Received March 6, 2021, accepted March 19, 2021, date of publication March 23, 2021, date of current version April 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068267

Evolutionary Algorithm-Based and Network Architecture Search-Enabled Multiobjective Traffic Classification

XIAOJUAN WANG[1], XINLEI WANG[1], LEI JIN[1], RENJIAN LV[1,2], BINGYING DAI[3], MINGSHU HE[1], AND TIANQI LV[1]

[1]School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]North China Institute of Computing Technology, Beijing 100083, China
[3]Department of Statistics, Colorado State University, Fort Collins, CO 80523, USA

Corresponding author: Lei Jin (jinlei@bupt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61871046.

ABSTRACT Network traffic classification technology plays an important role in network security management. However, the inherent limitations of traditional methods have become increasingly obvious, and they cannot address existing traffic classification tasks. Very recently, neural architecture search (NAS) has aroused widespread interest as a tool to automate the manual architecture construction process. To this end, this paper proposes NAS based on multiobjective evolutionary algorithms (MOEAs) to classify malicious network traffic. The main purpose is to simplify the search space by reducing the spatial ratio and number of channels of the model. In addition, the search strategy is changed in the effective search space, and the utilized strategies include EAs with the nondominated sorting genetic algorithm with the elite retention strategy (NSGA-II), strength Pareto evolutionary algorithm (SPEA-II) and multiobjective particle swarm optimization (MOPSO) to solve the formulated multiobjective NAS. Through comprehensive comparison of the population convergence times, model accuracies, Pareto optimality sets, model complexities and running speeds of the strategies, it is concluded that the model based on NSGA-II search has the best performance. The experimental results of the current machine learning algorithms and artificial learning methods based on the network are compared, showing that our method achieved better classification performance on two public datasets with a lower computational complexity, as mainly measured by FLOPs. Our approach is able to achieve 99.806% and 99.369% F1-score with 11.501 MB and 4.718 MB FLOPs on both IDS2012 and ISCX VPN dataset respectively.

INDEX TERMS Deep learning, multiobjective, neural architecture search, traffic classification.

I. INTRODUCTION

Network traffic classification is an essential task that can be used to detect network intrusion or provide appropriate network service. Recently, traffic classification has become increasingly challenging for several reasons. First, network applications vary, and the types of network flows have increased immensely. Second, increasingly network applications use encryption protocols. Classical methods such as deep packet inspection (DPI) cannot resolve encrypted traffic. Third, real-time traffic classification requires a model with lower complexity and a better running speed to ensure the classification performance.

Port-based methods [1] and DPI [2] are not effective at addressing encrypted scenes. To handle this tough problem, feature-based methods [3] are proposed. After artificially extracting the relevant features of the network traffic, researchers use machine learning (ML) algorithms such as support vector machine (SVM) [4], [5] and random forest (RF) [6], [7] to fit traffic data. However, extracting features from network traffic requires relevant expert knowledge and may cause missing information. Deep learning (DL) is suitable for directly sending the original data into DL-based models. Previous research [8]–[10] has proposed well-designed convolution neural networks (CNN) for classification

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaoqing Pan.

problem of traffic flow. The rise of AutoML [11], [12] has inspired us to automatically construct a network that can be applied to traffic classification tasks. The advantages of AutoML are obvious. First, it can improve model performance. In addition, there will be much less human intervention in designing the network architecture.

In the previous research, only simple 1D and 2D convolution layers or fully connected layers have been used. Recently, some useful convolutional blocks, such as the bottleneck used in ResNet [13] and the dilated convolution [14], [15] used in segmentation task, have increased network performance in the computer vision field. Determining how to apply these useful network blocks to traffic classification tasks remains a tough problem. Consequently, researchers tend to use AutoML [34], [35] to handle it. The neural architecture search (NAS), which is suitable for our scene, is an important branch of AutoML. NAS seeks to automate the network architecture design process. Some work [16], [17] generally treats NAS as a single-objective optimization problem that only focuses on improving the effect of classification but ignores the high complexity of the structure. However, some network attacks (e.g., DDoS) gradually focus on the mobile end [57], so when the DL training model is implemented in resource-constrained mobile devices, it is necessary to consider whether the model meets the requirements of being lightweight from multiple perspectives [36]. Therefore, in order to balance the classification accuracy and complexity of the model, we use multiobjective optimization algorithms (MOAs), including the nondominated sorting algorithm with the elite retrenchment strategy (NSGA-II), strength Pareto evolutionary algorithm (SPEA-II) and multiobjective particle swarm optimization (MOPSO), to complete the search. Then the lightweight performance of the model is evaluated again from the number of network parameters, FLOPs, MAccs (multiply-accumulate operations), and other aspects to evaluate the effectiveness of the above mentioned MOAs. Furthermore, we add more predefined operation block structures into the search space of the NAS architecture to obtain improved performance. We test the searched model on two general network traffic datasets to verify the efficiency of the designed search space.

The following is our main work: We defined five blocks and popular operations to simplify the search space. In ablation experiments, we demonstrate that blocks with reduced spatial ratio and number of channels can be searched for models with lower computational complexity and higher F1-scores. Then, based on the selected search space, three multiobjective search strategies were used to determine the Pareto frontier of the population on the same dataset. Finally, we select the ideal architecture from the Pareto frontier and compare their F1-score, complexity, lightweight, and speed. In general, the best search strategy is determined from the overall search situation and the optimal architecture searched. The innovations of this paper are as follows:

(1). Simplifying the search space: We propose an improved search space that mainly includes changing the number of channels and the spatial ratio for the NAS based on NSGA-II (mainly to obtain the optimal search space). The ablation study shows the efficiency of these improvements.

(2). Optimizing the search strategy: We use different architecture search strategies based on evolutionary algorithms (EAs) to automatically generate the population of CNN architectures that approximate the Pareto set (PS), which can achieve low complexity and high weighted F1-score compared with existing hand-crafted neural networks.

(3). Evaluating the performance of the developed model: After comparing the performance and model complexity, the Pareto optimality (PO) based on three optimization algorithms is selected. The lightweight deployment of the optimal model is analyzed from different aspects.

The rest of this paper is arranged as follows. Section II introduces the related work of this paper. Section III shows details of the searched block structures and search strategies. Section IV verifies the effectiveness of our searched architectures. The conclusion is given in section V.

II. RELATED WORK

Considering that the proposed model of this paper seeks to clearly demonstrate malicious traffic classification based on a MOEA task, the literature review is divided into three parts. The first section covers traffic classification and the previous efforts by experts in this field. NAS and its introduction of important methods are reviewed in the second section. In the third section, we reorganize the relevant algorithms including the early classic and the state-of-the-art MOEAs, and analyze the application of NSGA-II, SPEA-II, and MOPSO algorithm mainly focuses on the convenience and high feasibility brought by its application in real world problems.

A. TRAFFIC CLASSIFICATION

At present, there are two main traffic classification methods: one is based on ML, and the other is based on neural networks. Al-Obaidy *et al.* [17] systematically evaluate the social media traffic classification performance of four supervised ML methods, namely, SVM, naive Bayes, C4.5, and MLP. The authors extracted 14 flow-based features including source and destination IP addresses. When the number of features is increased successively, the traffic classification results are also gradually improved. It is proved that the rules generated by C4.5 perform the best on the dataset based on flow-based features, with the highest accuracy of 86.33%. Considering the design of an intelligent system (tractor) to improve the analysis of malicious traffic, Muliukha *et al.* [18] mainly used RF and naive Bayesian classifiers to classify VPN connections and SSL traffic, respectively. The results showed that the average classification accuracy of the RF is 87.9%(±0.60%); however, there is a large difference in the accuracy when the naive Bayes classifier is used to classify different types of traffic, with the highest accuracy reaching 99.1379% and the lowest accuracy only 33.33%. The classification accuracy of ML depends largely on feature engineering, which requires a considerable amount of manual work. Features extracted manually are shallow features, and more global features and local features can improve the accuracy of assessment [58].

DL can automatically select features through each layer [59]. The learning ability of DL is far higher than that of ML, and DL can learn the model with higher complexity. CNN model is designed in [19] to detect malicious URL, which saves the step of feature extraction. To address end-to-end encrypted traffic classification, in [20], they used a 1D-CNN to automatically learn the characteristics of encrypted traffic, which is more suitable to the task of encrypted traffic classification than 2D-CNN. The results show that the classification performance score of the 1D CNN is generally higher than that of C4.5. Yin *et al.* [21] explored an intrusion detection system based on DL using a recurrent neural network (RNN-IDS). The results show that the performance of the RNN-IDS is better than those of traditional ML classification methods, including the J48, artificial neural network, RF, SVM, and other ML methods studied by former researchers, in binary and multiclassification tasks.

DeepDefense [49] combines the RNN and CNN to transform DDoS packet-based detection into window-based detection. Compared with traditional ML, the error rate is greatly reduced. However, DeepDefense's training requires a large number of parameters and takes a long time. In recent years, Lucid, which was addressed in [50], considers the lightweight deployment of resource constrained devices and uses a CNN to share the weight parameters of the convolution kernel to reduce the storage space required for the model. Experiments show that Lucid can achieve high classification accuracy using a low complexity model on the test dataset. The above researches show us that the traffic classification performance using DL is likely better than that of traditional ML, and the lightweight model should also be used as a standard to measure model performance.

B. NAS

A large number of ML applications rely on expert learning to preprocess data, accurately select features, select an appropriate model and optimize the superparameters. Even if DL is used to train the network, it is necessary to reselect the superparameters of the network for different datasets or even to design the network structure from scratch. As a result, there is a growing trend towards automated learning, which allows models to be applied without human intervention, and AutoML has emerged. In recent years, AutoML methods [39], [44], [45] have matured sufficiently and achieved performance comparable to that achieved by manually designed network architectures on certain tasks [52]–[54]. As a branch of AutoML, the main task of NAS is to select the appropriate network architecture.

NAS has become an emerging research trend in recent years since practitioners can no longer rely on manually designed networks and automatically generate task-dependent networks. The NAS method needs to parameterize the search space by defining the maximum number of layers and operations. Then, the network architecture is learned through different search strategies, and the models with good performance are selected. Finally, the candidate model or optimal model is evaluated accurately. Therefore, the following studies are focused on the search space, search strategy, and model evaluation.

The search space generally has two ways to define spatial structure: searching for overall structure (global search) or searching the cell structure and splicing cells together according to the large structure defined in advance (cell-based search). In earlier studies [46], [32], the global search space was designed as a chain structure or a skipping structure. However, it is often necessary to search all the components in the entire network architecture, which lacks flexibility. Using a cell-based [26], [51], [16], [39] search space is a common choice in development in order to achieve good robustness and effectiveness in the search structure. This method has a simple design and usually consists of many repeated cells used to form a larger architecture. As a result, the search space will be greatly reduced, and the search architecture is also reduced to searching for a single cell. The fine-grained search space (atomic blocks) [51] limits the optional search space for each block to the type of convolution, the number of output channels and the size of the convolution kernel. For the current traffic classification task, to reduce the complexity of the model and make the structure have better transferability between datasets, the cell structure is adopted in our design. The search strategies can be roughly divided into three categories, namely, gradient-based methods [22]–[24], reinforcement learning (RL) [46], [47], [26] and EA-based [18], [27]–[30] approaches. The model evaluation can be divided into two stages: search objectives during the search process and the evaluation metrics after the search. The former usually predicts the performance of candidate models through multiple or single objectives to determine whether to keep the model or extend it on this basis. The latter assess the optimal model obtained after the search (which may stack models to form a deeper network) and re-evaluates the performance of the model by making a fair comparison with a comparative model on the same training set or applying it to other datasets to prove its transferability.

At present, all work on NAS mainly focuses on reducing the network search space, searching for the search strategy that converges as fast as possible, and selecting the appropriate model evaluation methods. Building the search space (cell-based search) mainly considers the network topology and operation type. In previous studies, the types of operations are usually fixed to simplify the search space. However, the varied topology of the network should be the focus of the discussion. In this paper, to the traditional traffic classification operation, we add the related operations in the computer vision field to enhance the classification accuracy. The setup of our network is similar to the structure defined in [28]; that is, the connection between nodes in the block is an operation type, and each block is linearly connected along the depth. We limit the number of operations in the search space to a fixed value and simplify it to a block of only one layer according to the traffic classification task. We also designed a special network structure to reduce the number of channels and spatial ratio. In this way, the complexity of the current search structure has fully met our expectations. However, finding the classification accuracy comparable to [20] using a low-complexity structure has become a more noteworthy problem. In other words, the architecture found should always consider an implicit trade-off between the accuracy and redundant memory consumption. We introduce NSGA-Net, a genetic algorithm-based network architecture search method proposed by Lu *et al.* [27], which extracts PS through parent-generation crossover and inheritance. In this way, we can measure both the network complexity and accuracy to make a final choice. Inspired by this, we also considered two other optimization algorithms similar to NSGA-II to find a task-matching search strategy.

C. MOAs

With the increasing complexity of global optimization problems, experts pay more attention to the problem of multiobjective solution. In this kind of problem, the optimization objective function results in a sharp increase in the calculation amount under the mutual restraints of many factors. The deterministic algorithms such as planning algorithm and branch-and-bound algorithm have been unable to solve the complex production and living needs. The emergence of intelligent evolutionary algorithms (IEAs) solves this problem. It uses the genetic laws of simulated biology to solve highly complex nonlinear problems. According to different scientific techniques, multiobjective IEAs can be divided into Pareto-based, aggregation-based and indicator-based. [61] proposed SPEA-II on the basis of the original method, and the convergence and diversity of the solution set are better. Coello and Lechuga [62] proposed a Pareto-based MOPSO algorithm. In the iterative process, the external archive is used to store the non-dominated solution, and other population members are combined to guide the particles to fly. It has the advantages of easy implementation and fast convergence. The team of DEB [63] proposed NSGA-II, which has become a research hotspot of multiobjective optimization algorithm in recent years. Then, the DEB team abandoned the crowding distance strategy in NSGA-II, and introduced widely distributed reference points to deal with the optimization problem of more than three objectives in individual analysis, thus proposed NSGA-III [64]. NSGA-III solves the problem of poor convergence and less diversity of NSGA-II in the high dimensional objective optimization task. [65] proposes multiobjective evolutionary algorithm based on decomposition (MOEA/D) algorithm, which has better performance in high-dimensional multiobjective solution. With the increase of objective dimension in multiobjective optimization problems, EAs based on indicator evaluation framework has been developed, such as ISDE+ [66], HypE [67] and MAOEA/IGD [68].

NSGA-II, MOPSO, and SPEA-II are widely used in MOEAs. However, the performance on many multiobjective testing problems has not been tested in past studies. In fact, in the process of considering the application of MOEAs, we comprehensively compare the following situations: NSGA-III, MOEA/D and other algorithms need a large number of scale vectors or appropriate reference sets, which are not easy to be applied in real world problems [72]. Our task only contains two optimization objectives, which is not a high-dimensional objective optimization problem. Many of the algorithms proposed in recent years, such as ISDE+, MAOA/IGD and other indicator-based algorithms, are more concerned with the situation of high dimensional objectives, and this paper does not need too much attention. There is no fact to deny the effectiveness of MOPSO, NSGA-II and SPEA-II algorithms in solving practical problems. On the contrary, it is a common solution of multiobjective tasks to combine the experimental design method with these MOEAs. Because of the good performance of PSO in dealing with large-scale complex problems, researchers often use MOPSO to solve multiobjective optimization problems with high computational time [69], [70]. SPEA-II and NSGA-II are commonly used algorithms in MOEAs. [71] proposed SPEA-II as the basic framework to solve the Pareto solution set that maximizes the profit of smart grid. In [56], NSGA-II and SPEA-II algorithms are applied to the smart grid environment to manage the peak load scheduling problem. The result shows that both algorithms have good convergence, but NSGA-II performs better in time complexity and accuracy. Based on the above analysis, we use NSGA-II algorithm to implement the traffic classification problem based on NAS, and compare the results with SPEA-II and MOPSO.

III. METHODS

For traffic classification, subject to the enormous amount of real-time traffic and limited hardware resources in practical applications, we must consider the computational complexity of the model. Therefore, we should balance multiple objectives (e.g., predictive performance and computation complexity). In this paper, we adopt FLOPs [37] and F1-score to judge the models' computation complexity and classification effectiveness, respectively. Often, when multiple objectives are considered, there may not be a perfect solution that can reach the optimal score in all desired metrics. Therefore, we attempt to find a trade-off between the effectiveness and the complexity. We apply three optimization algorithms, namely, NSGA-II, SPEA-II and MOPSO, to architecture search, which can automatically generate a set of useful CNN models. [38] states that the network obtained by the NAS search based on cell search space converges faster and more stably. Therefore, we choose the architecture with smaller classification performance and fewer FLOPs under three optimization algorithms and compare the convergence of F1-score curve on the validation dataset for the different architectures. In addition, by combining the traffic classification characteristics, we design a group useful substructure that can improve the model performance. The rest of this section describes the details of our architecture.

A. DATA PROCESS

In this paper, traffic packets are presented using hexadecimal coding. We transform the original data into decimal coding, and the model analyzes the data directly. Compared with traditional artificial feature engineering, it contains more information for the classification model. The data processing is as follows:

1) DATA CLEANING

First, we should remove redundant and duplicate packages. One traffic packet generally consists of an Ethernet layer, a network layer, a transport layer, and an application layer. We remove the data of the Ethernet layer, which contains the MAC source/destination address, and the protocol version. In addition, IP addresses are useless. This information is not relevant to the network behavior.

2) DATA SPLIT

In this step, we split the traffic data with the same five-tuple of information (source/destination IP, source/destination port, and protocol). Traffic packets with the same five-tuple of information form a flow. We use the SplitCap tool to split the traffic.

3) VECTORIZATION

We use only 10 packets for a network flow. The network flows that have less than 10 packets are padded with zeros to reach a certain length. For every packet, we extract only 160 bytes of payloads. When a packet does not have 160 bytes, we pad zeros behind it. Finally, we transform every traffic flow into a 1600-dimensional vector. We can resize it into a 2D-vector that can be processed by CNN.

B. GENETIC SEARCH ARCHITECTURE

1) ENCODING

From a biological perspective, the neural network architecture can be viewed as a phenotype, and it is mapped from its genotype. Genetic operations such as crossover and mutation are conducted in the genotype space. The mapping relation between genotypes and phenotypes is called encoding in this paper. This idea of coding mapping is also mentioned in [60]. In other words, we should define an encoding of the neural network architecture.

Most existing CNNs can be viewed as a combination of computational blocks that define the layerwise computation (e.g., ResNet blocks [13], Inception blocks, and SE blocks). We follow the encoding method proposed by Xie and Yuille [32]. However, existing architectures (e.g., Alexnet [41] and GoogLeNet [42]) are searched using image processing, which requires deeper and larger networks for complex scenes. Traffic classification is easier to distinguish, and a deeper network may lead to overfitting. Therefore, we should reduce the search space for our task. First, we divide the whole classification model into several phases, as in [39], [40]. Each phase is a block that is composed of several basic operations. To encode the whole network, the blocks should be encoded first.

Block Encoding: Our proposed binary network representation encodes the network architecture into binary strings, which are used to specify whether the data flow flowing out of a node needs to reduce the number of channels or change the spatial ratio. A network architecture in the population is called a block, and a block consists of several nodes that describe the operations. Here, a node is a basic operation unit, which can be a single operation or a sequence of operations. In this way, we can represent a network with a combination of several blocks. The blocks can be viewed as a combination of basic operations. In this paper, for simplicity, our final network repeats the same blocks. Given the ith block x_i as a part, it is represented as $x_i = (x_i^O, x_i^N)$, where x_i^O and x_i^N represent the encoding of the set of operations and the traffic exchange nodes, respectively.

The operation set is defined by the search space. The set contains a total of 10 operations from which an operation is randomly selected to act on the outflow of the current node. In addition, for nodes with traffic outflows, we judge whether the node needs to change the number of channels by encoding. The kth node of the ith block in the network architecture is defined as x_i^k. Hence,

$$x_i^k = \begin{cases} CH_I, CH_O \rightarrow CH_I, CH_O/n_{BN}, \mathit{if} & k = 0, 1 \\ CH_O, CH_I \rightarrow CH_{O,I}/n_{BN}, & \mathit{otherwise}. \end{cases} \tag{1}$$

where $k = 0, 1, 2, \ldots, L$, and L represents the number of nodes in the block. If node x_i^k is connected to a node encoded as 0 or 1, the input channel CH_I of the operation that applies to the outflow of x_i^k does not change, whereas the output channel CH_O is reduced to $1/n_{BN}$. n_{BN} represents the number of empty nodes (that is, no operation is done on this node except *Concat* or *add*). However, if the current node is connected to the nodes other than 0 and 1, the number of input and output channels for the operation on the outflow traffic of the node is reduced to $1/n_{BN}$. Fig. 1 shows the complete network architecture composed of a block.

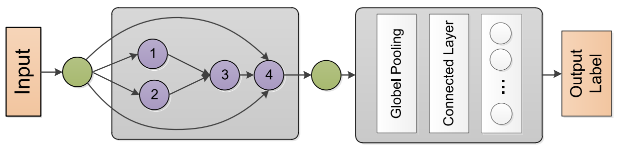


FIGURE 1. A classification network contains one block with 4 nodes.

Search Space: To make architecture searching easier, we should predetermine the input layer, output layer and connection layers between blocks. The total search space of a block is governed by the number of operations and the

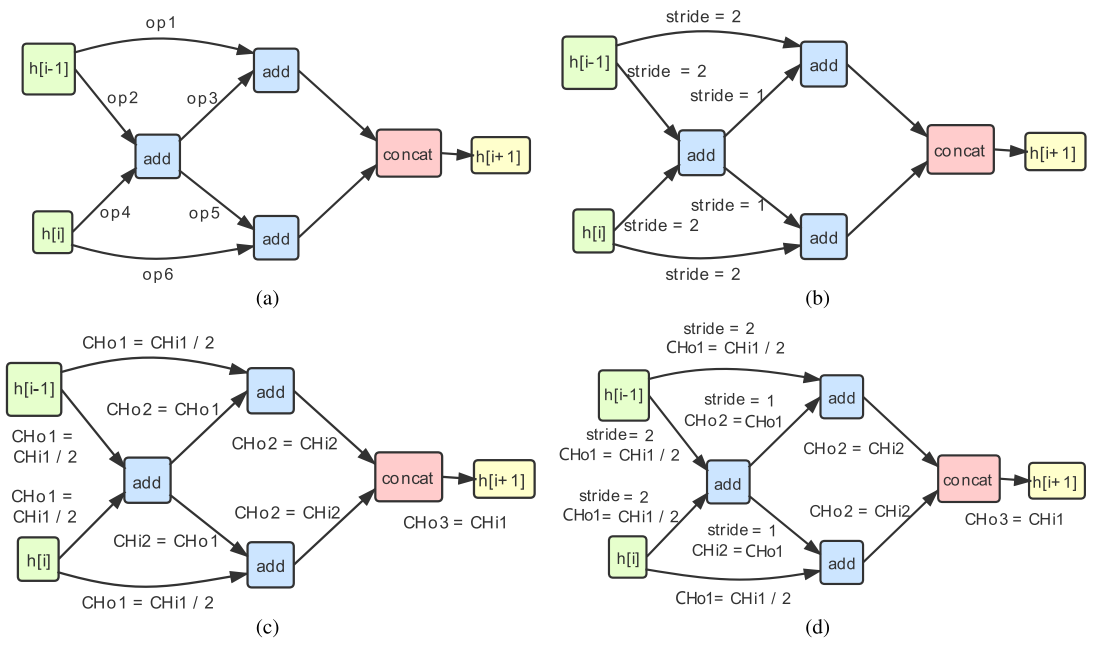


FIGURE 2. **(a) Architecture with full operation set. (b) Decrease in the spatial ratio. (c) Decrease in the number of channels. (d) Decrease in the spatial ratio and the number of channels.**

number of nodes. The search space of a block is:

$$\Omega_{x_i^k} = \prod_{m=1}^{m=n_o} ((m+1)^2) n_p^{2n_o}. \tag{2}$$

where n_p is the number of operations and n_o is the number of nodes in the block. Because we set the encoding for the different phases to be the same, the search space of a complete network is equivalent to a block.

To enrich our search results and improve network performance, we add some predefined architecture blocks. A detailed discussion of the proposed predefined architectures is provided in the next section.

2) PREDEFINED ARCHITECTURE BLOCKS

In this section, we predefine some useful blocks that have lower spatial ratio and fewer channels. We also tested the lightweight and running speed of the architecture and compared it with the hand-designed CNN model, proving that the architecture is lighter and faster in the predefined search blocks. In our paper, a normal block is a directed acyclic graph.

Original Architecture: First, as a comparison with the previous hand-crafted network, we use only the original operations such as 2D convolution and pooling. In Fig. 2, we show the complete structure of a block. The convolutional operations are the edges in the block. A node in a block has two inputs, and we use *add* to merge the two inputs. Finally, a *Concat* operation selectively combines the information of previous nodes to make an output. The original operation set consists of ReLUConvBN, MaxPooling, AvgPooling, and Skip. The detailed structures are shown in Fig. 3.

Architecture With the Full Operation Set: Recently, some more effective convolutional operations have been proposed. For example, in order to extract more detailed information and expand the receptive field without changing the spatial ratio of feature map, we add the dilated convolution operation to the full operation set. To solve the disappearing gradient problem in the training process, the ReLU nonlinear function is introduced as the activation function in the operation set, and batch normalization is added in some operations. SepConv uses two horizontal and vertical convolution kernels to replace the original convolution kernels. We add some new operations to the operation set to construct new blocks. Fig. 3 shows the detailed structures of the new operations.

Decrease in the Spatial Ratio: For a convolutional network, decreasing the spatial ratio of the feature maps can decrease the computation complexity and integrate low level information. Therefore, we change the stride of convolution operation in the operation set from 1 to 2. Then, the spatial sizes of the feature maps will be cut in half after passing a block. We show the difference between the original blocks and blocks with a decreased spatial ratio in Fig. 2.

Decrease in the Number of Channels: In addition to reducing the spatial ratio, reducing the number of channels in operations is effective. The different paths in blocks can be viewed as different information flows. Then, decreasing the number of channels of an information flow means compressing the information. Finally, the operation is used to combine all the information flows. In this setting, we keep the number of channels equal between the inputs and outputs of blocks. We call the node connected to the output node the null node. Then, we set the number of nodes between the input and output nodes to M so that the number of empty

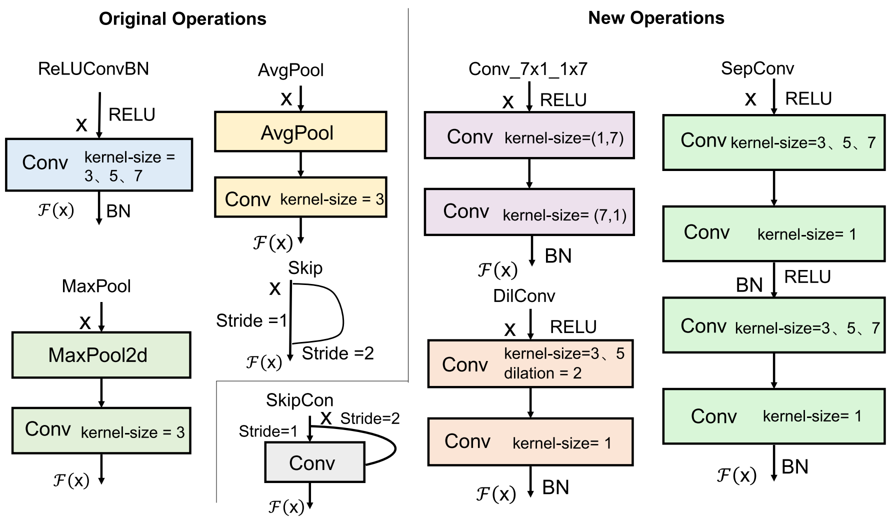


FIGURE 3. Detailed implementation of major operations. The left side contains common operations in traffic classification, while the right side adds rectangular convolution, deep separable convolution, dilated convolution, and skip connection.

nodes is $[1, 2, \ldots M - 1]$. Therefore, the number of output channels should be a common multiple of $[1, 2, \ldots, M - 1]$. Fig. 2 shows the detailed structure of a block with fewer channels.

Decrease in the Spatial Ratio and Number of Channels: Next, we can construct a block that has both decreased spatial ratios and fewer channels. Fig. 2 shows the structure of this type of block.

In addition to the above four design blocks, we hope to further expand the search space to find a better structure. Therefore, we design two groups of search spaces: fixed initial channels and searching for initial channels. Since the former simply specifies the original number of channels, the set value will be described in Section 4. Only the relevant principles of the latter are introduced here.

Searching for Initial Channels: The search space is expanded by modifying the number of initial channels from the original fixed value to a variable number of channels. The initial population length of the first generation was modified to $(Length_{fixed} + 1)$, where $Length_{fixed}$ is the initial population length under the condition of a fixed number of channels. The last value of the network architecture code represents the searchable channel value, which we set to 12, 24, 36, and 48.

3) SEARCH PROCEDURE

After thousands of years of evolution, the existence of all things in nature must be justified. People seek the optimal laws in nature and develop them into EAs. NSGA-II, MOPSO, and SPEA-II are MOEAs based on the optimization rules of nature. In the current task, we stipulate that every bit of an individual must be an integer; therefore, the integer programming problem is worthy of attention. When an individual in a population crosses and mutates, it is essentially a bit in an individual and a random number ranging from (0,1) that is added and subtracted. Therefore, the modified code is remapped to an integer by rounding. The details of these three optimization algorithms are provided below:

(1) NSGA-II

This is a process in which the solutions become gradually better. First, we randomly initialize several network architectures, which are called populations. In every iteration, new network architectures are generated by the parent networks sampled from the population, including mutation and selection processe [55]. Each network competes for both survival and reproduction (becoming a parent). After several generations, the networks in the population will obtain PO for this dataset. The procedures of mutation and selection in NSGA-II are described below:

Mutation: To enhance the diversity of the population (having different network architecture) and the ability to escape from local optima, we randomly flip the bit encoding. To prevent creating a completely different architecture, we restrict the number of flipped bits to one for each mutation operation.

Selection: After generating the network architecture, we use a unified metric to select the effective network. In the traffic classification task, we use the FLOPs and F1-score to select networks.

(2) SPEA-II

Compared with NSGA-II, SPEA-II has less efficient operations but is better able to address high-dimensional optimization problems. In addition, Herstein *et al.* [43] demonstrates that SPEA-II has better performance than NSGA-II in both constrained and unconstrained multiobjective optimization problems. The following are the key steps in SPEA-II, first initializing an archive set A_t for storing the historical generations and a population set P_t for the current generation. The environmental selection strategy is implemented in both A_t and P_t to generate the resulting generation A_{t+1} and P_{t+1}.

Then, the mutation operation is applied to the individuals of the newly formed A_{t+1} and P_{t+1}. The specific implementation process of the environment selection strategy is as follows:

Environmental Selection Strategy: Pareto optimal soluitons (nonnormalized soluitons) is characterized by the fact that it cannot weaken at least one other objective function while improving any objective function. The nondominated solutions are selected from P_t and E_t. The set of nondominant solutions is preserved in the next generation of the archived population E_{t+1}. If the number of solutions in the nondominant solution set of P_t and E_t is less than the size of E_{t+1}, the dominant individuals with the lower dominance level in P_t and E_t will be saved in E_{t+1} to maintain the population diversity. However, when the number of nondominant solutions is larger than the size of E_{t+1}, the truncation strategy based on k-nearest neighbors follows to maintain the size of the external population.

(3) MOPSO

In MOPSO, each solution of the multiobjective optimization problem is regarded as a particle, and each particle's current position is represented by a fitness function. Moreover, each particle can remember its own historical best position and choose the flight direction and distance based on its own experience and other particles in the population. To guide particles to the Pareto front (referred to as the global optimal), the position and velocity of particles need to be updated as follows:

$$v_i = \omega v_{io} + c_1 r_1 (x_{io}^P - x_{io}) + c_2 r_2 (x_{go}^G - x_{io}). \tag{3}$$

$$x_i = x_{io} + v_i. \tag{4}$$

where v_{io} and $x_{io}(0 \leq i<N)$ are the velocity and position of the current generation of particle i, respectively. x_{io}^P is the optimal position reached by the ith particle, and x_{go}^G is the optimal position reached by all particles in the population. c_1 and c_2 are two coefficients of acceleration, and r_1 and r_2 are two random numbers. ω is represented as follows:

$$\omega = \omega - \frac{t \times (\omega_{max} - \omega_{min}) + 1}{t_{max}}. \tag{5}$$

A large ω is good for the particle to jump out of the local optimum. However, a small ω is good for the convergence of the algorithm. We set the initial value of ω to 1 and use the above formula to reduce the value of ω linearly. t represents the current number of iterations, t_{max} represents the total number of iterations, ω_{max} is an all-one D-dimensional vector, and ω_{min} is an all-zero D-dimensional vector. D is the encoding length of each generation of particles. The initial velocity of the particle is random.

Similar to the previous two methods, MOPSO is also an iterative optimization algorithm, but there is no crossover or variation in the implementation process.

C. LIGHTWEIGHT MODEL

Due to storage space and power consumption limitations, the neural network model still faces great challenges in the storage and computing of embedded devices. We start with the basic convolution operation to reduce the computational complexity without sacrificing the network performance. MobileNet V1 and V2 and shuffle Net V1 use the FLOPs to evaluate the model dimensions, so we calculate the FLOPs for the final network searched by NSGA-net. We reduce the FLOPs in the following two ways: 1) Following MobileNet V1, we use the deep separable convolution instead of the standard convolution to reduce the number of parameters. 2) We design a network architecture to reduce the number of channels and the spatial ratio at the same time. Reducing the number of channels obtains a more lightweight model, and reducing the spatial ratio reduces the model complexity. The complexity of the evaluation model is measured using the FLOPs, MAccs, and model inference ability. In addition, the lightweight of the model is evaluated using the model parameters and memory usage.

IV. EXPERIMENTS

In this section, we will explain the parameter settings and implementation details of the network structure. The experiment is divided into three parts. First, the NSGA-II-based optimization algorithm is applied to the five search spaces previously designed, and the ablation experiment is conducted to select the one with the best effect. In this part, we focus on comparing which search space can find the better model, so we only choose NSGA-II as the search strategy. Second, the performances of the three optimization algorithms in the selected search space are compared, and then three structures are selected from the Pareto optimal set based on the three optimization algorithms. Finally, the lightweight and speed performance of the searched architectures are analyzed by comparing the MAccs and F1-score curves. For more details, see the following description:

A. IMPLEMENTATION DETAILS

Data Description: To verify the effectiveness of the search, we select two public datasets. We use the IDS2012 dataset to detect intrusion traffic. This dataset contains different types of malware traffic including Brute Force SSH, DDoS, HTTPDos, and relaying attacks. The difficulty of analyzing this dataset lies in the unbalanced distribution of different traffic flows, which can be shown in Table 1. To prove that the search strategy can search the network architecture with better performance than the manually designed on different datasets, we rerun the experiment on the ISCX VPN dataset. This dataset has 7 general categories of encrypted traffic, namely, web browsing, email, chat, streaming, file transfer, VoIP, and P2P, which are shown in Table 2. We split both original training sets to create our training (70%) and validation (30%) sets for the architecture search. Finally, when the architecture is completely retrained, and the test results are obtained by the original partitioned dataset.

Hyperparameters: The network architecture consists of only one block in which there is no repetitive pattern in each architecture. In our design, each block contains

TABLE 1. IDS2012 dataset description.

Class Name	Quantity	Percentage(%)
Normal1	8483064	95.323
Brute Force SSH	6964	0.78
DDoS	21121	2.37
HttpDoS	3482	0.39
Infiltrating Transfer	10044	1.13
TOTAL	998817	100

TABLE 2. ISCX VPN dataset description.

Class Name	Description	Quantity	Percentage(%)
Email	SMPTS, POP3S and IMAPS.	26844	14.94
Chat	ICQ, AIM, Skype, Facebook and Hangouts.	33978	18.92
Streaming	Vimeo and Youtube.	26682	14.85
File Transfer	Skype, FTPS and SFTP using Filezilla and an external service.	30000	16.7
VoIP	Facebook, Skype and Hangouts voice calls.	30000	16.7
P2P	uTorrent and Transmission (Bittorrent).	32130	17.89

6 operations and 5 nodes. Of the 5 nodes, 2 nodes are fixed nodes (No.0 and No.1 nodes which are marked in Fig. 1), and the remaining 3 nodes are intermediate connected nodes of 6 randomly generated operations. Finally, the output data streams of all intermediate connected nodes are spliced together as output results. Hence, we set the number of blocks to 1 and the number of nodes to 3 (except for two fixed input nodes). We set the random seeds as 0 and determine the initial population of the first generation using uniformly distributed random numbers to ensure that each experiment starts with the same random numbers to iterate. The number of generations is 10, and the population size is 20 for each generation; therefore, 200 offspring can be obtained per search, which takes approximately 2−3 days on an NVIDIA 1080Ti GPU in PyTorch.

Training Details: For each convolutional block, we train for 12 epochs with a batch size of 128 in both the training set and test set. Then, we use momentum stochastic gradient descent (SGD) to optimize the weight and adjust the learning rate with cosine annealing. At the end of a cyclical learning rate, the initial learning rate is 0.025, which can decrease to 0. We set the momentum to 0.9 and the weight decay to 3×10^{-4}. In NSGA-II, the crossover probability and mutation probability are set as 0.4. In SPEA-II, the archive size is set to 20.

B. ABLATION EXPERIMENTS

To study the influence of different numbers of input channels on the classification performance and complexity of the model, we design two groups of comparative experiments: a fixed initial channel and searching for the initial channel. In the experiment with a fixed number of initial channels, we conduct the ablation experiments in five different search spaces. We find that the structural results found in the search space based on the original operation are not ideal; therefore, in the search for the initial channel experiment, only the last four groups of search spaces are selected.

Table 3 shows the performance of the Pareto optimal solution representation model for each search space. The original architecture using original operations is significantly more complex. The model performance improved as more operations are added. Therefore, the new operation is also suitable for traffic classification.

The results of the decrease in the spatial ratio are shown in Table 3. It is clear that the F1-score increased whereas the FLOPs decreased. Reducing the spatial ratio of the feature graph can aggregate the intermediate features and reduce the computational complexity. According to the channel reduction data, we find that if only the number of channels is reduced, the performance of the model cannot be significantly improved. However, if the space ratio and channel are reduced at the same time, as shown in bold in the table, this search space can find a network structure with a higher F1-score.

To prove that our results are not due to chance, we will analyze the overall trend as follows. Fig. 4 shows the scatter plot obtained by NSGA-II based on the bi-objectives in different experimental groups. The figure clearly shows the improvement of the entire population under the four groups of ablation experiments. A large number of outliers appear in the experiment of the full operations, which is more clearly reflected in the box diagram in Fig. 5, indicating that many architectures with unsatisfactory bi-objectives are searched out in the operation of the full set. In the search space with only a reduced spatial ratio, the outliers of the F1-score and FLOPs are greatly reduced, while their metrics are improved. For the method of decreasing the number of channels, the F1-score is improved and the outliers are reduced, whereas the model tends to be more complicated. However, the method of decreasing the spatial ratio and the number of channels can greatly improve F1-score. In addition, the initial number of input channels can vary. We can add the initial input channel to the search space. The result of increasing the number of initial input channels for searching is shown in Fig. 6. We can see that the value of the FLOPs in the four groups of the search space is significantly increased compared with that of the fixed initial channels, and the overall F1-score is also improved. Whether viewed from the individual comparison in PO (Table 3) or the overall search results (Fig. 6), there is a significant decrease in the FLOPs after reducing the spatial ratio, and the F1-score is also increased after reducing the spatial ratio and number

TABLE 3. The best structure in the two group experiments (The results after training for 12 epochs).

		FLOPs(MB)	F1-score(%)	Accuracy(%)	Precision score(%)	Recall score(%)
Fixed initial channels	Original architecture	33.005	99.227	99.223	99.201	99.252
	Architecture with full operation set	18.413	98.968	98.961	98.931	99.005
	Decrease in spatial ratio	7.133	99.448	99.446	99.430	99.464
	Decrease in channels	18.873	98.870	98.863	98.832	98.917
	Decrease in spatial ratio & channels	**11.501**	**99.574**	**99.573**	**99.560**	**99.586**
Searching for initial channel	Architecture with full operation set	198.068	99.680	99.679	99.670	99.688
	Decrease in spatial ratio	44.467	99.660	99.656	99.650	99.671
	Decrease in channels	204.058	99.564	99.563	99.550	99.577
	Decrease in spatial ratio & channels	**52.762**	**99.787**	**99.786**	**99.780**	**99.792**

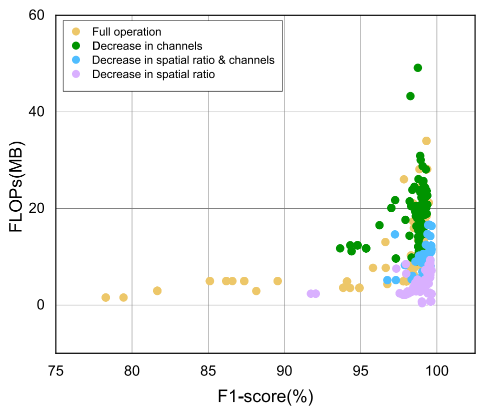


FIGURE 4. Improvement of the bi-objective with the population in four sets of ablation experiments based on the IDS2012 dataset.

of channels. The above results show that the search space designed by us is effective at reducing the complexity and improving the classification performance. This result is more intuitive in Fig. 7. As the number of initial channels increases, the F1-score and FLOPs improve to varying degrees.

The above experimental results are summarized as follows: in the variable initial channel method, although the F1-score of the model is relatively high, the complexity also increases more. In addition, we find that both the complexity and the F1-score of the model are ideal when the number of channels is 12. Therefore, in the case that the number of channels is fixed to 12, we consider extending the search space to the full set of operations and conducting subsequent experiments on the operating space with a reduced number of channels and spatial ratio.

C. CONCLUSION REGARDING THE SEARCH STRATEGIES

In this section, we adopt different optimization algorithms to decrease the spatial ratio and channel search space to select the best search strategy. We adopt 3 optimization algorithms introduced in Section II, which can automatically search structures that approximate the Pareto front between the error (error = 100 - F1 score) and complexity of malicious traffic classification tasks. It is clear from Fig. 8b that 10 generations of structures are searched by using MOPSO, and the FLOPs of each generation are very low (the FLOPs of the initial generation are lower than the other two). MOPSO performs well in the task of searching low-complexity architectures. However, in terms of the number of structures in the PS, as shown in Fig. 8a, the NSGA-II algorithm has more than the other two (the red dots represent Pareto optimal outcomes). From 8c, we observe that SPEA-II cannot search out a structure with less than 2.5 Mb FLOPs, and the number of POs is also the least.

Because the complexity of the structure is generally low and the implementation of the MOPSO algorithm is simpler, as shown in Table 4, the shortest running time of the three optimization algorithms is achieved by the MOPSO algorithm, followed by NSGA-II, and SPEA-II needs the longest running time. Fig. 9 compares the distribution of total number of POS of each generation under the bi-objectives obtained after the search of the three optimization algorithms. NSGA-II is more likely to search for individuals with high F1-scores, while MOPSO tends to search for individuals with low FLOPs. In addition, Table 4 shows that in the PO searched by the three optimization algorithms, NSGA-II searches for the highest F1-score and MOPSO searches for the lowest FLOPs. In addition, the POs searched by NSGA-II and SPEA-II are relatively clustered while the POs searched by MOPSO are more dispersed from Fig. 9. We analyzed the possible reasons as follows: NSGA-II and SPEA-II have genetic crossover and mutation, and the information shared between the populations is close, so the number of outliers in the models searched by these two algorithms is relatively small. MOPSO, on the other hand, follows a single information sharing mechanism. Although the particles can converge at a fast speed, they are not closely related to each other, so there are many outliers.

To further compare the performance of the PO searched by the three algorithms, the F1-score curve measured on the validation dataset is depicted. Fig. 10 shows that the searched block based on NSGA-II has a short shock at approximately the 40th epoch, but it tends to be stable as training progresses; meanwhile, the F1-score curves of the other two models have a large amplitude shock. In addition, the NSGA-II searched

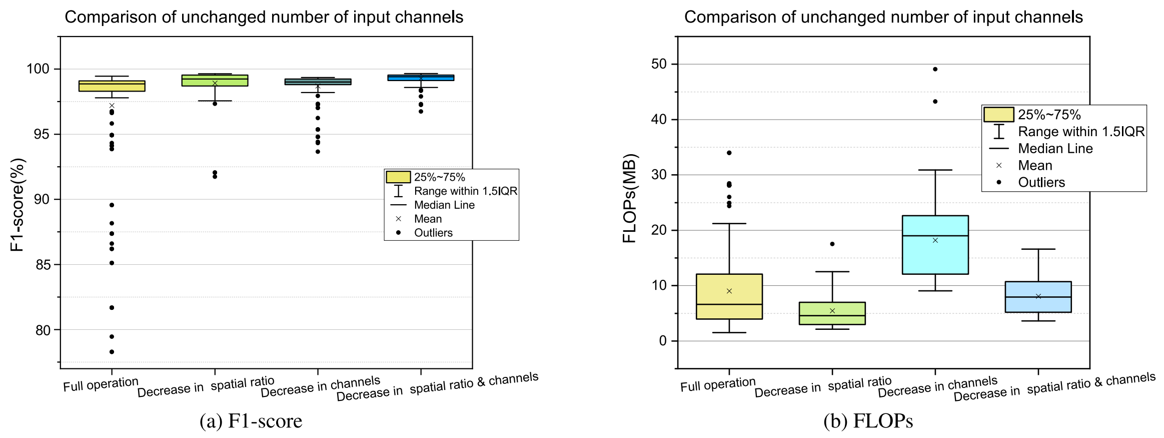


FIGURE 5. Box plots of the four groups of experiments with fixed numbers of input channels based on the IDS2012 dataset.

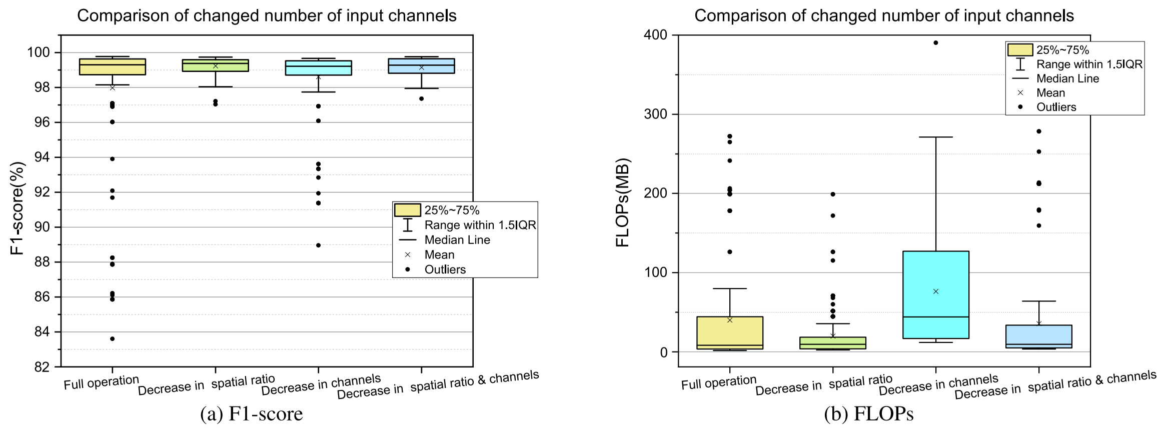


FIGURE 6. Box plots of the four groups of experiments with the number of searched input channels based on the IDS2012 dataset.

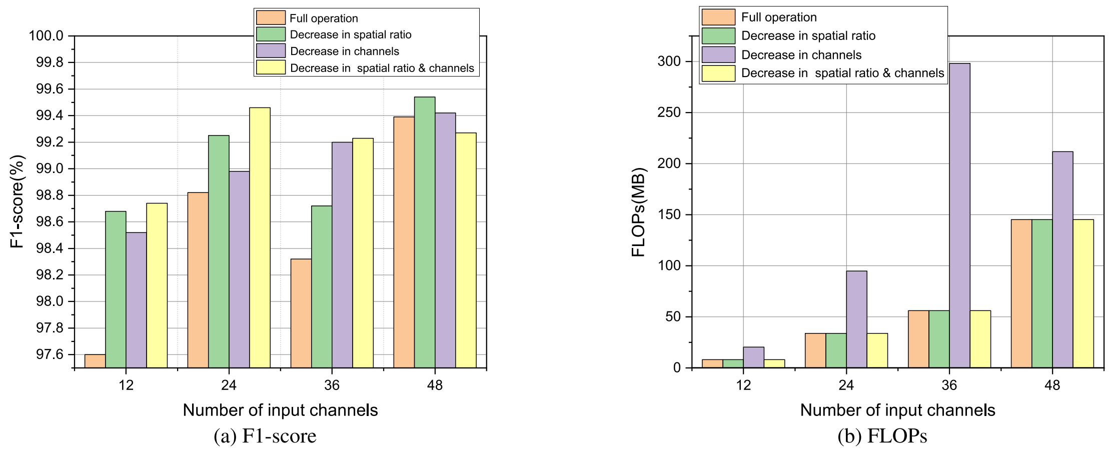


FIGURE 7. Bar charts of the four groups of experiments with the number of searched input channels based on the IDS2012 dataset.

model tends to have small convergence errors and small damped oscillations after the 50th epoch of training, while the other two models only converge after approximately the 70th epoch. This result indicates that the model obtained by NSGA-II has better stability and convergence than the other models.

D. EVALUATION OF THE ARCHITECTURE

The F1-score of our searched architecture based on NSGA-II has exceeded that of the architecture designed by hand. However, in malicious traffic detection, it is necessary to stop malicious traffic in time via real-time detection. In addition to the F1-score, the computational complexity is another

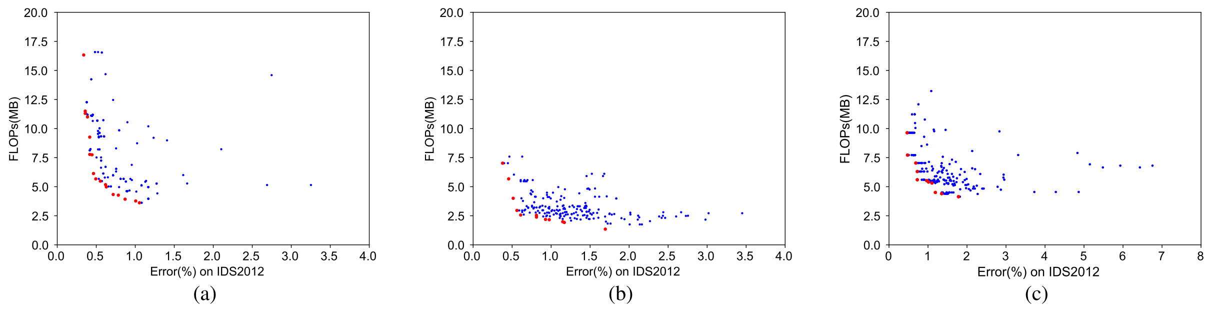


FIGURE 8. (a) Trade-off frontier of NSGA-II (the Pareto optimal set contains 17 optimal solutions). (b) Trade-off frontier of MOPSO (the Pareto optimal set contains 12 optimal solutions). (c) Trade-off frontier of SPEA-II (the Pareto optimal set contains 11 optimal solutions) based on the IDS2012 dataset.

TABLE 4. Comparison with different optimization methods. In this table, each optimal solution is selected from the Pareto optimal solution set, and the F1-score is the result of 120 epochs of training based on the IDS2012 dataset.

Methods	F1-score(%)	FLOPs(MB)	GPU Hours
MOPSO	99.699	2.959	10
NSGA-II	99.806	11.501	20
SPEA-II	99.738	5.597	25

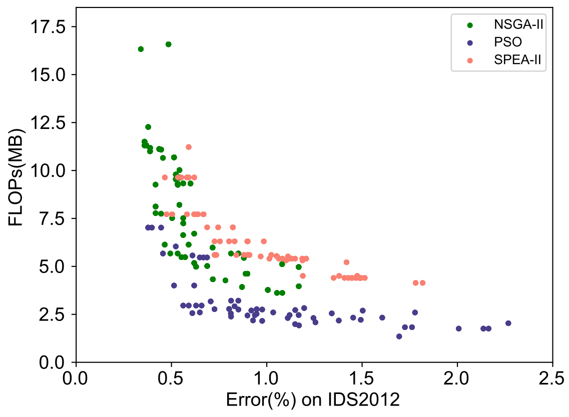


FIGURE 9. Pareto optimal outcomes of the optimization algorithms based on the IDS2012 dataset.

TABLE 5. Comparison of the complexity, numbers of parameters and speeds of the search models under different EA algorithms.

Methods	Params(KB)	MAcc(MB)	Flops(MB)	GPU Speed (Batches/sec.)
NSGA-II	25.510	22.18	11.3	28.962
MOPSO	9.557	5.68	2.96	30.789
SPEA-II	11.009	10.71	5.59	20.034
CNN	52.106	54.05	26.99	2.012

important metric to be considered by the CNN network. To meet this requirement, the performance of the three searched models in terms of the model complexity and speed is evaluated below.

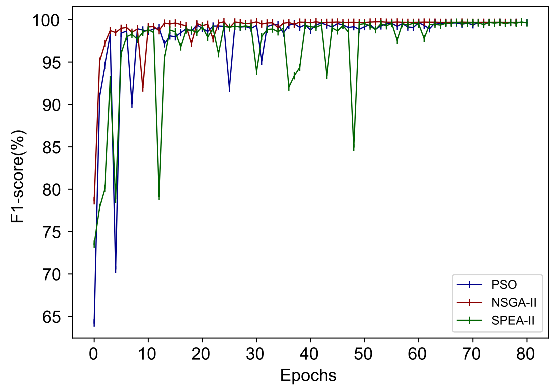
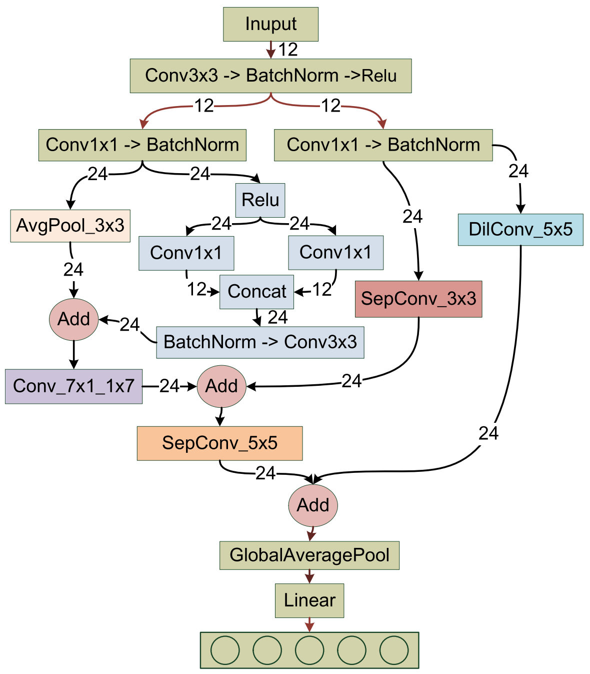


FIGURE 10. Validation F1-score (%) curves of the architectures based on MOPSO, NSGA-II, and SPEA-II on IDS2012 dataset during training.

When evaluating the complexity of the model, that is, the amount of calculations of the model, we use a naïve solution method. In the previous design of the search space, there are many convolution operation blocks in which a large number of point multiplication and accumulation forms are included. Therefore, in addition to the FLOPs, we use the MAccs to measure the amount of calculations of the model. The lightweight of the model was evaluated by calculating the number of parameters of the model. To make the model as light as possible, we reduce the number of input channels into the convolution kernel. To evaluate the model speed, we use a more intuitive comparison method to compare the results of the inference experiments.

The final comparison results are shown in Table 6. It is clear that the model searched using NAS is better than the model designed by hand in terms of the model complexity and operating speed.

In Table 6, the MAccs and FLOPs can be used to indirectly evaluate the model speed, and we directly measure the running speed of the model on the GPU using inference experiments. Intuitively, the higher the FLOPs of the model are, the higher the complexity of the model, which will lead to a slower running speed. The model based on MOPSO has the lowest complexity and the fastest running speed. The speed of the search model based on NSGA-II is slightly lower than



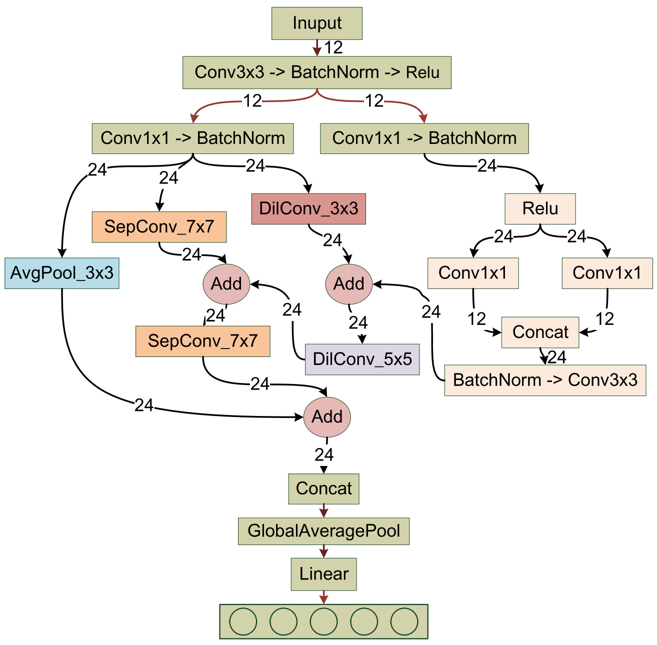


FIGURE 11. Optimal architecture searched based on NSGA-II on the IDS2012 and ISCX VPN datasets.

TABLE 6. Comparison between the Pareto optimal solution obtained by NSGA-II search in our designed search space (decreased spatial ratio and number of channels) and other classification algorithms on the IDS2012 and ISCX VPN datasets.

Dataset	Method	FLOPs(MB)	F1-score(%)	Precision score(%)	Recall score(%)
IDS2012	CNN	26.995	98.96	98.96	98.96
	CNN + Metric Learning	28.363	99.71	99.71	99.71
	KNN	-	95.862	95.932	95.825
	LR	-	54.191	74.827	60.890
	RF	-	97.679	97.616	97.754
	DT	-	97.258	97.317	97.204
	XGBoost	-	97.637	97.813	97.489
	NAS	**11.501**	**99.806**	**99.802**	**99.809**
ISCX VPN	CNN	12.372	95.43	95.58	95.48
	CNN + Metric Learning	12.995	98.56	98.53	98.53
	KNN	-	98.56	98.53	98.53
	LR	-	68.12	67.19	67.19
	RF	-	23.99	20.25	22.80
	DT	-	85.57	85.21	84.91
	XGBoost	-	84.88	84.04	84.16
	NAS	**4.718**	**99.369**	**99.374**	**99.321**

that of NSGA-II. However, the model based on SPEA-II is not the most complex, but the running speed is the lowest. in the research on ShuffleNet V2 [36] stated that the model complexity is not the only factor affecting the speed of the model, and the memory access cost (MAC) is also important. In other words, packet convolution can reduce the parameters of the model, but it can slow down the running speed. There are many packet convolutions in the model searched by SPEA-II (more than the other two models), which decreases the model speed.

E. COMPARISON WITH OTHER MODELS

The above analysis shows that it is the best to choose the search space with a reduced space ratio, fewer channels, and use NSGA-II as the search strategy. Therefore, we train the searched architecture from scratch on the training set. The ML classification algorithms (KNN, LR, RF, DT, and XGBoost) and two manually designed CNN network models are trained on the IDS2012 and ISCX VPN datasets. The results are shown in Table 5. The table shows that the network obtained from the search has good performance in terms of the classification and model complexity, so the designed search space is applicable and the search strategy performs well in this task. Fig. 11 shows the best architecture searched.

V. CONCLUSION

This paper studies traffic classification models based on NAS. To conduct the network search, we designed traffic classification network search architectures using NSGA-II, and compared the results with SPEA-II and MOPSO. First, we conducted ablation experiments to prove that the designed search space is effective, and the search space that reduces the spatial ratio and the number of channels can search for PO based on their FLOPs and F1-score. Second, the search strategy is changed in the optimal search space, and the accuracies and search times of the models are evaluated based on three optimization algorithms. It is concluded that the model searched by NSGA-II has the highest F1-score, the MOPSO search model can save time and energy, and the models searched by SPEA-II has the lowest F1-score. Finally, we designed inference experiment under the same GPU condition and prove that the MOPSO search model has the lowest complexity and the fastest running speed. When there is little difference in the search time and complexity, we believe that the weighted F1-score is the most important metric, and the number of optimal solutions in the PS obtained by NSGA-II is the largest. Hence, NSGA-II is more suitable than the other two optimization algorithms for the traffic classification problem. In all of the cases mentioned above, the weighted F1-score and speed of the resulting model exceeded that of the artificially designed model.

ACKNOWLEDGMENT

The authors are grateful to the editor and anonymous reviewers for their suggestions in improving the quality of the paper. AJE has refined this article before its first submission, and relevant supporting material is provided in ''Supplental File for Review''. This work was supported by the National Natural Science Foundation of China (61871046).

REFERENCES

[1] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *Proc. 4th ACM SIGCOMM Conf. Internet Meas. (IMC)*, Sicily, Italy, 2004, pp. 121–134.

[2] N. Weng, L. Vespa, and B. Soewito, "Deep packet pre-filtering and finite state encoding for adaptive intrusion detection system," *Comput. Netw.*, vol. 55, no. 8, pp. 1648–1661, Jun. 2011.

[3] D.-W. Kim, G.-Y. Shin, and M.-M. Han, "Analysis of feature importance and interpretation for malware classification," *Comput., Mater. Continua*, vol. 65, no. 3, pp. 1891–1904, 2020.

[4] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at Internet scale," in *Proc. Netw. Distrib. Syst. Secur. Symp.* San Diego, CA, USA: IEEE Computer Society, 2016, pp. 1–15.

[5] R. Dubin, A. Dvir, O. Pele, and O. Hadar, "I know what you saw last minute—Encrypted HTTP adaptive video streaming title classification," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 3039–3049, Dec. 2017.

[6] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 63–78, Jan. 2018.

[7] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing Android encrypted network traffic to identify user actions," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 114–125, Jan. 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2012, vol. 25, no. 2, pp. 1097–1105.

[9] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$Deep-Full-Range$: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[10] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[11] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, IL, USA, Aug. 2013, pp. 847–855.

[12] H. Wallach, "Computational social science & computer + social data," *Commun. ACM*, vol. 61, no. 3, pp. 42–44, 2018.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[14] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding convolution for semantic segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Lake Tahoe, NV, USA, Mar. 2018, pp. 1451–1460.

[15] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 636–644.

[16] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proc. Int. Conf. Learn. Represent.* Vancouver, BC, Canada: Vancouver Convention Center, 2018, pp. 13–25.

[17] F. Al-Obaidy, S. Momtahen, M. F. Hossain, and F. Mohammadi, "Encrypted traffic classification based ML for identifying different social media applications," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, Edmonton, AB, Canada, May 2019, pp. 1–5.

[18] V. A. Muliukha, L. U. Laboshin, A. A. Lukashin, and N. V. Nashivochnikov, "Analysis and classification of encrypted network traffic using machine learning," in *Proc. 23rd Int. Conf. Soft Comput. Meas. (SCM)*, St. Petersburg, Russia, May 2020, pp. 194–197.

[19] C. Luo, S. Su, Y. Sun, Q. Tan, M. Han, and Z. Tian, "A convolution-based system for malicious URLs detection," *Comput., Mater. Continua*, vol. 62, no. 1, pp. 399–411, 2020.

[20] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Beijing, China, Jul. 2017, pp. 43–48.

[21] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[22] S. Richard, P. Charles, and S. Dawn, "Differentiable neural network architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)* Vancouver, BC, Canada: Vancouver Convention Center, 2018, pp. 1–4.

[23] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, 2018, pp. 1–13.

[24] R. Luo, F. Tian, T. Qin, and T. Liu, "Neural architecture optimization," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2019, pp. 1–12.

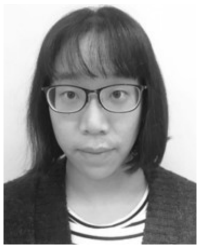
[25] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 2815–2823.

[26] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.

[27] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, Prague, Czech Republic, Jul. 2019, pp. 419–427.
[28] J. Jiang, F. Han, Q. Ling, J. Wang, T. Li, and H. Han, "Efficient network architecture search via multiobjective particle swarm optimization based on decomposition," *Neural Netw.*, vol. 123, pp. 305–316, Mar. 2020.
[29] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 4780–4789.
[30] Z. Lu, K. Deb, E. Goodman, B. Wolfgang, and N. B. Vishnu, "NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Glasgow, U.K.: Newcastle Univ., 2020, p. 3.
[31] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 2902–2911.
[32] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 1388–1397.
[33] F. E. Fernandes Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm Evol. Comput.*, vol. 49, pp. 62–74, Sep. 2019.
[34] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," 2019, *arXiv:1908.00709*. [Online]. Available: http://arxiv.org/abs/1908.00709
[35] R. Elshawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," 2019, *arXiv:1906.02287*. [Online]. Available: http://arxiv.org/abs/1906.02287
[36] N. Ma, X. Zhang, H. T. Zheng, and S. Jian, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, 2018, pp. 116–131.
[37] P. Molchanov, S. Tyree, T. Karras, A. Timo, and K. Jan, "Pruning convolutional neural networks for resource efficient inference," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, 2017, pp. 1–17.
[38] Y. Shu, W. Wang, and S. Cai, "Understanding architectures learnt by cell-based neural architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, 2020, pp. 1–21.
[39] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1–14.
[40] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 2423–2432.
[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9.
[43] L. M. Herstein, Y. R. Filion, and K. R. Hall, "Evaluating the environmental impacts of water distribution systems by using EIO-LCA-based multiobjective optimization," *J. Water Resour. Planning Manage.*, vol. 137, no. 2, pp. 162–172, Mar. 2011.
[44] C. Li, X. Yuan, C. Lin, M. Guo, W. Wu, J. Yan, and W. Ouyang, "AM-LFS: AutoML for loss function search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 8409–8418.
[45] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, vol. 112, no. 11, pp. 815–832.
[46] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*. Toulon, France: Palais des Congrès Neptune, 2017, pp. 1–16.
[47] G. Bender, H. Liu, B. Chen, G. Chu, S. Cheng, P.-J. Kindermans, and Q. V. Le, "Can weight sharing outperform random architecture search? An investigation with TuNAS," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 14311–14320.
[48] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Vienna, Austria, Jul. 2018, pp. 1595–1598.
[49] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Hong Kong, May 2017, pp. 1–8.
[50] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del-Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for DDoS attack detection," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 2, pp. 876–889, Jun. 2020.
[51] J. Mei, Y. Li, X. Lian, X. J. Jin, and L. J. Yang, "AtomNAS: Fine-grained end-to-end neural architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, 2020, pp. 1–14.
[52] D. H. Song, C. Xu, X. Jia, and Y. Y. Chen, "ESR-EA: Efficient residual dense block search for image super-resolution," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 12007–12014.
[53] L. W. Yao, H. Xu, W. Zhang, X. D. Liang, and Z. G. Li, "SM-NAS: Structural-to-modular neural architecture search for object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 12661–12668.
[54] C. Jiang, H. Xu, W. Zhang, X. Liang, and Z. Li, "SP-NAS: Serial-to-parallel backbone search for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11860–11869.
[55] A. Abayomi-Alli, S. Misra, L. Fernández-Sanz, O. Abayomi-Alli, and A. R. Edun, "Genetic algorithm and tabu search memory with course sandwiching (GATS_CS) for university examination timetabling," *Intell. Automat. Soft Comput.*, vol. 26, no. 3, pp. 385–396, 2020.
[56] P. K. Ray, S. Nandkeolyar, C. S. Lim, and I. N. W. Satiawan, "Demand response management using non-dominated sorting genetic algorithm II," in *Proc. IEEE Int. Conf. Power Electron., Smart Grid Renew. Energy (PESGRE)*, Cochin, India, Jan. 2020, pp. 1–6.
[57] H. Chen and S. Kuo, "Active detecting DDOS attack approach based on entropy measurement for the next generation instant messaging app on smartphones," *Intell. Automat. Soft Comput.*, vol. 25, no. 1, pp. 217–228, 2019.
[58] L. Shen, X. Chen, Z. Pan, K. Fan, F. Li, and J. Lei, "No-reference stereoscopic image quality assessment based on global and local content characteristics," *Neurocomputing*, vol. 424, pp. 132–142, Feb. 2021, doi: 10.1016/j.neucom.2020.10.024.
[59] Z. Pan, X. Yi, Y. Zhang, B. Jeon, and S. Kwong, "Efficient in-loop filtering based on enhanced deep convolutional neural networks for HEVC," *IEEE Trans. Image Process.*, vol. 29, pp. 5352–5366, 2020.
[60] Z. Pan, X. Yi, Y. Zhang, H. Yuan, F. L. Wang, and S. Kwong, "Frame-level bit allocation optimization based on video content characteristics for HEVC," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 16, no. 1, pp. 1–20, 2020.
[61] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Proc. Evol. Methods Design Optim. Control Appl. Ind. Problems*, 2001, pp. 95–100.
[62] C. A. C. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. Congr. Evol. Comput.*, Honolulu, HI, USA, Dec. 2002, pp. 1051–1056.
[63] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[64] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.
[65] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[66] T. Pamulapati, R. Mallipeddi, and P. N. Suganthan, "$I_{SDE}+$—An indicator for multi and many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 346–352, Apr. 2019.
[67] Z. Li, X. Wang, S. Ruan, Z. Li, C. Shen, and Y. Zeng, "A modified hypervolume based expected improvement for multi-objective efficient global optimization method," *Struct. Multidisciplinary Optim.*, vol. 58, no. 5, pp. 1961–1979, Nov. 2018.
[68] Y. Sun, G. G. Yen, and Z. Yi, "IGD indicator-based evolutionary algorithm for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 173–187, Apr. 2019.
[69] X. He, X. Fu, and Y. Yang, "Energy-efficient trajectory planning algorithm based on multi-objective PSO for the mobile sink in wireless sensor networks," *IEEE Access*, vol. 7, pp. 176204–176217, 2019.
[70] H. Mofid, H. Jazayeri-Rad, M. Shahbazian, and A. Fetanat, "Enhancing the performance of a parallel nitrogen expansion liquefaction process (NELP) using the multi-objective particle swarm optimization (MOPSO) algorithm," *Energy*, vol. 172, pp. 286–303, Apr. 2019.

[71] S. Khalid, A. Ishfaq, and K. Mohammad E., "An evolutionary approach to optimize data center profit in smart grid environment," in *Proc. 2nd Int. Conf. Data Intell. Secur. (ICDIS)*, South Padre Island, TX, USA, Jun. 2019, pp. 89–96.

[72] M. Ohki, "Effectiveness of NSGA-II with linearly scheduled Pareto-partial dominance for practical many-objecitve nurse scheduling," in *Proc. 7th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Prague, Czech Republic, Jun. 2020, pp. 581–586.



XIAOJUAN WANG received the Ph.D. degree in electronic science and technology from the Beijing University of Posts and Telecommunications. She is currently an Associate Professor with the School of Electronic Engineering, Beijing University of Posts and Telecommunications. Her research interests include deep learning, complex networks, and human gesture recognition.

XINLEI WANG received the B.E. degree in communication engineering from the Shandong University of Science and Technology, Shandong, China, in 2019. She is currently pursuing the M.A.Sc. degree with the Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include deep learning and computer network security.

LEI JIN received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree. His research interests include complex networks and deep learning.

RENJIAN LV received the B.S. degree in software engineering from Beihang University. He is currently pursuing the Ph.D. degree in information security with the Beijing University of Posts and Telecommunications. Since 2006, he has been working with the North China Institute of Computing Technology, as an Engineer and the Director of the Research Department. He has early 15 years' experience on technology and industry development research related to ICT technologies, such as wireless networks, embedded systems, information security, and the Internet of Things. He was a Junior Expert of the EU-China IOT Advisory Group, and an Expert in the ISO/IEC JTC1/WG10.

BINGYING DAI received the M.S. degree in statistics from Xiamen University, in 2017. She is currently pursuing the Ph.D. degree in statistics with Colorado State University. Her research interests include network analysis and machine learning.

MINGSHU HE received the B.E. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree. His research interests include network security, anomaly detection, and machine learning.

TIANQI LV received the B.E. degree in electronic information engineering from Yanshan University, Hebei, China, in 2015. He is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include deep learning and artificial intelligence.

• • •