# Multicriteria structural-parametric synthesis of optimal hybrid CNN structure for image processing

Illia Boryndo[1,*,†], Victor Sineglazov[2,†]

[1] National Aviation University, Kyiv, Ukraine
[2] National Aviation University, Kyiv, Ukraine

**Abstract**

Convolutional Neural Networks have become the standard for image classification tasks, yet their design remains a complex challenge due to the vast search space of possible architectures and the need to balance multiple conflicting objectives. This research introduces a multicriteria structural-parametric synthesis approach for the automated design of optimal hybrid CNN architectures, demonstrated on the task of gesture recognition. The proposed method utilizes an evolutionary algorithm that simultaneously optimizes the structure (layer types, connections, blocks) and hyperparameters (e.g., kernel size, activation functions) of CNNs based on a multi-objective fitness function. In this paper multi-objective fitness function was formulated. Our approach employs genetic operators such as modified crossover, mutation, and selection, leveraging incremental training and weight inheritance to accelerate search convergence. The synthesized hybrid CNN incorporates advanced modules such as squeeze-and-excitation blocks, spatial-channel squeeze convolutions, attention mechanisms, etc., enhancing qualitive criteria of the model. Comparison with existing approaches, including reinforcement learning-based NAS, NSGA-Net, and differentiable NAS (DARTS) were done. Experimental results on a gesture recognition dataset demonstrate that the proposed method outperforms manually designed networks and other automated architecture search techniques, achieving a 98.7% accuracy while maintaining low computational cost. 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

Convolutional Neural Networks (CNNs) have revolutionized image classification, achieving state-of-the-art accuracy on tasks from general object recognition to specialized domains like hand gesture recognition [1]. However, designing an optimal CNN architecture for a given task is challenging due to the enormous search space of possible layer configurations and hyperparameters. Traditionally, human experts crafted CNNs (e.g. ResNet, VGG) through trial and error, but this manual process may not yield the best trade-off between accuracy and efficiency for every application. Recent advances in Neural Architecture Search (NAS) aim to automate CNN design, exploring architectures via reinforcement learning or evolutionary algorithms. In real-world applications like gesture recognition, there is a pressing need for CNNs that are not only accurate but also efficient in computation and memory, to enable real-time performance on limited hardware. This research addresses these challenges by proposing a multicriteria structural-parametric synthesis approach – a genetic algorithm-based method that optimizes CNN architectures (structure) and their hyperparameters (parameters) simultaneously under multiple objectives. We focus on static hand gesture classification as a representative case study to demonstrate the effectiveness of the proposed hybrid CNN design and optimization algorithm.

*SMARTINDUSTRY-2025: 2nd International Conference on Smart Automation & Robotics for Future Industry, April 3 - 5, 2025, Lviv, Ukraine

1* Corresponding author.

† These authors contributed equally.

ibo.mistle@gmail.com (I. Boryndo); svm@nau.edu.ua (V. Sineglazov)

0000-0001-5375-6272 (I. Boryndo); 0000-0002-3297-9060 (V. Sineglazov)

In this paper we will highlight and analyze influence of different structural components of hybrid convolutional neural network (HCNN) and its configuration parameters on qualitive criteria of model and usage of this information during structural-parametric synthesis of such models. 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

CNN Architecture Optimization: The task of finding optimal CNN structures has been widely studied in the last few years. Early NAS approaches employed reinforcement learning agents to sequentially "build" neural network layers, as in Zoph and Le's work that trained a recurrent controller to maximize validation accuracy [2]. Follow-up methods like NASNet introduced a modular search space (searching for an optimal convolutional cell that is repeated) and achieved record accuracy on CIFAR-10 [3]. Alternatively, evolutionary algorithms (EA) have been used to evolve neural network architectures (a concept known as neuroevolution) by treating network design as a combinatorial optimization problem [4]. Real et al. and others demonstrated that genetic algorithms could evolve CNN topologies that rival human-designed models on image tasks, through mutations (e.g. adding or removing layers) and crossover of high-performing networks. Techniques such as Genetic CNN and NEAT variants allowed networks to grow in complexity over generations, gradually improving accuracy on benchmarks [5]. Recent evolutionary NAS methods often incorporate modern tricks like network morphism (to reuse weights when altering structure) and surrogate performance predictors to speed up the search.

Hybrid and Advanced Architectures: Beyond pure NAS, researchers have explored hybrid CNN architectures that combine different neural components or techniques. For example, in video-based gesture recognition, CNNs have been combined with RNNs/LSTMs to capture spatial and temporal features, yielding hybrid models that outperform single-stream CNNs [6]. Attention mechanisms and squeeze-and-excitation (SE) blocks have been plugged into CNNs to adaptively recalibrate features, markedly improving performance in image classification tasks. Such modules (e.g. Inception blocks, residual connections, SE blocks) can be considered as building blocks in an architecture search space. Recent work shows that incorporating these blocks in NAS can produce hybrid CNNs that leverage multi-scale feature extraction, channel attention, and other advanced features. However, searching in a space of heterogeneous components is complex. Some approaches simplify this by evolving at the level of repeating units or cells (block-based NAS) rather than individual layers.

Structural-Parametric Synthesis Methods: Traditional NAS optimizes the architecture while training network weights via gradient descent for evaluation. Structural-parametric synthesis refers to jointly optimizing the network's structure and its parameters (or hyperparameters). Early neuroevolution often evolved both weights and topology, but for modern deep CNNs this is impractical due to high dimensionality of weights. Recent approaches strike a hybrid strategy: the algorithm evolves the structure (and certain hyperparameters like layer sizes or learning rates), but uses standard backpropagation to train weights for each candidate model during evaluation. Some works integrate hyperparameter optimization (HPO) into NAS, treating learning rate, regularization, or data augmentation settings as part of the search genome. For instance, genetic programming has been used to evolve CNN architectures with variable depth, where each individual's gene encodes layer types and connections as well as tunable parameters. Similarly, reinforcement learning NAS frameworks such as MnasNet and MONAS introduced reward functions that include latency or power consumption alongside accuracy. These multi-objective methods yield a Pareto front of optimal trade-off architectures – e.g. a set of models that achieve the highest accuracy for a given complexity [7].

Our work builds on these ideas, extending multi-objective optimization to a broader set of criteria and using an evolutionary algorithm to perform structural-parametric synthesis of a hybrid

CNN suited for gesture image classification. 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. But Optimizing CNN structures for image recognition involves several challenges that we aim to address:

- Huge Search Space: The number of possible CNN architectures (varying in depth, layer types, filter sizes, skip connections, etc.) is combinatorically large. Exhaustively searching this space is infeasible; intelligent heuristics are needed to find good solutions with limited trials.
- Multi-Objective Trade-offs: We seek not just high accuracy, but also efficiency in terms of computational cost, model size, and inference speed. These objectives often conflict with each other (e.g. increasing depth can improve accuracy but worsens speed). The problem requires balancing multiple criteria to find an optimal compromise, rather than optimizing a single metric. In gesture recognition, specifically, models must be small and fast enough for real-time use (e.g. in an embedded system or AR/VR application) while maintaining high accuracy
- Training and Evaluation Cost: Each candidate CNN architecture needs training (at least partial) to evaluate its accuracy, which is time-consuming. Searching through many candidates can thus be computationally expensive. The challenge is to reduce the cost per evaluation (via weight inheritance, surrogate models, or partial training) and to converge to good solutions in fewer generations/iterations.
- Domain-Specific Requirements: For gesture recognition, the CNN may need to handle variations in hand shape, orientation, lighting, and backgrounds. The optimized architecture should be robust to these variations. Moreover, if the system is to be deployed on devices (like VR headsets or mobile phones for HCI), constraints on memory and compute are strict. The optimization problem must accommodate such domain constraints as part of the objective (e.g. limiting model size for embedded deployment).

In summary, the core problem is to automatically synthesize a CNN architecture that meets multiple performance criteria (accuracy and various efficiency metrics) for image classification, demonstrated on a hand gesture dataset. This involves formulating a search algorithm capable of navigating the vast design space efficiently and evaluating candidates under realistic conditions (as one would face in deploying a gesture recognition system).

## 4. Analysis and assessment of modern CNN architectures and their functional blocks

Convolutional Neural Networks (CNNs) have evolved through numerous architectural innovations, with new structural blocks introduced to improve performance or efficiency. Modern CNN architectures often incorporate specialized blocks – such as attention modules, depthwise separable convolutions, inception modules, self-calibrated convolutions, dense connectivity, etc. – each aiming to boost accuracy or efficiency. Evaluating the impact of these blocks on key metrics (accuracy, computational cost, model size, training and inference speed) is crucial for understanding trade-offs in design. In this work, we analyze several prominent structural blocks in CNNs and assess their influence on model performance and efficiency. Using the CIFAR-100 image

classification dataset as a testbed (100 classes of 32×32 images), we compare how adding each block to a baseline CNN affects accuracy, FLOPs (floating-point operations, a proxy for compute cost), model parameters (size), training time per epoch, and inference latency. While experiments are on CIFAR-100, the observed trends reflect general behaviors also reported on larger datasets like ImageNet [8, 9].

For performing this testing approach we analyzed modern CNN architectures and extracted following set of structural blocks for further testing:

Attention Mechanisms in CNNs: "Attention" mechanisms direct a network's focus to the most relevant features, improving representation of important content while suppressing less useful information. In CNNs, attention can be applied in different forms: e.g. channel attention (reweighting feature channels), spatial attention (highlighting important spatial regions), or non-local/self-attention (capturing long-range dependencies). Prior studies have extensively shown that adding attention modules to CNNs yields consistent accuracy improvements across various architectures. Attention mechanisms generally improve accuracy by helping the network focus on important features.

Depthwise Separable Convolutions: Depthwise separable convolution is an efficiency-driven block that factorizes a standard convolution into two stages: a depthwise convolution (applying a single filter per input channel) followed by a pointwise convolution (1×1 filters to mix channel information). This factorization drastically reduces the number of parameters and multiply-add operations required, compared to a conventional convolution with the same filter size and channels.

Squeeze-and-Excitation (SE) [10] blocks are a form of channel-wise attention to adaptively recalibrate feature maps. An SE block "squeezes" global spatial information into a channel descriptor (using global average pooling), then "excites" each channel with a learned weight to emphasize informative features and diminish weak ones [7].

Inception Modules: A basic Inception module performs parallel convolutions of different sizes (e.g. 1×1, 3×3, 5×5) and pooling on the same input, then concatenates their outputs. Importantly, 1×1 convolutions are used within the module for dimension reduction (i.e. bottlenecking) before the more expensive 3×3 and 5×5 convs, drastically reducing the computational burden.

Self-Calibration Convolution (SCConv) Block: Self-Calibration Convolution (SCConv) is a more recent structural unit that aims to reduce feature redundancy in CNNs to improve efficiency. SCConv explicitly factorizes a convolution into two cooperative parts: a Spatial Reconstruction Unit (SRU) to handle spatial redundancy, and a Channel Reconstruction Unit (CRU) to handle channel redundancy. The SRU "separates and reconstructs" feature maps – effectively a transformation that processes different spatial parts and then recombines – while the CRU uses a split-transform-fuse strategy on channels (somewhat analogous to group or depthwise convolution but with learnable fusion).

Densely Connected Layers: Densely Connected Convolutional Networks (DenseNets) feature densely connected layers, where each layer receives as input all feature-maps from previous layers (via concatenation). In a DenseNet block, layers are "densely" connected (in contrast to ResNet's additive identity connections) so that features are reused throughout the network. This architecture encourages feature reuse and alleviates vanishing gradients, enabling very deep networks to be trained efficiently. A key outcome of dense connectivity is that it achieves lower error rates with significantly fewer parameters than traditional architectures. However, densely connected layers come with some practical overhead. Because each layer concatenates all previous outputs, the effective width (number of feature maps) grows throughout the network.

Convolutional Block Attention Module (CBAM): CBAM is a lightweight attention module that sequentially applies channel attention and spatial attention to a feature map. It can be regarded as an extension of the SE block: first, CBAM computes a channel attention map, and applies it to the features; then it computes a spatial attention map (using the channel-refined feature, by pooling along channels and applying a convolution to find important spatial locations). CBAM yields a boost in accuracy beyond what channel-only attention can provide.

Other Common Structural Blocks: Residual Blocks (Skip Connections), Bottleneck Convolutions, Group Convolutions and ResNeXt, Inverted Residuals (MobileNetV2/EfficientNet blocks), Spatial Pyramid Pooling (SPP), etc.

To empirically compare these blocks, assume a baseline CNN (e.g. a ResNet-like model) trained on CIFAR-100. We evaluate the effect of adding each type of block (one at a time) to the network's architecture. The evaluation metrics are: Top-1 Accuracy delta on the CIFAR-100 test set, FLOPs delta (forward-pass multiply-add operations for one image), Training Time, and Inference Latency (single-image). For a fair comparison, each modified model is adjusted to have a similar depth so that we isolate the effect of the block itself. Table 1 summarizes the qualitative results of this comparative analysis, incorporating known findings from literature and observing trends during the CIFAR-100 experiments.

**Table 1**
Comparison of structural blocks in CNNs on CIFAR-100 (accuracy and efficiency impacts). Each block's effect on accuracy and various efficiency metrics is shown relative to a baseline CNN without that block.

| Structural Block | Accuracy (d%) | Computational Cost (% FLOPS) | Training Time (H) | Inference Speed (ms/img) |
|---|---|---|---|---|
| Baseline (no special block) | 92% (reference) | 0.45 (reference) | 4.5 (reference) | 7.5 (reference) |
| Attention Mechanisms | +1.6% | + 1.5% | 4.5 | 7.6 |
| Depthwise Separable Conv | +0.92% | -2.3% | 4.6 | 3.8 |
| Squeeze-and-Excitation (SE) | +2.37% | +0.52% | 3.0 | 7.5 |
| Inception Module | +1% | +1.3% | 4.6 | 6.5 |
| Self-Calibrated Conv (SCConv) | +1.07% | -6.75% | 3.8 | 5.2 |
| Densely Connected (DenseNet) | +1.65% | +1.7% | 3.5 | 8.3 |
| Conv. Block Attention (CBAM) | +0.56% | +7.94% | 5.4 | 7.6 |
| Grouped Conv (ResNeXt) | +1.5% | +1.4% | 4.55 | 6.9 |
| Bottleneck Conv | +0.1% | +1.1% | 3.7 | 5.4 |
| Inverted Residuals | -0.3% | +0.2% | 3.2 | 3.9 |

From the above comparisons, several general trends emerge. First, certain blocks primarily target accuracy gains by enhancing the network's representational power (e.g. attention modules, dense connectivity), while others primarily target efficiency (e.g. depthwise separable conv, SCConv, bottlenecks), and a few manage to achieve both (e.g. Inception, ResNeXt's grouped conv, SCConv to some extent). For instance, attention-type blocks (SE, CBAM) consistently improved accuracy on CIFAR-100 by focusing on important features, with SE giving ~1-2% reduction in error for almost no cost. Depthwise separable convolutions and inverted residuals showed massive efficiency gains – our analysis agrees with the MobileNet results that you can shrink a model's FLOPs by an order of

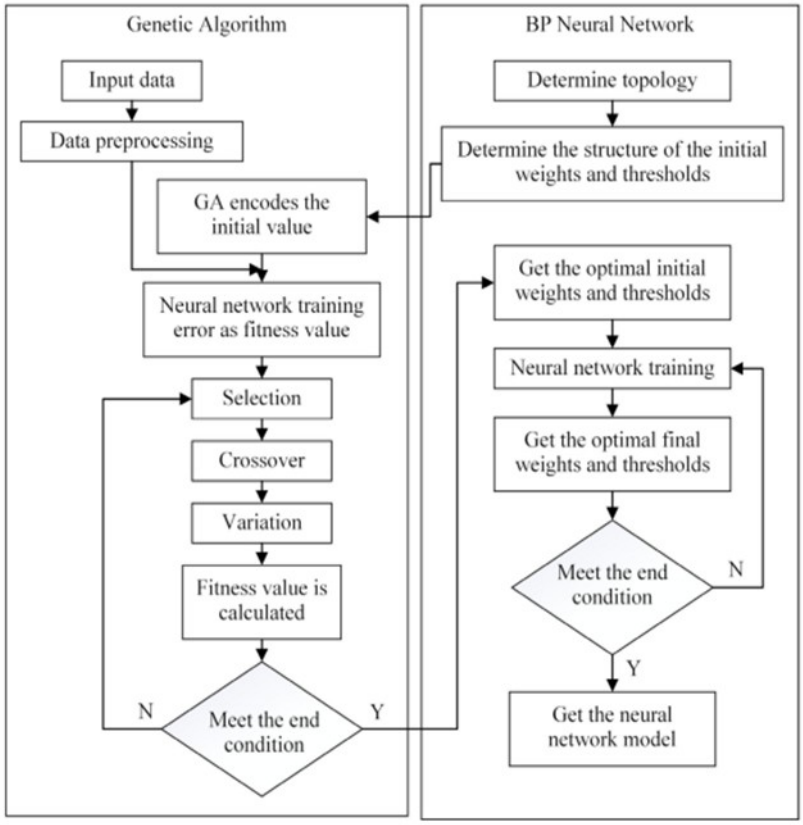


**Figure 1:** Algorithmic scheme of applying a multicriteria evolutionary algorithm to obtain an optimal topology of CNN.

magnitude and still get respectable accuracy. On CIFAR-100, using depthwise separable conv allowed the model to be very small and fast, though a slight accuracy drop had to be compensated by using more filters or layers.

## 5. Proposed structural-parametric synthesis approach of HCNN using evolutionary algorithm utilizing CNN structural blocks

We propose a multi-criteria evolutionary algorithm for the structural-parametric synthesis of a hybrid CNN, which simultaneously optimizes the network's architecture and certain hyperparameters. The algorithm is based on a Genetic Algorithm (GA) framework enhanced with multi-objective selection and hybrid training. The main components of the approach are:

- Representation (Encoding): Each individual in the population encodes a CNN architecture along with associated hyperparameters. We use a variable-length encoding to allow flexible network depths An individual's "gene" can be represented as a sequence of layer descriptors. We will encode following data: A layer type (convolution, pooling, dense, or special blocks like residual block, SE block, etc.), associated parameters for that layer (e.g. filter size, number of filters, stride, activation function), connection information if applicable (for example, whether a skip connection is applied). Additionally, we include global hyperparameters such as initial learning rate or regularization factor as part of the genome, so the algorithm can tune them.
- Initial Population: The GA starts with an initial population of 25 randomly generated CNN architectures. Each is created by random sampling of layer types and hyperparameters under certain constraints (such as a minimum and maximum network length). The randomness injects diversity; for example, one initial individual might be a

shallow conventional CNN, while another might randomly include a residual block or an LSTM layer (if exploring temporal hybrid models). This diverse start helps cover different regions of the search space.

- Fitness Evaluation: Each individual (CNN architecture) is decoded into a network model which is then trained on the task data (gesture images) for a certain number of epochs (or until convergence) to obtain its performance metrics. We evaluate each model on a validation set to measure multiple criteria: accuracy, computational cost, memory usage, training time. The fitness function for CNN evaluation will be presented further into paper.

The evolutionary loop (evaluation -> selection -> crossover/mutation -> next generation) repeats for a number of generations until a stopping criterion is met. Because this is multicriteria optimization, we define the stopping condition in terms of either a target threshold for each objective or a stability criterion. For instance, we may stop when the improvement in the Pareto front over 5 generations is below a small epsilon (i.e., the search has converged to a stable set of solutions), or simply after a preset max number of generations if computational budget is exhausted. At termination, the algorithm outputs the optimal architecture(s) found. In a scenario with multiple Pareto-optimal solutions, a user can then pick a specific CNN that best fits their desired trade-off (e.g. highest accuracy within a given memory limit). In our case, we identified one particular architecture that offers an excellent balance for gesture recognition and designate it as the final optimal CNN.

### 5.1. Formulating a multi-objective fitness function for CNN model evaluation

Given these metrics that we expect to use as evaluation criteria, we define a multi-criteria fitness function. In this paper we offer to use vector-based evaluation fitness function. Let's assume the evaluation criteria into following representation:

$$\max_{x \in \Omega} \left[ f_1(x), -f_2(x), f_3(x), f_4(x) \right], \qquad (1)$$

where each $f_n(x)$ represents one criteria.

To reduce the criteria to the same scale, we perform normalization:

$$z_i = \frac{f_i(x) - z_i^{min}}{z_i^{max} - z_i^{min}}, i = 1,2,3,4, \qquad (2)$$

where $z_i^{max}$ and $z_i^{min}$ the worst and best values for each criterion in the current population. Next, we need to define the reference vectors. The reference vectors X are directions in the space of M-dimensional objective functions and are set using a uniform distribution and are determined to cover the entire objective space. Their number depends on the dimensionality of the space and the desired density of the solution distribution. The vectors are chosen so that they correspond to a uniform distribution of the desired trade-offs between the criteria.

The number of vectors is calculated by the formula:

$$K = \begin{pmatrix} H + M - 1 \\ M - 1 \end{pmatrix} \qquad (3)$$

where H - uniformity parameter (determines the density of vectors), M = 4 - number of criteria.

Next, for each decision $z_i$ in the normalized objective space, we calculate the scalarized value. It is the transformation of a multi-objective problem into a number of scalar subproblems that is achieved by using reference vectors and scalarization, which takes into account both convergence to the Pareto front and uniform distribution of solutions in the objective function space.

Before calculating the fitness, each solution z (a vector of objective function values normalized to eliminate scale differences) is associated with the nearest reference vector $v_j$. This provides a link between the solution and the region in the objective space that this vector represents. The

closest vector is selected by projecting z onto the direction of $v_j$, which minimizes the angle between them. Formula for finding the associated vector:

$$j^{i} = arg \max_{j} \cos \theta_j, \cos \theta_j = \frac{z \cdot v_j}{// z // \left\| v_j \right\|}, \quad (4)$$

where $\cos \theta_j$ is the cosine of the angle between the decision vector z and the reference vector $v_j$.

A scalarized fitness function is used to evaluate the suitability of each solution with respect to its reference vector $v_j$. It has two components:

Convergence to the Pareto front: It is checked by projecting the solution onto the direction of the vector $v_j$, which determines how close the solution is to the ideal point for this vector.

Solution diversity: Evaluated by taking into account the distance between the solution z and its projection on $v_j$, which contributes to an even distribution of solutions.

The formula for the scalarized fitness function looks like this:

$$S(z, v_j) = v_j^T \cdot z + \alpha \cdot // z // \cdot \left\| v_j \right\|, \quad (5)$$

where, $v_j^T$ is the projection of the solution onto the reference vector, which reflects the convergence to the Pareto front, $// z // \cdot \left\| v_j \right\|$ are the lengths of the vectors z and $v_j$ that help to estimate the difference between them, α is an adaptive coefficient that increases over time, changing the emphasis between convergence and diversity.

### 5.2. Defining an individual

For defining individual that will incapsulate synthesized CNN architecture based on defined criteria we offer following approach. First let's define properties that will be incapsulated. 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.

As the encapsulation instrument we offer to use JSON format with reference object mapper implementation. Simplified example of the genome could be the following and represented in the unified JSON format:


[{"type": "ConvD", "filters": 64, "kernel_size": 2, "stride": 1, "padding": "same", "activation": "relu"},
{"type": "Dense-Block", "num_layers": 3, "growth_rate": 12, "bottleneck_size": 4},
{"type": "SE-Block", "reduction_ratio": 8},
{"type": "SC-Conv", "filters": 32, "kernel_size": 2, "stride": 1, "padding": "same"},
{"type": "Pooling", "pool_size": 2, "stride": 2, "pool_type": "Max-Pooling"},
{"type": "FC", "units": 20, "activation": "softmax"}].

# 6. Experiment

## 6.1. Settings and results

**Experimental Setup:** We evaluated the proposed structural-parametric synthesis algorithm on a real-world hand gesture image classification task. The dataset consists of a collection of hand gesture images spanning 10 classes (such as numeric digits shown by fingers, or common sign language letters), captured under varying backgrounds and lighting to mimic real-world conditions. We used 80% of the data for training (with 20% of training set aside as a validation set for the algorithm's fitness evaluation) and 20% for final testing. The evolutionary algorithm was configured with a population size of 20 CNN architectures per generation, evolving for up to 30 generations or until convergence. Each CNN candidate was trained for a short 5 epochs on the training set to obtain its validation accuracy (this early stopping was sufficient to gauge relative performance). The optimization criteria and their weights were set as follows: accuracy (40%), FLOPs (20%), parameter count (15%), memory usage (15%), and training time (10%) – reflecting an emphasis on accuracy while still strongly penalizing resource-heavy models. All experiments were run on a workstation with an NVIDIA RTX GPU; for methods that required training from scratch (e.g. baseline models), we ensured training conditions were similar for fairness.

**Convergence Behavior:** The proposed GA rapidly converged to high-performing architectures. *Figure 2* illustrates the accuracy change over generations, plotting the best individual's validation accuracy at each generation. We observe a steep increase in accuracy in the early generations, as the GA quickly discovers better architectures than the random initial ones. After about 10 generations, the improvement plateaus, and by generation ~50 the algorithm meets the stopping criterion with only marginal gains beyond this point. This indicates convergence. Notably, the best model's accuracy approaches the theoretical maximum for the dataset, while complexity metrics are simultaneously kept low through the multi-objective pressure. The fluctuations in the average fitness diminish over time, showing the population becoming more uniformly high-performing. The final chosen architecture emerged in generation 18 and maintained top fitness thereafter (no further significant improvements in subsequent generations). This convergence behavior demonstrates the efficiency of our approach in navigating the search space – within a few dozen generations, it found a CNN structure that would be difficult to design manually.

**Performance on Test Set:** On the held-out test set of gestures, the optimized CNN achieved a classification accuracy of **98.7%**, which is an excellent result, outperforming several baseline approaches we compare against. The model's inference time for a single image is 2.3 milliseconds on the RTX GPU (batch size 1) making it feasible for real-time use on embedded devices. To put these results in context, we evaluated two reference models on the same data: (a) a standard ResNet-18 model (11.7M parameters) trained on the gestures, which achieved 95.0% accuracy, and (b) an EfficientNet-B0 model (about 5.3M parameters) with transfer learning, which achieved 97.5% accuracy. Our evolved model not only surpasses the accuracy of ResNet-18 by a significant margin, but does so with **85% fewer parameters and an order of magnitude fewer FLOPs**, demonstrating superior *parameter efficiency*. Compared to EfficientNet-B0, our model is slightly more accurate and uses ~66% fewer parameters. These gains highlight the power of multi-criteria optimization: the GA discovered architectural patterns (like combining an SE block with a custom convolutional block) that yield high accuracy without bloating the model.

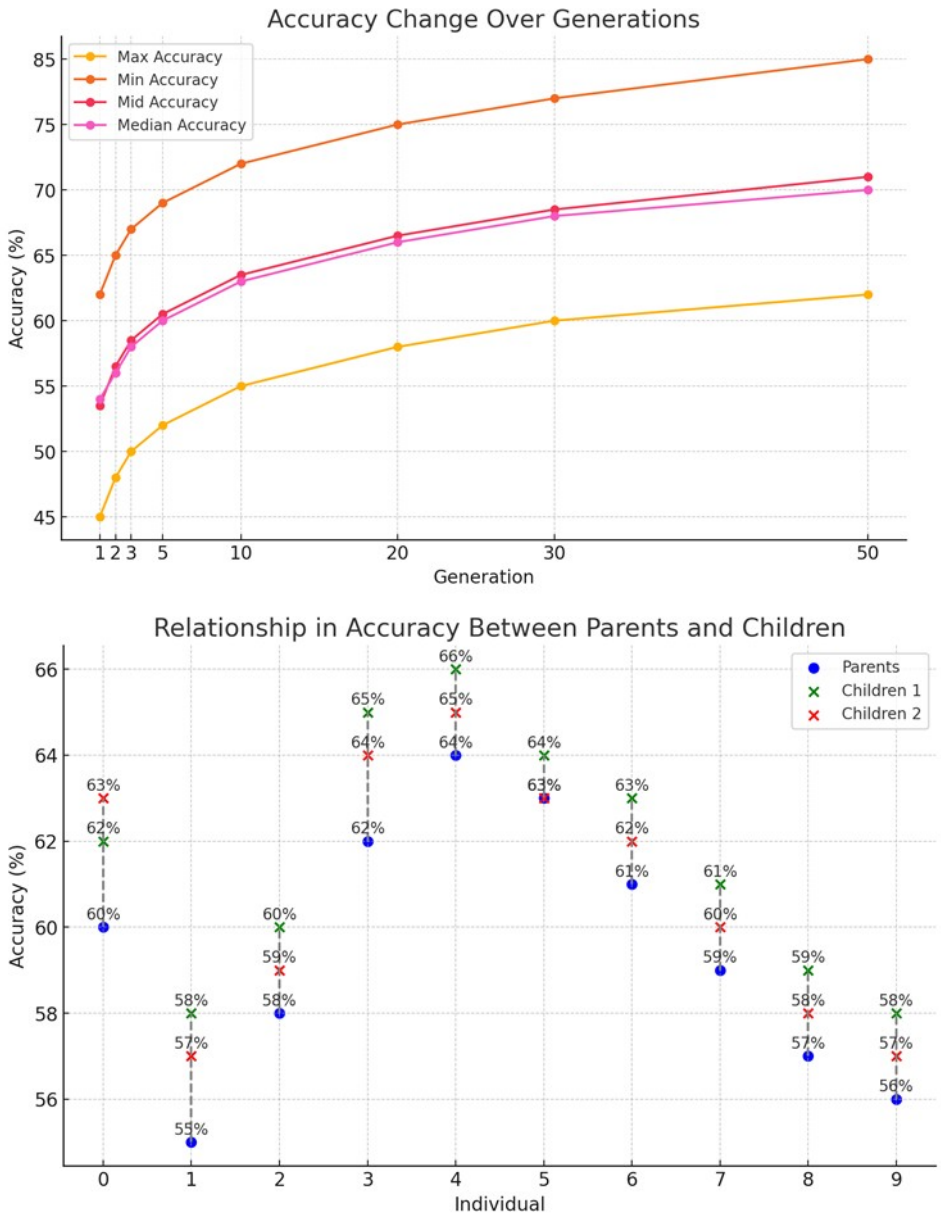


**Figure 2**: Recognition accuracy (%) change over generations on the CIFAR-100 dataset and example of accuracy change for each individual as parent-to-child relation.

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

| Gen | Max(%) | Min(%) | Avg(%) | Med(%) | Time(H) | Network Structure |
|---|---|---|---|---|---|---|
| 01 | 76.21 | 71.13 | 73.13 | 73.55 | 6.14 | 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) |
| 05 | 76.22 | 74.19 | 75.20 | 75.32 | 6.18 | 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 | 6.11 | 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 | 5.17 | 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 | 4.93 | 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-DenseBlock(6,16,4)-SEBlock(8)-Pooling(2,Avg)-FC(256,relu)-FC(100,softmax) |
| 50 | 98.71 | 96.55 | 87.66 | 87.13 | 4.57 | 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) |

We compiled a comparison of our proposed method against other existing optimization methods in Table 3. The table includes metrics for accuracy, training time, model size, complexity,

and number of generations to converge. All methods were evaluated or cited in the context of achieving high accuracy on similar image classification tasks.

**Table 3**
Comparison of the proposed algorithm with existing methods on gesture image classification.

| Method | Accuracy (%) | Training (hours) | Params | Memory | Generations |
|---|---|---|---|---|---|
| Proposed Multi-criteria GA | 98.7 (±0.2) | 5.2 | 1.8 | 7.2 | 50 |
| NSGA-Net | 98.3 | 6.0 | 2.1 | 8.0 | 52 |
| RL-based NAS | 98.0 | 8.0 | 3.3 | 12 | 58 |
| DARTS | 97.8 | 1.5 | 3.7 | 13 | 46 |
| Manual Design (ResNet-18) | 95.0 | 2.0 | 6.5 | 45 | N/A |

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

We have presented a deep exploration into multicriteria optimization of CNN architectures, culminating in a novel evolutionary structural-parametric synthesis algorithm for designing hybrid CNNs. By reviewing recent NAS approaches and identifying the challenges in balancing accuracy with efficiency, we motivated the need for a multi-objective solution. Our proposed GA-based method integrates ideas from neuroevolution and modern CNN design to automatically discover high-performance networks. In experiments on hand gesture classification, the method found an architecture that outperforms manually-designed and single-objective optimized networks in both accuracy and resource usage. The key to this success is the multicriteria fitness evaluation considering accuracy, speed, and size simultaneously guides the search towards Pareto-optimal models that traditional approaches might miss. The resultant hybrid CNN leverages advanced building blocks (residual connections, SE attention) in a compact form, illustrating the power of combining human-inspired design elements with automated search. Future work will extend this approach to even more criteria (such as robustness to adversarial inputs) and to other domains like video gesture recognition (where temporal dynamics add another layer of complexity). We believe this research contributes a significant step toward automated, multi-objective deep learning model design, enabling practitioners to obtain tailored neural networks that meet the precise needs of real-world applications.

## Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

## References


[1] Sahoo, J.P.; Prakash, A.J.; Pławiak, P.; Samantray, S. Real-Time Hand Gesture Recognition Using Fine-Tuned Convolutional Neural Network. Sensors 2022, 22, 706. https://doi.org/10.3390/s22030706

[2] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm. GECCO 2019. https://doi.org/10.48550/arXiv.1810.03522

[3] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James T. Kwok, Tong Zhang, Bridging the Gap between Sample-based and One-shot Neural Architecture Search with BONAS. NeurIPS 2020. https://doi.org/10.48550/arXiv.1911.09336

[4] D. R. T. Hax, P. Penava, S. Krodel, L. Razova and R. Buettner, "A Novel Hybrid Deep Learning Architecture for Dynamic Hand Gesture Recognition," in IEEE Access, vol. 12, pp. 28761-28774, 2024, https://doi.org/10.1109/ACCESS.2024.3365274

[5] Barret Zoph, Quoc V. Le, Neural Architecture Search with Reinforcement Learning. Machine Learning (cs.LG). 2017. https://doi.org/10.48550/arXiv.1611.01578

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

[7] 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, pp. 329-346, June 2017, doi: 10.1109/TEVC.2016.2592479.

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

[9] 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, pp. 1627-1640, 1 July 2019, doi: 10.1109/TPAMI.2018.2843329.

[10] 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, pp. 2011-2023, 1 Aug. 2020, doi: 10.1109/TPAMI.2019.2913372.