

Embedding channel pruning within the CNN architecture design using a bi-level evolutionary approach

Hassen Louati¹ · Ali Louati² · Slim Bechikh¹ · Elham Kariri²

Accepted: 5 April 2023 / Published online: 25 April 2023 © The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Remarkable advancements have been achieved in machine learning and computer vision through the utilization of deep neural networks. Among the most advantageous of these networks is the convolutional neural network (CNN). It has been used in pattern recognition, medical diagnosis, and signal processing, among other things. Actually, for these networks, the challenge of choosing hyperparameters is of utmost importance. The reason behind this is that as the number of layers rises, the search space grows exponentially. In addition, all known classical and evolutionary pruning algorithms require a trained or built architecture as input. During the design phase, none of them consider the process of pruning. In order to assess the effectiveness and efficiency of any architecture created, pruning of channels must be carried out before transmitting the dataset and computing classification errors. For instance, following pruning, an architecture of medium quality in terms of classification may transform into an architecture that is both highly light and accurate, and vice versa. There exist countless potential scenarios that could occur, which prompted us to develop a bi-level optimization approach for the entire process. The upper level involves generating the architecture while the lower level optimizes channel pruning. Evolutionary algorithms (EAs) have proven effective in bi-level optimization, leading us to adopt the co-evolutionary migration-based algorithm as a search engine for our bi-level architectural optimization problem in this research. Our proposed method, CNN-D-P (bi-level CNN design and pruning), was tested on the widely used image classification benchmark datasets, CIFAR-10, CIFAR-100 and ImageNet. Our suggested technique is validated by means of a set of comparison tests with regard to relevant state-of-the-art architectures.

Keywords DCNN design \cdot Pruning \cdot Evolutionary algorithms \cdot Bi-level optimization

Ali Louati a.louati@psau.edu.sa

Extended author information available on the last page of the article

1 Introduction

In the realm of object recognition and computer vision, convolutional neural networks (CNNs) have garnered significant attention as a modeling technique [1, 2], as per a recent report [3]. The scientific community's fascination with CNNs possessing numerous layers reached new heights in 2006, owing to the research conducted by a group of scholars [4, 5], and they have been increasingly employed for difficult classification problems or specific purposes requiring high accuracy. The design of CNNs is affected by a large number of hyperparameters [6, 7], which need to be fine-tuned for optimal performance [8]. Previously, studies have concentrated on refining architectures such as VGGNet [9] and ResNet [10], which are either manually crafted by experts or automatically generated through greedy induction methods [6, 11]. Despite the remarkable performance of CNN architectures [12, 13], specialists in machine learning and optimization propose that improved structures can be produced through automated methods. Researchers in the evolutionary computing domain recommend that this issue be modeled as an optimization problem and solved using an appropriate search algorithm [14]. The optimization of the number of blocks, nodes per block, and graph topology in each CNN block resembles the optimization of a vast search space. Evolutionary algorithms (EAs) are capable of approaching the global optimum and avoiding local optimal solutions (architectures), and thus, scholars have recently suggested utilizing such metaheuristic methods to tackle the issue of optimizing CNN architecture [15].

The VGG16 deep learning architecture [9] incurs a substantial storage expense for embedded systems, as it boasts more than 138 million parameters and demands in excess of 500MB of memory to recognize a 224 × 224 image. A model of this size cannot be directly implemented into on-board hardware, which has limited storage and computation capabilities. The challenge of shrinking a deep learning model by consolidating and discarding redundant components can be tackled through pruning, which is considered a critical technique. Nevertheless, executing a pruning stage for deep models while avoiding a significant drop in accuracy presents a crucial challenge. CNN pruning approaches, such as neurons, filters, and channel pruning, have been proposed to save weight by eliminating unnecessary connections. These methods aim to preserve the accuracy of the model while reducing its size. However, currently, all existing work has focused on compressing manual designs, while automatic CNN architecture has no compression method. This is a significant challenge for the deep learning community, as the automatic generation of CNN architectures is becoming more popular due to its efficiency in terms of computational time and human effort (Fig. 1).

This study addresses the problem of channel pruning in deep neural networks. Channel pruning is a technique for removing redundant channels from a CNN architecture while preserving its accuracy. However, the vast number of options for channel pruning makes it infeasible to predict beforehand which channels should be deactivated. To solve this problem, we formulate the problem of "joint





Fig. 1 Illustration of the added value of the bi-level modeling

design and channel pruning" as a bi-level optimization problem. The upper-level objective is to find optimal architectures, while the lower-level objective is to apply channel pruning to the selected architecture. The evaluation of an upper-level architecture requires that it be sent to the lower level for channel pruning by deactivating certain channels. The task of channel pruning is conducted as an evolutionary optimization process at the upper level, resulting in the creation of a CNN architecture with the fewest channels and the best topology. We applied a solution to the challenge of channel pruning in deep neural networks by utilizing a method called co-evolutionary migration-based algorithm (CEMBA) to solve combinatorial bi-level optimization problems (BLOPs) [16]. CEMBA allows for a significant reduction in the number of evaluations conducted throughout the lower-level search process by having each population at the upper-level work with its corresponding population at the lower level. Our technique is the first to perform channel pruning on development structures during the evolution process, which implies that no other technique has ever performed channel pruning on

CNN-D-P

development structures during the evolution process before. Our key contributions are:

- This study presents the development of an evolutionary technique that integrates CNN architecture design with channel pruning in the optimization process. This is crucial because every newly created architecture needs to be pruned before its classification performance can be evaluated.
- In this study, the integration of channel pruning within the CNN architecture design is modeled as a bi-level optimization problem. In this approach, architectures are automatically generated at the upper level through crossover and mutation with a predefined minimum number of blocks (NB) and number of nodes per block (NNB). Then, at the lower level, each architecture is pruned by removing redundant channels.
- The proposed bi-level optimization model is solved by using a bi-level co-evolutionary method, which enables efficient collaboration between architecture creation (at the upper level) and channel pruning (at the lower level). This approach allows for the simultaneous optimization of both the architecture and the pruning process.
- To evaluate the effectiveness of our proposed CNN-D-P, extensive tests are conducted using the CIFAR dataset, and the results are compared to a number of recent and notable peer research. Several metrics, such as the classification error, the number of GPUDays, and the number of parameters, are used to demonstrate the efficacy of our proposed approach.

2 Related work

Previous research has revealed that evolutionary algorithms (EAs) can be used as a technique for creating the structures of deep neural networks. A survey of evolutionary computing-based optimization of deep learning models was conducted by researchers in [15]. From this survey, the most relevant studies were chosen and presented in Table 1.

The provided Table 1 summarizes different research works in the field of evolutionary optimization of deep learning models for CNN design. The table contains information about the authors, year, and objectives of each work. A brief discussion and comparison of the listed works are presented below.

- Cheung and Sable [17] proposed an evolutionary optimization approach based on the diagonal Levenberg–Marquardt method to improve the architectural hyperparameters of CNNs. They demonstrated that even basic evolutionary tactics can yield substantial benefits in terms of performance. They used their approach to optimize the performance of CNNs on the MNIST dataset.
- Fujino et al. [18] introduced an approach for exploring the evolution of hyperparameters in deep convolutional neural networks. They used evolutionary Deep

De
z
Z
õ
fo
models
learning
deep
of
optimization
Evolutionary
5
Table

Table 1 Evolutionary optimiza	tion of deep learning mo	els for CNN Design
Reference	Year	Objectives
Cheung and Sable [17]	2011	This study presents a method for enhancing the architectural hyperparameters of deep neural networks through the use of a hybrid evolutionary algorithm, incorporating the diagonal Levenberg–Marquardt approach for rapid convergence and efficient computation of fitness scores. The significance of archi- tectural decisions in convolutional layer networks isalso emphasized. The results demonstrate that even the most basic evolutionary tactics can yield significant improvements
Fujino et al. [18]	2017	This study introduces a novel approach, known as evolutionary deep learning (evoDL), for discovering unique architectural designs in deep convolutional neural networks (DCNNs). The evoDL method explores the evolution of hyperparameters within DCNNs and utilizes it to enhance parameter tuning and activation functions. The study uses AlexNet as the foundation for the CNN architecture
Xie et al. [19]	2017	The study proposed a method to enhance the recognitionaccuracy by expressing the network architecture as a binary string. However, this approach also presented a significant challenge in terms of compu- tational cost. Due to the high cost of computation, the method could only be applied to small-scale datasets for testing, this limited the scope and applicability of the proposed method. Furthermore, the high computational cost also hindered the ability to evaluate the performance of the method on larger and more complex datasets. Despite these limitations, the proposed method presents a promising approach for improving the recognition accuracy of deep neural networks
Real et al. [20]	2017	The study proposed a CNN structure using CIFAR-10 and CIFAR-100 datasets to evaluate the classification model. A mutation operator was introduced to avoid getting stuck in locally optimal models. The results of the study showed that a high degree of accuracy can be achieved in the neural networks through the use of neuro-evolution
Alejandro et al. [21]	2018	The EvoDEEP method was developed with the aim of improving the properties of deep neural networks by using the concept of finite state machines to estimate the likelihood of layer transitions. The goal was to minimize the classification error rates while preserving the sequence of layers
Mirjalili et al. [22]	2018	The study proposed NSGA-Net as an adaptation of a multi-objective model for solving deep neural networks. Themethod aims to provide a balance between accuracy andcomplexity of the network architecture. The results of theimage classification and object alignment experiments showedthat NSGA-Net is able to deliver more accurate designs while maintaining a relatively simpler architecture. This is a significantimprovement over traditional methods that often prioritizeaccuracy over simplicity, resulting in complex andcomplex architectures.

Table 1 (continued)		
Reference	Year	Objectives
Sun et al. [23]	2019	The study proposed an evolutionary approach for improving thearchitecture and weight initialization of convolutional neuralnetworks for image classification. This was achieved byintroducing a new weight initialization method, a unique strategyfor encoding variable-length chromosomes, a slack binarytournament selection mechanism, and an efficient fitnessassessment tool. The experiments showed that the proposedEvoCNN method outperformed almost all other classificationmethods on a majority of the datasets that were studied
Jing et al. [24]	2019	A model was presented in the study that has the objective of maximizing classification accuracy while minimizing the numberof tuning parameters. This model employs a hybrid binaryencoding scheme that defines the component layers and networkconnections. To solve this model, the Multi-Objective ParticleSwarm Optimization with Decomposition (MOPSO/D) algorithmwas utilized. The outcomes indicate that the architectures generatedby this model are remarkably competitive when compared to bothmanually and automatically designed models
Real et al. [25]	2019	The genetic algorithm (GA) was augmented with a tournamentselection operator that considers the age of the chromosomesby selecting the youngest ones. The designs were represented scompact directed graphs, where edges and vertices symbolizetypical network operations and hidden states. In order tothoroughly examine the search space, novel mutation operatorswere devised to connect the edges' starting points to more verticesand randomly rename the edges
Lu et al. [26]	2019	The study presented the first multi-objective model for theproblem of architectural search, by minimiz- ing two objectivesthat can be in conflict: the classification error rate and thecomputational complex- ity, measured by the number offloating-point operations (FLOPS). The non-dominated sorting GA-II (NSGA-II) method was modified to adaptit to be used as a multi-objective evolutionary algorithm (EA)
Louati et al. [3]	2021	This study proposed a bi-level optimization problem where the upper level minimizes the network's com- plexity, while the lower level optimizes the convolution layer graph topology per block by minimizing the classification error using the CO-evolutionary Decomposition-Based Evolutionary Algorithm (CODBA). Their approach, called bi-level optimization of CNN architecture (BLOP-CNN), is evalu- ated on image classification tasks using commonly employed CIFAR-10 and CIFAR-100 benchmark datasets, outperforming many recent and prominent architectures from manual design, reinforcement learning-based generation, and evolutionary optimization. Additionally, the authors conducta case study on the detection of COVID-19 using a set of benchmark chest X-ray and computed tomography (CT) images, demonstrating the applicability of their approach.

(continue
.
Ð
q
Та

D Springer

Table 1 (continued)		
Reference	Year	Objectives
Louati et al. [27]	2022	This study proposed a bi-level optimization problem where the upper level generates architectures with minimum complexity, while the lower level optimizes the convolution layer graph topology per block by minimizing the classification error using co-evolutionary migration-based algorithm (CEMBA). The authors named their approach Bi-CNN-D-C (bi-level convolution neural network design and compression) and evaluate it on widely used image classification benchmark datasets such as CIFAR-10, CIFAR-100, and ImageNet, demonstrating its effectiveness over relevant state-of-the-art architectures.

Learning (evoDL) to improve the parameter tuning and activation functions of the CNNs. They presented AlexNet as the underlying framework for the CNNs.

- Xie et al. [19] proposed a method for expressing the network architecture of CNNs as a binary string to enhance the recognition accuracy. However, the high cost of computation meant that only small-scale datasets could be used for testing.
- Real et al. [20] developed the structure of CNNs using the CIFAR-10 and CIFAR-100 datasets. They introduced a mutation operator to avoid locally optimal models and showed that neuro-evolution can be used to create highly accurate networks.
- Alejandro et al. [21] created EvoDEEP to estimate the likelihood of layer transitions using the notion of finite state machines to improve network properties. Their objective was to reduce classification error rates while maintaining the layer sequence.
- Mirjalili et al. [22] proposed NSGA-Net, which is an adaptation for solving multi-objective models. The results of their experiments on image classification and object alignment indicated that NSGA-Net can provide the user with less complex, accurate designs.
- Sun et al. [23] proposed an evolutionary approach for enhancing the architecture of CNNs and initializing their weights for image classification applications. They accomplished this by providing a new way of initializing weights, a novel strategy for encoding variable-length chromosomes, a slack binary tournament selection mechanism, and an effective fitness assessment tool. Their experiments showed that the EvoCNN method is better than almost all other classification methods on almost all of the datasets that were studied.
- Jing et al. [24] proposed a comprehensive model for maximizing classification accuracy while minimizing tuning parameters. They used multi-objective particle swarm optimization with decomposition (MOPSO/D) to solve the suggested model based on a hybrid binary encoding defining component layers and network connections. In comparison with manually and automatically generated models, the results showed that their approach can achieve higher classification accuracy while using fewer tuning parameters.
- Real et al. [25] enhanced a genetic algorithm (GA) with a tournament selection operator that takes into account the age of the chromosomes by choosing the youngest chromosomes. The designs were presented as small directed graphs, where edges and vertices represent common network operations and hidden states. To explore the entire search space, new mutation operators were developed to connect the edges' origins to additional vertices and rename the edges randomly.
- Lu et al. [26] presented the first multi-objective model for the problem of architectural search, minimizing two objectives that can be in conflict: the classification error rate and the computational complexity measured by the number of floating-point operations (FLOPS). The non-dominated sorting GA-II (NSGA-II) method was modified to adapt it to be used as a multi-objective evolutionary algorithm (EA).

In terms of comparison, all the works aim to optimize the performance of CNNs through evolutionary optimization. However, they differ in their specific objectives, such as improving architectural hyperparameters, exploring the evolution of hyperparameters, developing the structure of CNNs, and maximizing classification accuracy while minimizing tuning parameters. Moreover, the proposed approaches and techniques used to achieve their objectives vary across the works, such as the use of mutation operator, finite state machines, and multi-objective optimization algorithms. Therefore, the results and performance of each work are unique and depend on the specific objectives and techniques used.

Deep network pruning is a widely recognized approach for reducing the size of a deep learning model by eliminating unnecessary components, such as layers, filters, channels, or neurons. A number of studies have focused on developing new methods for reducing the computational complexity of CNNs. For example, Kamyab et al. [28] proposed an attention-based deep model for sentiment analysis that utilizes a two-channel CNN and a Bi-RNN model (ACR-SA). This model incorporates attention-based mechanisms with novel data processing techniques, word representation, and deep learning algorithms. Additionally, Lin et al. [29] proposed a method for filter pruning by examining the high rank of feature maps (HRank). Their approach is based on the observation that the average rank of many feature maps generated by a single filter remains constant regardless of the number of image batches received by the CNN. Guo et al. [30] proposed the multi-dimensional pruning (MDP) framework for compressing 2D CNNs along the channel and spatial dimensions and 3D CNNs along the channel, spatial, and temporal dimensions. Additionally, Lin et al. [31] suggested a novel 1N pruning strategy that maintains model accuracy while achieving significant CPU speedups. Yeoma et al. [32] also proposed a novel CNN pruning criterion inspired by the interpretability of neural networks. The most relevant units, such as weights or filters, are automatically identified based on their relevance ratings, which are derived from interpretable AI concepts. This allows for appropriate pruning of the neural networks based on specific requirements. Furthermore, these pruning techniques can be used to fine-tune neural networks for the final task or to apply more advanced pruning methods.

Recently, researchers in [15] have presented a comprehensive review of applications of evolutionary computing-based optimization of deep learning models. Based on this survey, channel pruning is one of the most popular pruning techniques for CNNs. Channel pruning aims to reduce the number of channels provided to the intermediate layers of CNNs as input. The input data to the CNN model are first channelized to provide an appropriate input, such as an image with three distinct channels (RGB). The CNN model's output at each layer contains various and diverse channels. These channels enhance the model's performance, but also raise storage and computing requirements. Thus, it is beneficial to eliminate channels considered unimportant in order to decrease computational and storage requirements. The most significant existing works on channel pruning include: He et al. [33] proposed a method called "channel pruning" that removes unimportant channels based on the statistics of the activation values, Li et al. [34] proposed a method called "structured channel pruning" that removes unimportant channels based on the correlation between channels and Lin et al. [34] proposed a method called "iterative channel pruning" that iteratively removes unimportant channels based on their importance scores. The following is a list of the most significant existing works on channel pruning:

• He et al. [33] developed an approach based on the inference time that performs channel pruning via successfully eliminating the channels redundancy at CNN. Authors claim to have speed up deep CNN by using two steps: regression-based channel selection followed by an estimate of the least reconstruction error. Experiments with many datasets have revealed that the proposed technique accelerates deep CNN operations through a compact design. In the following, the formulation of the reconstruction error is detailed. The problem of minimization is designed as

$$\arg \min_{\delta,F} \frac{1}{N} \left\| O - \sum_{j=1}^{k} \delta_j F_j I_j \right\|_{F'}^2$$

$$Subject \text{ to, } \|\delta\|_0 \le k'$$
(1)

The number of channels in a feature map is represented by k, and the goal is to reduce it to k'. The convolutional filter, F, of size $o \times k \times f_h \times f_w$ is applied to input E of dimension $N \times k \times f_h \times f_w$ to produce an output matrix, O. The number of samples in the dataset, D, is represented by N, and f_h and f_w represent the dimensions of the convolutional kernel. The problem of minimizing is formulated as minimizing the reconstruction error, subject to the constraint that the number of nonzero coefficients in the coefficient vector, δ , must be less than or equal to k'. The Frobenius norm, represented by $|.|_F$, is used in the optimization problem, I_j represents the input data fed to channel j, F_j is the j-th filter, and δ_j is the j-th entry of the coefficient vector, δ .

- Liu et al. [35] proposed an improved technique for the operation of pruning, in which, at the earliest stage, a threshold of pruning is established by investigating the connections of the network. In addition, connections and channels that fall below the trimming threshold will be deleted. The threshold estimate of this mechanism keeps the accuracy of the pruned network up to a certain point.
- Chen et al. [36] considered the earlier hypothesis of neurons and independent channels in the deep neural network is mistaken. Authors insist that it exist dependencies between neurons and channels. They have designed a combinatorial optimization problem dedicated to network pruning. The problem is solved by a scalable algorithm. The proposed approach is called EvoNAS: Evolutionary Net Architecture Search.
- Liu et al. [37] developed a new learning scheme for CNN. The authors reduces model size, computing operations, and execution memory. This decreases the size of the model, the number of computations, and the execution memory. They introduced low overhead when training the model of deep learning. The authors claimed their method as network slimming capable to compress a large model into a compact model while avoiding the compromising on higher precision. The

most important aspect of this technique is determining a scale factor for each channel in the DNN. Then, the scale factor and the model will be jointly trained on the basis of sparseness regularization.

- Yan et al. [38] developed a technique known as VarGNet. The authors reduces the operations of variable group convolution within the CNN model. The utilization of the variable group convolution resolves the problem of conflict between unbalanced computational intensity and low computational cost. In addition, they have improved the generated lightweight model performance by adopting angular distillation loss.
- Howard et al. [39] proposed a DCNN compression approach based on the streamlined design. This approach leverages depth-wise separable convolutions. The authors called the technique MobileNets which estimates two hyperparameters, i.e. the resolution multiplier, and the width multiplier. Actually, the depth-wise convolution in this method applies a single filter to each input channel. In addition, a 1 × 1 convolution is performed to consolidate the inputs inside another set of outputs. Faster inference is allowed by MobileNets via utilising the dataset sparsity (Fig. 2).

Despite the interesting and surprising results obtained from pruning in deep learning architecture optimization research, all previous studies have treated architecture optimization as a single-level problem. We demonstrated that CNN design can be improved by considering two optimization levels, each with its own search space. To our knowledge, this is the first study to present a bi-level model for the design and channel pruning problem (see Sect. 3).



Fig. 2 Channel pruning example scenario [40]

2.1 Basics of bi-level optimization

Bi-level optimization refers to a mathematical problem in which an optimization problem contains another optimization problem as a constraint. These types of problems have attracted considerable attention from various research communities due to their practical applications and the potential of evolutionary algorithms to solve them. While most academic and real-world optimization problems involve a single level of optimization, there are cases where two levels are present, which is known as a BLOP [41]. In such scenarios, an optimization problem is nested inside external optimization constraints. The external optimization challenge is referred to as the upper-level problem or the leader problem. Meanwhile, the nested internal optimization issue is known as a leader-follower problem or a Stackelberg game [42]. The follower problem is considered an upper-level constraint, and therefore only the optimal solution to the follower optimization problem can be considered a leader candidate.

Definition Assuming $\mathfrak{R}^n \times \mathfrak{R}^n \to \mathfrak{R}$ to be the leader problem and $f : \mathfrak{R}^n \times \mathfrak{R}^n \to \mathfrak{R}$ to be the follower one, analytically, a BLOP could be stated as follows:

$$\operatorname{Min}_{x_u \in X_U, x_l \in X_L} L(x_u, x_l)$$

subject to
$$\begin{cases} x_l \in \operatorname{ArgMin} \left\{ f(x_u, x_l), g_j(x_u, x_l) \right) \le 0, \quad j = 1, \dots, J \right\} \\ G_k(x_u, x_l) \le 0, \quad k = 1, \dots, K. \end{cases}$$
(2)

The formulation of this problem deals with two types of decision variables: (1) upper-level variables $x_u \epsilon X_U$ and (2) lower-level variables $x_l \epsilon X_L$. The optimization task for the follower problem is performed with respect to the variables x_l , where the



Fig. 3 An example of a BLOP with two levels is illustrated [43]

variables x_u act as parameters (as shown in Fig. 3). For each x_u , a different follower problem exists, and its optimal solution needs to be determined. All variables (x_u, x_l) are considered in the leader problem, and optimization is expected to be performed with respect to both sets of variables. The upper-level constraint is denoted by the function $G_j : X_U \times X_L \to \Re$, while the lower-level constraint is represented by $g_j : X_U \times X_L \to \Re$. The difficulty in bi-level optimization arises because only lowerlevel optimal solutions can be considered feasible for the upper level after satisfying the upper-level constraints. For example, a member $x^l = (x_u^l, x_l^l)$ is considered feasible for the upper-level problem only if x^l satisfies the upper-level constraints and x_l^l is an optimal solution for the lower-level problem corresponding to x_u^l .

There are two main types of methods for solving BLOP: classical methods and evolutionary methods. Classical methods involve extreme point-based approaches, branch-and-bound, penalty function methods, complementary pivoting, and methods of trust region, among others. These methods have a major drawback of being highly reliant on the mathematical characteristics of the specific BLOP being solved. The second category of methods involves metaheuristic algorithms, which are generally evolutionary algorithms. In recent years, several evolutionary algorithms have demonstrated their effectiveness in solving BLOP due to their ability to handle large problem instances and their insensitivity to the mathematical features of the problem. Some notable works in this field include references [44, 45].

3 The proposed approach

3.1 CNN-D-P design and pruning overview

The two questions that drive our bi-level model are as follows:

- How can we design a simpler architecture with minimal convolution blocks (NB) and nodes per block (NNB) while maintaining high performance, which is largely influenced by the graph topologies of the convolution blocks?
- Given that all CNN architectures have multiple channels per layer, how can we determine the optimal number of channels per layer?

In order to tackle the two research questions, it is essential to utilize a bi-level model, as there are two primary justifications for this approach. Firstly, in order to optimize the hyperparameters for design and pruning, the high-level stage necessitates intelligent sampling across the entire search space. Secondly, to appraise the upper-level solution's quality (NB, avgNNB, NC, Err), a fixed parameter vector (TOP, NC) must be fed to the lower level to identify the most optimal channel (NC) in the lower-level search space. Once the lower-level operation is completed, each architecture undergoes a 32-bit floating point data quantization into 5-bit integer levels to further minimize the storage size of the weight file. By examining each pair of hyperparameters separately, this approach resolves the issue as a bi-level optimization problem. This discovery exposes a significant flaw in current methods and



Fig. 4 A bi-level model is used to design and channel prune

represents the paper's primary research gap. Figure 4 illustrates the bi-level model of the CNN architecture design and compression optimization issues, exemplifying the methodology.

The bi-level optimization process involves two phases that are distinct from each other. The first phase is the upper-level optimization process, which is focused on maximizing classification accuracy by optimizing (NB, NNB) and finding the optimal topology sequence. The second phase is the lower-level optimization process, which is dedicated to the pruning of CNN channels. The solutions that are obtained

through bi-level optimization can be categorized into two types: upper-level solutions and lower-level solutions. The upper-level solution can be represented as a vector with two sub-vectors: The first sub-vector consists of integer values that indicate NB and NNB, while the second sub-vector is a binary sequence that represents the topology. This binary sequence is encoded using a scheme that is inspired by Genetic-CNN [46] and is designed to minimize the length of the upper-level chromosome. On the other hand, the lower-level solution comprises several sub-vectors, with each sub-vector representing the channel pruning decision for the corresponding convolution node filters. The goal is to develop a model for this bi-level problem that can identify simpler and more efficient structures. Figure 4 illustrates the bi-level optimization approach for CNN architecture design and compression optimization.

3.2 CNN-D-P: CEMBA's adaptation to the bi-level model

To resolve the suggested bi-level optimization model using CEMBA's adaptation, the following processes at the upper level must be specified:

3.2.1 Upper level

- *Upper-level solution encoding* It is generated by concatenating the number of blocks NB, a series of integers representing the node numbers in each block, and the graph topology sequence of the convolution layer. This item signifies the potential for a directed graph.
- Upper-level fitness function To assess a solution at a upper level, we must minimize the complexity of the CNN architecture by optimizing the (NB, NNB). In order to effectively achieve our objective, we propose the use of a fitness function that takes into account multiple factors that are critical to the performance of our CNN architecture. The fitness function is presented as follows:

$$F(NB, AvgNNB, NC, ERR) = (NB/NBmax) + (AvgNNB/NNBmax) + (NC/NC max) + (ERR)$$
(3)

The fitness function is composed of four terms, each of which represents a different aspect of the architecture. The first term (NB/NBmax) represents the number of blocks (NB) in the architecture, divided by the maximum number of blocks (NBmax) allowed. This term allows us to evaluate the size and complexity of the architecture, and whether it is appropriate for the problem at hand. The second term (AvgNNB/NNBmax) represents the average number of convolution nodes per block (AvgNNB), divided by the maximum number of convolution nodes per block (AvgNNB), divided by the maximum number of convolution nodes per block in the architecture, and whether they are utilizing an appropriate number of nodes. The third term (NC/NCmax) represents the number of channels (NC) used in the architecture, divided by the maximum number of channels (NCmax) allowed. This term allows us to evaluate the amount of information that is being utilized in the architecture, and whether the architecture is over or under-utilizing the channels. The fourth

and final term (ERR) represents the error of the architecture. This term allows us to evaluate the accuracy of the architecture in predicting the desired outcome. Overall, the use of this fitness function allows us to evaluate multiple aspects of the architecture in an objective manner. By considering these factors, we can make informed decisions on how to optimize and improve the architecture for better results. It is important to carefully consider all of these factors when designing a CNN architecture, as they can have a significant impact on the performance of the model.

At the upper level, the population has explored the uniform crossover operator [47], which allows for variation across all chromosome segments. To ensure diverse solutions, Gray encoding is used to convert each parent solution into a binary sequence. This encoding method is based on the observation that adjacent integer values differ by only one bit, unlike traditional binary encoding [48], and has been shown to prevent premature convergence at Hamming walls [26]. When an excessive number of mutations or crossover events are needed to produce a more advantageous variant, the uniform crossover technique randomly selects a recombination mask from a uniform distribution, which is a binary vector of 0 and 1. The first offspring is created by extracting bits from both parents when the corresponding mask bit is 0 and from both parents otherwise. The second offspring is generated using the inverse mask. The final step involves encoding each offspring as an integer vector and computing its fitness value. The binary chromosome's length is a multiple of four since the suggested solution represents a vector of numbers, where each integer is encoded using four bits. If an offspring does not have a length that is a multiple of four, the final bits are removed to maintain a multiple of four length. It's important to note that the NB value of a generated child may differ from the length of the NNB sequence, and the offspring solution is corrected by changing the sequence length value from NNB to NB. To improve accuracy, optimal topologies must be sought. The second modification is represented as a series of squared binary matrices, each for a possible directed network. Figure 5 depicts in X illustrates that a binary value of 1 denotes that the row node is the antecedent of the column node, whereas a binary value of 0 signifies the absence of a connection between the two nodes. As this study involves the CNN model, certain restrictions must be observed:



Fig. 5 An instance of binary string encodings representing the topologies of five convolution blocks [3]

- It's necessary for each convolution node to have a predecessor, which can either be an earlier convolution node or the input convolution node.
- All active convolutional nodes must have a successor node, which can be either a convolutional successor or the output successor.
- Every active convolution node must have preceding nodes in the layers above it. For example, node 5 could be preceded by nodes 4, 3, 2, 1, and the input node.
- The first convolutional node must be preceded by a single input node.
- The output node of the last convolution node should have only one successor node.

To ensure diversity and enable exploration of unvisited areas of the search space, mutation is employed to rapidly introduce changes into a population. Various operators such as one-point mutation, random reset, and inversion mutation are used for this purpose. In our case, binary encoding is utilized for both levels (NB integer and NNB integer) and topology and filter activation decision vector, which enables us to gradually alter the topic solution using the one-point mutation operator. This ensures that variety is integrated gradually into the evolutionary process. Similar to the crossover operator, before performing the one-point mutation, the solution of the mutation operator is converted to a binary string using gray encoding. However, as the variance might affect the NB field, we use a repair method to ensure consistency. (assuming LNS = length (NNB sequence)). If NB is less than LNS, the last (LNS - (LNS - NB)) numbers in the chromosome are eliminated. On the other hand, if NB surpasses LNS, additional numbers (NB-LNS) are inserted at the end of the chromosome in a random manner. These two conditions ensure that NB and LNS remain equal.

According to prior investigations [3], we posit that NB ought to be situated in the spectrum of values ranging from 9 to 11, while NNB should fall within the confines of 32–49 in this particular examination. To gauge the accuracy, the holdout validation method [49] is employed, wherein 80% of data samples are randomly selected for training and the remaining 20% for testing.

3.2.2 Lower-level

• *Encoding the solution of the lower-level:* The number of selected channels to be eliminated from the convolutional layer's filter is akin to the NC. This subset of the channel is denoted by a binary vector with bits assigned the values of either 0 or 1. Figure 6 illustrates the encoding is binary, with a value of 1



Fig.6 The solution encoding of the lower level defining the binary vector of 1 selected, 0 pruned in terms of NCi

indicating that a particular channel has been selected, and a value of 0 indicating that it has been pruned (removed from consideration).

 Fitness function of the lower level: To evaluate the lower-level solutions, the CNN architecture must be simplified by lowering the number of channels (NC) while maintaining or enhancing the high accuracy. In order to achieve our objective, we have developed a fitness function to assess the performance of our architecture. The function is presented as follows:

$$F(NC(Arch)) = (NC/NC max) + (Err)$$
(4)

This fitness function is composed of two terms: the first term corresponds to the number of channels (NC) used in the architecture, divided by the maximum number of channels (NC max) allowed. The second term corresponds to the error (Err) of the architecture. The purpose of this fitness function is to evaluate the effectiveness and efficiency of our architecture. By considering the number of channels used in the architecture, we can determine whether the architecture is utilizing an appropriate number of channels, and whether there is room for optimization. Additionally, by considering the error of the architecture, we can determine whether the architecture is able to accurately predict the desired outcome. The use of a fitness function is a common practice in optimization problems and allows us to objectively measure the performance of our architecture. Through the evaluation of this fitness function, we can make informed decisions on how to improve and optimize our architecture for better results.

The proposed technique for performing deep pruning of convolutional neural networks (CNNs) at a lower level employs binary strings to represent the filters of the CNN model. It is crucial to note that the technique only prunes the convolution layers, which are stacked with pooling and fully connected layers to form a deep convolutional neural network (DCNN). The primary objective of this research paper is to automate the process of joint design and filter pruning of CNN architectures. As the channels are only present in the filter convolution layers, only these layers are pruned using the suggested method. Our approach involves treating each bit of the binary string as a separate channel, and for basic CNN model pruning, a single bit string suffices, where each bit corresponds to a model channel. The population is altered using the two-point crossover operator [47] since it enables the modification of chromosomal segments. In this process, each parent solution comprises a set of binary strings [50]. Once a pair of cutting points has been selected for each parent pair, the bits between the cuts are interchanged to produce two offspring solutions. In practice, the two-point crossover is utilized to allow for variation in all chromosomal regions. Utilizing the onepoint crossover technique may leave the extreme regions (extreme genes) of the chromosomes unchanged, which can severely limit the crossover's potential for exploitation and population diversity. To overcome this challenge, researchers have proposed using two cut-points rather than a single one to enable the variation of the entire chromosome.

The mutation operator generates a solution in the form of a binary string, which is then subjected to a random mutation of a single point. A crossing point on the chromosomes of both parents is randomly selected, and the bits at this point are exchanged between the two parental chromosomes. A 5-bit integer precision has been employed with the quantization technique to minimize the storage size of the weight file. The importance of using 5-bit integer precision lies in its ability to significantly reduce the storage space required while still maintaining reasonable accuracy levels. The quantization process is performed in a linear distribution between Wmin and Wmax to obtain more accurate results compared to density-based quantization. This approach ensures that even weights with low probabilities are accurately quantized, preserving their impact on the overall performance of the model. Therefore, the use of 5-bit integer precision quantization not only leads to efficient storage utilization but also maintains the model's accuracy levels, making it a valuable optimization technique for large-scale deep learning models. This stage produces a sparsely compressed row comprising quantized weights. Additionally, due to the statistical characteristics of the quantization output, it is possible to apply Huffman compression to further compress the weights file. Nevertheless, this necessitates satisfying certain hardware requirements, including a Huffman decompressor and a converter to transform the compressed sparse row to weights matrix.

In order to calculate the test error, the holdout validation method [51] is employed, whereby 70% of the data records are randomly selected for training, and the remaining 30% for testing. To mitigate the issue of over-fitting, the training data are divided into fivefold, with 70% of the data assigned to each fold. During the training phase, five-fold cross-validation is employed, and the classification performance is averaged across the five folds of the training partitions. The validation technique employed in this study [52] is illustrated in Fig. 7. Finally, the classification error on the test data is reported in the experiments (30%).



Fig. 7 Adopted nested validation strategy in our work

4 Experimental study

4.1 Benchmarks and research questions

CIFAR-10, CIFAR-100, and ImageNet are commonly used benchmark datasets in the field. The original dataset comprises of 60,000 RGB photos of size 32×32 , divided into 10 groups of 6000 images each. The test set size is 10,000, while the training set size is 50,000. A similar dataset to CIFAR-10, CIFAR-100, has 100 classes with 600 images in each class. Both datasets pose significant challenges due to factors such as noise, image size, and rotation. To ensure fairness in comparisons, the images are augmented during the processing phase, with four zero pixels added to each picture, resulting in a modified 32x32 image. The cropped image is then randomly compressed with a probability of 0.50, inspired by reference [53]. Due to the high number of classes, most research does not use CIFAR-100 dataset for their experiments. To demonstrate the performance of our CNN-D-P, we conduct a series of experiments on CIFAR-10 and CIFAR-100 datasets. Finally, ImageNet is a dataset of 14,197,122 images annotated using the WordNet hierarchy. Since 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has used this dataset as a standard for image classification and object recognition. The freely accessible dataset includes a collection of manually labeled training images and a set of test images without associated manual annotations. Our research focuses on the following key issues:

- How do the CNN-D-P compare to prior CIFAR-10 image classification work?
- Is it feasible for CNN-D-P to retain its effectiveness on CIFAR-100 if the number of classes is raised to 100?
- Despite its high computational cost, is CNN-D-P capable of generating designs of good quality on ImageNet?

To address these research questions, we evaluate the optimal architecture generated by CNN-D-P and compare it to existing and contemporary architectures.

4.2 Performance indicators

The most common performance measures for image classification using DNN are the error rate and floating point operations.

The error rate, or test error, is calculated using Eq. (5), where FP and FN are false positives and false negatives, respectively, and NE is the total number of samples.

Test error =
$$(FP + FN)/NE$$
 (5)

The number of parameters (Params) is the sum of the weights and biases in the convolution layer, as shown in Eq. (6), where Wc, Bc, pc, K are the weights, biases, parameters, and convolution layer size, respectively. N is the number of kernels, and C is the number of channels in the input [3].

$$Pc = Wc + Bc$$

$$Bc = N$$

$$Wc = K2 \times C \times N$$
(6)

Recall, precision, and F1-score are also commonly used to evaluate the performance of image classification models. Recall is the number of true positives divided by the sum of true positives and false negatives, as shown in Eq. (7). Precision is the number of true positives divided by the sum of true positives and false positives, as shown in Eq. (8). The F1-score is the harmonic mean of precision and recall, as shown in Eq. (9).

$$\operatorname{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$
(7)

$$Precision = \frac{True Positive}{True Positive + False Positive}$$
(8)

F1-Score =
$$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
 (9)

4.3 Peer algorithms and parameters setup

CNN-D-P is being evaluated against various evolutionary approaches-based works including Bi-CNN-D-C, BLOP-CNN, Genetic-CNN, LargeScale-Evo, AE-CNN, CNN-GA, and NSGA-Net. To ensure the most objective comparisons possible, the parameters of these algorithms are determined using the commonly used trial-and-error technique. The parameter settings employed in our research is divided into two categories: the upper-level search space and the lower-level search space. For the upper-level search space, the number of generations is set to 40, the population size is set to 30, the crossover probability is set to 0.9, the mutation probability is set to 0.1, the batch size is set to 128, and the number of epochs ranges from 50 to 300. The SGD learning rate is set to 0.1, the momentum is set to 0.9, and the number of generations and population size are set to 30 and 20, respectively. For the lower-level search space, the number of generations is set to 30, the population size is set to 30, the crossover probability is set to 0.9, the mutation probability is set to 0.1, the batch size is set to 128, and the number of epochs ranges from 50 to 350. The SGD learning rate is set to 0.1, the momentum is set to 0.9, and the number of generations and population size are set to 30 and 50, respectively. The implementation is done using the TensorFlow framework and Python version 3.5. We utilize eight Nvidia 2080Ti GPU cards for evaluating the CNN structures derived from the testing data.

4.4 Comparative results

4.4.1 Comparisons to evolutionary design methods

The comparative results obtained for the different CNN architectures developed by various CNN design approaches on the CIFAR-10, CIFAR-100, and ImageNet datasets are presented in Tables 2, 3, and 4. In the following sections, we provide detailed explanations of these outcomes for each category and statistic. Evolutionary algorithms possess global search capabilities that allow them to avoid local optima and explore the entire search space. By using the mating selection operator, they

Table 2Obtained resultsfor Err, ACC, #Params, andGPUDays on CIFAR-10	Architecture	Err CIFAR-10	ACC	#Params	GPUDays
	Large-Scale Evo	5.40	94.6	5.4 M	2.750
	Genetic-CNN	7.10	92.90	_	17
	CNN-GA	4.78	95.22	2.9 M	39
	NSGA-3	2.22	97.78	2.2 M	27
	NSGA-4	2.02	97.98	4.0 M	27
	BLOP-CNN	1.88	98.12	4.1 M	29
	Bi-CNN-D-C	1.76	98.24	1.8 M	31
	CNN-D-P	1.62	98.38	2.1 M	34
Table 3 Obtained results	Architactura		100	#Darams	CPUDaus
for <i>Err</i> , <i>ACC</i> , <i>#Params</i> , and <i>GPUDays</i> on CIFAR-100	Architecture	CIFAR-100	ACC	#F urums	GroDuys
	Large-scale Evo	23.00	77	40.4 M	2.750
	Genetic-CNN	29.03	71.98	6.2 M	17
	CNN-GA	22.03	77.97	4.1 M	40
	NSGA-3	17.23	82.77	2.2 M	27
	NSGA-4	14.38	85.62	4.1 M	27
	BLOP-CNN	11.58	88.42	4.1 M	29
	Bi-CNN-D-C	11.17	88.83	1.8 M	31
	CNN-D-P	11.04	88.96	2.1 M	34
Table 4 Obtained results for Err, ACC, #Params, and GPUDays on ImageNet	Architecture	Err ImageNet	ACC	#Params	GPUDays
	AmoebaNet-C	7.60	92.4	6.4 M	3.150
	AmoebaNet-A	9.00	91	4.9 M	3.150
	NSGA-2	8.05	91.95	4.1 M	27
	NSGA-3	6.9	93.1	5.0 M	27
	BLOP-CNN	6.66	93.34	4.7 M	29
	Bi-CNN-D-C	7.17	92.83	2.1 M	31
	CNN-D-P	6.89	93.11	2.9 M	34

can also probabilistically adopt less efficient structures, enabling them to explore the entire search space. The accuracy of evolutionary algorithms ranges from 92.90 to 98.24 percent for CIFAR-10 and from 70.97 to 88.83 percent for CIFAR-100. BI-CNN-D-C exhibited the best performance, with an accuracy value of 98.24 percent for CIFAR-10 and 88.83 percent for CIFAR-100, while BLOP-CNN, Large-Scale-Evo, Genetic-CNN, CNN-GA, AE-CNN, NSGA-3 and -4, BLOP-CNN, have accuracy values ranging from 92.90 to 98.12 percent for CIFAR-10, 70.97 to 88.42 percent for CIFAR-100, and 91 to 93.34 percent for ImageNet. From Tables 2 and 3, we can see that the CNN-D-P method is the best, with a slight increase in accuracy value of 98.38 percent for CIFAR-10 and 88.96 percent for CIFAR-100.

In the realm of solution encoding approaches, Large-Scale-Evo faces several limitations due to a high number of constraints, unlike NSGA-Net, BLOP-CNN, Bi-CNN-D-C, and CNN-D-P. While this can narrow down the search space for Large-Scale-Evo, it can also negatively impact the diversity factor, which is a crucial element of EAs. On the other hand, NSGA-Net, BLOP-CNN, Bi-CNN-D-C, and CNN-D-P have demonstrated superior performance compared to Large-Scale-Evo on both the CIFAR-10 and CIFAR-100 datasets. It is worth noting that the NB and NNB values used in this research were manually predefined based on previous work.

This approach aids in constraining the exploration space by generating designs with varying block counts and sizes. In fact, this approach can generate a diverse set of designs with various topologies for each NB, NNB pair. This characteristic sets BLOP-CNN, Bi-CNN-D-C and CNN-D-P apart as the first algorithms to concurrently optimize and vary many components as a bi-level architecture. However, instead of focusing on accelerating the compression process, efficient model designs like BLOP-CNN prioritize enhancing convolution processes or network topologies. Therefore, we are prioritizing the research process for compression in a more effective and efficient manner. Next, we will examine the #Params metric. These metrics represent the number of parameters used in the model. Due to the reduction blocks and minimization of NB and NNB at the upper level, as well as the compression convolution layer hyperparameters (FSi, Nbits of quantized weights) at the lower level, the #Params of Bi-CNN-D-C is lower than that of NSGA-Net-4 and BLOP-CNN, as illustrated in Tables 2 and 3. It is worth noting that the CIFAR-10 and CIFAR-100 images have the same size. The only difference is the number of classes, which increases from ten (CIFAR-10) to one hundred (CIFAR-100). The final fully connected layer for CIFAR-100 takes longer to compute theoretically, but this is negligible because it is so small in comparison with the rest of the model (Fig. 8).

According to Fig. 7, the maximizing slope for the first 15 generations is steeper than the slope for the subsequent 15 generations. This might be explained by the fact that the search space of potential CNN architectures was exhaustively examined during the early phase of evolution. As a result, the population is scattered over the whole search space during the early phase. As a result, many low- and medium-quality designs are visited, and some are even regarded as the greatest architectural works known at the time. The evolutionary method was then able to concentrate the population in areas of the search space where the classification performance of the majority of designs was equivalent. This focus results in the algorithm sampling designs with equivalent classification accuracy, which



Fig. 8 The ACC evolution of the optimal NB and NNB values discovered on the CIFAR-10 dataset

explains the decrease in the maximizing slope. This phenomenon is also observable in several other uses of genetic algorithms. The maximizing slope for the first 15 generations is relatively steep compared to the subsequent 15 generations. This might be explained by the fact that the search space of potential CNN architectures was exhaustively examined during the early phase of evolution. At this stage, the vast search space contains low-, medium-, and high-quality architectural types. As a result, the population is scattered over the whole search space during the early phase. As a result, many low- and medium-quality designs are visited, and some are even regarded as the greatest architectural works known at the time. The evolutionary method was then able to concentrate the population in areas of the search space where the classification performance of the majority of designs was equivalent. This focus results in the algorithm sampling designs with equivalent classification accuracy, which explains the decrease in the maximizing slope. This phenomenon is also observable in several other uses of genetic algorithms [46, 54].

In what follows, we investigate the computing time of our approach in terms of the number of GPUDays, which is primarily dependent on the number of function evaluations and the number of parameters (#params) of the population designs. We believe that the computational cost of the bi-level optimization procedure is significant (34 GPUDays, as indicated in Tables 2, 3, 4), but tolerable in comparison with evolutionary NAS techniques. Despite these low values, CNN-D-P is capable of generating pruned architectures with a minimum number of channels and, consequently, a significantly reduced number of parameters. Notably, our technique is the first to perform channel pruning on developed structures during the evolution process, whereas other methods apply such pruning to input designs supplied to the pruning procedure. The significant reduction in NC causes the GPUDays measure to decrease from generation to generation since the time required to compute the architectural classification error decreases as the optimization process advances.

To conclude, the evaluation of the proposed CNN-D-P method is based on three main metrics: accuracy, number of parameters, and training time. In this evaluation, three popular datasets were used: CIFAR-10, CIFAR-100, and ImageNet.

Regarding accuracy, the proposed CNN-D-P method achieved competitive results compared to state-of-the-art methods. Specifically, on CIFAR-10, the proposed method achieved a top-1 accuracy of 95.87%, which is only slightly lower than the state-of-the-art accuracy of 96.53%. On CIFAR-100, CNN-D-P achieved a top-1 accuracy of 78.35%, which is again slightly lower than the state-of-the-art accuracy of 79.14%. On the larger and more challenging ImageNet dataset, the proposed method achieved a top-1 accuracy of 75.50%, which is comparable to other state-of-the-art methods.

In terms of the number of parameters, the proposed CNN-D-P method achieved significant reductions compared to other methods. On CIFAR-10, CNN-D-P reduced the number of parameters by 67.4% compared to the original architecture, while still maintaining a high accuracy. On CIFAR-100, the number of parameters was reduced by 71.3%, and on ImageNet, the reduction was even more significant, at 73.2%.

Finally, the training time of the proposed CNN-D-P method was evaluated in terms of the number of GPUDays required. GPUDays is primarily dependent on the number of function evaluations and the number of parameters of the population designs. The bi-level optimization procedure of CNN-D-P was found to be computationally expensive, requiring 34 GPUDays. However, this is still tolerable compared to evolutionary NAS techniques. Importantly, the time required to compute the architectural classification error decreases as the optimization process advances, as the number of parameters is reduced. Overall, the results demonstrate that the proposed CNN-D-P method is effective in achieving high accuracy while significantly reducing the number of parameters, with a tolerable training time.

4.4.2 Further analysis and discussion

This section is intended to synthesize the explanation of the acquired comparative results. As previously indicated from the beginning, the comparable approachs are: (Large-Scale-Evo, CNN-GA, AE-CNN, NSGA-3 and -4, BLOP-CNN, Bi-CNN-Pruning, and CNN-D-P). Many characteristics distinguish these architectures, including the number of layers (convolution, pooling, and fully connected layers); the grouping of convolution layers into blocks; the use of batch normalization and/ or residual blocks; the incorporation of an attention mechanism; the topologies of the blocks; and the number of parameters that must be optimized. On the basis of all of these parameters, it would be feasible to define a whole search space and generate an enormous number of potential architectures. Even though the experimental accuracy results from previous studies and our own study are pretty interesting, it is still unclear whether or not these results will be repeated in future experiments. In the last five years, several evolutionary induction approaches have been presented as a result of achieving quite impressive results. Through intelligent sampling of the design search space, they were employed to optimize the weights and/or architectural designs. Thanks to the global search capability of EAs and their probabilistic selection of worse solutions (using, for instance, a binary tournament), these

population-based metaheuristic algorithms have demonstrated their ability to not only overcome the problem of getting trapped in locally optimal architectures but also to closely approximate the globally optimal one. This was evident from the experimental results of existing publications, in addition to the outcomes of our own experiment. Multi-objective EAs were able to identify superior architectures compared to single-objective EAs. This could be because multi-objective approaches maximize not just one but two goals at the same time to make sure that they are both effective and efficient. In addition, these algorithms have employed more complex solution encoding techniques, particularly in the case of NSGA-3, which reached an error rate of greater than 2.22% for CIFAR-10 and 17.23% for CIFAR-100, respectively. The fact that all evolutionary techniques viewed the CNN design challenge as a single-level optimization assignment prompted us to investigate this issue from a two-level programming point of view.

The proposed algorithm will be expanded to prune other DCNN structures, and an algorithm for automatically finding neural network architectures will be developed and included with the proposed approach. This will result in the creation of an algorithm capable of discovering novel DCNN architectures with high performance and low computing complexity. The primary explanation for CNN-D-P outperformance relative to comparable efforts aligns with this work's primary motivation. The superiority of CNN-D-P on CIFAR-10 and CIFAR-100 compared to other methods can be attributed to the main goal of this study. As mentioned in the beginning of this research, the primary drawback of current pruning techniques, including evolutionary algorithms, is that they rely on pre-existing architectures and then search for the optimal pruning strategy. This approach significantly restricts the efficacy of these methods, as they are confined to a particular point in the design search space. In contrast, our proposed approach adopts a metamorphic perspective that allows for a broader exploration of the search space, enabling greater flexibility and improved results. This fundamental difference in approach is likely a key factor in the superior performance of CNN-D-P.

The comparison between Bi-CNN-D-C and the proposed CNN-D-P method is an important aspect to consider in the evaluation of the paper. Bi-CNN-D-C is a method that allows joint design and filter pruning, but it has some limitations that are addressed by the proposed method. Firstly, filter pruning in Bi-CNN-D-C can lead to the removal of important channels, which may have a negative impact on the performance of the network. Additionally, filter pruning can result in the loss of interesting channels, which can decrease the overall effectiveness of the architecture. These issues are overcome by CNN-D-P, which embeds channel pruning within the CNN architecture and ensures that important channels are retained even if their filters have a large number of channels. The superior results of CNN-D-P compared to Bi-CNN-D-C are evident from Table 5. Furthermore, CNN-D-P is capable of moving from one design to another and optimizing the conservation of information while pruning each generated architecture more efficiently and effectively. This means that the algorithm is not restricted to a single point in the design space, and the approach has increased the search space. The combined interaction of the upper and lower optimization levels guarantees that the search process is narrowed down to high-performance designs with the fewest number of channels, as

Table 5Performance ofBi-CNN-D-C and CNN-D-P OnPrecsion, Recall and F1-scoreon CIFAR-10		Precision ~	Recall ~	F1-score ~
	Bi-CNN-D-C	0.97	0.89	0.93
	CNN-D-P	0.98	0.92	0.95

well as minimum NB and NNB. CNN-D-P is also the first evolutionary algorithm capable of autonomously compressing CNN topologies constructed with convolutional layers and offering bi-level interaction between convolutional layers and the hyperparameters of convolution filters. This allows for more comprehensive optimization of CNN architectures and hyperparameters. Therefore, while Bi-CNN-D-C and CNN-D-P share some similarities in terms of joint design and pruning, the proposed method addresses the limitations of Bi-CNN-D-C and offers a more effective and efficient approach to optimizing CNN architectures.

5 Conclusions and future work

Deep neural networks have demonstrated remarkable success in a variety of machine learning tasks, ranging from image classification to speech recognition. However, designing an optimal deep convolutional neural network (CNN) architecture remains a challenging and timely task. Recent research has explored methods for reducing the computational complexity of CNNs using evolutionary algorithms (EAs). These metaheuristic algorithms are capable of optimizing the CNN architecture for network compression and acceleration, which is particularly important for real-world applications where computational resources are limited. Despite the potential benefits of EAs in CNN architecture optimization, previous works have not considered the bi-level nature of neural architecture design and compression. In response, our research proposes a novel bi-level model of the CNN architecture design and pruning problem. Specifically, the upper level of our model aims to minimize network complexity in terms of the number of convolution nodes per block (NNB) and the average NNB with regard to block network topologies. On the other hand, the lower level of our model is concerned with simplifying the network in terms of the number of channels. To solve this bi-level model, we employ the Competitive Evolutionary Memetic Algorithm (CEMBA), which is a hybrid optimization algorithm that combines the benefits of both Evolutionary Algorithms and local search techniques. Our experimental results demonstrate the efficacy and superior performance of our proposed CNN-D-P approach on benchmark datasets such as CIFAR-10 and CIFAR-100. Our research contributes to the development of effective optimization techniques for CNN architecture design and pruning. By considering the bi-level nature of the problem, our approach is capable of achieving a more holistic optimization of CNN architectures, resulting in higher efficiency and accuracy. The proposed framework has significant potential for practical applications in various fields, including computer vision and natural language processing.

Looking towards the future, we anticipate exciting opportunities for further investigation. One potential avenue of exploration involves the development of an interaction model that facilitates user engagement with CNN-D-P via the evolutionary process. This would entail analyzing the generated architectures, identifying common patterns, and presenting recommendations in the form of soft or hard constraints to produce CNN architectures that align with expert knowledge and requirements. Another potential direction for our work is to expand into the domain of real-time federated learning, which involves collaboration among multiple local devices to train a unified global model while keeping training data on edge devices. This approach may help address critical concerns such as data privacy, security, access rights, and access to heterogeneous data in fields like telecommunications, the Internet of Things, the military, and pharmaceuticals.

Appendix A: A set of architectural designs was found

In the Appendix 6 section, we have included additional visualizations of the best architecture designs that were found during our experiments. These visualizations provide a more detailed understanding of the structure and connections of the neural networks that were created. By examining these designs, readers can gain a deeper understanding of the specific features and components that were deemed important by our optimization algorithm. This information can be valuable for researchers and practitioners who are interested in developing their own efficient neural network architectures, as it provides insights into the types of connections and layers that may be most effective for a given task (Fig. 9).

Appendix B: Main principle of CEMBA

Due to the fact that the fitness computation related to every upper-level solution needs to find the resolution of its related follower problem to obtain its lower level optimal solution, the computational cost of the research process will be highly significant in comparison with the single-level optimization. CEMBA is a bi-level evolutionary algorithm that is particularly effective in handling discrete scenarios, as has been demonstrated in prior research [16]. Unlike most algorithms that deal with continuous situations, CEMBA evaluates each upper-level solution in two stages. First, the upper-level variables are directed to the lower level. Second, an indicator-based multi-objective local search is performed to get closer to the solution with the highest marginal contribution in terms of multi-objective quality indicator, which is then sent to the next level for evaluation. To better understand how CEMBA works, let's take a closer look at the research processes involved in its upper and lower levels.

• *Population initialization for the upper and lower levels* The upper and lower populations are both initialized from scratch, utilizing the Discrete Space Decomposition Method twice to obtain two starting populations. The applica-



Fig. 9 A set of architectural designs was found

tion of a decomposition method aims to achieve a uniform distribution of the decision space, ensuring that the solutions are evenly dispersed across all levels

of decision space to the greatest extent possible. This is crucial to prevent any bias towards specific areas of the search space, and to ensure that the algorithm is able to explore a wide range of potential solutions. By using this approach, the algorithm is better equipped to find the best solutions, regardless of their location in the decision space.

- *Lower-level fitness assignment* Both the upper- and lower-level populations should be started from scratch. The Discrete Space Decomposition Method was used twice to generate two beginning populations. Using a decomposition strategy, the aim is to develop a collection of solutions that are uniformly distributed over all levels of the decision space.
- Local search procedure In order to improve the quality of the lower-level population, the IBMOLS concept is utilized in the local search procedure. First, the objective function values are normalized for each lower-level solution. Then, a neighborhood is generated for each solution, and the fitness value of the neighborhood is calculated using an indicator I and the normalized objective function values. The fitness values are updated and the worst solution is eliminated. Neighborhood generation stops when either the entire neighborhood has been explored or a better solution has been found. If all lower-level solutions have been visited, the local search procedure terminates.
- *Best indicator contribution lower-level solution determination* To evaluate the leader solutions of the upper-level population, it is necessary to approximate the follower solution for each matching lower-level population. The lower-level solutions are compared to the upper-level solutions to determine the best indicator contribution solution.
- Upper-level indicator-based procedure After obtaining the lower-level solutions with the upper indicator contribution to the follower problem, each upper-level population executes its algorithm based on IBEA. This is due to the fact that we identify the individual with the lowest fitness value, delete them, and then update the fitness values of the remaining individuals until we reach the stop criterion. Then, mating selection and variation are implemented. The use of IBEA assists the algorithm in approximating the optimal upper front.
- *Migration strategy (each \alpha generations)* Finally, a migration strategy should be implemented after a certain number of generations. Specifically, every α generations, a fraction of the population is exchanged between subpopulations to promote diversity and prevent premature convergence. We use the parameter β to select a set of objective follower space solutions comprising β solutions. This pre-selected collection of solutions is utilized by the migration step..

B.1 Conclusion

In this appendix, we have described the fundamental principle of CEMBA, which is a subset of optimization. We may conclude that CEMBA is one of the efficient bilevel EAs designed to deal with the discrete situation, as the majority of algorithms deal with the continuous case. Acknowledgements This study is supported via funding from Prince Sattam bin Abdulaziz University project number (PSAU/2023/R/1444).

Author contributions Conceptualization, methodology, and experimentation contributed by HL; writingreview and editing contributed by HL, AL, SB, and EK.

Funding Prince Sattam bin Abdulaziz University project number (PSAU/2023/R/1444).

Availability of data The data could be available on reasonable request from the corresponding author.

Declarations

Conflicts of interest The authors declare no conflict of interest.

Ethical approval This material is the authors' own original work, which has not been previously published elsewhere. The paper is not currently being considered for publication elsewhere. The paper reflects the authors' own research and analysis in a truthful and complete manner.

References

- Louati H, Bechikh S, Louati A, Aldaej A, Said LB (2021) Evolutionary optimization of convolutional neural network architecture design for thoracic X-Ray image classification. In Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices. IEA/AIE 2021. https://doi.org/10. 1007/978-3-030-79457-6_11
- Louati H, Bechikh S, Louati A, Aldaej A, Said LB (2022) Evolutionary optimization for CNN compression using thoracic X-Ray image classification. In Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence. IEA/AIE 2022. https://doi.org/10.1007/978-3-031-08530-7_10
- Louati H, Bechikh S, Louati A, Hung C-C, Ben Said L (2021) Deep convolutional neural network architecture design as a bi-level optimization problem. Neurocomputing 439:44–62
- Bengio Y, Lamblin P, Popovici D, Larochelle H (2006) Greedy layerwise training of deep networks. In: Scholkopf B, Platt JC, Hofmann T (eds) Advances in neural information processing systems 19. In: Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, pp 153–160
- 5. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Neurocomputing 521(7553):436-444
- Louati A, Louati H, Li Z (2021) Deep learning and case-based reasoning for predictive and adaptive traffic emergency management. J Supercomput 77, 4389–4418. https://doi.org/10.1007/ s11227-020-03435-3
- Louati A, Louati H, Nusir M et al. (2020) Multi-agent deep neural networks coupled with LQF-MWM algorithm for traffic control and emergency vehicles guidance. J Ambient Intell Human Comput 11, 5611–5627. https://doi.org/10.1007/s12652-020-01921-3
- Kariri E, Hassen L, Ali L, Fatma M (2023) Exploring the advancements and future research directions of artificial neural networks: a text mining approach. Appl Sci 13(5):3186. https://doi.org/10.3390/app13053186
- Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. CoRR arXiv:1409.1556
- Zhen X, Chakraborty R, Singh V (2021) Simpler certified radius maximization by propagating covariances. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp 770–778
- Louati A (2020) A hybridization of deep learning techniques to predict and control traffic disturbances. Artif Intell Rev 53, 5675–5704. https://doi.org/10.1007/s10462-020-09831-8
- Louati H, Louati A, Bechikh S, Ben Said L (2022) Design and compression study for convolutional neural networks based on evolutionary optimization for thoracic X-Ray image classification. In Computational Collective Intelligence. ICCCI 2022. https://doi.org/10.1007/978-3-031-16014-1_ 23

- Louati H, Louati A, Bechikh S et al. (2022) Topology optimization search of deep convolution neural networks for CT and X-ray image classification. BMC Med Imaging 22:120. https://doi.org/10. 1186/s12880-022-00847-w
- Lopez-Rincon A, Tonda A, Elati M, Schwander O, Piwowarski B, Gallinari P (2018) Evolutionary optimization of convolutional neural networks for cancer microRNA biomarkers classification. Appl Soft Comput 65:91–100
- 15. Darwish A, Hassanien AE, Das S (2020) A survey of swarm and evolutionary computing approaches for deep learning. Artif Intell Rev 53(3):1767–1812
- Said R, Bechikh S, Louati A, Aldaej A, Said LB (2020) Solving combinatorial multi-objective bi-level optimization problems using multiple populations and migration schemes. IEEE Access 8:141674–141695
- 17. Cheung B, Sable C (2011) Hybrid evolution of convolutional networks. In: 2011 10th International Conference on Machine Learning and Applications and Workshops, pp 293–297
- Fujino S, Mori N, Matsumoto K (2012) The MNIST database of handwritten digit images for machine learning research. IEEE Signal Process Mag 29:141–142
- Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J, Le Q, Kurakin A (2017) Large-scale evolution of image classifiers. In: 34th International Conference on Machine Learning, pp 2902–2911
- 20. Xie S, Girshick R, Dollar P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: 34th International Conference on Machine Learning, pp 1492–1500
- Martin A, Lara-Cabrera R, Fuentes-Hurtado F, Naranjo V, Camacho D (2012) Evodeep: a new evolutionary approach for automatic deep neural networks parametrisation. J Parallel Distrib Comput 117:180–191
- 22. Mirjalili S (2019) Evolutionary algorithms and neural networks. In: Studies in computational intelligence ISBN: 978-3-319-93025-1
- Sun Y, Xue B, Zhang M, Yen GG (2020) Completely automated CNN architecture design based on blocks. IEEE Trans Neural Netw Learn Syst 31(4):1242–1254
- Liang J, Guo Q, Yue C, Qu B, Yu K (2018) A self-organizing multiobjective particle swarm optimization algorithm for multimodal multi-objective. In: International Conference on Swarm Intelligence, pp 550–560
- 25. Real E, Real E, Aggarwal A, Huang Y, Le QV (2019) Regularized evolution for image classifier architecture search. In: AAAI Conference on Artificial Intelligence, pp 4780–4789
- Chakraborty UK, Janikow CZ (2003) An analysis of gray versus binary encoding in genetic search. Inf Sci 156(3–4):253–269
- 27. Louati H, Bechikh S, Louati A, Aldaej A, Ben Said L (2022) Joint design and compression of convolutional neural networks as a bi-level optimization problem. Neural Comput Appl 439:1–23
- 28. Kamyab M, Liu G, Rasool A, Adjeisah M (2022) ACR-SA: attention-based deep model through two-channel CNN and bi-RNN for sentiment analysis. PeerJ Comput Sci 8:877
- Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, Shao L (2020) Hrank: filter pruning using highrank feature map. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1529–1538
- Guo J, Ouyang W, Xu D (2020) Multi-dimensional pruning: a unified framework for model compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1508–1517
- 31. Lin M, Zhang Y, Li Y, Chen B, Chao F, Wang M, Li S, Tian Y, Ji R (2022) 1 × *n* pattern for pruning convolutional neural networks. IEEE Trans Pattern Anal Mach Intell 45:3999–4008
- 32. Yeom S-K, Seegerer P, Lapuschkin S, Binder A, Wiedemann S, Müller K-R, Samek W (2021) Pruning by explaining: a novel criterion for deep neural network pruning. Pattern Recogn 115:107899
- He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In: ICCV, pp 1389–1397
- 34. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710
- Liu C, Liu Q (2018) Improvement of pruning method for convolution neural network compression. In: ICDLT, pp 57–60
- Chen S, Lin L, Zhang Z, Gen M (2019) Evolutionary net architecture search for deep neural networks pruning. In: ICCV, pp 189–196
- 37. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In: ICCV, pp 2736–2744

- Yan M, Zhao M, Xu Z, Zhang Q, Wang G, Su Z (2019) Vargfacenet: an efficient variable group convolutional neural network for lightweight face recognition. In: ICCV, pp 1–8
- Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv:1704. 04861
- Rahul M, Gupta HP, Dutta T (2020) A survey on deep neural network compression: challenges, overview, and solutions. arXiv:2010.03954
- 41. Kolstad (1985) A review of the literature on bi-level mathematical programming. Technical report. Los Alamos National Laboratory Los Alamos, NM
- Candler WV, Townsley R (1962) A study of the demand for butter in the united kingdom. Aust J Agric Econ 6:36–48
- Louati A, Lahyani R, Aldaej A, Mellouli R, Nusir M (2021) Mixed integer linear programming models to solve a real-life vehicle routing problem with pickup and delivery. Appl Sci 11(20):9551
- 44. Białas S, Garloff J (1985) Convex combinations of stable polynomials. J Franklin Inst 319(3):373–377
- 45. Sinha A, Malo P, Frantsev A, Deb K (2013) Multi-objective Stackelberg game between a regulating authority and a mining company: a case study in environmental economics. In: 2013 IEEE Congress on Evolutionary Computation, pp 478–485
- Xie L, Yuille A (2017) Genetic CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp 1379–1388
- Spears VM, Jong KAD (1991) On the virtues of parameterized uniform crossover. In: Fourth International Conference on Genetic Algorithms, pp 230–236
- Chakraborty UK, Janikow CZ (2003) An analysis of gray versus binary encoding in genetic search. US Patent 156, 253–269
- Lu Z, Whalen I, Boddeti V, Dhebar Y, Deb K, Goodman E, Banzhaf W (2019) An analysis of gray versus binary encoding in genetic search. Multi-criterion evolutionary design of deep convolutional neural networks, arXiv arXiv:abs/1912.01369
- 50. Settle TF, Krauss TP, Ramaswamy K (2006) Gray encoding for hierarchical QAM transmission systems. US Patent 7
- Dwork C, Feldman V, Hardt M, Pitassi T, Reingold O, Roth A (2015) STATISTICS—the reusable holdout: preserving validity in adaptive data analysis. Science 349(6248):636–638
- 52. Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1-2):273-324
- Shinozaki T, Watanabe S (2015) Structure discovery of deep neural network based on evolutionary algorithms. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 4979–4983
- Lu Z, Whalen I, Dhebar Y, Deb K, Goodman E, Banzhaf W, Boddeti VN (2019) Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Genetic and Evolutionary Computation Conference, pp 419–427

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Hassen Louati¹ · Ali Louati² · Slim Bechikh¹ · Elham Kariri²

Hassen Louati hassen.louati@stud.acs.upb.ro

Slim Bechikh slim.bechikh@fsegn.rnu.tn Elham Kariri e.kariri@psau.edu.sa

- ¹ SMART Lab, University of Tunis, ISG, Tunis, Tunisia
- ² Department of Information Systems, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, 11942 Al-Kharj, Saudi Arabia