Balanced Neural Architecture Search and Its Application in Specific Emitter Identification

Mingyang Du, Xikai He, Xiaohao Cai and Daping Bi

***Abstract*—The performance of a single neural network can vary unexpectedly corresponding to different classification tasks, and thus the network with fixed structure may lack flexibility and often lead to overfitting or underfitting. It is urgent, also the main objective of this paper, to deal with the limitation of the fixed neural network structure on classifying radar signals in different electromagnetic environments. We in this paper propose a variable network architecture search (NAS) mechanism, called *balanced-NAS framework*, and apply it in specific emitter identification (SEI) to greatly improve the flexibility of model searching. In the proposed balanced-NAS framework, a "block-cell" structure and a controller based recurrent neural network (RNN) are utilized to design models automatically according to external environment. In particular, a balance function is also proposed and utilized in the balanced-NAS framework, acting on the RNN controller to take both the validation accuracy and computational budget into consideration while searching for ideal models. The efficiency of the searching process is further enhanced by exploiting a progressive strategy to design simple and complicate child models where unpromising ones after each evaluation process are obsoleted to release searching space. Simulations and experiments indicate that the proposed balanced-NAS framework is extremely efficient and outperforms the conventional algorithms in classifying radar signals in different environments.**

***Index Terms*—Specific emitter identification, time-frequency distribution, neural architecture search, radar, signal processing.**

I. Introduction

SPECIFIC emitter identification (SEI) is the process of discriminating or identifying different emitters based on the radio frequency fingerprint features extracted from the received signals [1]. The application of SEI techniques has been intensively studied in military communication, radar system, cognitive radio and self-organized network [2]. However, the increasing complexity of the spatial spectrum makes this task more complicated than ever before. For example, sensors deficiency and low signal-to-noise ratio (SNR) environments may cause measurement loss or error, which finally results in poor performance [3]. Therefore, SEI in the complex electromagnetic environment is inherently an urgent and extremely challenging problem. In this paper, we endeavor to propose new efficient methods to pave the way for SEI in complex electromagnetic environments.

M. Du, X. He and D. Bi are with College of Electronic Engineering, National University of Defense Technology, Huangshan Road No. 460, Hefei, China (e-mail: dumingyang17@nudt.edu.cn).

X. Cai is with School of Electronics and Computer Science, University of Southampton, University Road, Southampton, SO17 1BJ, UK (e-mail: x.cai@soton.ac.uk).

Pulse description word [4] is one of the most common and easiest ways to describe radar signal inter-pulse characteristics which mainly gather parameters including radio frequency (RF), pulse amplitude (PA), time-of-arrival (TOA), direction-of-arrival (DOA) and pulse width (PW). When the signal density in the electromagnetic environments is low and the majority of radars are conventional systems, the classification using inter-pulse features is feasible [5]. However, in modern radar systems, more sophisticated signal waveforms are used and thus only inter-pulse features may not be enough to separate and classify those received pulses. To identify the radar emitter in such environments, we need to explore the detailed structure inside each pulse, called intra-pulse features [6]. This is because an emitter has its own electrical signal structure inside each of its transmitted pulses due to both intentional and unintentional modulations [7]. Time-frequency analysis is one of the most effective approaches to extract intra-pulse features of radar signals. There are some conventional techniques such as short-time Fourier transform (STFT) [8], wavelet transformation [9], Wiger-Ville distribution [10], and Choi-Williams distribution [11]. Some state-of-the-art SEI techniques based on these time-frequency analyses are proposed in e.g. [12] [13]. The basic procedures include extracting the time-frequency distribution of the given signal, applying morphological processing on the extracted time-frequency images and finally designing classifiers to produce the outputs [14]. In the following, we briefly recall some popular classifiers which are related to the focus of this paper.

Some conventional classifiers like support vector machine, k-nearest neighbor and random forest have broadly been applied in SEI research [15] [16]. However, they have limited capacity to express complex functions. Moreover, for complicate classification problems, their generalization abilities are rather limited. Machine learning system has the advantages of self-organizing and self-learning in the course of the solution to recognition tasks. Their usefulness has been proved in many applications such as speech, image, and video recognition. Models like convolutional neural network (CNN) and recurrent neural network (RNN) employ a data-driven approach to learn discriminative features from raw sensor data to infer complex, sequential and contextual information in a hierarchical manner [17]. Much novel SEI research work based on CNN has been developed in recent years. Their high accuracies (up to 90%) on distinguishing different radar signals among a large scale of samples are strongly proved [18] [19].

Recently, some state-of-the-art neural networks like Inception-v3 and ResNet-152 are utilized to classify time-frequency distribution images [20]–[22]. In these works, SEI is

regarded as a problem in image classification. However, some discrepancies between SEI and image classification cannot be overlooked. For example, data sets (like CIFAR-10, CIFAR-100, ImageNet, SPORT8, MIT67 and FLOWERS102) in the image classification field are fixed. On the contrary, data sets in SEI are variable since radar signal samples might come from different electromagnetic environments and vary in size. Note that the performance of a single network can be different when dealing with variable data sets. More specifically, on the one hand, a deep network might have a good performance when faced with an extremely complicated data set with many categories and samples. On the other hand, it might experience overfitting because of the vanishing gradient when the target data set is quite small, with a great waste of computation resource as well.

Considering the SNR as a factor which generates difference between data sets in SEI, it is much easier to qualitatively analyze and recognize those signals in the high rather than low SNR environment. The reason is obvious, i.e., the noise will corrupt the true radar signals and thus cause many difficulties. Therefore, a much deeper network might work well with high noise level, whereas a narrower network would be adequate for low noise level. This implies that fixed structure networks might not be suitable for the SEI task. In addition, the procedure of designing networks manually is time consuming and error-prone. All these factors motivate us to design a mechanism that is capable of automatically constructing an optimized network structure for different environments.

Nowadays, there is a growing interest in computer vision about automated neural architecture search (NAS) methods [23]. The state-of-the-art techniques about NAS usually fall into one of the two categories: evolutionary algorithms (see e.g. [24] [25]) or reinforcement learning (see e.g. [26] [27]). Although these methods could be able to learn network structures that outperform manually designed architectures, they need considerable computation resources. For instance, the reinforcement learning method in [27] trains and evaluates 20,000 models across 500 P100 GPUs over 4 days. It is obviously unrealistic to afford such huge price in SEI. Several directions to accelerate the model searching and evaluating procedures to improve the efficiency are as follows. The work in [28] proposed an efficient NAS (ENAS) which uses parameter sharing to eschew training every child model from scratch to convergence, and notably, is 1000 times less expensive than the standard NAS. On the CIFAR-10 data set, for example, ENAS finds a novel architecture and achieves 2.89% test error, which is on a par with the 2.65% test error of the expensive method in [27]. The work in [29] proposed a simple iterative approach called Hillclimbing for NAS (NASH). At each step, it trains the resulting child networks with short optimization runs of cosine annealing, and moves to the most promising child network. NASH searches and trains a single network on CIFAR-10 with an error rate below 6% in roughly 12 hours on a single GPU. In [30], a progressing NAS (PNAS) was proposed which uses heuristic search to find the optimized cell structures, beginning with simple models to complex ones, and pruning out bad structures as searching ongoing. PNAS learns a model or surrogate function which can predict the performance of a structure without training it. This approach could achieve the state-of-the-art classification accuracy on CIFAR-10 and ImageNet, and it is 8 times faster than computing all child models. All of these above-mentioned NAS methods provide a great potential for SEI.

In this paper, we challenge the SEI problem in different electromagnetic environments. The main contributions are as follows.

1) A variable NAS mechanism – called balanced-NAS framework – based on the "block-cell" mode is proposed as an alternative of the fixed structure networks in conventional SEI algorithms such that the flexibility of the model searching is greatly improved. The details about the block and cell can be found in section IV. This mechanism can generate considerable promising network structures automatically without manpower, and thus possesses high potential to find more outstanding models. Moreover, different effects of the block and cell are tested and compared in the experiment, which provides guidances for adding network complexity when tackling more difficult classification tasks.
2) A novel balance function is proposed to trade off the accuracy and efficiency (computational cost) of a network in NAS. The amount of trainable parameters in neural networks is adopted as a metric to evaluate the efficiency, plus the validation accuracy as another criterion to ensure the output of the network also meets the accuracy requirement. Under the two criteria, the learned balance function would stimulate the newly constructed network to choose the most optimized models with both high accuracy and high efficiency. We emphasize that this high level of accuracy and efficiency is very hard, if not impossible, to achieve by a standard approach which only uses a network with fixed structure for SEI in different electromagnetic environments.
3) The proposed method is evaluated on a radar signal data set containing seven categories which are commonly used in modern radar systems. It includes linear frequency modulation (LFM) with positive and negative slope, sine frequency modulation, polyphase codes (P1, P2, P3, P4), which have the similar time-frequency feature (ridge line distribution). Moreover, different degrees of environmental noise are added, which brings more difficulties to classification. The experimental results show that the proposed method can design/select the most optimized models which could reach a state-of-the-art accuracy and simultaneously take the cheapest computational cost.

The reminder of this paper is organized as follows. Firstly, we introduce the radar signal model, radar system and the STFT approach in Section II. The previous methods in NAS about the search space, search strategy and performance estimation are then recalled in Section III. The new NAS framework with variable structure network and progressive search strategy is designed in Section IV. The novel balance function is proposed in Section V to trade off the accuracy and efficiency when searching and constructing networks.

Finally, Section VI reports extensive experiments to evaluate and compare the proposed method with the state-of-the-art techniques. The conclusion drawn from this paper and future work is given in Section VII.

II. Preliminary

To start, we first recall the representation of modeling discrete radar signals and introduce the radar system. After that, we review the ways of extracting time-frequency features from radar signals using discrete STFT. Finally, several typical radar signals are analyzed and compared in different environments.

A. Radar signal

In discrete setting, the radar signal u can be modeled as discrete time complex samples [31]. Let $m \in \mathbb{Z}$ and $T \in \mathbb{R}^+$ be the sample index and the sampling interval, respectively. Then the m-th sample of the radar signal $u(mT)$ reads

$$u(mT) = a(mT)\exp(j\theta(mT)) + n(mT), \tag{1}$$

where $a(mT)$ is the signal envelope within the pulse interval, $\theta(mT)$ is the instantaneous phase and $n(mT)$ denotes the additive Gaussian white noise.

B. Radar system

A typical radar system generally consists of RF signal reception unit, parameter measurement and deinterleaving unit, parameter estimation unit and emitter identification unit [1], see the flow chart in Fig. 1. In detail, a set of RF down-converters firstly translates received RF signals to the intermediate frequency. Next, the intersected radar signals are deinterleaved, followed by parameter estimation including e.g. RF, PW, PA, TOA, angle-of-arrival (AOA) consisting of pulse description word. Other feature parameters of emitters like pulse repetition interval and antenna scan style will also be estimated. After that, the emitter can be recognized by comparing these parameters with the known threat database.

In deep-learning-based methods, the intermediate process like parameter estimation which extracts hand-engineered features could be omitted. The raw radar signals after some preprocessing operations are formed as input data and fed to deep neural networks. In this paper, we assume that the received signals have been deinterleaved in advance, and are supposed to be transformed into images through the time-frequency analysis approach which is introduced in the following subsections. After that, at the emitter identification stage, the proposed method will search the most optimized neural networks, which are both accurate and efficient, according to different input radar signals.

C. Short time Fourier transformation (STFT)

STFT, one of the most classical techniques in signal processing, has been widely used in e.g. audio signal [32], radar signal [33], fingerprint image [34], and electroencephalography [35]. It maps a non-stationary signal onto a two-dimensional plane with time and frequency to analyze the frequency components and variations of signals. In discrete setting, it divides signals (e.g. radar signals) into small sequential or overlapping data frames and then fast Fourier transformation is applied to each one, with output containing both the temporal and spectral information; see e.g. [36], [37] for more details. The time-frequency distribution of the m-th input discrete radar signal, $X(mT, f)$, is defined by

$$X(mT, f) = \sum_{k=0}^{N-1} u(kT)h^*((k-m)T)\exp(-j\frac{2\pi f}{N}kT), \tag{2}$$

where f is the frequency component, $h^*(\cdot)$ is the time window function, and N is the length of $u(mT)$.

The STFT uses $h^*(kT)$ to intercept $u(mT)$ and conducts Fourier transformation on the intercepted part. By shifting k, the central position of $h^*(kT)$ is shifted and the Fourier transformation at different time is attained. Therefore, the STFT compensates the Fourier transformation's lack of location analysis capability. The time-frequency resolution of STFT techniques is inversely related to the window length. Increasing the window length will increase the frequency resolution and simultaneously reduce the frequency tracking capability of the representation.

D. STFT time-frequency images

In this paper, we extract the time-frequency feature of radar signals to classify different emitters. By STFT, the time-series radar signal can be transferred into 2D images which contain individual time-frequency information. However, noise is one of the most normal factors, which influences the classification accuracy. Generally, the SNR in signal processing is used to measure the level of a desired signal to the degree of noise power [38]. It is defined as $10\log_{10} P_s/P_n$, where P_s and P_n are the effective powers of the signal and the noise, respectively. We consider the LFM with positive slope and M-bit polyphase codes including P1, P2, P3 and P4 radar signals with different SNR. LFM is one of the most common intra-pulse modulation types where the frequency varies linearly with time. The M-bit polyphase codes like P1, P2, P3 and P4, which modulate the signal phase by some specific codes, are listed in Table I, where $p, q = 1, 2, ...M, M = 64$.

Fig. 2 gives the time-frequency images of the 64-bit polyphase codes transferred by STFT. The discriminating features are clearly visible when the noise is free, see the top row of Fig. 2. Specifically, polyphase codes and LFM have the similar time-frequency distribution which contains several ridge lines in the plane. For the polyphase code signal, during a single coding cycle, P3 code has two main ridge lines and several minor ridge lines, whereas codes P1, P2 and P4 all have one main ridge line and several minor ridge lines. During two coding cycles, P3 code has three main ridge lines, while codes P1, P2 and P4 all have two main ridge lines. For the LFM signal, since the frequency varies linearly with time, there is a single oblique line in the time-frequency distribution plane. However, in the bottom row of Fig. 2, we can see that it becomes extremely complicated to extract the aforementioned features and distinguish these signals when the SNR drops to -10dB.

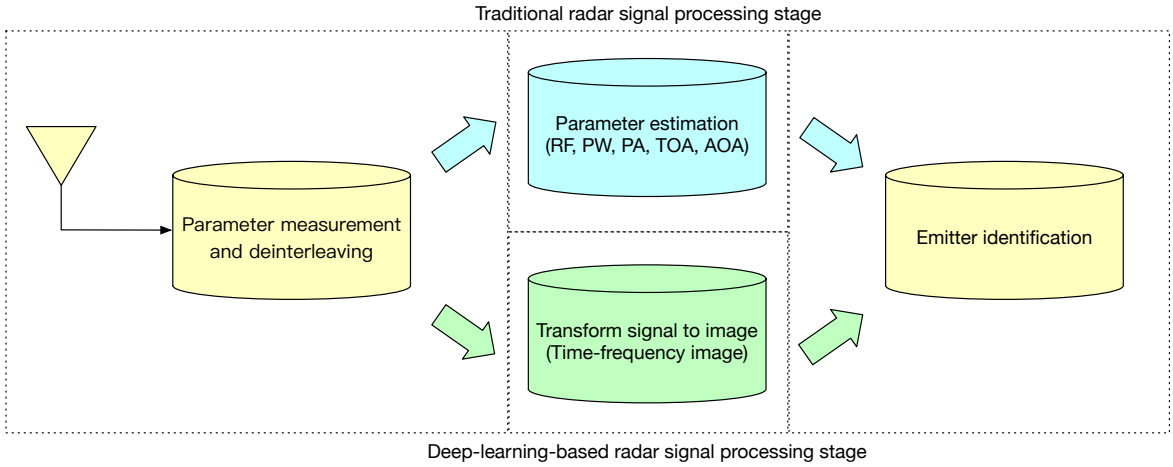


Fig. 1. Radar system flow chart which illustrates the comparison between the traditional method (light blue part) and the proposed deep-learning-based method (light green part). The deep-learning-based method in this paper does not need the parameter estimation step where the features estimated are replaced by high-level features extracted by the hidden layers in deep models.

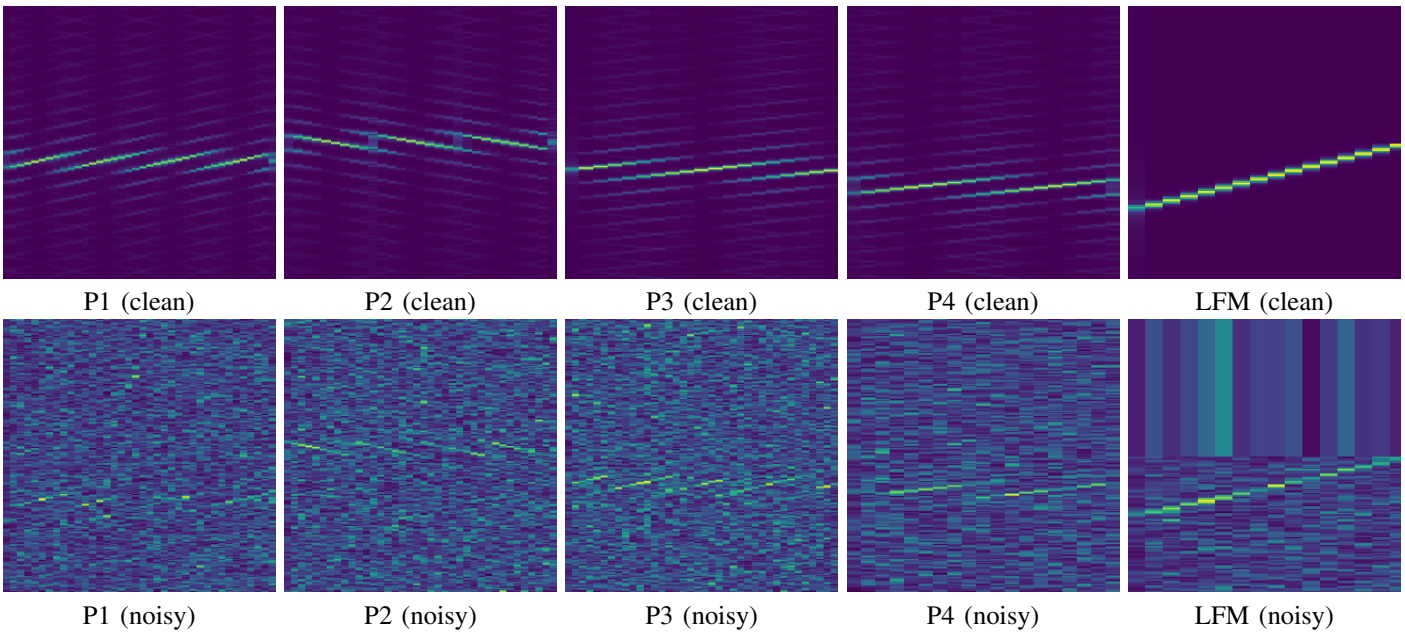


Fig. 2. Examples of some time-frequency distribution images transferred from 64-bit polyphase codes of radar signals. The top row is in the noise-free environment and the bottom row is in the situation of SNR $= -10$dB. Columns from left to right give the P1, P2, P3, P4 and LFM radar signals, respectively.

TABLE I
EXAMPLES OF 64-BIT POLYPHASE CODES

Phase	Formula
P1	$-\frac{\pi}{M}[(M-(2q-1))][(q-1)M+p-1]$
P2	$-\frac{\pi}{2M}[2p-1-M][2q-1-M]$
P3	$-\frac{\pi}{M}(p-1)^2$
P4	$-\frac{\pi}{M}(p-1)^2-\pi(p-1)$

III. RELATED WORK

Currently employed architectures for SEI were mostly developed manually with experts' advice, which is time consuming and error-prone. Automated NAS methods therefore have been attracting more and more interests. The intrinsic NAS can be categorized into three dimensions, i.e., the search space, search strategy and performance estimation [25]. In this section we briefly recall some widely-utilized techniques in terms of these aspects which are in relation to the work here.

A. Block-cell structure and search space

A block is a mapping from two input tensors to one output tensor. A cell is a convolutional neural network, which contains a certain number of blocks and maps one tensor to another, see e.g. Fig. 5. The size of a cell's output tensor depends on the stride and padding therein.

Let $\mathcal{I}_i$ be the input set containing all the outputs of previous cells. Let $\mathcal{O}$ be the set of operations on set $\mathcal{I}_i$, and $\mathcal{C}$ be the set of actions on set $\mathcal{O}$. Here $\mathcal{C}$ contains the elementwise addition and concatenation. Let $\mathcal{B}_b$ be the b-th block in a single cell, where $b = 1, 2, ..., B$. For example, a block $\mathcal{B}_b$ in [27] is

composed of five tuples (I_1, I_2, O_1, O_2, C), where $I_1, I_2 \in \mathcal{I}_i$ are the inputs to the block, $O_1, O_2 \in \mathcal{O}$ are the operations applied to the inputs I_1 and I_2, and $C \in \mathcal{C}$ is the action of combining O_1 and O_2 which generates the output of the block. In other words, the output of the block (I_1, I_2, O_1, O_2, C) here is operations executed by either addition or concatenation.

The search space defines which architectures are under consideration in principle. The choice of the search space therefore largely determines the complexity and difficulty of the given searching problem. Many NAS techniques use the "block-cell" mechanism to construct basic units of a full CNN [26], [27]. This mechanism exploits knowledge from previous networks which contain a specific number of components such as convolutional layers, pooling layers, and fully connected layers. More specifically, all convolutional networks in the search space are "cells" with identical structure but different weights. A full CNN is a stack of cells at specific times. Searching for the optimized network architecture is therefore reduced to find the best cell structure, which has a great promotion in terms of efficiency compared to designing a whole network directly.

B. Search strategy and performance estimation

The search strategy details how to explore the search space. It is supposed to find well-performed architecture quickly and avoid premature convergence to local optimizations. Many different search strategies can be used to explore the space of a neural architecture, including the random search (RS), Bayesian optimization [39], evolutionary method [40] and reinforcement learning (RL) [26]. For example, the controller in [26] samples child networks with a certain probability among the search space. These child networks are then trained to converge to a validation accuracy as a reward to update the controller by proximal policy optimization (or Q learning [41]); the main framework is akin to the diagram shown in Fig. 3. The work in [42] made a comparison between the RS, RL and evolutionary method, and concluded that the RL and evolutionary method, with nearly equal test accuracy, outperform the RS.

Cell structures are searched directly in the full search space in [43], and thus it is difficult to navigate in such a huge space and there is lack of prior knowledge to recognize what model is better. The work in [30] proposed a progressive NAS method which searches cell structures in a hierarchical order, i.e., beginning with the simplest models to complex ones. For example, at the beginning all possible cell structures are with fixed $B = 1$ (i.e., every cell has only one block) in the queue. Every model in the queue will then be trained from scratch. After that the cell structures will be expanded with $B = 2$ (i.e., the number of blocks in every cell rises to 2) and analogous procedures are repeated until the predefined stopping criterion is satisfied.

It is generally unaffordable to train child models individually and evaluate their performance since the scale of child models is considerable. The design procedure of the control scheme shown in Fig. 3 is based on the property of the series of recurrent networks. The controller in NAS regarding

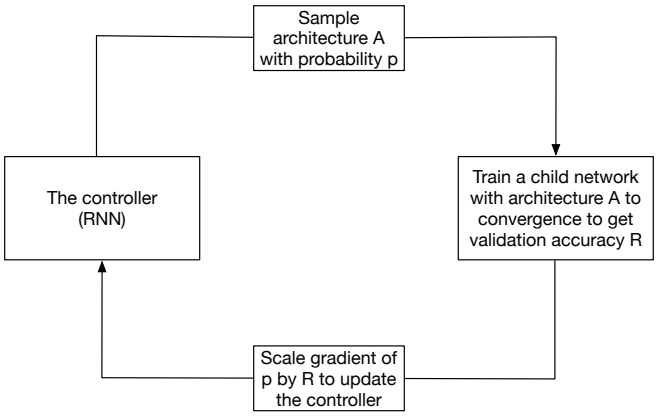


Fig. 3. NAS framework. Starting from the state at the top of the diagram, a child network with an architecture called 'A' is sampled with probability p from the search space. This child network will then be trained on the given data set from scratch. The obtained validation accuracy named R is viewed as a "reward signal" and used to update the controller. Iteratively, the controller can give higher probabilities to those architectures which have higher accuracy.

the quantitative analysis should have capacity to traverse the search space and generate arbitrary child models. Since the child models' complexity in the search space covers a large range, their corresponding strings will possibly possess variable length. RNN has advantages in processing the variable-length string and thus is naturally taken into consideration in the NAS framework. A long short-term memory (LSTM) – one of the most representative RNNs – was adopted e.g. in [30] as the controller to generate convolutional architectures.

More specifically, the learning scheme of the LSTM controller is illustrated in Fig. 4. Firstly, child models are transferred into a series of input strings by one-hot encoding and fed into the LSTM controller. Then, a one-unit fully connected layer following the LSTM embedded layer outputs a series of scalar values representing the probable validation accuracy of the input child models. By backward propagating the gradient of the loss between the predicted accuracy and real accuracy gained from the training procedure, the weights and bias in the controller will be optimized to get more accurate estimations. Consequently, candidate child models in the search space are to be divided into two groups, i.e., one used for training on the real data to update the LSTM contributes a smaller proportion, the other whose accuracy will be attained by estimation contributes a larger proportion. Finally, the LSTM approximates the true performance of each child model in the search space, which greatly improves the overall efficiency compared to training all child models from scratch.

Note that in this mechanism, the algorithm can be stimulated to find the most accurate models since it considers the accuracy only as the reward to train the LSTM and update weights. Other important factors, like efficiency, are not yet considered to design networks.

IV. VARIABLE STRUCTURE NETWORK

Since a network with fixed structure may lack flexibility when conducting classification tasks on different data sets, we in this section focus on variable structure network and

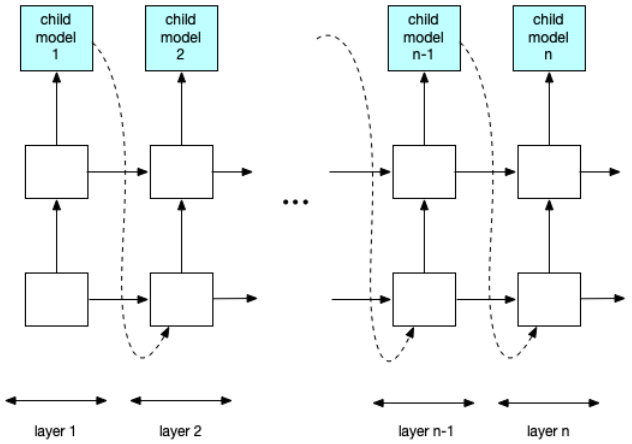


Fig. 4. LSTM controller. Each child model is encoded first and fed into the LSTM controller with its validation accuracy to repeatedly generate more promising individuals through back-propagation.

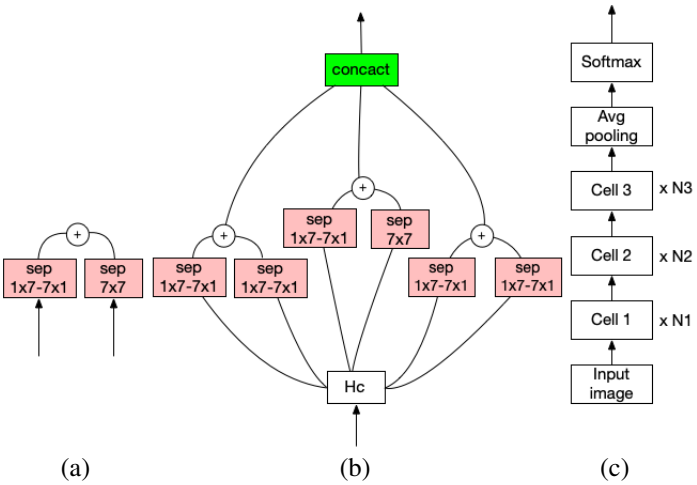


Fig. 5. Block-cell structure. (a): a block; (b): a cell which contains three blocks; and (c): a full CNN. Specifically, there is an average pooling layer before the softmax layer.

construct one with progressive search strategy, which will be used for SEI.

We first explicitly define the operation set $\mathcal{O}$ used in our network. Let $\mathcal{O}$ contain the following eight operations: 1) three depthwise separable convolutions called 'dc' whose sizes are 3x3, 5x5 and 7x7, respectively; 2) 1x7 followed by 7x1 convolution called '1x7-7x1-c'; 3) 3x3 dilated convolution called '3x3-d'; 4) 3x3 average pooling called '3x3-ap'; 5) 3x3 max pooling called '3x3-mp'; and 6) identity. The above generated set $\mathcal{O}$ contains less operations than that in [27] (i.e., containing 13 functions) since those used in [27] not directly related to the main focus of this work are abandoned here.

The "block-cell" structure introduced in Section III is used for our construction, see Fig. 5 as an example. Fig. 5 (a) represents a block consisting of two inputs, two operations (i.e., 7x7-dc and 1x7-7x1-c) and one combination operator (i.e., addition). Fig. 5 (b) represents a cell structure consisting of three blocks with structure similar to the one shown in Fig. 5 (a). Here, we describe this cell structure as {1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 7x7-dc, 1x7-7x1-c, 1x7-7x1-c}. A full CNN is then constructed by stacking this kind of cell for specific times, with a global average pooling before the softmax layer, see Fig. 5 (c).

The search space size of this kind of structure can be measured explicitly. For example, the size of the b-th block $\mathcal{B}_b$ is $|\mathcal{B}_b| = |\mathcal{I}_b|^2 \times |\mathcal{O}|^2 \times |\mathcal{C}|$, where $|\mathcal{I}_b| = 2+b-1$, $|\mathcal{O}| = 8$ and $|\mathcal{C}| = 2$. Therefore, for $b = 1$, there are $|\mathcal{B}_1| = 2^2 \times 8^2 \times 2 = 512$ child network structures. When $B = 5$, the total number of possible cell structures is $|\mathcal{B}_{1:5}| = \Pi_{i=1}^{5} |\mathcal{B}_i| = 1.8 \times 10^{16}$, which is too large to train them all. We therefore propose to only choose the top $K \ll |\mathcal{B}_{1:B}|$ child models sorted by accuracy for training and updating the controller, see Fig. 6, where M_{bj} indicates the j-th child model in the b-th block.

As discussed in Section III, the searching strategies proposed so far only feed back the validation accuracy as the reward to the controller to update the weight of each child model. This mechanism is able to find the most accurate models in the search space but overlooks other important metrics like efficiency of the network. In practice (e.g. in SEI)

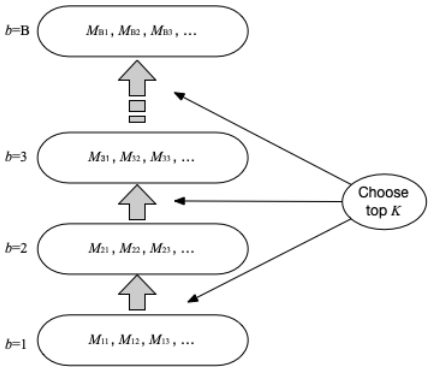


Fig. 6. Child models selection. In the NAS algorithm, considering the computational cost, only the top K child models in terms of high accuracy at each searching stage will be chosen.

high real-time performance is as important as the accuracy since actions like electronic jamming and combat effectiveness destruction are needed to be conducted immediately after classifying emitters.

To overcome this shortcoming within the current searching strategies, we propose a new search strategy which modifies the current reward principle by addressing both the accuracy and efficiency simultaneously. Compared to the framework shown in Fig. 3, a metric evaluating the efficiency of child models at each searching procedure is also considered in the NAS framework. The proposed NAS framework, coined as *balanced-NAS framework*, is shown in Fig. 7. The integration of the validation accuracy and network efficiency will be utilized to evaluate the potential of every candidate child model. The new reward value R is used to replace the former accuracy to update the weights and bias in the LSTM controller.

In the next section we propose a novel way to evaluate the model accuracy and efficiency. Briefly speaking, a balance function is designed to trade off different metrics. Again, the output of the proposed balance function will be a composite reward to update the controller.

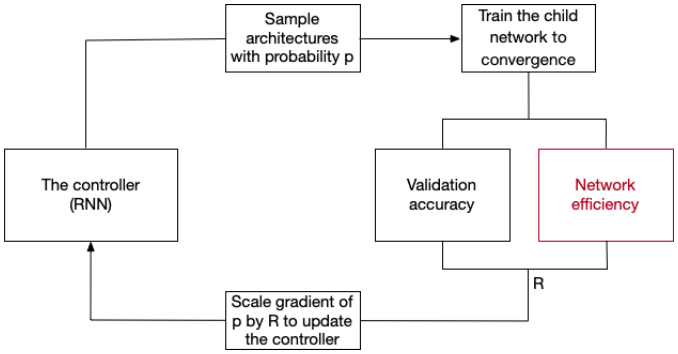


Fig. 7. The proposed balanced-NAS framework. To update the controller, the balanced-NAS framework ensures the efficiency and validation accuracy of every child model are both evaluated.

V. Proposed Balance Function

To optimize the search strategy in NAS and generate the most suitable architectures for various data sets, we propose a multivariate function, called *balance function* and denoted by $F(x, y)$, to replace the solely accuracy-based reward to train the controller, where $x > 0$ and $y > 0$ respectively represent the accuracy and the efficiency of a single network. The output value of this balance function $F(x, y)$ is regarded as "scores"; the higher the scores, the better the model performs. For the convenience of analysis, the balance function is split into two parts, $F_1(x)$ and $F_2(y)$, which denote the accuracy function and efficiency function, respectively. Note that $F_1(x)$ and $F_2(y)$ can also be viewed as the marginal distributions of the multivariate function $F(x, y)$ respectively with respect to y and x. For variable x, we directly use the validation accuracy at every epoch. The variable y, representing the efficiency of a network, can be measured by using the computational complexity (e.g. the number of trainable parameters or FLOPs) of a network. In this work the number of trainable parameters is selected as the criterion to evaluate the efficiency. More precisely, for CNNs, we mainly take the number of parameters regarding convolutional layers and fully connected layers into consideration [44], which can be computed by

$$n = \sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2, \tag{3}$$

where l is the index of the convolutional layer, d is the depth or the number of convolutional layers, n_l is the number of the filters in the l-th layer, s_l is the length of the filter, and m_l is the spatial size of the feature map (output size). Analogous to the accuracy x whose value range is $[0, 1]$, for the i-th child model in a specific search space, the normalized number of trainable parameters, say $y^{(i)}$, is defined as

$$y^{(i)} = \frac{n^{(i)}}{n_{\max}}, \tag{4}$$

where n_i is the number of trainable parameters of the i-th child model and $n_{\max}$ denotes the maximum number of trainable parameters of individual models in a specific search space.

TABLE II
Examples for the choices of the functions $F_1(x)$ and $F_2(y)$.

Function	$F_1(x)$	$F_2(y)$
Linear function	$ax + b, a > 0$	$ay + b, a < 0$
Exponential function	$a^x, a > 1$	$a^y, a < 1$
Power function	$x^n, n > 0$	$y^n, n < 0$
Sigmoid function	$\frac{1}{1+e^{-x}}$	$\frac{1}{1+e^{y}}$

A. Basic properties

The balance function $F(x, y)$ should follow the following two basic properties which reflect the relationship between the accuracy and the efficiency, i.e.,

$$F(x, y^{(i)}) > F(x, y^{(j)}), \quad \forall y^{(i)} < y^{(j)}; \tag{5}$$

$$F(x^{(i)}, y) < F(x^{(j)}, y), \quad \forall x^{(i)} < x^{(j)}. \tag{6}$$

The first inequality implies that the scores of more complicated models are lower than simpler models corresponding to the same accuracy, and the second inequality implies that more accurate models have higher priorities given the same efficiency. Accordingly, $F_1(x)$ and $F_2(y)$ are increasing and decreasing functions, respectively. Table II gives some functions (e.g., linear functions, exponential functions, power functions, and Sigmoid function) which can be used for functions $F_1(x)$ and $F_2(y)$.

It is also worth investigating the gradient of functions $F_1(x)$ and $F_2(y)$. Obviously, $F_1'(x) > 0$ and $F_2'(y) < 0$. A brief discussion about the comparison between $|F_1'(x)|$ and $|F_2'(y)|$ is given below. I) If $|F_1'(x)| > |F_2'(y)|$, this means that the weight of accuracy is larger than the efficiency in the balance function $F(x, y)$, and then the final result of the NAS will prefer the accuracy to the efficiency; II) if $|F_1'(x)| < |F_2'(y)|$, analogously, the final result of the NAS will prefer the efficiency to the accuracy; and III) if $|F_1'(x)| = |F_2'(y)|$, then the final result of the NAS will weigh the efficiency and the accuracy equally, which might be the category we shall use more often in practice if no prior knowledge is available.

The above three properties suggest that when the accuracy is already high (e.g. above 90%), it is wise to focus on searching more efficient models among those high-accurate ones, and thus the weight of efficiency is supposed to be larger than that of accuracy, i.e., $|F_1'(x)| < |F_2'(y)|$. On the other hand, if the accuracy is low (e.g. below 60%), it is necessary to continue searching more accurate models, and in this regard the weight of accuracy should be stressed compared to the efficiency, i.e., $|F_1'(x)| > |F_2'(y)|$.

All these properties are necessary for the balance function. The most natural way of defining $F_1(x)$ and $F_2(y)$ is to set

$$F_1(x) = x, \quad F_2(y) = y^{-1}. \tag{7}$$

The natural way of generating the balance function using $F_1(x)$ and $F_2(y)$ is

$$F(x, y) = F_1(x) * F_2(y) \tag{8}$$

or

$$F(x, y) = F_1(x) + F_2(y). \tag{9}$$

Note that the main focus of this paper is to show the excellent performance of the balance function in enabling the NAS algorithm to learn perfect models. The investigation of searching optimized pairs of $F_1(x)$ and $F_2(y)$ for the balance function for specific problems will be left for future work.

B. Performance evaluation

We define the following metrics which are useful to validate the performance of a model and gauge the influence of the balance function on choosing the most optimized models.

Let τ represent the accuracy and efficiency ratio of the i-th child model, which is calculated by

$$\tau = x_i / y_i, \tag{10}$$

where x_i and y_i are the accuracy and the generalization value of the computational cost (efficiency), respectively. A model with high value τ means it has the high accuracy and low computation on average, and therefore it is advisable to search the neural network possessing the largest τ.

Let sequences $A = \{a_1, a_2, ..., a_N\}$ and $A' = \{a'_1, a'_2, ..., a'_N\}$ represent models in a search space with descending order in terms of accuracy and the cost computed in (10), respectively. For example, if $i < j$, then $x_{a_i} \geq x_{a_j}$ and $\tau_{a'_i} \geq \tau_{a'_j}$. Let x_{A_M} denote the average accuracy of the top M models in A, i.e.,

$$x_{A_M} = \frac{1}{M} \sum_{i=1}^{M} x_{a_i}, \tag{11}$$

which measures the overall accuracy of the M candidate models. Let Δ denote the maximum accuracy difference between sequences A and A', which is computed by

$$\Delta = |x_{a_1} - x_{a'_1}|. \tag{12}$$

Note that Δ above can be used to verify the influence of the balance function on the accuracy, i.e., the larger the magnitude Δ, the higher the influence of the balance function on the accuracy.

VI. Experimental results

In this section, we test the proposed method in terms of learning the most optimized cell structures. The radar signal data sets detailed below are generated by Matlab. All the experimental tests are run on Jetbrains Pycharm 2020 with a CPU of Intel(R) Core(TM) i9-9900k @ 3.60GHz and two GPUs of NVIDIA GeForce RTX 2080Ti.

A. Data sets

In previous works e.g. [11] [18] on SEI, synthetic data set consisting of a large scale of different types of radar signals is generally used to test methods' performance. Time-series radar signals are firstly generated according to modulation types, and are then transformed into the time-frequency domain. The data set used in the following contains more signal categories with much more different environments compared to the aforementioned one. More specifically, the data set contains 7 different radar signals which can be split into two categories: frequency modulation (FM) and phase modulation (PM). For FM signals, compared to the previous work in [11] which uses positive slope LFM only, they are augmented here by adding negative slope LFM and Sine FM. For PM signals, there are 64-bit polyphase code signals including P1–4. Each signal type in a single environment has 2,000 samples, thus 14,000 in total of the data set. Five different electromagnetic environments are formed based on the SNR set to -10dB, -8dB, -5dB, -1dB and noise free, respectively. We name this data set "SIGNAL-5". The details of the signal parameters like the range of the pulse width, carrier frequency and band width for the FM signals are given in Table III. The settings of code duration time, sample frequency and band width for polyphase code signals are given in Table IV.

TABLE III
FREQUENCY MODULATION SIGNAL PARAMETERS

	Pulse width	Carrier frequency	Band width
FM	20–30us	20MHz	70–80MHz

TABLE IV
64-BIT POLYPHASE CODE SIGNAL PARAMETERS

	Duration time	Sample frequency	Band width
Polyphase code	2.5ms	15kHz	0.83–1.67kHz

B. Simulations

According to the illustration in Fig. 1, the received radar signal should be deinterleaved firstly. Thus, the aforementioned data set is the group of intersected radar signals coming from different emitters. After that, data preprocessing is conducted to transform the time-series signal data to the time-frequency distribution image data which will then be fed into our proposed balanced NAS.

For the data preprocessing, the Hamming window is applied in STFT and the length of each segment is 256. The time-frequency distribution image is in RGB format with size $512 \times 512 \times 3$, which is clipped to the size of 64×64 using the nearest neighbor interpolation and then fed into CNN for the sake of computation reduction. The whole data set is separated into training, validation and test sets, which account for 60%, 20% and 20% samples, respectively, with batch size set to 64.

For the RNN controller implementation, we use a single LSTM layer, where the number of cells in the controller is 100. An embedding layer with output dimension 20 encodes the candidate child models to a fixed-size vector before LSTM cells. A densely-connected layer with one unit follows the LSTM to output a scalar representing the predicted accuracy of the input child model. We minimize the sum-of-squares loss between the predicted accuracy from LSTM and the real accuracy in the training process by using gradient descent to update the controller. The Adam optimizer is used with learning rate 10^{-3} initially and then exponential decay.

For the NAS structure, the maximum number of cells and blocks of the network structure is respectively set to 2 and 3 unless specific statement. During the searching process, we set

$K = 64$ for stage one and $K = 10$ for the rest of stages. There are two types of cells which respectively have the number of filters 32 and 64, with stride set to 2. The learning rate for the full CNN is 10^{-3} initially with cosine decay. To construct the full CNN from cells, there is a global average pooling layer before the final dense layer. About the training epochs, an early stopping mechanism is implemented, i.e., the training stage terminates when the validation accuracy is larger than 0.999 or the increase of accuracy is less than 0.01% between two consecutive epochs.

The sigmoid function and power function are used to construct the balance function $F(x, y)$, i.e.,

$$F_1(x) = 1/(1 + e^{-10(x-0.5)}), \tag{13}$$

$$F_2(y) = -y^2, \tag{14}$$

$$F(x, y) = F_1(x) + F_2(y). \tag{15}$$

It is straightforward to verify that $F(x, y)$, $F_1(x)$ and $F_2(y)$ all meet the requirements of the basic properties discussed in Section V-A, see also Fig. 8 for the shape of these functions. As shown in Fig. 8 (a), with the variable x (accuracy) increasing, the sigmoid function $F_1(x)$ increases slowly first, then abruptly in the middle and steadily when x is close to 1. This property is ideal for our requirement since when the accuracy is low (e.g. 40%), the scores of models will increase fast with respect to the accuracy; otherwise the scores of models will increase slowly with respect to the accuracy and in this case the emphasis on the efficiency will be taken.

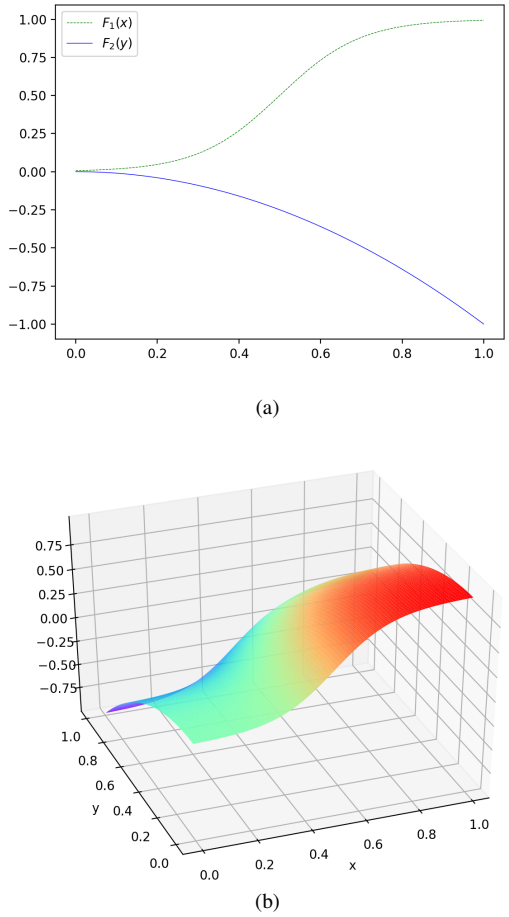


Fig. 8. Function curves. (a): $F_1(x) = 1/(1 + e^{-10(x-0.5)})$ (green line) and $F_2(y) = -y^2$ (blue line); (b) $F(x, y) = F_1(x) + F_2(y)$.

C. Results

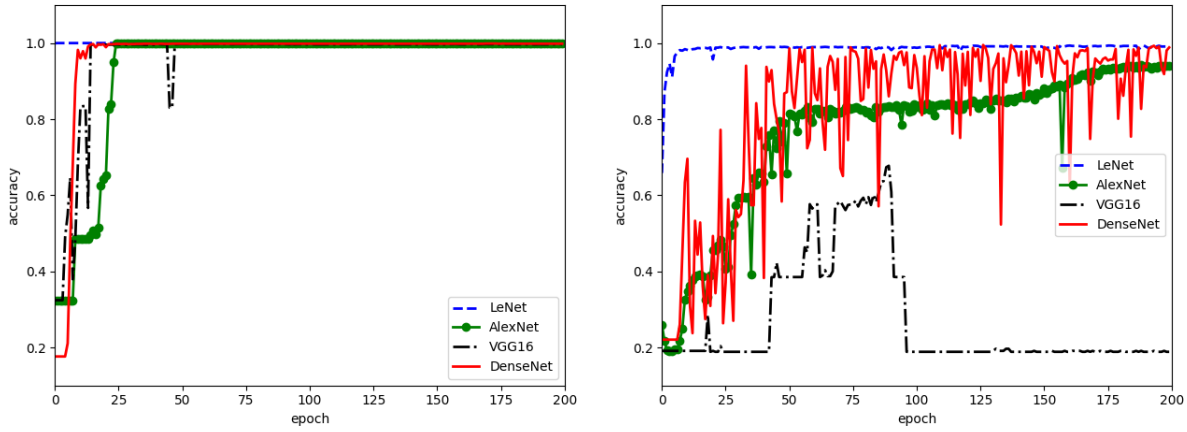
In the following the performance of four traditional and representative neural networks (i.e., LeNet, AlexNet, VGG16 and DenseNet) is first tested. We then explore the different effects between cell and block units in CNNs. After that, we report the performance of the proposed balanced NAS in searching the most optimized cell structures in different SNR. Finally, we conduct the robustness and stability analysis of the proposed method, and discuss its relationship and difference from the reinforcement learning methods like [26], which emphasizes the excellent performance of the proposed balanced NAS.

1) Performance of the CNNs with fixed structure: The performance of applying four conventional CNNs (i.e., LeNet, AlexNet, VGG16 and DenseNet, whose hyper parameter settings, including e.g. the number of filters, kernel size, strides and learning rate, follow the ways given in the seminal papers) to classify radar signals in different SNR circumstances is shown in Fig. 9. We can see that when the noise is free, these networks all perform perfectly, i.e., being able to achieve nearly 100% accuracy in a few epochs. However, when the radar signals with SNR down to -10dB, their accuracy suffer a significant degradation or sharp fluctuation except for LeNet. The possible reason why LeNet performs the best in this scenario might be due to the scale and diversity of the signal data set. A deeper model will more easily suffer from overfitting in the RF signal image data set, which contains a smaller scale and less categories compared to large data sets like ImageNet. In addition, the intense noise distraction could also bring instability to the identification. To further validate the performance of VGG16 and DenseNe, Fig. 10 presents the results via parameter optimization, e.g. adjusting the learning rate and decreasing the number of units in the fully connected layers. We see that the performance of deeper models such as VGG16 and DenseNet can be enhanced in this manner.

Note that the total number of trainable parameters of LeNet is 16,816,741 (i.e., the least one among these four models) with input image size of 64×64. However, the parameters of LeNet is still too large and it is too shallow (only two convolutional layers) to extract much deeper features and will not enable to tackle complicated data sets with considerable samples. In sum, the fixed-structure network generally is not appropriate for variable data sets, which motivates us to design automatic mechanisms to search the most optimized models regarding different environments.

2) Performance of the block-cell structure: We now test the NAS method [30] which uses the accuracy only as the reward to search optimized models. To construct the full CNN, we first stack 2 cells ($C = 2$) in series, where each has 3 blocks ($B = 3$).

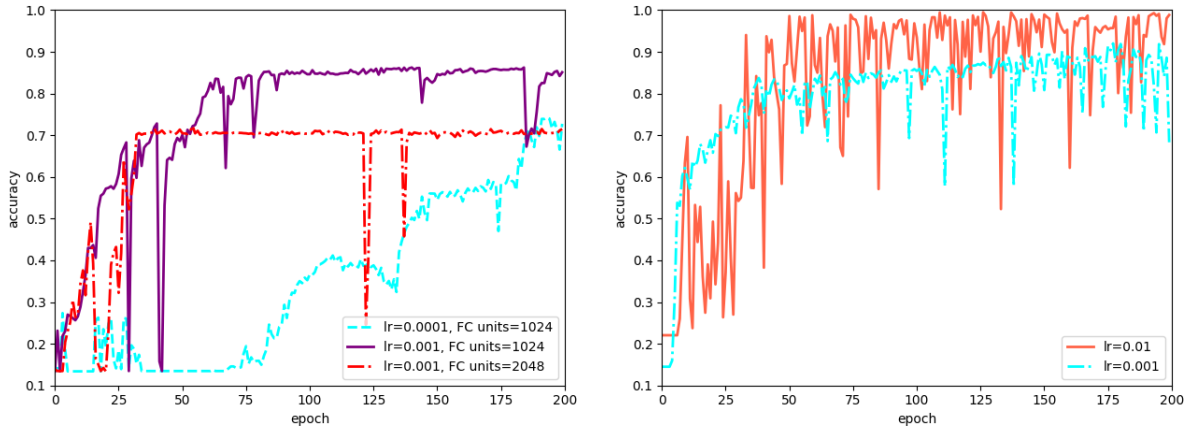
The accuracy of the top 10 models on the data set with SNR set to -1dB, -5dB, -8dB, -10dB and noise free is shown in Fig. 11. It shows that all the models can achieve



(a) Noise free data

(b) Data with SNR $= -10$dB

Fig. 9. Performance of four conventional CNNs (i.e. LeNet, AlexNet, VGG16 and DenseNet) with fixed structure in the SEI. (a): Model accuracy on noise free data; (b) model accuracy on the data with SNR $= -10$dB. The plots show that when the noise level is low, all these models can achieve nearly 100% accuracy within a few epochs; however, when the noise level is high, all these models suffer seriously except for LeNet.



(a) VGG16

(b) DenseNet8

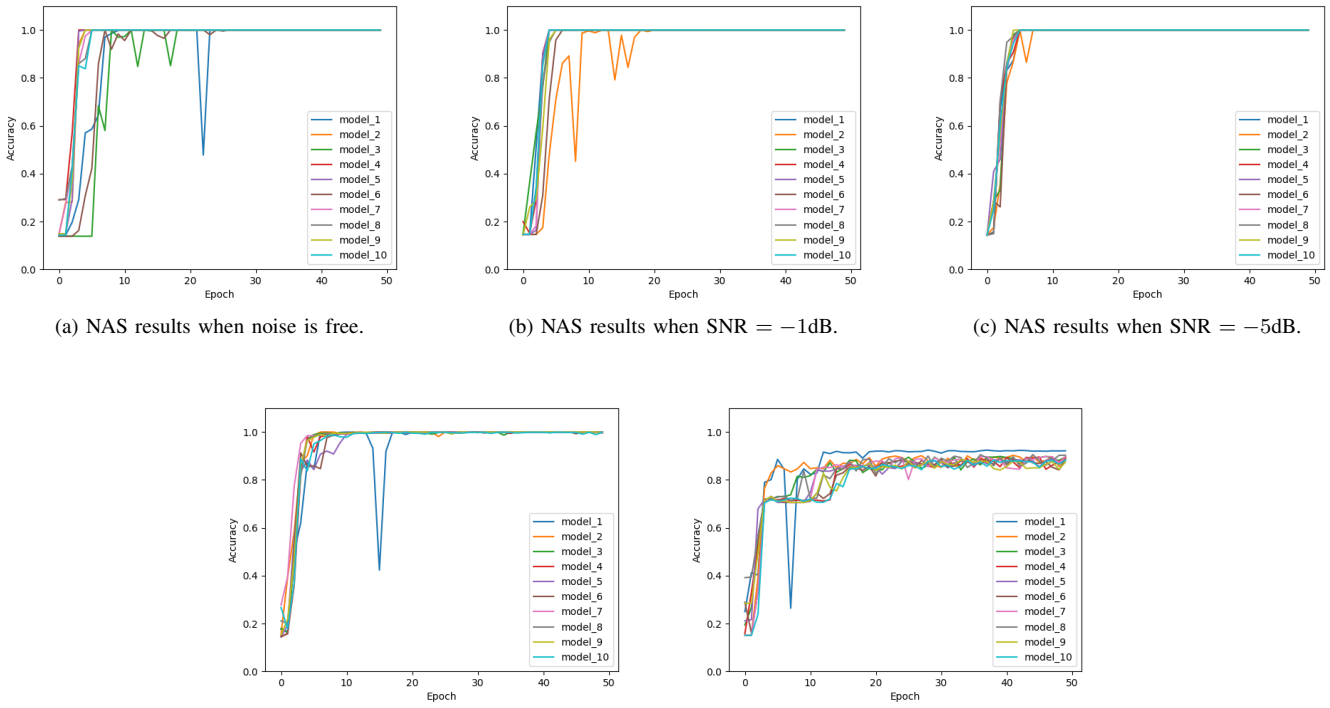
Fig. 10. Performance of the VGG16 and DenseNet8 with different hyper parameters on the data with SNR $= -10$dB. (a): Model accuracy of VGG16 with different *lr* (learning rates) and different number of units in the last three fully connected (FC) layers; (b): model accuracy of DenseNet with different learning rates. It indicates the performance of these two deeper models has an obvious enhancement compared to that in Fig. 9 with default hyper parameter settings.

high accuracy within a few epochs when the SNR is higher than -10dB; however, when the SNR is down to -10dB, the accuracy of these models decreases by more than 10%. This is obviously due to the challenge that the time-frequency distribution images are corrupted more seriously when the SNR decreases (see Fig. 2).

We now test the influence of adding more cells and blocks into the network structure. Note that, according to the aforementioned analysis (e.g. Fig. 5), adding more cells can be viewed as deepening the network, and adding more blocks can extract more features in each hidden layer. Investigating the effect of this kind of setting could provide us guidance regarding constructing optimized neural architecture. In the following we implement further experiments focusing on the most challenging case, i.e., SNR set to -10dB.

We first increase the number of cells with fixed number of blocks, and then increase the number of blocks with fixed number of cells. Fig. 12 (a) gives the accuracy of the top 10 models with the number of blocks fixed to 3 but the number of cells increased from 2 to 4. The structure of the most accurate model (i.e., model 1) in Fig. 12 (a) is shown in Fig. 13. It is worth highlighting that the total number of trainable parameters of model 1 in Fig. 12 (a) is 1,587,077, which is much less than that of LeNet (16,816,741) aforementioned. The best accuracy, compared to the results given in Fig. 11 (e), increases from 88.8% to 99.2%. Similarly, all other models also have about 10% increase in terms of accuracy.

Fig.12 (b) gives the accuracy of the top 10 models with the number of cells fixed to 2 but the number of blocks increased from 3 to 5. The highest accuracy on this occasion is 89.2%, which has a slight increase compared to the result 88.8% given in Fig. 11 (e), but far less than the 99.2% accuracy shown in Fig. 12 (a). This indicates that adding depth of the network is more effective than adding the number of blocks in this scale



(a) NAS results when noise is free.

(b) NAS results when SNR $= -1$dB.

(c) NAS results when SNR $= -5$dB.

(d) NAS results when SNR $= -8$dB.

(e) NAS results when SNR $= -10$dB.

Fig. 11. Top accuracy of the models selected by the NAS method without using the balance function in different environments; the network structure has 2 cells where each has 3 blocks. (a)–(e) present the results on the data set with no noise, SNR set to -1dB, -5dB, -8dB and -10dB, respectively. It shows that when the SNR is higher than -10dB, all the models can nearly achieve perfect accuracy within a few epochs. However, when the SNR is down to -10dB, their accuracy decreases by more than 10%.

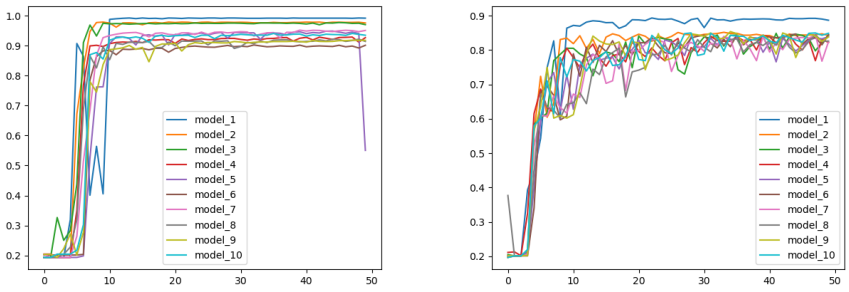
of samples.

3) Performance of the proposed balance function: For the comparison of the NAS method with and without the proposed balance function, two extreme cases – noise free and SNR set to -10dB – are considered.

For the noise free case, top 5 models in terms of their scores (say set A_5') rather than accuracy are selected by the NAS method with the proposed balance function. Their performance are shown in bold type in Table V, where "Para." represents the number of trainable parameters. Table V also lists the top 5 models (say set A_5) given in Fig. 11 (a) which are selected in terms of accuracy. It shows that the balance function can find cell structures which have both high accuracy and efficiency. The computational cost decreases nearly more than 10 times when using the balance function, with the accuracy comparable to the conventional algorithm (i.e., the average accuracy of A_5 and A_5' are respectively $x_{A_5} = 1.0$ and $x_{A_5'} = 0.996$, and the maximum accuracy difference between A_5 and A_5' is $\Delta = 0.34\%$). The τ values corresponding to the best models selected by the NAS method without and with the balance function are 2.46 and 31.94, respectively, which shows a huge improvement of the model selected by using the proposed balance function.

For the case of the SNR set to -10dB, similar to the setting in Table V, Table VI gives the results of the NAS method without and with the proposed balance function. Note that here the top 5 models selected without using the balance function are those shown in Fig. 11 (e). It is clear that the proposed balance function dramatically prompts the searching performance of the NAS algorithm in terms of both accuracy and efficiency. Specifically, the accuracy of the best models without and with the balance function has been improved from 88.8% to 92.35% (i.e., $\Delta = 3.55\%$), and the τ value corresponding to the best models selected by the NAS method without and with the balance function are respectively 2.45 and 14.92. This implies that in challenging cases when applying the balance function the overall accuracy of the searching results could also be highly increased rather than sacrificing a fraction of the accuracy for the improvement of the efficiency. Therefore, the balance function in some cases can assist the NAS algorithm in searching the most optimized model possessing both high accuracy and efficiency.

A further comparison of the NAS algorithm with and without the proposed balance function in terms of accuracy and efficiency is implemented for networks with different blocks. Fig. 14 shows the comparison for networks with the number of blocks set to 1 to 8, where the best models selected with and without the proposed balance function are compared. It is observed that when more blocks are added, the accuracy of the models selected with or without the balance function is very similar with no dramatic increase (see red lines in Fig. 14), which is consistent with the previous conclusion that adding more number of blocks may not boost the accuracy. Regarding the efficiency of the selected models with or without the



(a) (b)

Fig. 12. Top accuracies of the models selected by the NAS method without using the balance function; SNR is set to -10dB. (a): Network structure with the number of blocks fixed to 3 but the number of cells increased from 2 to 4; (b): network structure with the number of cells fixed to 2 but the number of blocks increased from 3 to 5. It shows that the highest accuracy in (a) reaches 99%, which is about 10% higher than the highest one in (b).

TABLE V
TOP 5 ACCURATE MODELS WHEN NOISE IS FREE. THE BOLD PARTS ARE RESULTS AFTER APPLYING THE BALANCE FUNCTION.

Model index	Model structure	Accuracy	Para.	Score
model 1	1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 7x7-dc	1	257,157	0.829
	3x3-dc, 3x3-ap	**99.66%**	**19,717**	**0.981**
model 2	1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 7x7-dc, 1x7-7x1-c	1	257,157	0.829
	3x3-ap, 3x3-dc	**98.37%**	**19,717**	**0.980**
model 3	1x7-7x1-c, 1x7-7x1-c, 7x7-dc, 1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c	1	257,157	0.829
	1x7-7x1-c, 3x3-mp	**1**	**30,533**	**0.973**
model 4	1x7-7x1-c, 1x7-7x1-c, 7x7-dc, 1x7-7x1-c	1	196,421	0.863
	1x7-7x1-c, 3x3-ap	**1**	**30,533**	**0.973**
model 5	7x7-dc, 1x7-7x1-c	1	135,685	0.900
	3x3-mp, 1x7-7x1-c	**1**	**30,533**	**0.973**

TABLE VI
TOP 5 ACCURATE MODELS WHEN SNR $= -10$DB. THE BOLD PARTS ARE RESULTS AFTER APPLYING THE BALANCE FUNCTION.

Model index	Model structure	Accuracy	Para.	Score
model 1	1x7-7x1-c, 5x5-dc, 1x7-7x1-c, 1x7-7x1-c, 5x5-dc, 1x7-7x1-c	88.8%	229,189	0.838
	3x3-dc, 3x3-dc	**92.35%**	**39,109**	**0.929**
model 2	1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 5x5-dc, 1x7-7x1-c	85.3%	205,701	0.847
	3x3-dc, 3x3-dc, 3x3-dc, 3x3-dc, 3x3-dc, 3x3-mp	**91.7%**	**97,925**	**0.919**
model 3	1x7-7x1-c, 1x7-7x1-c, 5x5-dc, 1x7-7x1-c, 5x5-dc, 1x7-7x1-c	86.0%	229,189	0.835
	3x3-dc, 3x3-dc, 3x3-dc, 1x7-7x1-c	**72.8%**	**89,029**	**0.914**
model 4	1x7-7x1-c, 5x5-dc, 1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c, 5x5-dc	86.1%	229,189	0.835
	3x3-dc, 3x3-dc, 3x3-dc, 3x3-dc, 3x3-dc, 3x3-dc	**85%**	**117,317**	**0.902**
model 5	1x7-7x1-c, 1x7-7x1-c, 5x5-dc, 1x7-7x1-c, 1x7-7x1-c, 1x7-7x1-c	85.2%	205,701	0.847
	3x3-dc, 3x3-dc, 3x3-dc, 5x5-dc	**92.75%**	**112,517**	**0.900**

balance function (see blue lines in Fig. 14), it shows that the proposed balance function can indeed control the computation amount well when adding more blocks into the neural network, whereas without using the balance function, the computation amount of the selected models is increasing greatly. This demonstrates the great importance of the proposed balance function in NAS in terms of balancing the accuracy and efficiency.

4) Robustness and stability analysis: To validate the robustness of the proposed balanced-NAS framework, a simple but effective white-box attack, called Fast Gradient Sign method (FGSM) [45], is used to add some intentional and imperceptible perturbation on the clean data set. We adopt the adversarial training as the defense strategy to strengthen the robustness of those searching results. The research work in [46] finds that fine-tuning the candidate models with adversarial training for a few epochs can obviously promote the performance. The adversarial accuracy is served as the robustness indicator of the network. Moreover, we develop the flow of the adversarial training, which is illustrated in Fig. 15. It firstly generates some adversarial examples by Projected Gradient Descent [47] to augment the original clean samples. After training the optimized models by our balanced NAS, we fine-tune them for a few epochs on the augmented data set. Fig. 16 shows the performance comparison of the top 5 optimized models before attack, after attack and after fine-tune, where the data sets with noise free and SNR set to -10dB are showcased. The comparison indicates that the accuracy of the selected models drops significantly after the attack, see the orange lines in Fig. 16. This accuracy drop can be eliminated immediately by fine-tuning on the adversarial examples, which clearly demonstrates the robustness of the balanced NAS. Fig.

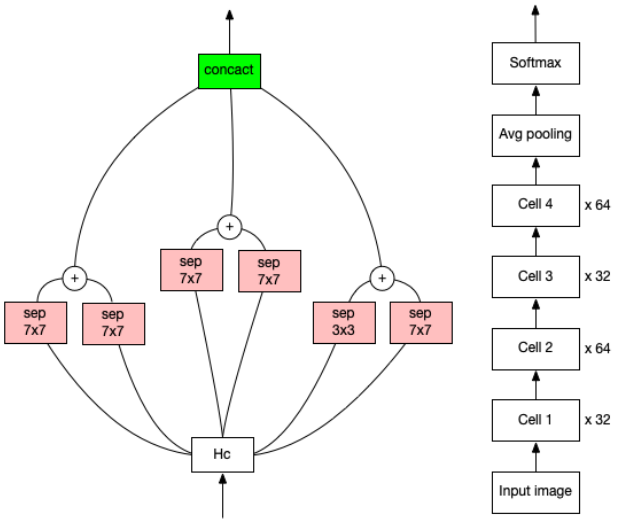


Fig. 13. The structure of model 1 in Fig. 12 (a), i.e., the most accurate model with $B = 3, C = 4$ and accuracy 99.2%. The left and right figures show the cell structure and the full CNN, respectively.

16 (b) also shows that, after adversarial training for the case of SNR $= -10$dB, the accuracy achieved could even surpass that achieved before the FGSM attack.

To validate the stability of the proposed balanced-NAS framework, we show the standard deviation of the accuracy and computation amount of the searching results after repeating the experiments with random seeds. It is illustrated in e.g. [48], [49] that the skip-connection in deep architectures often leads to the most rapid error decay during optimization, which might bring instability. In our setting, the single-path paradigm is used in the search space, which is more efficient and less error-prone. We execute the standard NAS and our balanced NAS 10 times on the data with SNR $= -10$dB and compare the top 10 candidate models in each run (i.e., 100 candidate models in total for comparison). In the meantime, we also randomly sample 100 child models from the search space and test their performance as the baseline. The distribution of the accuracy among these models is displayed in Fig. 17. It shows that i) the random search baseline verifies the effectiveness of the searching strategy in NAS; and 2) the proposed balanced NAS can indeed find the cheapest models whose accuracy is also comparable to that obtained by the standard NAS. To quantify the stability, the mean and standard deviation of the *accuracy* and the normalized *computation amount* are computed, i.e., $[0.824\pm0.12, 0.246\pm0.153]$, $[0.895\pm0.002, 0.515\pm0.170]$ and $[0.843 \pm 0.005, 0.076 \pm 0.047]$ corresponding to the random search baseline, the standard NAS and our balanced NAS, respectively. This clearly shows the great stability of the proposed balanced-NAS framework. Moreover, it is worth emphasizing again that the candidate models searched by the proposed balanced-NAS framework have less trainable parameters up to 7 times compared to the standard NAS at the price of a slight accuracy decrease (i.e., 5.8%).

5) Further discussion: Finally, a brief discussion is conducted regarding the relationship and difference between the proposed method and the standard reinforcement learning methods like [26]. The similarity is that they both utilize an RNN policy to sequentially sample a string representing the neural architectures where the generation of a neural architecture can be viewed as an agent's action [23]. As for their difference, the agent's reward in the work like [26] is only based on the accuracy of the trained architecture, whereas our proposed strategy takes both the accuracy and efficiency into consideration by leveraging the proposed balance function whose excellent performance has been validated in the previous sections. Moreover, comparing to the local search manner used in [26], we adopt the heuristic search inspired by [30], which evaluates the candidate child models from simple to complex and abandons unpromising individuals at the searching stage to further promote the overall efficiency.

VII. Conclusion

In this paper, a new NAS mechanism called balanced-NAS framework was designed for SEI as an alternative of the fixed neural structure in conventional approaches, with the benefit that not only the accuracy but also the efficiency is considered as rewards to the RNN controller for model searching in neural networks. A balance function was also proposed to manage the accuracy and efficiency in NAS. It is motivated by the real-time performance requirement of SEI that the neural network in SEI should possess high accuracy and efficiency simultaneously. Experimental results demonstrated that the new NAS mechanism exploiting the block-cell structure and RNN controller can design models comparable to the state-of-the-art manually-designed networks. Moreover, the proposed balance function can optimize the search strategy and can find models with both high accuracy and efficiency, which outperform the ones searched by the standard network in which only the accuracy criterion is used. The robustness and stability of the proposed balanced-NAS framework have also been validated quantitatively. Potential future works could consider other neural network structures except for the block-cell-based search space. It is also of great interest to analyse inter-pulse radar features in addition to the ones with time-frequency distribution.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant no. 61671453).

References

[1] K. I. Talbot, P. R. Duley, and M. H. Hyatt, "Specific emitter identification and verification," *Technol. Rev.*, vol. 11, pp. 113–133, 2003.
[2] Z. Zhang, K. Long, and J. Wang, "Self-organization paradigms and optimization approaches for cognitive radio technologies: a survey," *IEEE Commun. Lett.*, vol. 20, no. 2, pp. 36–42, 2013.
[3] G. Revillon, A. Mohammad-Djafari, and C. Enderli, "Radar emitters classification and clustering with a scale mixture of normal distributions," *IET Radar, Sonar Navig.*, vol. 13, no. 1, pp. 128–138, 2018.
[4] C.-S. Shieh and C.-T. Lin, "A vector neural network for emitter identification," *IEEE Trans. Antennas Propag.*, vol. 50, no. 8, pp. 1120–1127, 2002.
[5] C.-M. Lin, Y.-M. Chen, and C.-S. Hsueh, "A self-organizing interval type-2 fuzzy neural network for radar emitter identification." *Int. J. Fuzzy Syst.*, vol. 16, no. 1, 2014.

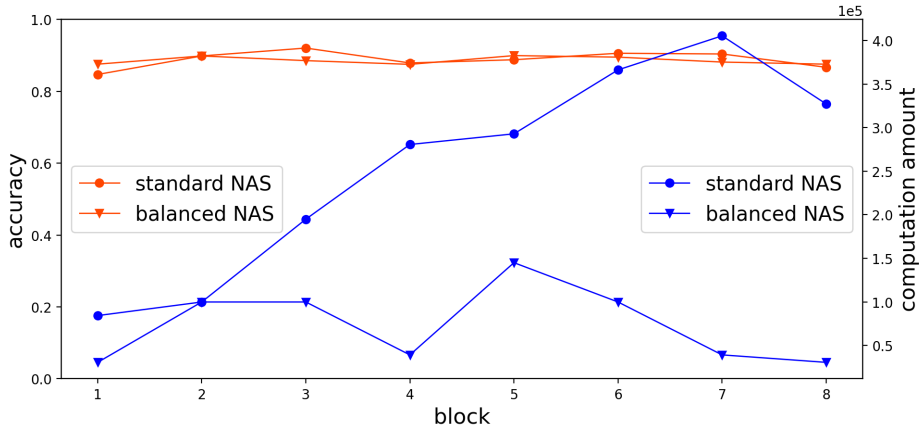


Fig. 14. Performance comparison of the NAS method with and without the proposed balance function, where the number of blocks is set from 1 to 8, the number of cells is fixed to 2, and SNR $= -10$dB. The lines in red and blue show the comparison corresponding to the accuracy and the computation amount, respectively.

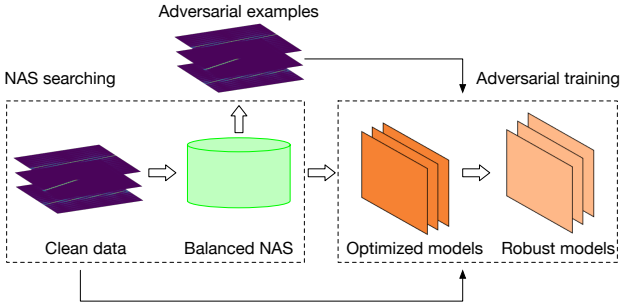
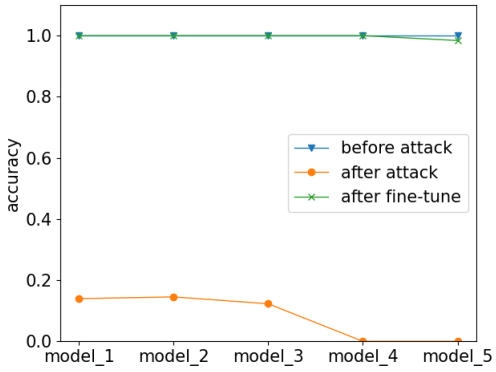
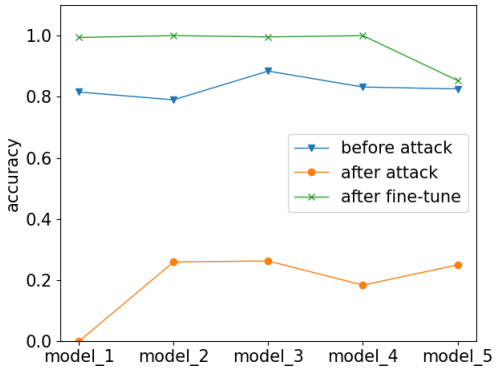


Fig. 15. The flow chart of the adversarial training for our proposed balanced NAS. During the searching process, adversarial examples are generated though FGSM, which augments the data. Then, the robustness of the optimized models is strengthened by fine-tuning on the augmented data set.



(a)



(b)

Fig. 16. Robustness performance analysis of the balanced NAS in terms of accuracy before adversarial attack, after attack and after fine-tune through the adversarial training. (a)–(b): Accuracy of the top 5 optimized models selected by the balanced NAS (i.e., Tables V and VI) for identifying the radar signals with noise free and SNR $= -10$dB, respectively.

[6] A. Kawalec and R. Owczarek, "Radar emitter recognition using intra-pulse data," in *Int. Conf. Microwaves Radar Wireless Commun.*, vol. 2. IEEE, 2004, pp. 435–438.

[7] L. Ding, S. Wang, F. Wang, and W. Zhang, "Specific emitter identification via convolutional neural networks," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2591–2594, 2018.

[8] G. López-Risueño, J. Grajal, and A. Sanz-Osorio, "Digital channelized receiver based on time-frequency analysis for signal interception," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 3, pp. 879–898, 2005.

[9] C. Bertoncini, K. Rudd, B. Nousain, and M. Hinders, "Wavelet fingerprinting of radio-frequency identification (rfid) tags," *IEEE Trans. Consum. Electron.*, vol. 59, no. 12, pp. 4843–4850, 2011.

[10] J. Lundén and V. Koivunen, "Automatic radar waveform recognition," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 1, pp. 124–136, 2007.

[11] M. Zhang, L. Liu, and M. Diao, "Lpi radar waveform recognition based on time-frequency distribution," *Sensors*, vol. 16, no. 10, pp. 1682–1701, 2016.

[12] S. Liu, X. Yan, P. Li, X. Hao, and K. Wang, "Radar emitter recognition based on sift position and scale features," *IEEE Trans. Circuits Syst. II*, vol. 65, no. 12, pp. 2062–2066, 2018.

[13] U. Satija, N. Trivedi, G. Biswal, and B. Ramkumar, "Specific emitter identification based on variational mode decomposition and spectral features in single hop and relaying scenarios," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 581–591, 2019.

[14] J. Zhu, Y. Zhao, and J. Tang, "Automatic recognition of radar signals based on time-frequency image character," *Def. Sci. J.*, vol. 63, pp. 308–314, 2013.

[15] I. Jordanov, N. Petrov, and A. Petrozziello, "Supervised radar signal classification," in *Proc. Int. Jt. Conf. Neural Networks*. IEEE, 2016, pp. 1464–1471.

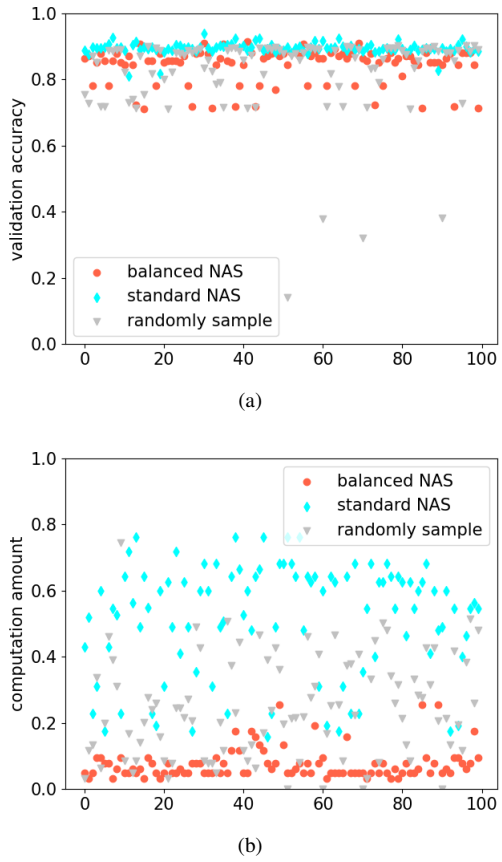


Fig. 17. Stability comparison in terms of (a) validation accuracy and (b) computation amount between the random sample baseline, the standard NAS and our balanced NAS on the data with SNR $= -10$dB. For each method, the top 10 models are selected in each of the 10 independent executions.

[16] P.-j. Chun, I. Ujike, K. Mishima, M. Kusumoto, and S. Okazaki, "Random forest-based evaluation technique for internal damage in reinforced concrete featuring multiple nondestructive testing results," *Comput. Biol. Med.*, vol. 253, pp. 119 238–119 247, 2020.

[17] D. Ravi, C. Wong, B. Lo, and G. Yang, "Deep learning for human activity recognition: A resource efficient implementation on low-power devices," in *IEEE Int. Conf. Wearable Implant. Body Sens. Netw.*, 2016, pp. 71–76.

[18] S.-H. Kong, M. Kim, L. M. Hoang, and E. Kim, "Automatic lpi radar waveform recognition using cnn," *IEEE Access*, vol. 6, pp. 4207–4219, 2018.

[19] M. Zhang, M. Diao, and L. Guo, "Convolutional neural networks for automatic cognitive radio waveform recognition," *IEEE Access*, vol. 5, pp. 11 074–11 082, 2017.

[20] Q. Guo, X. Yu, and G. Ruan, "Lpi radar waveform recognition based on deep convolutional neural network transfer learning," *Symmetry*, vol. 11, no. 4, pp. 540–553, 2019.

[21] C. Wang, J. Wang, and X. Zhang, "Automatic radar waveform recognition based on time-frequency analysis and convolutional neural network," in *IEEE Int. Conf. Acoust. Speech Signal Process Proc.*, 2017, pp. 2437–2441.

[22] C.-W. Tan and A. Kumar, "Accurate iris recognition at a distance using stabilized iris encoding and zernike moments phase features," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3962–3974, 2014.

[23] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[24] 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 *Int. Conf. Mach. Learn.* JMLR. org, 2017, pp. 2902–2911.

[25] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

[26] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[27] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.* IEEE, 2018, pp. 8697–8710.

[28] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[29] T. Elsken, J.-H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," *arXiv preprint arXiv:.04528*, 2017.

[30] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Lect. Notes Comput. Sci.* Springer, 2018, pp. 19–34.

[31] K. Cui, W. Wu, J. Huang, X. Chen, and N. Yuan, "Doa estimation of lfm signals based on stft and multiple invariance esprit," *Int. J. Electron. Commun.*, vol. 77, pp. 10–17, 2017.

[32] P. Magron, R. Badeau, and B. David, "Model-based stft phase recovery for audio source separation," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 6, pp. 1095–1105, 2018.

[33] I. Djurović, V. Popović-Bugarin, and M. Simeunović, "The stft-based estimator of micro-doppler parameters," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 3, pp. 1273–1283, 2017.

[34] S. Chikkerur, A. N. Cartwright, and V. Govindaraju, "Fingerprint enhancement using stft analysis," *Pattern Recognit.*, vol. 40, no. 1, pp. 198–211, 2007.

[35] M. K. Kıymık, İ. Güler, A. Dizibüyük, and M. Akın, "Comparison of stft and wavelet transform methods in determining epileptic seizure activity in eeg signals for real-time application," *Comput. Biol. Med.*, vol. 35, no. 7, pp. 603–616, 2005.

[36] J. Zhang, F. Wang, O. A. Dobre, and Z. Zhong, "Specific emitter identification via hilbert–huang transform in single-hop and relaying scenarios," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1192–1205, 2016.

[37] L. Durak and O. Arikan, "Short-time fourier transform: two fundamental properties and an optimal implementation," *IEEE Trans. Signal Process.*, vol. 51, no. 5, pp. 1231–1242, 2003.

[38] T. Yuan, W. Deng, J. Tang, Y. Tang, and B. Chen, "Signal-to-noise ratio: A robust distance metric for deep metric learning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 4815–4824.

[39] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Int. Conf. Mach. Learn.*, 2013, pp. 115–123.

[40] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. Genet. Evol. Comput. Conf.*, 2017, pp. 497–504.

[41] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.* IEEE, 2018, pp. 2423–2432.

[42] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," *arXiv preprint arXiv:1703.01041*, 2017.

[43] L. Xie and A. Yuille, "Genetic cnn," in *Proc. IEEE Int. Conf. Comput. Vision*. IEEE, 2017, pp. 1379–1388.

[44] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.* IEEE, 2015, pp. 5353–5360.

[45] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[46] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When nas meets robustness: In search of robust architectures against adversarial attacks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 2020, pp. 631–640.

[47] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[48] X. Chu, B. Zhang, J. Li, Q. Li, and R. Xu, "Scarlet-nas: bridging the gap between stability and scalability in weight-sharing neural architecture search," *arXiv preprint arXiv:1908.06022*, 2019.

[49] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Lect. Notes Comput. Sci.*, 2019, pp. 1294–1303.