www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(4), 2026, 134-151. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2026.v17i4.1213

Automatic design of convolutional neural network architectures using multi-objective metaheuristics for classification of medical databases

Erick Franco-Gaona[1], *Maria Susana Avila-Garcia**[1], *Ivan Cruz-Aceves*[2], *Arturo Hernandez-Aguirre*[3]

[1] Departamento de Estudios Multidisciplinarios, División de Ingenierías, Campus Irapuato-Salamanca Universidad de Guanajuato.
[2] SECIHTI-Centro de investigación en Matemáticas (CIMAT).
[3] Centro de investigación en Matemáticas (CIMAT).
Email address(es): e.francogaona@ugto.mx susana.avila@ugto.mx ivan.cruz@cimat.mx artha@cimat.mx

Abstract. Convolutional neural networks (CNNs) have been used for different classification tasks, achieving superior performance to machine learning techniques. However, it is necessary to modify the hyperparameters, as their performance depends on the value assigned to them, and the empirical design of CNNs requires a lot of time and effort. Therefore, in this work, we propose a generic methodology, Neural Architecture Search (NAS), to automatically find architectures that classify different medical databases from literature, generating new models that maximize efficiency through the F1 score and minimize the number of parameters. The results of a single-objective (GA) and multi-objective (NSGA-II) genetic algorithm that generated a single architecture for three different medical databases were compared, obtaining an F1 score of 2.9 and 2.831 out of a total of 3 (1 for each database) as the best performance result, respectively. Although results show that the GA got better results than the NSGA-II, the latter took a sixth of the execution time to find the solution compared to the GA. The proposed generic method, with hyperparameter encoding in predefined ranges, allows binary medical databases (e.g., pneumonia or no pneumonia) to be classified efficiently with a single architecture.
Keywords: neural architecture search; convolutional neural network; medical dataset classification; multi-objective optimisation; genetic algorithm; NSGA-II; automated CNN design; hyperparameter optimisation; F1-score optimisation; model complexity reduction; binary medical classification; evolutionary computation.

Article information

Received: Aug 25, 2025
Accepted: Jun 11, 2026

1 Introduction

In the Mexican healthcare system, the use of assisted software for the analysis and classification of medical images has become increasingly important, especially in the face of a shortage of specialists in radiology and diagnostic imaging in many regions of the country. In 2018, 147,910 specialists were registered for a population of 123,518,272, of which only 69% had current certification, resulting in a rate of 119 specialists per 100,000 population (Heinze-Martin et al., 2018). This limited availability of specialized personnel highlights the need for technological tools to support clinical diagnosis. Systems based on artificial intelligence and deep learning make it possible to detect patterns in medical images with high precision, facilitating the early identification of diseases such as cancer, pulmonary or cardiovascular conditions. In addition to reducing evaluation times, these solutions reduce the workload of medical staff and contribute to improving the quality of diagnosis. In a resource-constrained environment, this type of technology represents a key opportunity to expand access to more efficient and equitable healthcare services.

Convolutional neural networks (CNNs) have established themselves as one of the most effective tools for image classification (Krichen, 2023; Zewen Li et al., 2021). Unlike traditional approaches (Kuo et al., 2021; Patel et al., 2020; Zhang et al., 2022), which rely on manual feature design, CNNs automatically extract relevant information from the data during training. In its early layers, the network identifies simple features such as edges and contours, while deeper layers build more abstract and complex representations from these features (Khan et al., 2018). Thanks to this hierarchical learning capability, CNNs can easily adapt to

different types of images and tasks, without requiring human intervention to define which image attributes are important to consider for classification.

Convolutional neural networks (CNNs) require the correct configuration of a significant set of hyperparameters to achieve efficient training. These are values that are set prior to training and directly influence both the architecture and performance of the model (Probst et al., 2019). Common ones include the number of convolutional layers, filter size, and number of pooling layers, among others. Determining which hyperparameters to adjust and which values to assign is not a simple task, since the architecture design can be viewed as a combinatorial problem, where the number of possible solutions grows rapidly as a function of the number of hyperparameters considered. To avoid this combinatorial problem, some authors use pre-trained networks such as VGG16 (Shuying & Weihong, 2015) and VGG19 (Simonyan & Zisserman, 2015); however, it is not always possible to have an extensive database, and there is a probability that the model will be overfitted. It is common to try to manually find the best possible architecture, i.e., by testing different architectures and keeping the best one found. This method requires a lot of time and human effort (Jang & Son, 2019).

Therefore, it is important to have a method that automatically generates CNN architectures and finds an efficient solution. Neural architecture search (NAS) is a branch of machine learning and artificial intelligence that aims to automate the design of neural network architectures (Kang et al., 2023). Its main goal is to discover highly efficient network structures for specific tasks with minimal human intervention (Ren et al., 2021). NAS uses optimization techniques to automatically explore the search space and evaluate different neural network architectures to provide the best solution found based on one or more chosen evaluation metric. Common optimization techniques include random search, evolutionary algorithms, and gradient-based optimization techniques, etc. In the literature, there is some consensus on the search space and evaluation strategies employed in NAS. However, the way in which the search strategy is approached varies according to each author's approach. For example, in works (Amou et al., 2022; Atteia et al., 2022; Borgli et al., 2019; Monteiro et al., 2020), Bayesian optimization is resorted to in order to improve the classification of medical images, achieving significant advances with respect to the state of the art. In contrast, other studies (Gaspar et al., 2021; Kilichev & Kim, 2023; Mostafa et al., 2020) adopt genetic algorithms (GA) and evolutionary computation techniques, highlighting their effectiveness on multipurpose datasets and evidencing better performance compared to previous solutions. The choice of approach depends largely on the researcher's preferences and available computational resources. One of the main advantages of population algorithms, such as genetic algorithms, is their high capacity for parallelization, which allows a considerable reduction in execution times.

This work is based on (Franco-Gaona et al., 2025), where an architecture was generated for each database using single-objective algorithms. The difference, apart from the multi-objective approach, is the generation of a generic architecture for multiple databases. Unlike previous works that generate unique solutions for unique problems, this work proposes:

1. A multi-objective NAS methodology that resolves the conflict between accuracy and computational complexity.
2. The demonstration that there is an optimal generic architecture capable of maintaining competitive performance in multiple medical diagnoses simultaneously, reducing the need for independent search processes for each new database.

The conceptual motivation for adopting a multi-objective approach via NSGA-II lies in the inherent conflict between model capacity and efficiency. While a single-objective Genetic Algorithm (GA) can achieve high F1-scores by expanding the architectural search space, it often leads to 'over-parameterized' models that are computationally expensive and prone to overfitting. In contrast, the multi-objective search allows for a critical exploration of the trade-offs: identifying architectures that sacrifice marginal increments in precision for significant reductions in complexity. This balance is essential for medical applications where a generic architecture must be robust enough to handle diverse datasets while remaining lightweight for deployment on limited hardware.

This study is divided into two parts, the generation of a generic convolutional architecture for three binary medical databases using GA and the comparison with a multi-objective Non-dominated Sorting Genetic Algorithm II (NSGA-II) to reduce the network parameters. This is done because the problem with a single-objective GA is that it can generate architectures with millions of parameters, which generates computationally more complex models to train. Due to the similarity of the algorithms, it is expected that the results will be efficient with the advantages offered by each technique. As a first part of the research the main contributions of this study are the following:

• A novel method to automatically generate convolutional neural network architectures for classification in binary databases with a single architecture.

• The present method was tested on three medical databases of different sizes obtaining a single architecture for all of them with a cumulative F1-score of 2.831 after 20 runs as best result.

2 Experimental procedures

This section presents the concept of optimization and each step to be followed in this work. The general model on which the proposed method is based and the optimization technique selected for the automatic search of architectures is presented. Finally, the proposal of this research is described.

The word "optimum" comes from the Latin optimus, which means "the best" or "the ultimate ideal". In this sense, optimizing refers to the process of improving a system, process or solution to get as close as possible to that ideal state. In the field of computer science, engineering and decision making, optimization seeks to find the best possible alternative among a set of options, given certain constraints and objectives. Solving an optimization problem involves following a series of structured steps to approach the problem in a systematic and efficient manner (Talbi, 2009):

1. **Formulate the problem:** In this initial phase, a general description of the problem to be solved is made. This formulation may be preliminary or imprecise, but it must clearly identify the internal and external factors involved, as well as the objective or objectives to be achieved. It is important to define whether it is a single-objective or multi-objective optimization, and what constraints should be considered.
2. **Model the problem:** Once the problem has been identified, it is translated into mathematical or computational terms by means of a formal model. It is common to rely on existing models documented in literature, which facilitates the adoption of proven approaches. This stage also involves making certain simplifications of reality so that the problem is computationally tractable, without losing the key elements that affect the solution.
3. **Optimize the problem:** With the model defined, the most appropriate optimization method or algorithm is selected, which can be exact (such as linear or integer programming) or heuristic/metaheuristic (such as genetic algorithms, simulated annealing or swarm algorithms). The objective is to find a solution that maximizes or minimizes the objective function, considering the given constraints. Depending on the type of problem, the solution obtained may be optimal or suboptimal.
4. **Implement a solution:** The proposed solution is validated by testing in real or simulated environments. This stage involves the participation of the decision-maker, who evaluates whether the solution is feasible and acceptable from a technical, economic or social point of view. If satisfactory, the solution is adopted, and its performance is monitored in practice. If it does not meet expectations, it may be necessary to return to earlier stages to adjust the model or rethink the problem formulation.

2.1 Experimental procedures

NAS can be formulated as an optimization problem, whose main objective is to find a neural network architecture that maximizes classification performance on a specific task, such as medical image classification. Formally, in single objective optimization an architecture $a \in \mathcal{A}$ is searched for in a previously defined search space A that maximizes F1-score on a test set:

$$\underset{a \in \mathcal{A}}{MAX}\ f(a) = F1\ score(a) \tag{1}$$

On the other hand, when an optimization problem involves multiple objective functions, we speak of multi-objective optimization or multi-criteria decision making. In this context, it is not appropriate to focus only on one of the objectives while ignoring the others. Different solutions can generate conflicts between the different objectives, known as trade-offs. It is possible that a solution may be optimal for one objective, but not for the others. Therefore, that solution cannot be considered as the best for the problem in general, which makes it necessary to seek a balance between all the objectives involved (Correa-Florez et al., 2008). Given a search space $\mathcal{A}$, the objective is to find an architecture $a \in \mathcal{A}$ that optimizes a set of objective functions:

$$\underset{a \in \mathcal{A}}{MIN}\ f(a) = (f_1(a), f_2(a), \ldots, f_k(a)) \tag{2}$$

Where:
$\mathcal{A}$: search space (e.g. all possible combinations of layers, block types, etc.)
a: a candidate architecture
$F(a)$: vector of objective functions to minimize (can be transformed into maximization if desired).
$f_i(a)$: the i-th evaluation metric (e.g., validation error, number of parameters, FLOPs, latency, etc.).
k: number of objectives.

The selection of these objectives is not arbitrary; it responds to the need for sustainable AI in healthcare, where the trade-off between the F1-score and the number of parameters ensures that the resulting generic architecture is not only accurate but also architecturally efficient for real-world clinical implementation.

To solve the above problem, some criteria must be defined to determine which solutions are of efficient quality and which are not. In multi-objective optimization problems, there is not always a single solution that is the best in all objectives at the same time. So, a set of solutions that represent the best possible compromises among the objectives is sought. That set is called the Pareto front. Goldberg in 1989 first proposed to use the concept of the Pareto front as a selection parameter (Goldberg, 1989). To address the above, the concept of dominance is incorporated, which facilitates the ranking of different solutions and helps to identify suitable options, considering both the existence and the magnitude of the *M* objectives of the problem. A solution x_1 is said to dominate another solution x_2 if the following conditions are met:

1. Solution x_1 is not worse than x_2 on all objectives.
2. Solution x_1 is strictly better than x_2 on at least one objective

If at least one of the conditions is not satisfied, it is said that solution x_1 does not dominate the solution x_2. A solution is non-dominated if no other solution dominates it. All the non-dominated solutions (the best ones) are in front 1, known as the Pareto front and the rest to front 2 or subsequent. Formally the Pareto set is described in equation 3.

$$\mathcal{P} = \{a \in \mathcal{A} | \nexists a' \in \mathcal{A}: F(a') \prec F(a)\} \tag{3}$$

This type of problem requires clearly defined objectives, decision variables and associated constraints. The purpose is to design an architecture that achieves the highest possible performance - measured through metrics such as accuracy, F1-score or sensitivity - on a given data set. Decision variables in NAS include structural and configuration aspects of the model, such as the number and type of layers (convolutional, pooling, dropout, fully connected), filter size, activation functions, number of neurons per layer, and other hyperparameters such as learning rate or batch size.

Additionally, various constraints, both computational and practical, must be considered. For example, it is common to impose limits on the total number of model parameters, the maximum training or inference time, the efficient use of hardware resources (CPU or GPU), or the available memory, especially when seeking to deploy the model on resource-limited devices. The focus of this work is multi-objective optimization, in which we simultaneously seek to maximize the F1-score and minimize the size of the model.

2.2. Model the problem

NAS has evolved along with the advances in deep neural networks. To cope with the complexity of searching architectures, it is necessary to automate the design process using machine learning (Elsken et al., 2019). The steps of NAS are described below, and the general methodology is shown in Figure 1.

1. Definition of the search space: A set of possibilities that represents all the neural network configurations that can be evaluated is defined. This space ranges from the number of layers to their specific type (such as convolutional, recurrent or dense layers), as well as other key aspects such as how they are connected to each other, activation functions, and training parameters such as learning rate. In essence, this step defines the boundaries within which the best group of architectures will be sought.
2. Automatic generation of architectures: From the previously defined space, algorithms capable of proposing new network structures are used. These can be generated randomly, following predefined rules, or by more advanced techniques such

as evolutionary algorithms, gradient search or reinforcement learning methods. The goal is to efficiently explore different possible combinations to identify promising designs.

3. Performance evaluation: Each of the proposed architectures is tested using a validation or test set, evaluating its ability to solve a specific task, such as image classification or time series. Performance is measured through relevant metrics, such as accuracy, F1 score or number of parameters, depending on the specific problem.
4. Model update: Each of the proposed architectures is tested using a validation or test set, evaluating its ability to solve a particular task, such as image classification or time series. Performance is measured through relevant metrics, such as accuracy, F1 score or number of parameters, depending on the specific problem.
5. Choosing the optimal architecture: After completing a given number of iterations or reaching a stopping criterion, the best performing architecture is selected. This final network represents the best solution found by the automatic search system within the defined space and can be used for real tasks or as a starting point for further refinement.

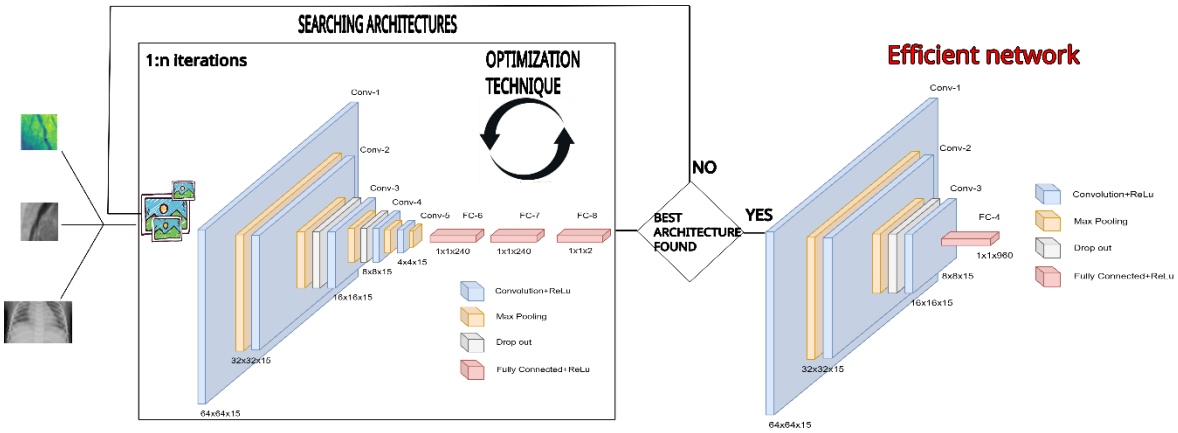


Figure 1. Flow of the NAS process in a generic way. During the search process using an optimization technique, a random architecture is initially defined; at the end of the process, an efficient architecture is obtained for all case studies.

The search space in the NAS case is the hyperparameters set before training. The hyperparameters and ranges considered for this study are shown in Table 1. To define the ranges we considered the literature where it has already been demonstrated that certain values work adequately for medical image classification (Gaspar et al., 2021; Lee et al., 2021; Monteiro et al., 2020; Singh et al., 2021). The hyperparameters were encoded in a binary array of size 29 containing each of them, and the range of values depends on the length of each hyperparameter; consequently, the complexity of the problem is $O(2)^{29}$. If the hyperparameter has only 2 options, an array space is used where 0 represents the first option and 1 the second option. If the hyperparameter can have a range of values, that range is explicitly defined by converting from binary to decimal.

Table 1. Selected hyperparameters and ranges based on literature.

Hyperparameters	Ranges or values
Kernel size	3x3 / 5x5
Stride size	1 / 2
Drop out layers	0-3
Pooling layers	0-7
Batch size	8/16/24/32
Number of filters	5-63
Convolutional layers	1-15
Number of neurons	4-31
Full connected layers	1-7
Learning rate	0.001/0.01/0.1/0.02

2.3. Optimize the problem

Evolutionary Algorithms are search and optimization procedures that emulate the mechanisms of natural evolution that operate by giving a higher probability of survival and reproduction to those individuals more adapted to the environment. In terms of the problem to be solved, an individual represents a potential solution to the problem, the set of individuals is called population. These algorithms work on populations of solutions that evolve from generation to generation by means of evolutionary operators (Engelbrecht, 2007). Generally, the evolutionary search is determined by the following steps:

1. Encode the solutions in binary arrays known as chromosomes.
2. An evaluation function is defined for the individuals.
3. Initialization of the population
4. Selection of the genetic operators
5. Replication of the operators

If the problem has constraints, they should be considered when generating solutions. For example, in the NAS problem there must be at least 1 convolutional layer, and the number of pooling layers is restricted by the size of the input image. In general, in evolutionary algorithms it is common for solutions to be represented as binary chains known as chromosomes. A chromosome is equivalent to an individual and is composed of genes. The value assigned to each gene is called allele (Engelbrecht, 2007). An example of this type of encoding can be seen in Figure 2. A good encoding should represent a decision vector of the solution space. Moreover, it should not generate incorrect individuals when applying evolutionary operators, especially if there are constraints.

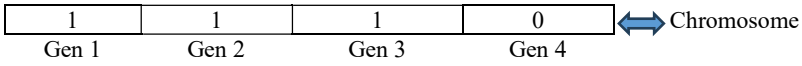


Figure 2. Example of a 4-gene chromosome. Each chromosome is equivalent to a solution in the search space

GAs are a class of optimization algorithms inspired by the theory of biological evolution and genetics. These algorithms are used to solve combinatorial optimization problems, handling multiple solutions at the same time to be evaluated, which, collectively, are known as population. They can only optimize one objective at a time. The basic process of a genetic algorithm is summarized in Algorithm 1. An initial population of candidate solutions is created randomly or using some heuristic method. Each candidate solution in the population is evaluated using an objective function that measures its quality in terms of the solution to the optimization problem. Candidate solutions are selected from the current population to act as parents in the next generation. Solutions with better quality have a higher probability of being selected, but some degree of randomness may be included to encourage diversity. Genetic operators, such as crossover and mutation, are applied to selected parents to generate new offspring. Crossover involves combining parts of two or more solutions to create a new solution, while mutation introduces random changes to a solution. Each solution of the new offspring is evaluated using the objective function. The solutions are selected to form the next generation, using some selection method based on solution quality and possibly other criteria such as diversity and elitism. The process is repeated for a fixed number of generations or until some stopping criterion is met, such as reaching a satisfactory solution or running out of computational resources.

Algorithm 1. Pseudocode of genetic algorithms.

Input: Population size N, selection percentage P_s, mutation P_m, number of generations G

Output: Best individual I^*

Start population P with N individuals;
for $g = 1$ *to* G **do**
    Evaluate each individual's performance in P;
    Select individuals P_s of P for crossover;
    Apply crossover and mutation;
return best individual found I^*;

On the other hand, if a multi-objective problem needs to be solved, there is a genetic algorithm for this known as Non-dominated Sorting Genetic Algorithm II (NSGA-II). The NSGA-II introduced by Deb (Deb et al., 2002), generates an initial population P_t of size *N*, where each individual represents a possible solution to the problem. Each individual is evaluated with respect to all the objective functions. For example, if a problem with two objectives is being solved, F1 and F2 are calculated for each solution. The population is sorted into different fronts following the non-dominance model, where the pareto front contains the best solutions. Within each front, a diversity measure called crowding distance is calculated. This distance estimates how close the solutions are to each other. It is used to maintain diversity in the population. Parents are randomly selected from each front for crossing using a binary tournament: Two individuals are chosen at random. The one with the lower front ranking is preferred (e.g., Front 1 wins over Front 2). If they are on the same front, the one with the greater crowding distance is chosen. The calculation of crowding distance is illustrated in equation 4. From the selected parents, genetic operators are applied: Crossover: combining parts of two solutions to form offspring. Mutation: making small random changes in an individual to explore new areas of the solution space. This generates *N* offspring (population Q_t). The current population P_t and the new population of offspring Q_t are combined into R_t of size *2N*. Non-dominated front ordering is applied to R_t. The best *N* individuals are selected to form the new population P_{t+1}, starting with Front 1, then Front 2, and so on. This is repeated for a given number of generations. This process is shown in Algorithm 2.

$$Distance = \sum_{m=1}^{M} \frac{f_m(i+1) - f_m(i-1)}{f_m^{max} - f_m^{min}} \tag{4}$$

Where:
M = total objective functions.
$f_m(i)$ = value of the objective function m for each individual
f_m^{max} & f_m^{min} = maximum and minimum value of target *m* on the front.
$*D_i = \infty$ if *i* is the first or last of the function f_m (keep extreme values).

Algorithm 2. Pseudocode of NSGA-II.

Input: Population size N, selection percentage P_s, mutation P_m, number of generations G
Output: Best individual *I**
Start population P with N individuals;
for $g = 1$ *to* G **do**
    Evaluate each individual's performance in P;
    Arrange individuals on each front;
    Select individuals for reproduction using crowding distance of P;
    Apply crossover and mutation;
return best individual found *I**;

2.4. Implement a solution

After completing the optimization process using NAS, the architecture that has shown the best overall performance is selected as the final solution, considering a weighted sum of target metrics, such as the F1-score and the number of network parameters. F1-score is calculated as the result of equation 4. Equation 5 shows the weighted sum for the selection of the architecture where x_1 and x_2 are the f1-score and the number of parameters respectively, and w_1 and w_2 have a value of 0.5, giving equal importance to both metrics. This step not only consists of choosing the most promising architecture but also subjecting it to more rigorous and detailed validation. It should be noted that during the search phase, the model was partially trained to reduce the computational load. However, in this final stage, the chosen architecture is fully trained using the entire available training set and applying a more intensive training regimen. Finally, an evaluation is carried out on an independent test set to measure its actual performance under conditions that simulate its end use.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

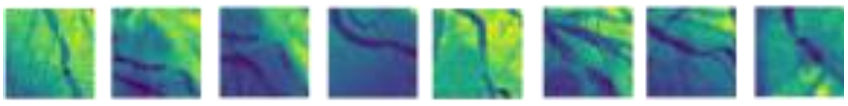
$$S = \sum_{1}^{n} w_i * x_i \tag{5}$$

2.5. Optimize the problem

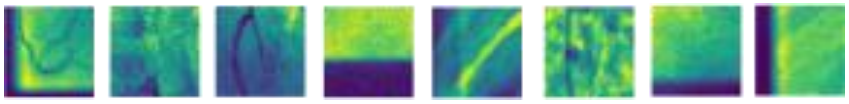
This section describes the databases employed to evaluate the proposed method and determine the most suitable architecture for each case. Table 2 provides a general overview of the characteristics of all datasets used in this study. To ensure consistency across experiments, all images were resized to 64×64 pixels, as variations in image size can significantly impact the resulting architecture. Following the table, representative image samples from each dataset are presented. These examples are shown at their original scale, which may result in some images appearing with reduced visual quality due to lower resolution.

Table 2. Description of medical databases taken from literature to test the generated architecture.

Database	Number of images	Description
Coronary stenosis (Antczak & Liberadzki, 2018)	250	The database consists of 250 patches taken from real angiograms and classified by experts. The patches show examples of arteries that are obstructed and unobstructed due to coronary stenosis. Positive and negative instances represent an equal 1:1 ratio.
Stenosis608 (Gil-Rios et al., 2024)	608	Database contains 180 expert-annotated digital images measuring 512 × 512 pixels. In total, a set of 608 patches was formed. The patches indicate arteries with or without coronary stenosis. Positive and negative instances, representing an equal 1:1 ratio.
Pneumonia (Melendez et al., 2015)	5856	The database consists of a total of 5,856 anteroposterior chest X-rays of pediatric patients between the ages of 1 and 5. The entire image is used for classification, rather than patches. The database has 4,273 positive images and 1,583 negative images, so it is randomly truncated to 1,583 for both classes.

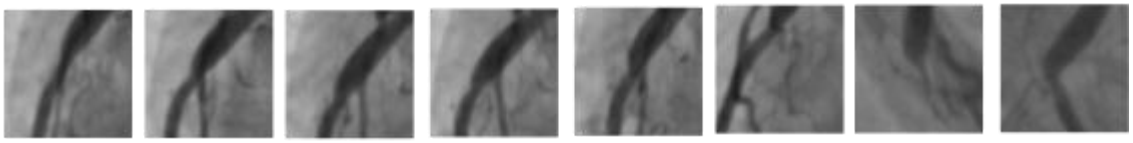


(**a**) Positive

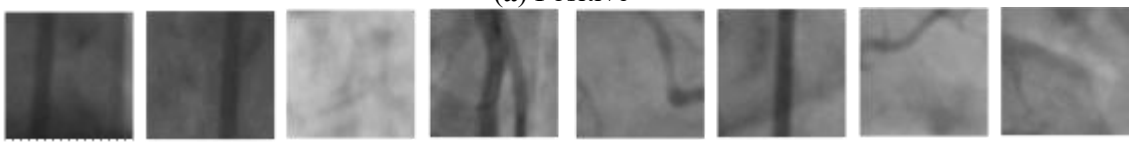


(**b**) Negative

Figure 2. Subsection (a) shows examples of stenosis, and subsection (b) shows examples of non-stenosis in actual angiograms (Antczak & Liberadzki, 2018).

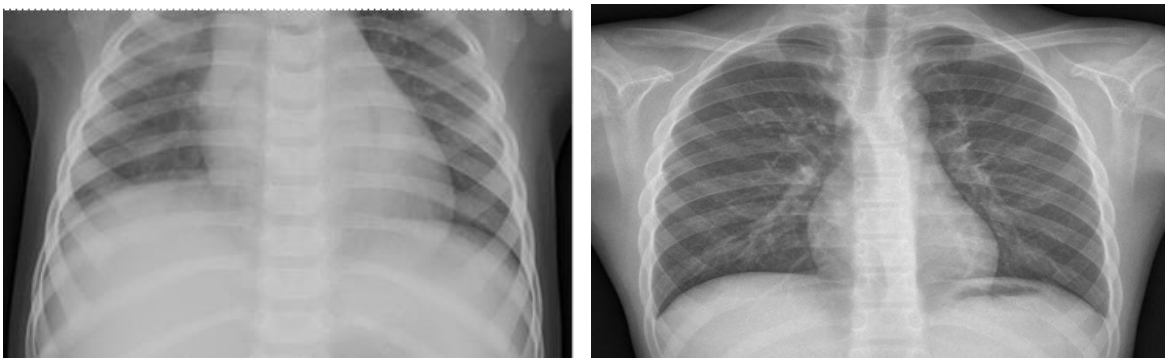


(**a**) Positive



(**b**) Negative

Figure 3. Subsection (a) shows examples of stenosis, and subsection (b) shows examples of non-stenosis in actual angiograms from Stenosis608 (Gil-Rios et al., 2024).



(**a**) Positive (**b**) Negative

Figure 4. Subsection (a) shows examples of lungs with pneumonia, and subsection (b) shows examples of healthy lungs from Melendez et al (Melendez et al., 2015).

3 Results

The experiments were conducted using a server equipped with the following specifications: 128 GB of RAM, an Intel Xeon Silver 4214 processor, and a 24 GB NVIDIA Titan RTX graphics card. This server compatibility and powerful tools that aided in the execution of population algorithms, such as Parallel Computing Toolbox, allowed the algorithms to take advantage of the cores and complete the process effectively and efficiently. In addition, MATLAB 2024b integrates with hardware such as GPUs (NVIDIA) to accelerate training. Building CNNs from scratch during algorithm execution was facilitated by Deep Learning Toolbox. A file was programmed to build the CNN with the instructions provided by each algorithm. Table 3 shows the initial parameters for GA and NSGA-II.

Table 3. GA and NSGA-II parameters in the NAS optimization process.

Algorithm	Parameters	Values
GA/NSGA-II	Generations	20
	Population size	10
	Selection rate / Pareto fraction	0.6
	Mutation rate	0.02

For the experiments, the databases were divided separately into 80% for training, 10% for validation, and 10% for testing. GA and NSGA-II were run 20 times on a training set with validation to check the robustness of the optimization techniques. Table 4 shows the statistical results based on the F1-score of the 20 runs, the recall, and the precision for the three databases. The results shown in Table 5 correspond to the test set and present the best results obtained for the three databases, considering the weighted sum and the F1-score and number of parameters separately.

Table 4. Overall F1-score statistical results for 20 runs for the three databases in training dataset.

Metric	Minimum	Maximum	Average	Median	Standard deviation
		GA			
F1-score	2.799	2.899	2.836	2.827	0.029
		NSGA-II			
F1-score weighted sum	2.014	2.831	2.483	2.568	0.261
Best F1-score	2.631	2.862	2.747	2.761	0.073

Table 5. Best specific and overall F1-score results for the three databases in testing set. Both NSGA-II F1-scores are on the Pareto front.

Stenosis	Stenosis608	Pneumonia	General	Parameters	Time to find solution (20 runs)
		GA			
0.999	0.95	0.949	2.9	909326	30h 50m 1s
		NSGA-II			
		NSGA-II weighted sum			
0.923	0.95	0.9589	2.831	28952	5h 5m 30s
		NSGA-II best F1-score			
0.917	0.984	0.962	2.862	227636	5h 5m 30s

As can be seen in the results in Table 5, the proposals have a low standard deviation, thus proving their robustness. As mentioned, the algorithms were tested 20 times with the three databases, using the same architecture but separately, accumulating the F1-scores. The means and medians are very close, suggesting that the distribution is approximately symmetrical, with no significant bias. The solutions shown in Table 5 for NSGA-II are valid solutions on the Pareto front, then any could be chosen. An architecture with fewer parameters could be used, still yielding efficient results. The time it takes to find a solution with GAs is considerable, unlike NSGA-II, since once it finds architectures with fewer parameters and a high F1-score, it propagates those architectures, which generates less computational cost.

To validate the reliability of the architectures found using NAS, an overfitting risk analysis was performed. This risk was quantified as the absolute difference between the performance obtained in the training set and the test set (F1 train - F1 validation) during the 20 independent runs of each algorithm. The distribution of these values is shown in Figure 5, where greater proximity to the ideal reference line (zero) indicates better generalization capacity of the model. Furthermore, the presence of outliers in this strategy suggests a notable instability in the search for robust solutions for medical databases. In contrast, variants of the NSGA-II multi-objective algorithm demonstrate significant superiority in terms of stability and generalization capacity. The Weighted

Sum variant reported risk levels closest to the ideal value of zero, keeping most of its results below a threshold of 0.04. By actively penalizing model complexity and reducing the number of parameters to only 28,952, this approach acts as an intrinsic regularizer that prevents overfitting. Although this variant shows some dispersion in its distribution, its central tendency reaffirms that a lightweight generic architecture is sufficient to effectively classify various binary pathologies.

Finally, the NSGA-II variant oriented towards the Best F1-score ranks as the most consistent strategy of the three evaluated, showing the most compact distribution and no significant outliers. With a median risk of 0.03 and an architecture of 227,636 parameters, this model achieves an optimal balance between learning capacity and diagnostic robustness. This analysis confirms that the transition from a single-objective to a multi-objective search not only drastically reduces the computational cost from 30 to 5 hours but also produces scientifically more reliable models for implementation in real clinical settings.

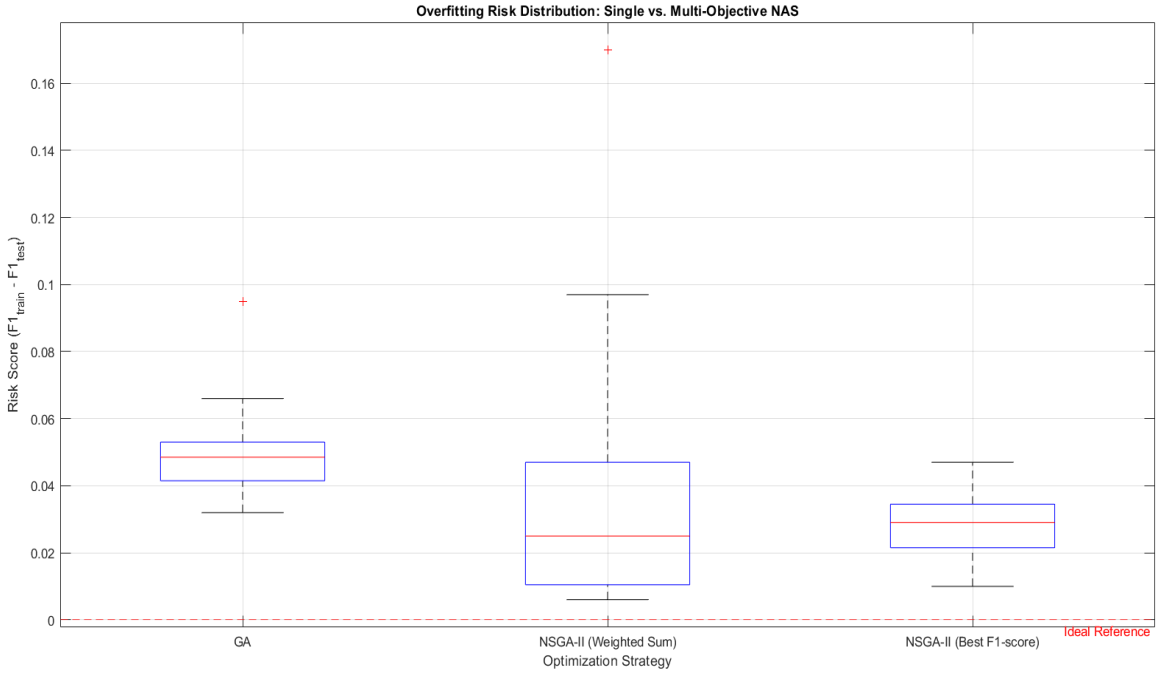


Figure 5. Box plot of overfitting risk distribution (F1 train - F1 test) for 20 runs. The multi-objective NSGA-II variants demonstrate a lower risk median compared to the single-objective GA, indicating superior generalization across the three binary medical databases. The Weighted Sum variant achieves the lowest overall risk, reinforcing the effectiveness of minimizing model complexity.

The normality analysis using the Shapiro-Wilk test in Table 6 reveals different behaviors depending on the search strategy used. For GA and NSGA-II (Best F1-score), p-values of 0.12 and 0.135 were obtained, respectively. As these are higher than the significance level alpha = 0.05, the null hypothesis is not rejected, confirming that their results follow a normal distribution. This indicates a high consistency in the ability of these algorithms to converge to solutions with similar performances in each run. In contrast, the NSGA-II (weighted sum) variant had a p-value of 0.029, indicating a non-normal distribution. This phenomenon is due to the nature of the trade-off imposed by the 50% weighted sum between F1-score efficiency and network size minimization. By giving equal weight to parameter reduction, the algorithm explores Pareto frontiers where some resulting architectures are extremely compact but show notable drops in the F1-score. This variability generates outliers toward the lower end of performance, which translates into a higher standard deviation (0.231) and breaks the symmetry of the normal distribution. Despite this lack of normality, the weighted sum approach remains valuable, as it achieves the lowest median overfitting risk of all techniques, as seen in the proximity of its median to the 'Ideal Reference' in the overfitting comparison.

Table 6. Statistical analysis of normality for F1-score results in the test set after 20 independent runs. The mean, standard deviation, and p-value obtained using the Shapiro-Wilk test are presented for each optimization strategy.

Algorithm	F1 average	Standard deviation	Shapiro-Wilk (p)	Distribution
GA	2.786	0.028	0.12	Normal
NSGA-II weighted sum	2.489	0.231	0.029	No normal
NSGA-II best F1-score	2.736	0.062	0.135	Normal

Tables 6, 7, and 8 compare the results obtained when using NSGA-II to find an architecture for all databases with literature. The authors proposed an architecture for a single database. The proposed NAS architecture using an NSGA-II is shown in Figure 6.

Table 6. Comparison of the best results in the literature for the stenosis database (Antczak & Liberadzki, 2018) with the proposed method on the test set.

Models	Accuracy	Precision	Recall	F1-Score
Features selection (Gil-Rios et al., 2024)	88%	NP	NP	NP
CNN stenosis (Antczak & Liberadzki, 2018)	90%	NP	NP	NP
Proposed NSGA-II weighted sum	92.2%	91.5%	93.2%	92.3%

*NP=not provided

Table 7. Comparison of the best results in the literature for the natural Stenosis608 database (Gil-Rios et al., 2024) with the proposed method in testing set.

Models	Accuracy	Precision	Recall	F1-Score
Feature selection with BUMDA (Gil-Rios et al., 2024)	92%	NP	NP	92%
Proposed NSGA-II weighted sum	95%	94%	96.1%	95%

Table 8. Comparison of the best results in the literature for the pneumonia database (Melendez et al., 2015) with the proposed method in testing set.

Models	Accuracy	Precision	Recall	F1-Score
CNN from scratch and data augmentation (Stephen et al., 2019)	93.73	NP	NP	NP
Lightweight deep learning architecture (Trivedi & Gupta, 2022)	97.09%	97%	98%	97%
CNN with weighted classifier (Hashmi et al., 2020)	98.43%	98.26%	99%	98.63%
Proposed NSGA-II weighted sum	95.9%	95%	96.7%	95.89%

The comparative analysis reveals that the proposed NSGA-II methodology provides a significant balance between classification performance and computational efficiency. While specific models in literature, such as Hashmi et al. (2020), report higher accuracy for pneumonia (98.43%) by utilizing weighted classifiers and pre-existing architectures, our generic model achieves a competitive F1-score of 0.9589 using only a fraction of the parameters. This indicates that the parameters in larger models may be redundant for binary medical classification tasks. Furthermore, the transition from a single-objective GA to a multi-objective NSGA-II reduced the execution time from 30 hours to approximately 5 hours, representing a 6x speedup in finding the optimal solution. This efficiency is rooted in the algorithm's ability to prioritize lightweight architectures on the Pareto front, which requires less computational power during the evaluation phase. Table 9 provides a critical analysis of the proposal with respect to the literature.

Table 9. Summary Table: Critical Analysis of the Proposal.

Aspect	Approaches in Literature	NAS Proposal (NSGA-II)	Critical Analysis
Specialization	Architectures designed for a single database	Generic architecture for multiple pathologies	Generalization reduces redesign effort and enables cross-diagnostics with a single structural configuration
Complexity	Models with millions of parameters or empirical design	Lightweight model with 28,952 parameters (weighted sum)	A 96% reduction in parameters compared to conventional GA facilitates deployment on limited hardware without significant loss of F1-score
Search Time	Extensive optimization processes (e.g., 30 hours with GA)	Efficient search in 5h 5m	The multi-objective approach accelerates convergence by propagating smaller models that are faster to train
Methodology	Use of transfer learning or manual data augmentation.	Training from scratch with automatic search	It demonstrates that the structure found by NAS is inherently robust for classifying 64x64 medical images without relying on pre-trained models

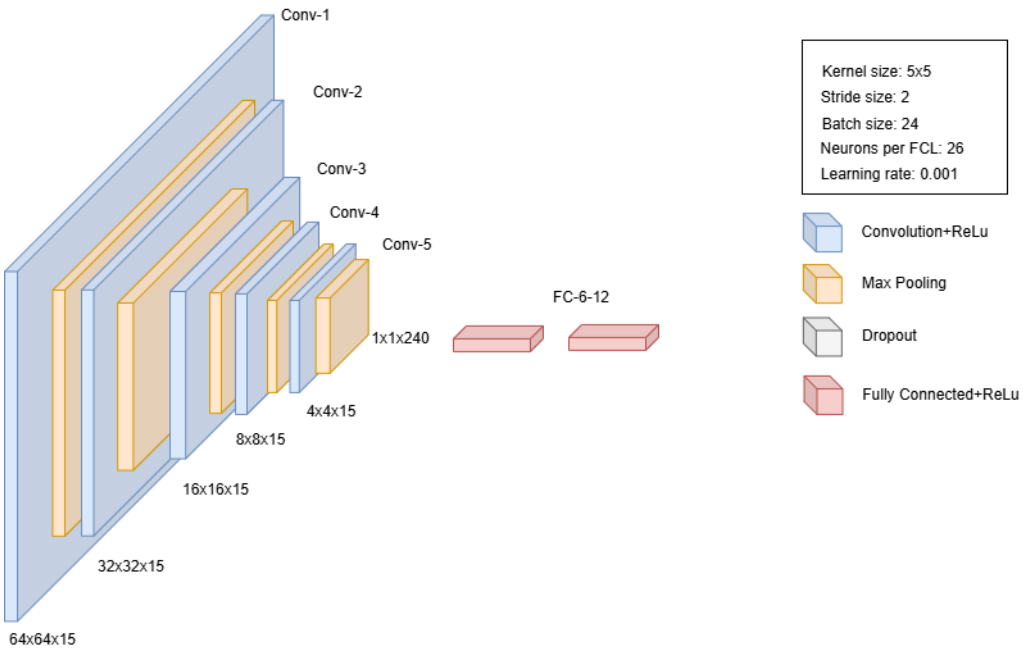


Figure 6. Architecture proposed by NAS using NSGA-II considering the weighted sum between F1-score and number of parameters for the classification of three binary databases.

While the proposed generic architecture demonstrates high robustness across multiple binary medical datasets, its application to other domains involves certain considerations. In multi-class classification problems, the current architectural search space may require expansion to account for increased decision boundary complexity. Furthermore, the 64x64 resolution used in this study, while efficient for the selected pathologies, might represent a limitation for medical conditions where diagnosis depends on high-frequency details. Finally, a potential risk of a generic approach is the trade-off in specificity; while it offers a versatile and lightweight solution, highly critical clinical environments might still require specialized models to capture domain-specific nuances that a generic configuration might overlook.

4 Conclusions

In this paper, a novel method for automatic neural architecture search based on a multi-objective NSGA-II algorithm for binary classification problems with a single CNN architecture was presented. The proposal helped to explore the search space with a complexity of $O(2^n)$ where n=29, finding architectures with optimal results and with fewer parameters. The hyperparameters were encoded in a binary array of size 29 and using NSGA-II, the best possible architecture was obtained for three binary medical databases. Furthermore, the proposed method was compared with multiple state-of-the-art methods, testing its efficiency in terms of accuracy and F1 score. In the experimental results, the proposed method with NSGA-II was run 20 times in training and achieved F1 scores 2.831 in weighted sum of the three databases on its best architecture with the test set, with individual results better than the literature for the most part. The state-of-the-art proposals, in addition to being empirically generated, attempted to solve database problems, such as imbalance, to achieve maximum efficiency, which required greater effort on the part of the authors. With the proposed method, this was not necessary, and the results were superior with reduced architectures. Unlike a GA that took 30 hours, 50 minutes, 1 second to find its solution, NSGA-II took only 5 hours, 5 minutes, 30 seconds, resulting in lower computational costs.

Ethical statement
Include an appropriate ethical statement in articles involving investigations on live subjects or where consent may be required.

References

Ait Amou, M., Xia, K., Kamhi, S., & Mouhafid, M. (2022). A novel MRI diagnosis method for brain tumor classification based on CNN and Bayesian optimization. *Healthcare, 10*(3), Article 494. https://doi.org/10.3390/healthcare10030494

Antczak, K., & Liberadzki, Ł. (2018). Stenosis detection with deep convolutional neural networks. *MATEC Web of Conferences, 210*, Article 04001. https://doi.org/10.1051/matecconf/201821004001

Atteia, G., Abdel Samee, N., El-Kenawy, E. S. M., & Ibrahim, A. (2022). CNN-hyperparameter optimization for diabetic maculopathy diagnosis in optical coherence tomography and fundus retinography. *Mathematics, 10*(18), Article 3274. https://doi.org/10.3390/math10183274

Borgli, R. J., Stensland, H. K., Riegler, M. A., & Halvorsen, P. (2019). Automatic hyperparameter optimization for transfer learning on medical image datasets using Bayesian optimization. In *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)* (pp. 1–6). IEEE. https://doi.org/10.1109/ISMICT.2019.8743779

Correa Flórez, C. A., Bolaños Ocampo, R. A., & Molina Cabrera, A. (2008). Algoritmo multiobjetivo NSGA-II aplicado al problema de la mochila [Multiobjective knapsack problem using the NSGA-II algorithm]. *Scientia et Technica, 14*(39), 206–211.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197. https://doi.org/10.1109/4235.996017

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research, 20*(55), 1–21. https://jmlr.org/papers/v20/18-598.html

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction* (2nd ed.). Wiley. https://doi.org/10.1002/9780470512517

Franco-Gaona, E., Avila-Garcia, M. S., & Cruz-Aceves, I. (2025). Automatic neural architecture search based on an estimation of distribution algorithm for binary classification of image databases. *Mathematics, 13*(4), Article 605. https://doi.org/10.3390/math13040605

Gaspar, A., Oliva, D., Cuevas, E., Zaldívar, D., Pérez, M., & Pajares, G. (2021). Hyperparameter optimization in a convolutional neural network using metaheuristic algorithms. In D. Oliva, E. H. Houssein, & S. Hinojosa (Eds.), *Metaheuristics in machine learning: Theory and applications* (Studies in Computational Intelligence, Vol. 967, pp. 37–59). Springer. https://doi.org/10.1007/978-3-030-70542-8_2

Gil-Rios, M. A., Cruz-Aceves, I., Hernandez-Aguirre, A., Hernandez-Gonzalez, M. A., & Solorio-Meza, S. E. (2024). Improving automatic coronary stenosis classification using a hybrid metaheuristic with diversity control. *Diagnostics, 14*(21), Article 2372. https://doi.org/10.3390/diagnostics14212372

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.

Hashmi, M. F., Katiyar, S., Keskar, A. G., Bokde, N. D., & Geem, Z. W. (2020). Efficient pneumonia detection in chest X-ray images using deep transfer learning. *Diagnostics, 10*(6), Article 417. https://doi.org/10.3390/diagnostics10060417

Heinze-Martin, G., Olmedo-Canchola, V. H., Bazán-Miranda, G., Bernard-Fuentes, N. A., & Guízar-Sánchez, D. P. (2018). Los médicos especialistas en México. *Gaceta Médica de México, 154*(3), 342–351. https://doi.org/10.24875/GMM.18003770

Jang, S., & Son, Y. (2019). Empirical evaluation of activation functions and kernel initializers on deep reinforcement learning. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 1140–1142). IEEE. https://doi.org/10.1109/ICTC46691.2019.8939854

Kang, J.-S., Kang, J., Kim, J.-J., Jeon, K.-W., Chung, H.-J., & Park, B.-H. (2023). Neural architecture search survey: A computer vision perspective. *Sensors, 23*(3), Article 1713. https://doi.org/10.3390/s23031713

Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). *A guide to convolutional neural networks for computer vision*. Morgan & Claypool. https://doi.org/10.2200/S00822ED1V01Y201712COV015

Kilichev, D., & Kim, W. (2023). Hyperparameter optimization for 1D-CNN-based network intrusion detection using GA and PSO. *Mathematics, 11*(17), Article 3724. https://doi.org/10.3390/math11173724

Krichen, M. (2023). Convolutional neural networks: A survey. *Computers, 12*(8), Article 151. https://doi.org/10.3390/computers12080151

Kuo, C.-H., Huang, E.-H., Chien, C.-H., & Hsu, C.-C. (2021). FPGA design of enhanced scale-invariant feature transform with finite-area parallel feature matching for stereo vision. *Electronics, 10*(14), Article 1632. https://doi.org/10.3390/electronics10141632

Lee, S., Kim, J., Kang, H., Kang, D.-Y., & Park, J. (2021). Genetic algorithm based deep learning neural network structure and hyperparameter optimization. *Applied Sciences, 11*(2), Article 744. https://doi.org/10.3390/app11020744

Li, S., & Ding, W. (2015). Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (pp. 730–734). IEEE. https://doi.org/10.1109/ACPR.2015.7486599

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems, 33*(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

Melendez, J., van Ginneken, B., Maduskar, P., Philipsen, R. H. H. M., Reither, K., Breuninger, M., Adetifa, I. M. O., Maane, R., Ayles, H., & Sánchez, C. I. (2015). A novel multiple-instance learning-based approach to computer-aided detection of tuberculosis on chest X-rays. *IEEE Transactions on Medical Imaging, 34*(1), 179–192. https://doi.org/10.1109/TMI.2014.2350539

Monteiro, T. G., Skourup, C., & Zhang, H. (2020). Optimizing CNN hyperparameters for mental fatigue assessment in demanding maritime operations. *IEEE Access, 8*, 40402–40412. https://doi.org/10.1109/ACCESS.2020.2976601

Mostafa, S. S., Mendonça, F., Ravelo-García, A. G., Juliá-Serdá, G., & Morgado-Dias, F. (2020). Multi-objective hyperparameter optimization of convolutional neural network for obstructive sleep apnea detection. *IEEE Access, 8*, 129586–129599. https://doi.org/10.1109/ACCESS.2020.3009149

Patel, C. I., Labana, D., Pandya, S., Modi, K., Ghayvat, H., & Awais, M. (2020). Histogram of oriented gradient-based fusion of features for human action recognition in action video sequences. *Sensors, 20*(24), Article 7299. https://doi.org/10.3390/s20247299

Probst, P., Boulesteix, A.-L., & Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research, 20*(53), 1–32. https://jmlr.org/papers/v20/18-444.html

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., & Wang, X. (2021). A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys, 54*(4), Article 76. https://doi.org/10.1145/3447582

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1409.1556

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015), Conference Track Proceedings*. https://arxiv.org/abs/1409.1556

Singh, P., Chaudhury, S., & Panigrahi, B. K. (2021). Hybrid MPSO-CNN: Multi-level particle swarm optimized hyperparameters of convolutional neural network. *Swarm and Evolutionary Computation, 63*, Article 100863. https://doi.org/10.1016/j.swevo.2021.100863

Stephen, O., Sain, M., Maduh, U. J., & Jeong, D.-U. (2019). An efficient deep learning approach to pneumonia classification in healthcare. *Journal of Healthcare Engineering, 2019*, Article 4180949. https://doi.org/10.1155/2019/4180949

Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. Wiley. https://doi.org/10.1002/9780470496916

Trivedi, M., & Gupta, A. (2022). A lightweight deep learning architecture for the automatic detection of pneumonia using chest X-ray images. *Multimedia Tools and Applications, 81*(4), 5515–5536. https://doi.org/10.1007/s11042-021-11807-x

Zhang, J., Li, Y., Tai, A., Wen, X., & Jiang, J. (2022). Motion video recognition in speeded-up robust features tracking. *Electronics, 11*(18), Article 2959. https://doi.org/10.3390/electronics11182959