SCConv, 64, 3x3]

5.2.5. Mutation of Individuals

Mutation introduces random changes to the genetic material of individuals to maintain diversity in the population and explore new regions of the search space. For proposed implementations we propose the following transformations of individuals:

- Parameter Mutation: Randomly alter the parameters of a structural block.

  Example: Change the number of filters in a Conv layer from 32 to 64.

- Block Mutation: Replace one structural block with another of the same type but with different parameters.

  Example: Replace [DenseBlock, 32, 3x3, relu] with [SCConv, 64, 5x5].

- Layer Addition/Removal: Randomly add or remove a structural block in the genome.

  Example: Add [SEBlock, 16] after a Conv layer or remove an existing DenseBlock.

- Swap Mutation: Swap the positions of two structural blocks.

  Example: Swap [DenseBlock, 4, 12, 4] with [SEBlock, 16].

6. Experiment

In the same way as other implementations of evolutionary algorithms, our implementation requires a large computational capability to perform the task that makes it hard to evaluate it on large and complex datasets such as "ILSVRC-2017". We decided to utilize CIFAR-100 to synthesize set of potentially good architectures and then migrate it to larger-scale environment.

6.1. Settings and results

Initial setup for the experiment. Dataset is "CIFAR-100" that contains of 32x32 color images. The task is to receive most suitable CNN model based on criteria described in fitness function definition section. Population size is set to 10, generations – 50, mutation rate – 5.5%, initial population - randomly generated CNN architecture with a selective mix of convolutional layers, Dense blocks, SE-blocks, SCConv blocks, attention-based modules, SRU, CRU, Batch Normalization layers, 1x1 Conv layers, Dropout layers, and Fully Connected layers. The results based on the described evolutionary algorithm is shown in Table 2. and accuracy change dynamics is shown on Figure 2.

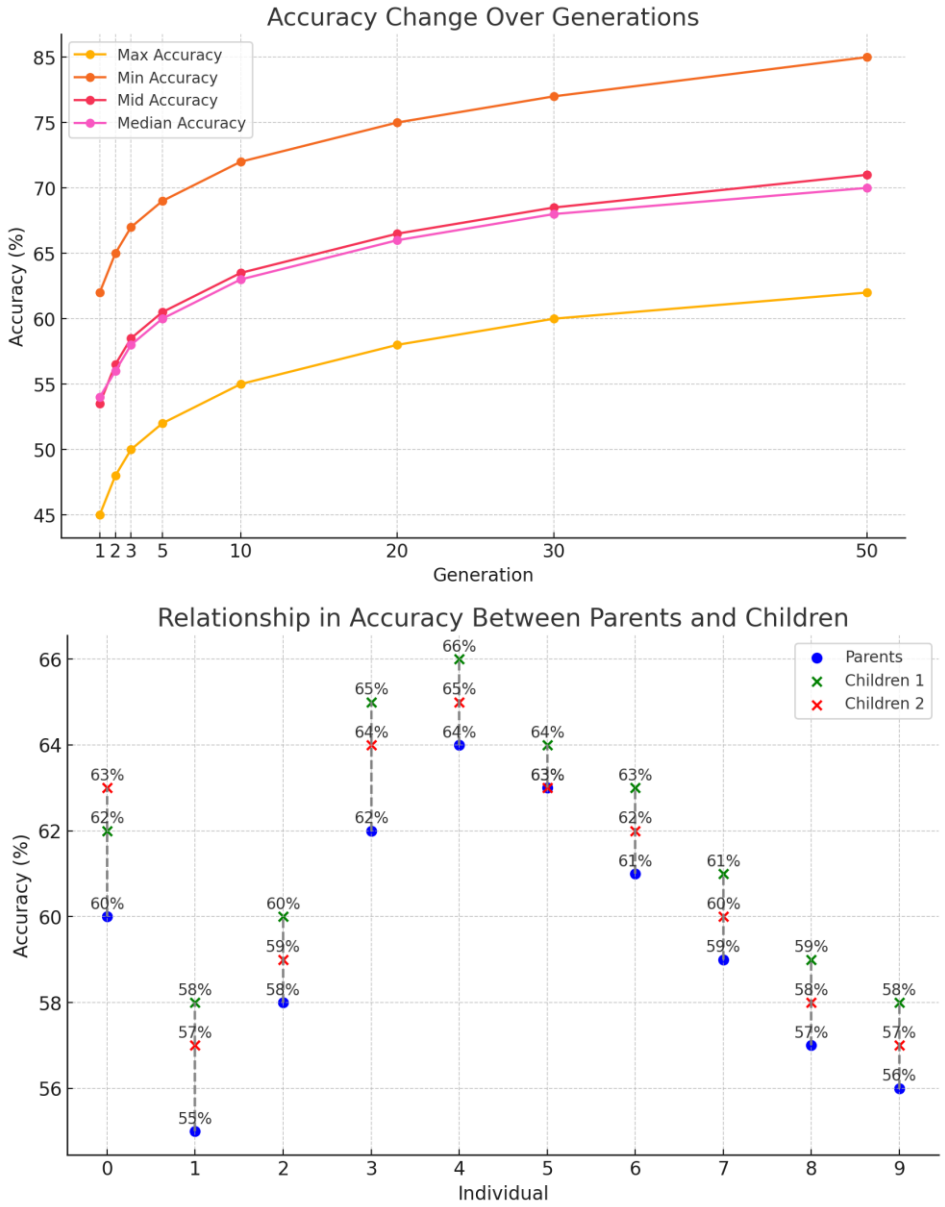

Figure 2: Recognition accuracy (%) change over generations on the CIFAR-100 dataset and example of accuracy change for each individual as parent-to-child relation

Table 2
Recognition accuracy (%) change over generations on the CIFAR-100 dataset

Gen	Max(%)	Min(%)	Avg(%)	Med(%)	Time(H)	Network Structure
01	75.13	72.53	73.83	73.61	6.31	Conv(64,5x5,relu)-SCConv(32,3x3)-SEBlock(8)-DenseBlock(6,16,4)-Pooling(2,Max)-Conv(128,3x3,relu)-Dropout(0.5)-BatchNorm-SRU(32,3x3)-SEBlock(16)-Pooling(2,Avg)-FC(512,relu)-FC(100,softmax)
02	75.65	72.75	74.2	74.48	5.93	Conv(32,3x3,relu)-DenseBlock(4,12,4)-SEBlock(16)-Conv(64,3x3,relu)-Pooling(2,Max)-Conv(128,3x3,relu)-Dropout(0.5)-BatchNorm-CRU(64,3x3)-

						SEBlock(8)-Pooling(2,Avg)-FC(256,relu)-FC(100,softmax)
05	76.22	74.19	75.20	75.32	6.02	Conv(64,3x3,relu)-SEBlock(16)-SCConv(32,3x3)-DenseBlock(4,12,4)-Pooling(2,Max)-Conv(128,3x3,relu)-Dropout(0.5)-BatchNorm-DenseBlock(6,16,4)-SEBlock(8)-Pooling(2,Avg)-FC(256,relu)-FC(100,softmax)
10	78.91	75.88	77.39	77.44	5.51	Conv(32,3x3,relu)-DenseBlock(4,12,4)-SEBlock(16)-Conv(64,3x3,relu)-Pooling(2,Max)-Conv(128,3x3,relu)-Dropout(0.5)-BatchNorm-SRU(32,3x3)-SEBlock(8)-Pooling(2,Avg)-FC(256,relu)-FC(100,softmax)
20	81.03	79.74	80.38	80.24	5.37	Conv(64,5x5,relu)-SCConv(32,3x3)-SEBlock(8)-DenseBlock(6,16,4)-Pooling(2,Max)-Conv(128,3x3,relu)-Dropout(0.5)-BatchNorm-CRU(64,3x3)-SEBlock(16)-Pooling(2,Avg)-FC(512,relu)-FC(100,softmax)
30	83.24	82.15	82.69	82.23	5.13	Conv(32,3x3,relu)-DenseBlock(4,12,4)-SEBlock(16)-Conv(64,3x3,relu)-Pooling(2,Max)-Conv(128,3x3,relu)-Dropout(0.5)-BatchNorm-DenseBlock(6,16,4)-SEBlock(8)-Pooling(2,Avg)-FC(256,relu)-FC(100,softmax)
50	84.91	83.69	84.05	84.13	4.87	Conv(64,3x3,relu)-SEBlock(16)-SCConv(32,3x3)-DenseBlock(4,12,4)-Pooling(2,Max)-Dropout(0.5)-BatchNorm-CRU(64,3x3)-Pooling(2,Avg)-FC(256,relu)-FC(100,softmax)

The maximum error percentage decreased significantly from 37% to 16% over 50 generations. This indicates that even the worst-performing models in the population improved significantly. The midpoint error rate saw a substantial improvement, reflecting overall population improvement.

7. Conclusions

The implementation of a multi-objective fitness function enabled a comprehensive evaluation of CNN architectures. This function incorporated normalized metrics for accuracy, inference time, memory usage, robustness under adversarial conditions, and scalability. By weighting these metrics appropriately, we could balance their contributions and achieve a holistic assessment of network performance

Our experimental results shown in Table 2 and Figure 2, obtained through training on the CIFAR-100 dataset, illustrated the significant impact of different structural blocks on HCNN

performance. By testing blocks both individually and in combination, we identified optimal configurations that enhanced feature representation and reduced computational load.

Based on the experimental results, we received optimal CNN architecture for recognition based on CIFAR-100 dataset. Due to computational and time restrictions, the experiment was executed in low-scale mode under limited number of generations and "CIFAR-100" dataset. It's considered to continue research within more complex conditions using more advanced hardware support in future.

In conclusion, this research advances the state-of-the-art in hybrid neural network design and optimization. The proposed hybrid learning algorithm and multi-objective optimization framework offer powerful tools for developing more efficient, robust, and scalable deep learning models.

References

[1] J. Li, Y. Wen and L. He, "SCConv: Spatial and Channel Reconstruction Convolution for Feature Redundancy," in: 2023 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 2023, pp. 6153-6162, doi: 10.1109/CVPR52729.2023.00596.

[2] S. Jiang and S. Yang, "A Strength Pareto Evolutionary Algorithm Based on Reference Direction for Multiobjective and Many-Objective Optimization," in: IEEE Transactions on Evolutionary Computation, vol. 21, no. 3, 2017, pp. 329-346. doi: 10.1109/TEVC.2016.2592479.

[3] M. Zgurovsky, V. Sineglazov, E. Chumachenko, "Classification and Analysis of Multicriteria Optimization Methods" in: Artificial Intelligence Systems Based on Hybrid Neural Networks, vol 904, 2021, pp.59-174, doi: 10.1007/978-3-030-48453-8_2.

[4] M. Meza-Sánchez, E. Clemente, M.C. Rodríguez-Liñán, G. Olague, "Synthetic-analytic behavior-based control framework: Constraining velocity in tracking for nonholonomic wheeled mobile robots" Information Sciences 501 (2019) 436-459, doi: 10.1016/j.ins.2019.06.025.

[5] Dürr, P., Mattiussi, C., Floreano, D., "Neuroevolution with Analog Genetic Encoding", in: Parallel Problem Solving from Nature, Springer, Berlin, Heidelberg, vol. 4193, 2006. doi:10.1007/11844297_68.

[6] N. Hansen, D. V. Arnold and A. Auger, "Evolution Strategies", 2018, doi: 10.1007/978-3-319-07124-4_13.

[7] V. Sineglazov, K. Riazanovskiy, A. Klanovets, E. Chumachenko, and N. Linnik, ''Intelligent tuberculosis activity assessment system based on an ensemble of neural networks'' Comput. Biol. Med. 147 (2022) 105800, doi: 10.1016/j.compbiomed.2022.105800.

[8] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", in: International Conference on Learning Representations, Jan 2017, pp. 1–15.

[9] C. Nagpal and S. R. Dubey, "A Performance Evaluation of Convolutional Neural Networks for Face Anti Spoofing," in: International Joint Conference on Neural Networks, Budapest, Hungary, 2019, pp. 1-8. doi: 10.1109/IJCNN.2019.8852422.

[10] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," in: IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, 2018, pp. 41-50. doi: 10.1109/TETCI.2017.2772792.

[11] M. K. Yadav and K. P. Sharma, "Intrusion Detection System using Machine Learning Algorithms: A Comparative Study," in: 2nd International Conference on Secure Cyber Computing and Communications, Jalandhar, India, 2021, pp. 415-420, doi: 10.1109/ICSCCC51823.2021.9478086.

[12] C. Cao, Y. Huang, Y. Yang, L. Wang, Z. Wang and T. Tan, "Feedback Convolutional Neural Network for Visual Localization and Segmentation", in: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 7, 2019, pp. 1627-1640. doi: 10.1109/TPAMI.2018.2843329.

[13] J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks" in: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 8, 2020, pp. 2011-2023. doi: 10.1109/TPAMI.2019.2913372.

[14] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1800-1807. doi: 10.1109/CVPR.2017.195.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift" ICML 37 (2015) 448-456.

[16] S. Bell, C. Zitnick, K. Bala, R. Girshick, "Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks" in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016 pp. 2874-2883, doi: 10.1109/CVPR.2016.314.
[17] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," ECCV (2016). doi: 10.1007/978-3-319-46484-8_29.
[18] V. Sineglazov, A. Kot, "Design of hybrid neural networks of the ensemble structure" Eastern-European Journal of Enterprise Technologies 77 (2022) 31-45, doi: 10.15587/1729-4061.2021.225301.