Pol-NAS: A Neural Architecture Search Method With Feature Selection for PolSAR Image Classification

Guangyuan Liu, Yangyang Li, *Senior Member, IEEE*, Yanqiao Chen, Ronghua Shang, *Member, IEEE*, and Licheng Jiao, *Fellow, IEEE*

***Abstract*—With the development of deep learning, more and more neural networks have been used in polarimetric synthetic aperture radar (PolSAR) image classification and obtain good results. As we all know, the performances of neural networks highly depend on well-designed neural architectures. Besides, the features input to neural networks also have a huge impact on the classification results. Both architecture design and feature selection are time consuming and require human expertise. Therefore, in this article, we propose a neural architecture search method with feature selection (Pol-NAS) for PolSAR image classification. It can automatically search and obtain a good architecture, including intracell and intercell structure and the number of layers in the search stage. Meanwhile, all the features commonly used in PolSAR data interpretation, rather than part of them, are input to the model in order to avoid selecting the size of an optimal feature subset, which is a hyperparameter and usually different for different models. Then, we propose the feature attention block (FA block) and redesign the stem layers by combining the FA block and the original stem layers. Thus, Pol-NAS can adaptively find the importance of each feature in the training stage by using the redesigned stem layers. With the help of Pol-NAS, we only need to prepare the data and wait for the classification results. Experimental results on three real PolSAR datasets show that the performance of Pol-NAS is better than that of state-of-the-art PolSAR image classification models.**

***Index Terms*—Feature selection, image classification, neural architecture search (NAS), polarimetric synthetic aperture radar (PolSAR).**

I. Introduction

POLARIMETRIC synthetic aperture radar (PolSAR) obtains information by actively transmitting and receiving electromagnetic waves. Therefore, compared with optical radar, it can work all day and in all weather conditions [1]. Owing to its advantages, more and more attention has been paid to PolSAR and the interpretation of PolSAR images. PolSAR image classification, as the basic task of PolSAR interpretation, has become a hot research field in recent years [2], [3]. PolSAR image classification models in the early days can be divided into three categories: 1) methods based on physical scattering mechanisms, such as target scattering vector model (TSVM) [4] and multiple-component scattering model (MCSM) [5]; 2) methods based on statistical distribution, such as complex Wishart distribution [6] and Wishart classification method; and 3) methods based on machine learning, such as k-nearest neighbor [7] and support vector machine (SVM) [8]. These methods play an important role in the early days.

With the development of deep learning, more and more deep learning models have been employed to do PolSAR image classification and other remote sensing tasks due to their good performances. For example, stacked autoencoders (SAEs) [9], convolutional neural networks (CNNs) [10], [11], [12], [13], [14], [15], graph convolutional networks [16], [17], fully convolutional networks (FCNs) [18], [19], [20], and mixed architecture [21] have been proposed and greatly improve the classification accuracy. However, these models have two problems. First, their performances highly depend on proper architecture designs, which are not theoretically guaranteed and usually designed by human experts. This process is time consuming and disgusting. Second, their performances are also affected by the input features, and optimal features of different models are also different. That is to say, if we use a deep learning model to classify a PolSAR image, we have to solve the two problems, as shown in Fig. 1.

In the beginning, given a new dataset, we need to determine the network architecture. We have three kinds of methods to solve this problem, but they all have many disadvantages because architecture designs are not theoretically guaranteed and usually determined by trial and error. First, solution 1 requires human expertise, and this process is time consuming. For example, assume that we need to design a network with five layers (not including softmax layer), and each layer has four candidate operations such as 3×3, 5×5 convolution, and so on. There will be $4^5 = 1024$ candidate architectures. If it takes 1 h to train a model and test its performance, we will need 1024 h to determine the architecture without human expertise. Besides, we still need to determine the input features after determining the architecture.

Manuscript received 7 July 2022; revised 29 August 2022 and 8 October 2022; accepted 11 October 2022. Date of publication 25 October 2022; date of current version 8 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61772399, Grant 62101517, and Grant 62176200, in part by the Key Research and Development Program in Shaanxi Province of China under Grant 2019ZDLGY09-05, and in part by the 111 Project. *(Corresponding author: Yangyang Li.)*

Guangyuan Liu, Yangyang Li, Ronghua Shang, and Licheng Jiao are with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Joint International Research Laboratory of Intelligent Perception and Computation, Collaborative Innovation Center of Quantum Information of Shaanxi Province, School of Artificial Intelligence, Xidian University, Xi'an 710071, China (e-mail: gyliu@stu.xidian.edu.cn; yyli@xidian.edu.cn; rhshang@mail.xidian.edu.cn; lchjiao@mail.xidian.edu.cn).

Yanqiao Chen is with the 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang 050081, China (e-mail: chenyanqiao2016@163.com).

The code is available at https://github.com/guangyuanLiu/Pol-NAS.

Digital Object Identifier 10.1109/JSTARS.2022.3217047

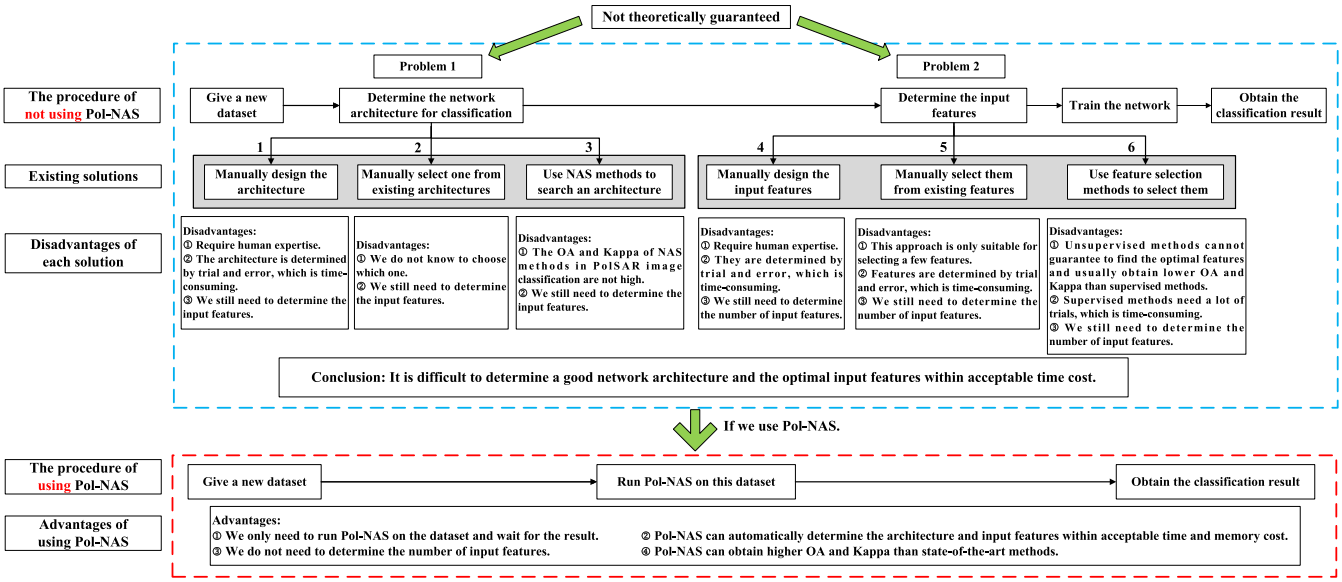


Fig. 1. Comparison between using and not using Pol-NAS for PolSAR image classification. NAS: Neural architecture search. OA: Overall accuracy. Kappa: Kappa coefficient.

TABLE I
ALL THE FEATURES IN POLSAR DATA

	Feature	Feature details	Dimension
Coherent decomposition	Target Scattering Vector Model	(alpha-s, phi-s, phi, tau-m)	4
	Pauli	(Odd, 0° Dbl, 45° Dbl)	3
Non-Coherent decomposition	Huynen	(T11, T22, T33)	3
	H/A/α	(entropy, anisotropy, alpha, lambda1, lambda2, lambda3)	6
	Holm	Holm1: (T11, T22, T33), Holm2: (T11, T22, T33)	6
	Freeman	Freeman2: (Vol, Ground), Freeman3: (Odd, Dbl, Vol)	5
	Cloude	(T11, T22, T33)	3
	Barnes	Barnes1: (T11, T22, T33), Barnes2: (T11, T22, T33)	6
	AnYang	AnYang3: (Odd, Dbl, Vol), AnYang4: (Odd, Dbl, Vol, Hlx)	7
	Yamaguchi	Yamaguchi3: (Odd, Dbl, Vol), Yamaguchi4: (Odd, Dbl, Vol, Hlx)	7
	VanZyl	(Odd, Dbl, Vol)	3
	Multiple-Component Scattering Model	(Odd, Dbl, Vol, Hlx, Dbl-Hlx, Wire)	6
Others		Span, θ_{null}_Re[T12], θ_{null}_Im[T12]	3
		10lg(Span), T22/Span, T33/Span, $\lvert T12\rvert/\sqrt{T11*T22}$, $\lvert T13\rvert/\sqrt{T11*T33}$, $\lvert T23\rvert/\sqrt{T33*T22}$	6
		$\mathbf{T}$ matrix: (T11, T22, T33, real(T12), img(T12), real(T13), img(T13), real(T23), img(T23))	9
Sum			77

Second, if we choose solution 2, we do not know which model has a good performance on the new dataset. Also, we still need to determine the input features. Third, although neural architecture search (NAS) [22], which aims to automatically search and design architectures, has been proposed recently and has good performances on many tasks, only a few NAS methods [23], [24] in PolSAR image classification are proposed. Furthermore, the improvements brought by these NAS methods in PolSAR image classification are limited. The accuracy of them is lower than that of some handcrafted models like in [19] and [20]. Also, they cannot automatically select the input features.

Similarly, all these methods (i.e., solutions 4–6) cannot solve problem 2 perfectly because they all have disadvantages. First, we usually do not use solution 4 because designing new features with good performances requires human expertise and is really difficult. Considering that many features have already been designed, we can select some features directly from Table I. But manually selecting features requires human expertise and is only suitable for selecting a few features because we do not know the complex coupling relationships among a large number of features. Besides, this approach is time consuming. For example, if we select five out of 77 features, there will be 19 million possibilities. It will take a long time to evaluate their performances. That is why feature selection by algorithms (i.e., solution 6) becomes popular. According to whether the label information is used or not, feature selection algorithms can be roughly divided into two categories: unsupervised and supervised methods. Unsupervised methods [25], [26] select a subset of features by exploiting the intrinsic information of them and do not need to train the network to test the performances of selected features many times. That is to say, unsupervised methods do not consider the relationship between input features and the network, and cannot guarantee to find the optimal features for a specific model. Oppositely, supervised methods [27], [28], [29] select the optimal subset of features by training the network to test the performances of different features many times. Obviously, unsupervised methods usually have lower algorithm complexity than supervised ones, but the performances of the former ones are usually worse than those of the latter ones. Besides, whichever method we use to do feature selection, we still need to determine the number of input features. Because the optimal feature subset is only true for a specific classification model in most cases, the optimal subsets of different models vary in size. For example, in [29], experimental results show that the optimal subsets for 2-D CNN and SVM contain 7 and 22 features, respectively.

Therefore, in order to simplify the above process and make NAS and feature selection for PolSAR image classification become easier, we propose a NAS method with feature selection called *Pol-NAS*, which can solve the two problems simultaneously. First, a NAS method for determining the optimal architecture in the search stage is proposed. Second, we use all the features rather than part of them as the input of the searched architecture, in order to avoid setting the number of input features. Third, considering that feature selection is essentially looking for the importance of each feature, which is similar to the attention mechanism [30], we propose the feature attention block (FA block). Then, we redesign the stem layers (the first three layers) of the searched architecture via combining the FA block and the original stem layers. Thus, the redesigned stem layers can find the importance of each input feature in the training stage. Finally, by means of the proposed NAS method and redesigned stem layers, we propose the NAS method with feature selection called *Pol-NAS*.

In summary, the greatest strength of *Pol-NAS* is that it can automatically find a good network structure and determine the importance of each input feature with acceptable time and memory cost. Given a PolSAR image, we only need to prepare the data and wait for the result. The contributions of this article are as follows.

1) A NAS method for PolSAR image classification is proposed. We redesign the search space and adopt the searching method based on the differentiable method for PolSAR image classification.
2) Considering that the number of network layers also affects the performance due to the existence of an overfitting phenomenon, we propose a greedy strategy to determine it automatically. We use several auxiliary cells in intermediate layers to predict the classification results and assign them "learnable weights," which can be updated in the architecture searching process. In the end, we determine the number of layers via selecting the layer with the strongest "learnable weight."
3) We use all the features as the input to avoid setting the number of input features. Then, we propose the FA block and redesign the stem layers of the searched architecture by combining the FA block and the original stem layers. Therefore, *Pol-NAS* can find the importance of each input feature in the training stage via the redesigned stem layers. The optimal architecture and the importance of each feature can be determined in one run.
4) Experiments on three real PolSAR images demonstrate that our method obtains better results than state-of-the-art methods for PolSAR image classification.

The rest of this article is organized as follows. The background knowledge is described in Section II. Section III introduces the proposed method *Pol-NAS* in detail. Section IV presents the experimental results. Finally, Section V concludes this article.

II. Background

Pol-NAS is a differentiable NAS method with feature selection for PolSAR image classification. Therefore, we briefly introduce the background knowledge about PolSAR data and commonly used features and differentiable NAS.

A. PolSAR Data and Commonly Used Features

A single pixel in the PolSAR image can be described by the scattering matrix, which can be formulated as

$$\mathbf{S} = \begin{bmatrix} S_{HH} & S_{HV} \\ S_{VH} & S_{VV} \end{bmatrix} \tag{1}$$

where S_{pq} represents the backscattering component of transmitting in p polarization and receiving in q polarization. $p, q \in \{horizontal, vertical\}$.

For the reciprocal backscattering case $S_{\mathrm{HV}} = S_{\mathrm{VH}}$, a complex vector representing the polarimetric scattering information can be obtained and represented as

$$\underline{\mathbf{k}}_{\mathbf{P}} = \frac{1}{\sqrt{2}} \begin{pmatrix} S_{HH} + S_{VV} & S_{HH} - S_{VV} & 2S_{HV} \end{pmatrix}^{\top} \tag{2}$$

where superscript $\top$ denotes the matrix transpose. A coherent matrix $\mathbf{T}$ can be formulated as

$$\mathbf{T} = \underline{\mathbf{k}}_{\mathbf{P}} \cdot \underline{\mathbf{k}}_{\mathbf{P}}^{*} = \frac{1}{2} \begin{bmatrix} |a|^2 & ab^* & ac^* \\ a^*b & |b|^2 & bc^* \\ a^*c & b^*c & |c|^2 \end{bmatrix} \tag{3}$$

where $a = S_{HH} + S_{VV}, b = S_{HH} - S_{VV}, c = 2S_{HV}$, and superscript * is a conjugate operator.

The vector consisting of the elements of coherent matrix $\mathbf{T}$ can be regarded as a kind of input features and adopted by a lot of methods. Besides, in order to better investigate the information contained in the scattering matrix $\mathbf{S}$ and the coherent matrix $\mathbf{T}$, a great number of features have been designed and can be mainly divided into coherent decomposition features, noncoherent decomposition features, and other features. Coherent decomposition features are obtained by decomposing the scattering matrix $\mathbf{S}$, which requires the scattering characteristics of the target to be steady and determinate, while noncoherent decomposition features are obtained by decomposing the coherent matrix $\mathbf{T}$, covariance matrix $\mathbf{C}$, Stokes matrix $\mathbf{K}$, or Muller matrix $\mathbf{M}$. Taking Pauli decomposition as an example, it decomposes the scattering matrix $\mathbf{S}$ into odd, 0° double, 45° double, and asymmetric reflection components, which are represented as follows:

$$\begin{aligned} \mathbf{S} &= \begin{bmatrix} S_{HH} & S_{HV} \\ S_{VH} & S_{VV} \end{bmatrix} \\ &= \frac{a}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{b}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ &\quad + \frac{c}{\sqrt{2}} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \frac{d}{\sqrt{2}} \begin{bmatrix} 0 & -j \\ j & 0 \end{bmatrix} \end{aligned} \tag{4}$$

where

$$\begin{aligned} a &= \frac{S_{\mathrm{HH}} + S_{\mathrm{VV}}}{\sqrt{2}}, \quad b = \frac{S_{\mathrm{HH}} - S_{\mathrm{VV}}}{\sqrt{2}} \\ c &= \frac{S_{\mathrm{HV}} + S_{\mathrm{VH}}}{\sqrt{2}}, \quad d = j\frac{S_{\mathrm{HV}} - S_{\mathrm{VH}}}{\sqrt{2}} \end{aligned} \tag{5}$$

$j = \sqrt{-1}$.

For the reciprocal backscattering case $S_{\mathrm{HV}} = S_{\mathrm{VH}}$, it decomposes the scattering matrix $\mathbf{S}$ into three components ($d = 0$).

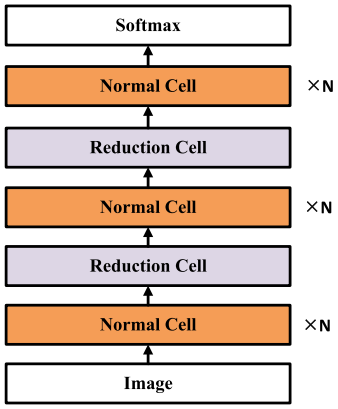


Fig. 2. NASNet search space. $\times N$ means repeat the cell for N times.

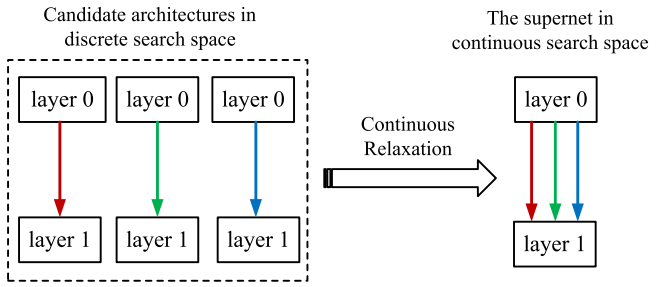


Fig. 3. Continuous relaxation of DARTS. The arrows in different colors between two layers denote different types of operations like 3×3 convolution, 5×5 convolution, etc.

Also, there are other handcrafted features carefully designed by human experts. We list all these commonly used features in Table I.

B. Differentiable Neural Architecture Search

Owing to the fact that there is no theoretical guarantee for designing the network structure and the manual designing process is time consuming and disgusting, NAS is developed to automatically design the network architectures. In the beginning, reinforcement learning [22], [31] and evolutionary algorithms [32], [33], [34] are employed to search good network architectures, but they require massive computation (i.e., thousands of GPU days) during the search process. Then, several approaches for reducing the cost while maintaining the quality of searched architectures have been developed. For example, NASNet [22] reduces the search cost by shrinking the size of search space. Specifically, it only searches two particular structures, called "normal cell" and "reduction cell," as shown in Fig. 2, of the network rather than the entire network. Normal cell and reduction cell can be regarded as the basic module, and we can combine them in a predefined way to build the entire network. Different normal cells have the same structure but different weights, so do reduction cells. Besides, ENAS [35] uses the weight sharing strategy to avoid training each candidate architecture from scratch to reduce the search cost.

However, they still need a large number of architecture evaluations for the search space is discrete. Luckily, Liu et al. [36] proposed to relax the discrete searching space to a continuous one, which is called continuous relaxation, as shown in Fig. 3. Then, we only need to train one supernet regardless of the number of candidate architectures, each of which is associated with a scalar called the architecture parameter. Besides, they proposed to model NAS as a bilevel optimization, which makes the supernet weights and architecture parameters can be alternatively optimized by the gradient descent algorithm. All these solutions proposed in [36] reduce the search cost from hundreds or thousands of GPU days to only a few GPU days. After the optimization process, we can decode and get the searched architecture by keeping the "arrow" with a maximal architecture parameter value. Finally, we train the searched architecture until convergence and obtain the model we want.

Owing to the good performance of differentiable NAS, numerous methods are developed based on it [36] and applied to many tasks, such as image classification [37], [38], semantic segmentation [39], [40], etc. The proposed *Pol-NAS* also belongs to the differentiable NAS method.

III. Proposed Method

In this section, our proposed *Pol-NAS* is presented in detail.

A. Search Space of Pol-NAS

Inspired by the state-of-the-art architectures [19], [20] in PolSAR image classification and some NAS methods [39], we propose a four-level hierarchical architecture search space, including intracell-, intercell-, layer-num-, and feature-level search, as shown in Fig. 4. Our goal is to find the intracell structure, intercell structure, the number of layers, and the importance of each input feature. Following the common practices, three "stem" layers are added in the beginning of the network. Their architectures are designed by human, as shown in Fig. 5. Intracell- and intercell-level search spaces are the same as those in [39]. We should note that we search the intracell structure, the intercell structure, and the number of layers in the search stage of *Pol-NAS* and search the importance of each input feature in the training stage.

1) Intracell-Level Search Space: We define *cell* to be a module, which is consistent with most NAS methods. A cell in layer l is a directed acyclic graph, which contains B blocks. The output tensor of cell in layer l is the concatenation of the blocks' output tensors $H_1^l, H_2^l, \ldots, H_B^l$. Each *block* has two branches, mapping two input tensors I_1 and I_2 to one output tensor H_i^l. Each branch chooses one operator O from the candidate operator set $\mathcal{O}$ and transforms its input tensor. The input tensor of each branch can be selected from the input tensor set $\mathcal{I}_i^l$ containing the outputs of the cell in previous layer H^{l-1}, the cell in previous layer H^{l-2} and all the previous blocks in the same cell. Candidate operator set $\mathcal{O}$ contains eight operators, which are widely used in CNNs and FCNs:

1) 3×3 atrous separable conv with rate 2;
2) 5×5 atrous separable conv with rate 2;
3) 3×3 depthwise-separable conv;

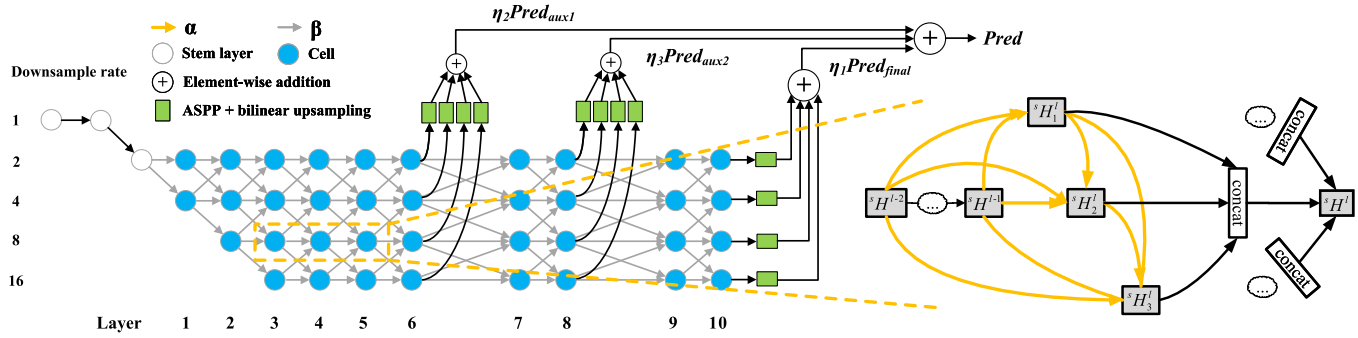


Fig. 4. Search space of PolNAS.

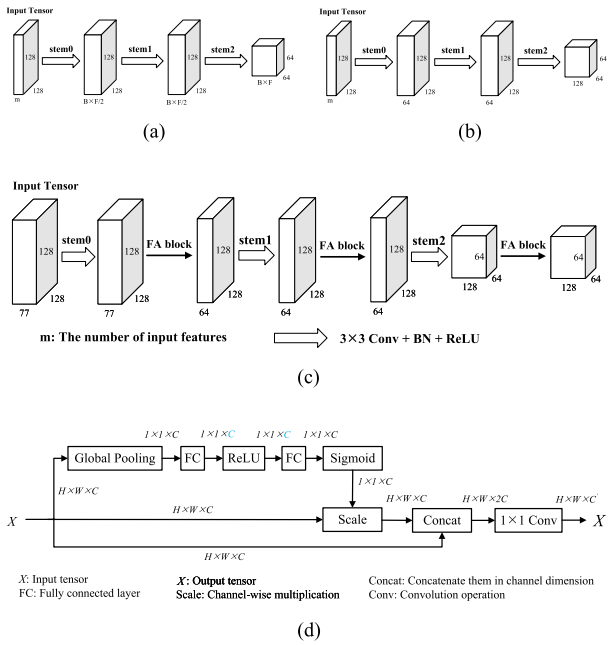


Fig. 5. (a) Stem layers in the search stage. (b) Original stem layers in the training stage. (c) Redesigned stem layers in the training stage. (d) FA block in redesigned stem layers.

4) 5×5 depthwise-separable conv;
5) 3×3 max pooling;
6) 3×3 average pooling;
7) skip connection;
8) no connection (zero).

In order to search the intracell structure via the differentiable method, we also use the continuous relaxation in [36]. The output of each block H_i^l is transformed from all the input tensors in $\mathcal{I}_i^l$, i.e.,

$$H_i^l = \sum_{H_j^l \in \mathcal{I}_i^l} O_{j\to i}\left(H_j^l\right). \tag{6}$$

Meanwhile, each $O_{j\to i}$ is approximated by its continuous relaxation $\bar{O}_{j\to i}$ denoted as

$$\bar{O}_{j\to i}\left(H_j^l\right) = \sum_{O^k \in \mathcal{O}} \alpha_{j\to i}^k O^k\left(H_j^l\right) \tag{7}$$

where

$$\sum_{k=1}^{|\mathcal{O}|} \alpha_{j\to i}^k = 1 \qquad \forall i, j \tag{8a}$$

$$\alpha_{j\to i}^k \geq 0 \qquad \forall i, j, k. \tag{8b}$$

We should note that the definition of $\alpha_{j\to i}^k$ is the same as that in [36] and [39]. They are scalars normalized by softmax and associated with each operator $O^k \in \mathcal{O}$. Then, the intercell update can be formulated as

$$H^l = \text{Cell}\left(H^{l-1}, H^{l-2}; \alpha\right). \tag{9}$$

2) Intercell-Level Search Space: Many NAS methods construct the entire network with the cell using a predefined pattern, which is designed by human experts. Considering the fact that the success of AutoDeepLab [39] and DCNAS [40], which both search how to stack the cell, and the intercell level variations of state-of-the-art models [19], [20] in PolSAR image classification, we can conclude that the stacking pattern of searched cell is also crucial to the performance of the entire network. Therefore, this pattern is searched in *Pol-NAS*.

The intercell-level search space is the same as that in [39], which designed the intercell-level continuous relaxation. All the tensors within a cell have the same spatial size. In other words, the height and width of output tensors of all the blocks within a cell are the same. The spatial sizes of output tensors of different cells in the same downsample rate are the same; otherwise, they are different. A scalar is also associated with each gray arrow in Fig. 4. Then, the intercell level update can be represented as

$$\begin{aligned} {}^s H^l = {} & \beta_{\frac{s}{2}\to s}^l \,\text{Cell}\left({}^{\frac{s}{2}}H^{l-1}, {}^s H^{l-2}; \alpha\right) \\ & + \beta_{s\to s}^l \,\text{Cell}\left({}^s H^{l-1}, {}^s H^{l-2}; \alpha\right) \\ & + \beta_{2\,s\to s}^l \,\text{Cell}\left({}^{2\,s} H^{l-1}, {}^s H^{l-2}; \alpha\right) \end{aligned} \tag{10}$$

where s denotes the downsample rate and $s = 2, 4, 8, 16$. l represents the layer number and $l = 1, 2, \ldots, L$. Because scalars β are normalized by softmax, so they meet the following conditions:

$$\beta_{s\to\frac{s}{2}}^l + \beta_{s\to s}^l + \beta_{s\to 2\,s}^l = 1 \qquad \forall s, l \tag{11a}$$

$$\beta_{s\to\frac{s}{2}}^l \geq 0 \quad \beta_{s\to s}^l \geq 0 \quad \beta_{s\to 2\,s}^l \geq 0 \qquad \forall s, l. \tag{11b}$$

3) Layer-num-Level Search Space: The overfitting phenomenon is a common problem, which means that the model performs well on training data but poorly on test data. It often occurs when the model is "big" (having a large number of parameters) with limited training data, which is very common in PolSAR image classification. Because the imaging mechanism of PolSAR data is different from that of optical images, so labeling PolSAR data accurately is difficult and usually needs field investigation. Actually, a model performing well on test data is the one we want. Therefore, some mechanisms should be employed to handle this problem. Usually, a model with a smaller Vapnik–Chervonenkis dimension [41], which is positive to the number of parameters, has a better generalization ability. The layer-num of a model is a critical factor in determining the number of parameters, and it is often set by trial and error or by human experience. Therefore, we choose to search the layer-num during the architecture search and propose a greedy strategy to decide how many layers should be used.

Inspired by Liu et al. [36] and Szegedy et al. [42], they add auxiliary classifiers in intermediate layers to increase the gradient signal (i.e., provide the gradient directly from the last layer rather than the one propagated back layer by layer, which can avoid gradient vanish to some extent that may affect the update of parameters in shallow layers) and provide additional regularization during training. And their losses are added to the total loss of the network with a discount weight (0.3 in [42] and 0.4 in [36]), which is formulated as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{final}}(y, Pred_{\text{final}}) + \lambda_1 \mathcal{L}_{\text{aux1}}(y, Pred_{\text{aux1}}) + \lambda_2 \mathcal{L}_{\text{aux2}}(y, Pred_{\text{aux2}}) \tag{12}$$

where $\mathcal{L}_{\text{total}}$ is the loss of the whole network. $\mathcal{L}_{\text{final}}$, $\mathcal{L}_{\text{aux1}}$, and $\mathcal{L}_{\text{aux2}}$ are the cross-entropy loss of the final, the first auxiliary, and the second auxiliary classifier, respectively. y is the ground truth. $Pred_{\text{final}}$, $Pred_{\text{aux1}}$, and $Pred_{\text{aux2}}$ are the result of last layer, the first auxiliary classifier, and the second auxiliary classifier, respectively. λ_1 and λ_2 are discount weights. With the help of auxiliary classifiers, a natural idea is to set the layer-num greedily with the best result given by the corresponding one. However, λ_1 and λ_2 are hyperparameters and have a huge impact on the performance of the trained model. If they are not properly set, the trained model will not perform well. In order to solve this problem, we choose to use auxiliary classifiers in another way. We define a scalar η associated with each classification result of them, which can be formulated as follows:

$$Pred = \eta_1 Pred_{\text{final}} + \eta_2 Pred_{\text{aux1}} + \eta_3 Pred_{\text{aux2}} \tag{13}$$

where $Pred$ is the classification result of the network. Then, only one cross-entropy loss is enough for training the network, which can be written as

$$\mathcal{L}_{\text{total}} = -\sum y \log(Pred). \tag{14}$$

η_1, η_2, and η_3 can be regarded as the architecture parameters and updated during the architecture search. η values are normalized by softmax. Therefore, they also meet the following conditions:

$$\eta_1 + \eta_2 + \eta_3 = 1 \tag{15a}$$

$$\eta_1, \eta_2, \eta_3 \geq 0. \tag{15b}$$

In *Pol-NAS*, we set the maximal layer-num $L = 10$. η_1, η_2, and η_3 are associated with the classification result of layer 10, layer 6, and layer 8, respectively. After the search process, we set the layer-num with the biggest η value and drop the subsequent layers. The maximal layer-num is set to 10 for two reasons. One the one hand, we set it according to some related work in NAS like AutoDeepLab [39] and DCNAS [40]. They set the layer-num in the range from 8 to 14, and these models are trained on datasets with plenty of labeled data. But we only have limited labeled training data in PolSAR image classification, as shown in Table III. If we set the layer-num larger, it may result in overfitting. This is also the reason why we set auxiliary classifiers to search the layer-num. On the other hand, it is set to 10 due to the GPU memory constraint. Therefore, the maximal layer-num is very unlikely to be larger than 10 for *Pol-NAS*. As for the positions of auxiliary classifiers, they are set by experience. In fact, we can also set them in layer 4 and layer 7. However, we believe a model with only four layers is too shallow.

4) Feature-Level Search Space: Input feature is another important factor that can influence the performance of the classification model. Most feature selection algorithms usually select an optimal feature subset, but the size of it for different classification models is different. On the one hand, the size is a hyperparameter that needs to be carefully configured. On the other hand, only when the classification model is determined can feature selection algorithms find the optimal subset for a specific classification model. Besides, the above process is time consuming and not friendly enough to the users.

In order to simplify the above process, we first choose to use all the features as the input of the classification model in both the search stage and the training stage, so we do not need to decide how many features should be selected. In the search stage, the goal is to find the optimal architecture including intracell and intercell architecture and layer-num. Then, in the training stage, we redesign the stem layers of our searched architecture with the help of the FA block proposed by us. Thus, the redesigned stem layers can automatically find and adaptively update the importance of each input feature during training. The redesigned stem layers and the FA block are shown in Fig. 5(c) and (d), respectively.

The FA block contains three branches. The upper two branches learn the importance of each channel (each input feature) and reweight the input features. Then, in order to improve the representation ability of input features, we concatenate the original and the reweighted features in channel dimension and utilize the 1×1 convolution to adjust the dimension of output features denoted as C', which is the same as the input dimension of the following layer. In a word, the FA block can be seen as an improved version of the SE block [30]. First, we remove the reduction ratio of the SE block to avoid tuning the hyperparameter. The dimension of output features of the first fully connected layer (FC) and ReLU are set to C, which is marked in blue. Second, we combine the original and the reweighted features to improve the representation ability of input features and give the model more autonomy.

B. Optimization

1) Architecture Search Stage: After introducing the continuous relaxation, these scalars associated with the four-level hierarchical architecture search space can be optimized by gradient descent. The bilevel optimization problem can be formulated as

$$\min_{\alpha,\beta,\eta} \quad \mathcal{L}_{trainB}\left(w^*(\alpha,\beta,\eta),\alpha,\beta,\eta\right) \tag{16a}$$

$$\text{s.t.} \quad w^*(\alpha,\beta,\eta) = \text{argmin}_w \mathcal{L}_{trainA}(w,\alpha,\beta,\eta) \tag{16b}$$

where $\mathcal{L}$ is the cross-entropy loss. We evenly split the training data into $trainA$ and $trainB$ as previous methods [36], [39] in order to avoid overfitting. Therefore, $\mathcal{L}_{trainA}$ and $\mathcal{L}_{trainB}$ are the loss calculated on $trainA$ and $trainB$, respectively. w is the learnable weight of the network. α, β, and η are scalars associated with intracell-level, intercell-level, and layer-num-level search spaces. Then, the optimization alternates between (17a) and (17b) until the stopping criterion

$$\nabla_w \mathcal{L}_{trainA}(w,\alpha,\beta,\eta) \tag{17a}$$

$$\nabla_{\alpha,\beta,\eta} \mathcal{L}_{trainB}(w,\alpha,\beta,\eta). \tag{17b}$$

2) Feature Importance Search in the Training Stage: After the search stage, we decode and train the searched architecture. Feature importance is searched and updated in the training process of decoded architecture with the help of our redesigned stem layers. The importance is saved in the FA block of each stem layer, which becomes a part of the trained model. Therefore, we do not need to do anything after the training process. The trained model is the one we want eventually.

C. Decoding the Searched Architecture

1) Intracell Architecture: The intracell structure consists of two parts: the two input tensors of each block and the operator of each branch. Following [36] and [39], the top-two strongest predecessors for each block are selected as the input tensors without considering "zero" connection, and the operator on each branch is decoded by retaining the strongest one.

2) Intercell Architecture: The intercell architecture indicates how to stack the searched cell, and β values contain all the information of our searched results. Actually, β values can be regarded as the "transition probability" between different "downsample rates" (spatial resolutions) across different layers. Our goal aims to find a path with maximal "transition probability" across the whole network. Therefore, following [39], the intercell architecture can be decoded by the Viterbi algorithm [43].

3) Layer-num: Considering the fact that the predicted result of the whole network in the search stage is made of the result of intermediate and last layers, we select the layer-num with biggest contribution (with maximal η value).

4) Feature Importance: Feature importance is updated in the training process of decoded architecture and saved in the FA block of each stem layer, which becomes a part of the trained model. Therefore, we do not need to decode it.

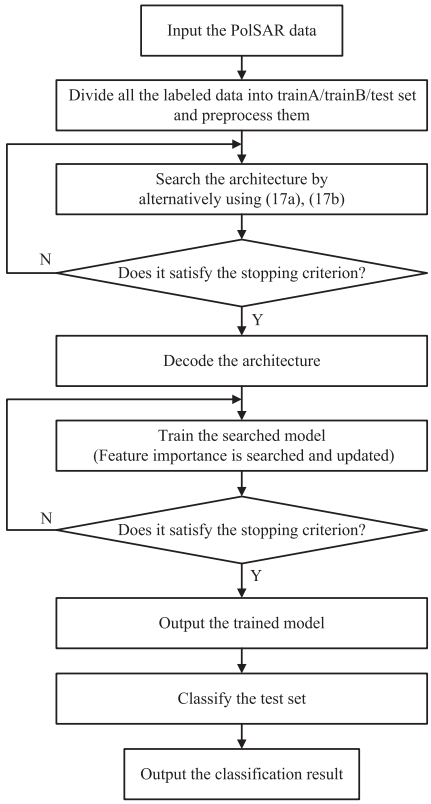


Fig. 6. Overall framework of *Pol-NAS*.

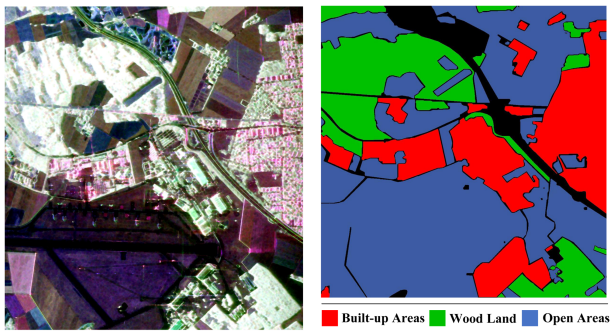


Fig. 7. (a) PauliRGB image of Oberpfaffenhofen. (b) Ground truth of Oberpfaffenhofen.

D. Overall Framework of Pol-NAS

The details of our proposed *Pol-NAS* have been introduced, and the overall framework is shown in Fig. 6.

IV. Experimental Results and Discussions

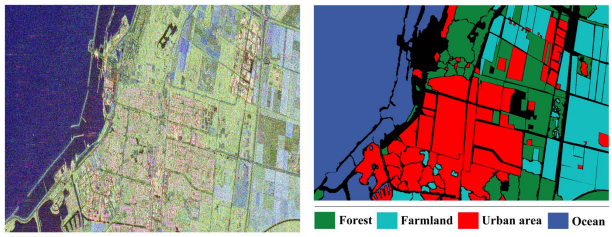
A. Dataset Description

In order to test the effectiveness of *Pol-NAS*, three real PolSAR datasets are used, and detailed information about them is listed in Table II.

Three PolSAR datasets are produced in Oberpfaffenhofen, Flevoland, and San Francisco, respectively. The PauliRGB images and the ground truths of them are shown in Figs. 7–9.

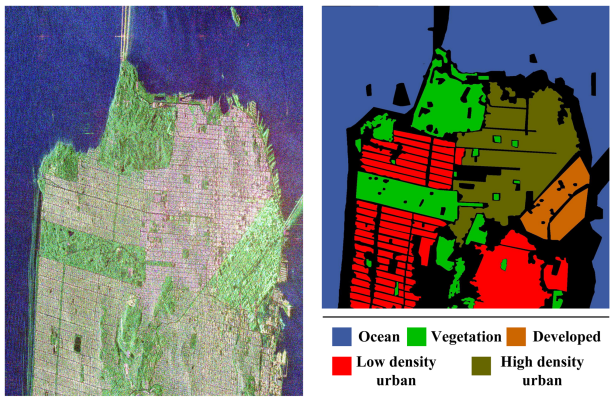
TABLE II
DETAILED INFORMATION ABOUT THREE POLSAR DATASETS

	Oberpfaffenhofen	Flevoland	San Francisco
Radar	E-SAR	Radarsat-2	Radarsat-2
Band	L	C	C
Year	1991	2008	2008
Resolution	3×2.2m	8×8m	10×5m
Polarimetric Information	Full polarimetric, multilook	Full polarimetric, multilook	Full polarimetric, multilook
Size	1300×1200	1000×1300	1800×1380
Classes	3	4	5



(a) (b)

Fig. 8. (a) PauliRGB image of Flevoland. (b) Ground truth of Flevoland.



(a) (b)

Fig. 9. (a) PauliRGB image of San Francisco. (b) Ground truth of San Francisco.

We should note that these datasets are widely used in PolSAR image classification, and these classes are defined by experts [15], [20], [29] not us. As for the differences between the woodland, forest, and vegetation that are very similar, we want to explain them here. The woodland and forest are areas covered with trees, but the forest is *thickly* covered. The vegetation is an area covered with trees and *other plants*.

B. Experimental Settings and Evaluation Criteria

1) Implementation Details of Pol-NAS:

1) Data preprocess: Patch size is 128×128. The refined Lee filter is adopted to reduce the speckle noise and z-score normalization is performed only when all the features are input to *Pol-NAS*. The divisions of $trainA$, $trainB$, and $test$ set of three PolSAR datasets in *Pol-NAS* are listed in Table III, which is set according to related work in PolSAR image classification like [14], [15], etc. And samples are randomly selected. The $test$ set in the search stage is the same as that in the training stage. Pol-NAS only uses $trainA$ and $trainB$ set in the search stage and the training stage. In other words, the $test$ set is only used for testing the classification performances of Pol-NAS and other comparison models in Table V.

TABLE III
DIVISIONS OF THREE POLSAR DATASETS IN POL-NAS

	Search Stage			Training Stage		
	trainA	trainB	test set	trainA	trainB	test set
Oberpfaffenhofen	0.5%	0.5%	99%	0.9%	0.1%	99%
Flevoland	0.5%	0.5%	99%	0.9%	0.1%	99%
San Francisco	0.1%	0.1%	99.8%	0.18%	0.02%	99.8%

2) In the search stage: The maximal layer-num L is set to 10, and a cell contains $B = 3$ blocks. Each cell with downsample rate s has $B \times F \times \frac{s}{2}$ output filters. Model capacity can be controlled by F, which is set to 6 in the search stage of *Pol-NAS*. We use a convolution with stride 2 to reduce the spatial size and double the number of filters for $\frac{s}{2} \rightarrow s$ connections. Bilinear upsampling and 1×1 convolution are used to increase the spatial size and halve the number of filters for $s \rightarrow \frac{s}{2}$ connections. The Atrous Spatial Pyramid Pooling (ASPP) module used in *Pol-NAS* is the same as that in [39], which contains only three branches including one 1×1 convolution, one 3×3 convolution with atrous rate $48/s$ and one image pooling [44]. Each branch in ASPP has $B \times F \times s/2$ output filters. $trainA$ and $trainB$ sets have the same size.

The architecture search optimization is performed for 80 epochs (the stopping criterion). Following [39], within the first 20 epochs, we only train the weights of the model because they found that the architecture may fall into the bad local optima when the weights are not well trained. In the remaining 60 epochs, weights are updated per epoch. But architecture parameters, α, β, and η are optimized only one epoch every ten epochs from epoch 20 to 79. In epochs that both weights and architecture parameters are optimized in, weights are optimized for one epoch first; then, architecture parameters are optimized for one epoch, i.e., not one iteration for weights and one for architecture parameters. Stochastic gradient descent (SGD) with cosine learning rate from 0.025 to 0.001, momentum 0.9, and weight decay 0.0003 is utilized to optimize weights. The Adam optimizer [45] with learning rate 0.001, $\rho = (0.9, 0.99)$, and weight decay 0.001 is employed to optimize architecture parameters. Batch size is set to 16 because of the GPU memory constraint. Horizontal and vertical flips are used to augment $trainA$ and $trainB$.

3) In the training stage: F is set to 8, which is different from that in the search stage. The ratio of $trainA$ and $trainB$ size is changed from 1:1 to 9:1. Because we have obtained

TABLE IV
DETAILED INFORMATION ABOUT THE BASELINE AND THREE EDITIONS OF POL-NAS

	Search Stage			Training Stage			
	Input Features		Layer-num Search	Input Features		Stem Layers	
	$\mathbf{T}$ matrix	All Features		$\mathbf{T}$ matrix	All Features	Original	Redesigned
Baseline	✓			✓		✓	
Pol-NAS-1	✓		✓	✓		✓	
Pol-NAS-2		✓	✓		✓	✓	
Pol-NAS-3		✓	✓		✓		✓

the searched architecture, we need to train the model until convergence. Therefore, $trainA$ is used to update the weights and $trainB$ is to select the best model in the training stage. Epoch is 100 and batch size is 16. The SGD optimizer with learning rate 0.001, momentum 0.9, and weight decay 0.0001 is adopted. Horizontal and vertical flips are only employed to augment $trainA$. Following [39], we adopt the encoder–decoder structure. The searched architecture is utilized as the encoder and the decoder is the same as the one in [39] and [46]. The low-level feature input to the decoder is from layer 3.

2) Evaluation Criterion: The overall accuracy (OA) and the Kappa coefficient [47] are used to evaluate the performances of different methods. The OA and the Kappa coefficient can be represented as follows:

$$\text{OA} = \frac{\sum_{i=1}^{r} x_{ii}}{N} \tag{18}$$

where N denotes the number of samples, r is the number of classes, and x_{ii} represents the number of samples belonging to the ith class in both the predicted result and the ground truth

$$\text{Kappa} = \frac{N\sum_{i=1}^{r} x_{ii} - \sum_{i=1}^{r} x_{i+} \cdot x_{+i}}{N^2 - \sum_{i=1}^{r} x_{i+} \cdot x_{+i}} \tag{19}$$

where x_{i+} denotes the number of samples belonging to the ith class in the ground truth. x_{+i} is the number of samples belonging to the ith class in the predicted result.

Besides, we also provide the search cost, training time, predict time, the memory cost for search [Mem. for Search (GB)], the memory cost for train [Mem. for Train (GB)], the number of parameters (Params), and the floating point operations (FLOPs) of each model. "h" represents hour and "s" denotes second. "GB" means gigabyte and "M" denotes million. "G" represents giga. We should note that the predict time is the time cost for predicting the whole image of each model. FLOPs is also the one for predicting the whole image.

C. Comparison Experiments and Results

In order to facilitate the description, we define four kinds of models including the baseline and three editions of *Pol-NAS* denoted as *Pol-NAS-1*, *Pol-NAS-2*, and *Pol-NAS-3*, respectively. Baseline is *Pol-NAS* without using layer-num search and redesigned stem layers (i.e., FA block). The differences between three editions of *Pol-NAS* are the input feature and stem layers adopted in the training stage. Detailed information about them is listed in Table IV.

In this subsection, the results of comparison experiments with handcrafted and autodesign models are presented first to prove the effectiveness of the searched architecture of *Pol-NAS-1*. The inputs of *Pol-NAS-1* and other methods are $\mathbf{T}$ matrix. Second, the complexity and the feature visualization results of different methods are provided. Third, ablation studies among the baseline, *Pol-NAS-1*, *Pol-NAS-2*, and *Pol-NAS-3* are performed to test the effectiveness of layer-num search and redesigned stem layers (i.e., FA block). Fourth, the searched architectures of *Pol-NAS* on three datasets are presented. Fifth, the generalization ability of searched architectures of *Pol-NAS-1* and *Pol-NAS-3* is investigated. Finally, we provide the results of different methods on data with various degradation variabilities.

All the comparison methods are implemented using Python 3.8 and PyTorch 1.7.0. All the experiments are run on a 3.30-GHz machine with 80.00-GB RAM and one Nvidia RTX 3090 GPU. All the results are the average ones of five independent runs. In Table V, $a \pm b$ denotes that the mean value is a and the standard deviation (std) is b. The best result is in bold and the second best one is underlined. "(-)" means the result of corresponding method is better than that of *Pol-NAS-1*, while "(+)" means it is worse. "$(c - d+)$" denotes that the results of c methods are better than that of *Pol-NAS-1*, while the results of d methods are worse.

1) Comparison Experiments With Handcrafted and Autodesign Models: Because *Pol-NAS-1* only uses $\mathbf{T}$ matrix as its input features and does not use the redesigned stem layers with the FA block in the training stage, *Pol-NAS-1* is selected and compared with several state-of-the-art methods in PolSAR image classification to test the effectiveness of searched architectures. These models include handcrafted and autodesign models, whose input features are also $\mathbf{T}$ matrix. SSAE-LSI [9], CV-CNN [12], PFDCN [13], DSNet [14], PolMPCNN [15], FCN [19], and CV-FCN [20] belong to handcrafted models. SAE-MOEA/D [23], PDAS [24], and CV-PDAS [24] are autodesign models. Considering that the input features of some models like PFDCN [13] and FCN [19] are not $\mathbf{T}$ matrix in the original papers, we provide additional results of them when they use $\mathbf{T}$ matrix as input features. The results of three PolSAR datasets with different methods are presented in Table V and Figs. 10–12. We can find that *Pol-NAS-1* has the best OA and Kappa, and the visualization results are almost the same as the ground truths.

Specifically, for Oberpfaffenhofen, *Pol-NAS-1* has the best mean values of OA and Kappa and obtains best mean results in most classes. FCN [19] obtains the second best mean values of OA and Kappa. The mean values of OA and Kappa of *Pol-NAS-1* are 0.49% and 0.82% higher than those of FCN [19], respectively. As for build-up area, the mean result of *Pol-NAS-1*

TABLE V
RESULTS OF THREE POLSAR DATASETS WITH DIFFERENT METHODS

		Handcrafted Models									Auto-design Models			
		SSAE-LSI	CV-CNN	PFDCN*	PFDCN	DSNet	PolMPCNN	FCN*	FCN	CV-FCN	SAE-MOEA/D	PDAS	CV-PDAS	Pol-NAS-1
Oberpfaffenhofen	Build-up areas	73.39 ± 1.52	81.93 ± 2.47	92.50 ± 0.50	88.97 ± 1.16	92.08 ± 1.46	86.17 ± 2.05	97.87 ± 0.62	**98.46 ± 0.15**	95.75 ± 0.72	75.52 ± 3.15	93.72 ± 1.00	93.47 ± 0.93	98.04 ± 0.50
	Wood land	83.69 ± 1.71	87.80 ± 1.81	97.09 ± 0.63	93.36 ± 0.80	96.52 ± 0.98	96.50 ± 1.36	98.27 ± 0.46	98.29 ± 0.18	96.55 ± 0.24	87.02 ± 0.83	96.47 ± 0.51	97.08 ± 0.61	**99.73 ± 0.10**
	Open areas	94.82 ± 0.70	97.47 ± 0.75	98.28 ± 0.25	97.87 ± 0.22	97.74 ± 0.45	97.99 ± 0.32	98.95 ± **0.14**	99.01 ± 0.15	98.19 ± 0.38	96.22 ± 0.32	98.27 ± 0.30	98.27 ± 0.27	**99.56** ± 0.20
	OA	87.40 ± 0.43	91.80 ± 0.40	96.62 ± 0.09	94.81 ± 0.18	96.11 ± 0.18	94.77 ± 0.29	98.55 ± 0.26	98.73 ± 0.11	97.27 ± 0.35	89.34 ± 0.48	96.80 ± **0.02**	96.85 ± 0.22	**99.22** ± 0.16
	Kappa	78.39 ± 0.80	85.82 ± 0.69	94.24 ± 0.15	91.12 ± 0.31	93.36 ± 0.30	90.99 ± 0.49	97.54 ± 0.45	97.84 ± 0.18	95.35 ± 0.60	81.70 ± 0.85	94.55 ± **0.04**	94.63 ± 0.38	**98.66** ± 0.27
	Search Cost (h)					-					**0.05 (-)**	6.00 (-)	6.96 (-)	17.04 (3-0+)
	Training Time (h)	**0.01 (-)**	0.09 (-)	2.04 (-)	2.06 (-)	0.2 (-)	0.63 (-)	0.73 (-)	0.77 (-)	4.03 (+)	**0.01 (-)**	0.72 (-)	1.86 (-)	2.64 (11-1+)
	Predict Time (s)	5.29 (-)	6.12 (-)	55.09 (+)	40.42 (+)	62.62 (+)	216.2 (+)	22.17 (-)	22.63 (-)	172.32 (+)	**4.73 (-)**	54.53 (+)	67.53 (+)	33.6 (5-7+)
	Mem. for Search (GB)					-					**2.62 (-)**	10.77 (-)	20.14 (+)	19.9 (2-1+)
	Mem. for Train (GB)	**1.29 (-)**	1.31 (-)	1.35 (-)	1.35 (-)	1.41 (-)	2.03 (-)	3.35 (-)	3.37 (-)	5.79 (+)	2.62 (-)	1.55 (-)	1.79 (-)	5.64 (11-1+)
	Params (M)	0.13 (-)	**0.02 (-)**	0.15 (-)	0.15 (-)	0.06 (-)	0.19 (-)	43.23 (+)	43.23 (+)	1.48 (-)	0.04 (-)	5.31 (+)	3.62 (+)	2.14 (8-4+)
	FLOPs (G)	392.27 (-)	**80.51 (-)**	2064.58 (-)	1859.55 (-)	620.97 (-)	38084 (+)	9128.34 (+)	9194.09 (+)	1678.89 (-)	118.26 (-)	39087.55 (+)	167378.77 (+)	5105.99 (7-5+)
Flevoland	Forest	67.28 ± 3.77	81.43 ± 1.99	83.23 ± 2.56	79.43 ± 3.38	85.10 ± 0.97	90.69 ± 1.16	95.01 ± 1.46	93.09 ± 1.02	90.91 ± 0.56	69.59 ± 1.09	85.81 ± 0.67	85.89 ± 1.17	**98.50 ± 0.31**
	Farmland	74.88 ± 4.59	89.80 ± 0.67	91.10 ± 0.83	87.27 ± 3.13	91.48 ± 0.45	92.34 ± 1.44	97.97 ± 0.29	96.62 ± 0.59	93.69 ± 0.56	76.09 ± 0.47	91.58 ± 0.78	91.26 ± 0.58	**98.74 ± 0.11**
	Urban area	72.33 ± 2.31	84.44 ± 1.38	90.55 ± 1.32	89.59 ± 2.86	89.67 ± 0.78	91.74 ± 1.74	97.32 ± 0.33	96.31 ± 0.69	96.11 ± 0.38	74.36 ± 1.06	89.20 ± 0.31	88.31 ± 1.27	**98.65 ± 0.18**
	Ocean	99.51 ± 0.07	**99.82 ± 0.03**	99.66 ± 0.14	99.68 ± 0.10	99.6 ± 0.06	99.53 ± 0.18	99.38 ± 0.06	99.45 ± 0.11	99.43 ± 0.10	99.65 ± 0.09	99.76 ± 0.05	99.70 ± 0.09	99.62 ± **0.02**
	OA	80.81 ± 0.45	90.19 ± 0.19	92.30 ± 0.19	90.55 ± 0.23	92.44 ± 0.15	94.18 ± 0.20	97.71 ± 0.25	96.82 ± 0.42	95.75 ± 0.09	82.11 ± 0.43	92.50 ± 0.19	92.18 ± 0.41	**98.96 ± 0.03**
	Kappa	73.94 ± 0.66	86.64 ± 0.25	89.52 ± 0.26	87.13 ± 0.33	89.73 ± 0.20	92.09 ± 0.27	96.89 ± 0.34	95.68 ± 0.57	94.22 ± 0.12	75.69 ± 0.57	89.82 ± 0.25	89.38 ± 0.55	**98.59 ± 0.04**
	Search Cost (h)					-					**0.05 (-)**	5.04 (-)	6.48 (-)	13.92 (3-0+)
	Training Time (h)	**0.01 (-)**	0.07 (-)	1.63 (-)	1.62 (-)	0.16 (-)	0.61 (-)	0.6 (-)	0.64 (-)	2.81 (+)	**0.01 (-)**	0.59 (-)	1.63 (-)	1.89 (11-1+)
	Predict Time (s)	4.34 (-)	5.14 (-)	43.41 (+)	33.22 (+)	52.03 (+)	210.9 (+)	18.3 (-)	18.73 (-)	190.14 (+)	**2.34 (-)**	44.78 (+)	80.18 (+)	29.04 (5-7+)
	Mem. for Search (GB)					-					**2.7 (-)**	8.76 (-)	16.12 (-)	19.9 (3-0+)
	Mem. for Train (GB)	**1.27 (-)**	1.28 (-)	1.33 (-)	1.33 (-)	1.37 (-)	1.96 (-)	3.4 (-)	3.42 (-)	5.93 (+)	2.7 (-)	1.63 (-)	2.25 (-)	4.49 (11-1+)
	Params (M)	0.13 (-)	**0.02 (-)**	0.15 (-)	0.15 (-)	0.06 (-)	0.25 (-)	43.23 (+)	43.23 (+)	1.48 (-)	0.04 (-)	5.38 (+)	28.4 (+)	3.15 (8-4+)
	FLOPs (G)	326.89 (-)	**67.09 (-)**	1721.06 (-)	1550.21 (-)	517.51 (-)	42315.55 (+)	7572.59 (+)	7627.08 (+)	1396.63 (-)	107.21 (-)	73214.31 (+)	447614.53 (+)	3003.81 (7-5+)
SanFrancisco	Ocean	99.93 ± 0.05	99.99 ± 0.01	99.99 ± 0.01	99.98 ± 0.01	99.97 ± 0.01	99.69 ± 0.29	99.96 ± 0.03	99.96 ± 0.02	99.93 ± 0.05	**100.0 ± 0.00**	99.98 ± 0.01	99.98 ± 0.02	99.98 ± 0.02
	Vegetation	84.25 ± 3.91	92.80 ± **0.22**	93.03 ± 0.60	91.85 ± 1.18	91.26 ± 0.62	91.57 ± 1.40	95.37 ± 1.42	94.21 ± 1.10	95.27 ± 0.52	88.63 ± 0.64	93.15 ± 0.57	94.23 ± 0.43	**96.94** ± 1.10
	Low density urban	72.38 ± 2.57	90.10 ± 2.21	95.20 ± 0.93	87.04 ± 1.98	95.84 ± 0.42	98.13 ± 0.92	98.64 ± 0.45	97.49 ± 0.57	99.35 ± **0.14**	79.79 ± 1.65	96.18 ± 1.27	94.52 ± 1.31	**99.67** ± 0.19
	High density urban	71.43 ± 5.55	93.14 ± 0.93	95.84 ± 1.16	84.06 ± 2.73	96.68 ± 0.25	95.73 ± 2.72	99.50 ± 0.18	97.90 ± 1.06	**99.69 ± 0.05**	79.17 ± 1.36	95.81 ± 1.10	96.91 ± 0.75	99.65 ± 0.12
	Developed	64.25 ± 7.73	85.84 ± 1.33	91.92 ± 2.72	89.47 ± 5.15	94.66 ± 1.29	93.52 ± 5.48	99.43 ± 0.35	97.66 ± 2.53	99.72 ± 0.32	70.23 ± 2.90	94.38 ± 0.80	93.19 ± 1.68	**99.99 ± 0.01**
	OA	86.38 ± 0.55	95.46 ± 0.39	97.09 ± 0.12	93.31 ± 0.26	97.22 ± **0.06**	97.33 ± 0.71	99.01 ± 0.25	98.30 ± 0.35	99.16 ± 0.08	89.92 ± 0.13	97.40 ± 0.08	97.33 ± 0.22	**99.47** ± 0.09
	Kappa	80.40 ± 0.77	93.45 ± 0.56	95.84 ± 0.17	90.46 ± 0.38	96.04 ± **0.09**	96.23 ± 1.00	98.57 ± 0.36	97.55 ± 0.51	98.79 ± 0.11	85.50 ± 0.19	96.28 ± 0.11	96.20 ± 0.31	**99.24** ± 0.14
	Search Cost (h)					-					**0.05 (-)**	1.2 (-)	1.92 (-)	27.12 (3-0+)
	Training Time (h)	**0.01 (-)**	0.03 (-)	0.79 (-)	0.79 (-)	0.51 (-)	0.26 (-)	1.26 (-)	1.27 (-)	9.30 (+)	**0.01 (-)**	0.28 (-)	0.61 (-)	4.39 (11-1+)
	Predict Time (s)	**7.41 (-)**	9.58 (-)	83.03 (+)	65.46 (+)	98.83 (+)	532.9 (+)	36.77 (-)	37.23 (-)	427.92 (+)	10.84 (-)	86.01 (+)	105.67 (+)	56.97 (5-7+)
	Mem. for Search (GB)					-					**2.86 (-)**	4.24 (-)	6.99 (-)	19.9 (3-0+)
	Mem. for Train (GB)	**1.25 (-)**	1.28 (-)	1.28 (-)	1.28 (-)	1.31 (-)	1.67 (-)	3.4 (-)	3.42 (-)	6.11 (+)	2.86 (-)	1.34 (-)	2.02 (-)	4.58 (11-1+)
	Params (M)	0.13 (-)	**0.02 (-)**	0.15 (-)	0.15 (-)	0.06 (-)	0.31 (-)	43.23 (+)	43.23 (+)	1.48 (-)	0.04 (-)	0.5 (-)	28.24 (+)	2.38 (8-3+)
	FLOPs (G)	624.62 (-)	**128.20 (-)**	3289.66 (-)	2963.19 (-)	988.91 (-)	101069.07 (+)	15232.22 (+)	15341.67 (+)	2904.97 (-)	180.17 (-)	65195.09 (+)	496992 (+)	10783.07 (7-5+)

* means the input features of the corresponding model are not $\mathbf{T}$ matrix in the original paper. We change its features to $\mathbf{T}$ matrix.

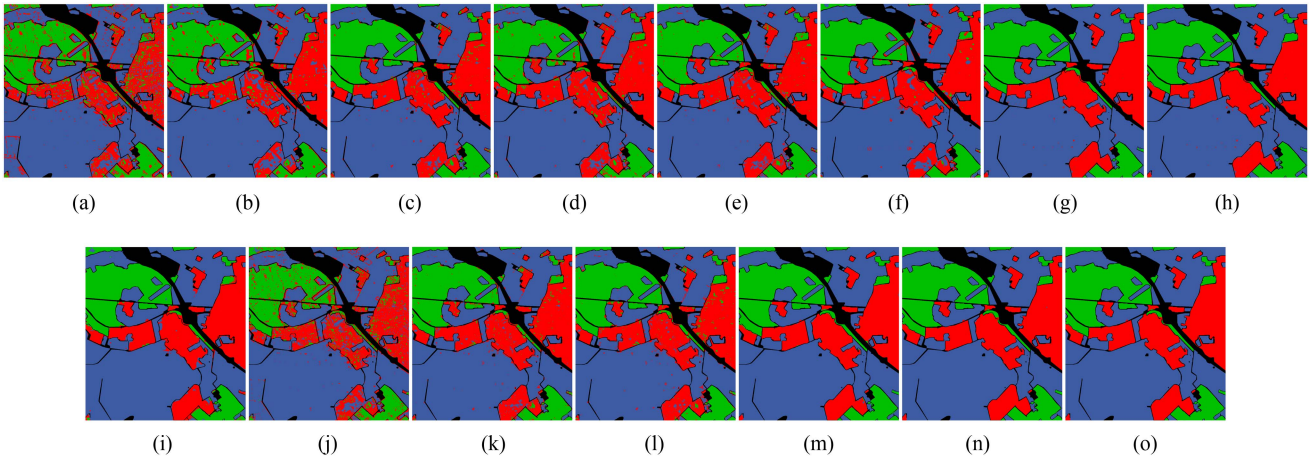


Fig. 10. Predicted results of (a) SSAE-LSI, (b) CV-CNN, (c) PFDCN*, (d) PFDCN, (e) DSNet, (f) PolMPCNN, (g) FCN*, (h) FCN, (i) CV-FCN, (j) SAE-MOEA/D, (k) PDAS, (l) CV-PDAS, (m) Pol-NAS-1, (n) Pol-NAS-2, and (o) Pol-NAS-3 on Oberpfaffenhofen dataset.

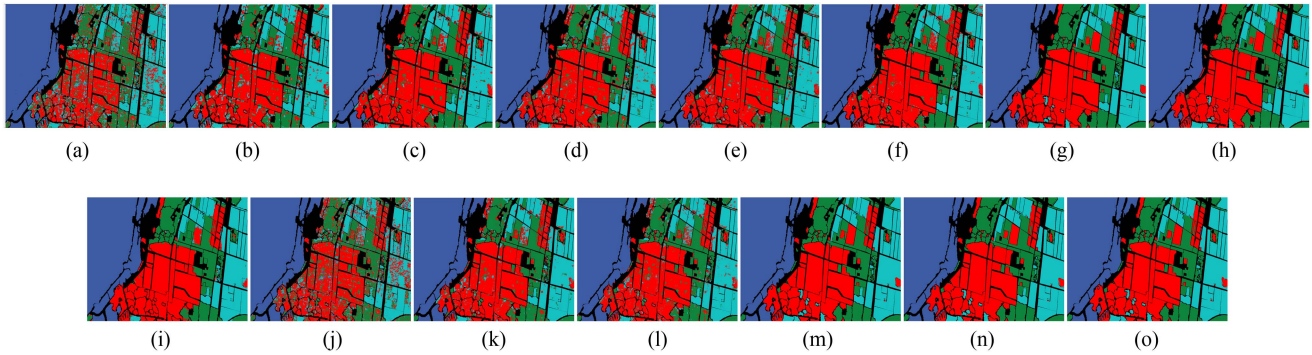


Fig. 11. Predicted results of (a) SSAE-LSI, (b) CV-CNN, (c) PFDCN*, (d) PFDCN, (e) DSNet, (f) PolMPCNN, (g) FCN*, (h) FCN, (i) CV-FCN, (j) SAE-MOEA/D, (k) PDAS, (l) CV-PDAS, (m) Pol-NAS-1, (n) Pol-NAS-2, and (o) Pol-NAS-3 on the Flevoland dataset.

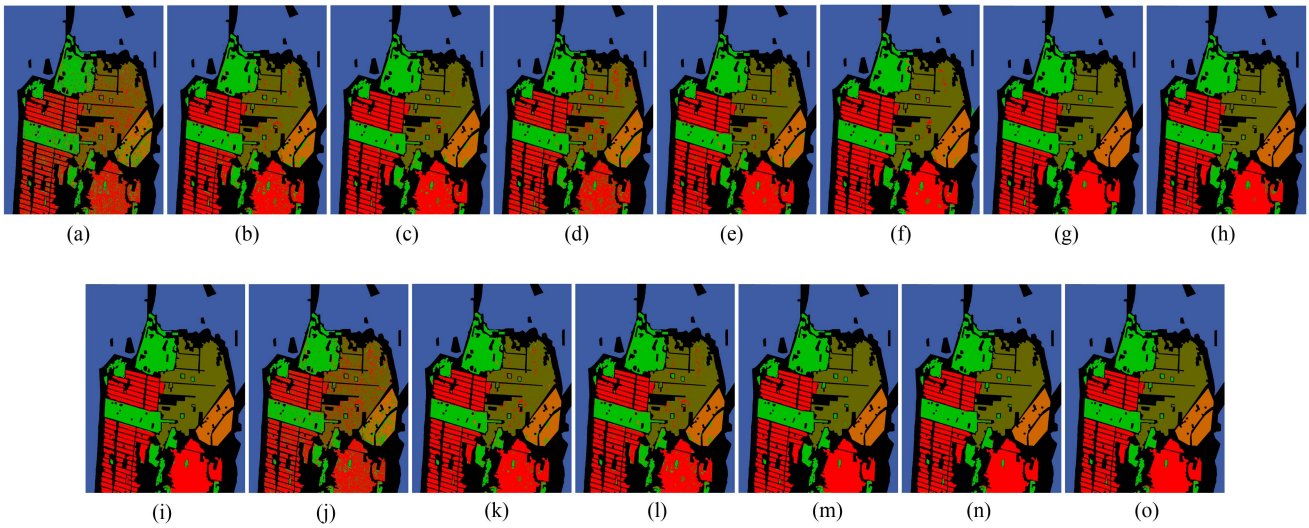


Fig. 12. Predicted results of (a) SSAE-LSI, (b) CV-CNN, (c) PFDCN*, (d) PFDCN, (e) DSNet, (f) PolMPCNN, (g) FCN*, (h) FCN, (i) CV-FCN, (j) SAE-MOEA/D, (k) PDAS, (l) CV-PDAS, (m) Pol-NAS-1, (n) Pol-NAS-2, and (o) Pol-NAS-3 on the San Francisco dataset.

is only 0.42% lower than that of FCN and much higher than that of others. Besides, compared with other methods, *Pol-NAS-1* has low standard deviations, which indicates that it has good classification stability. As for the time and memory cost, although the search cost of *Pol-NAS-1* is much higher than that of other autodesign models, the mean values of OA and Kappa of *Pol-NAS-1* are 9.88% and 16.96% higher at most and 2.37% and 4.03% higher at least. The training time, predict time, Mem. for Search, Mem. for Train, Params, and FLOPs of *Pol-NAS-1* are all between the best and the worst one, respectively, but *Pol-NAS-1* has the best mean value of OA and Kappa with a low standard deviation, which is also between the best and the worst one.

For Flevoland, *Pol-NAS-1* has the best mean values of OA and Kappa and obtains best mean results in most classes. FCN* [19] obtains the second best mean values of OA and Kappa. The mean values of OA and Kappa of *Pol-NAS-1* are 1.25% and 1.70% higher than those of FCN* [19], respectively. As for ocean, the mean result of *Pol-NAS-1* is only 0.20% lower than that of CV-CNN. Besides, compared with other methods, *Pol-NAS-1* has the lowest standard deviations, which indicates that it has good classification stability. As for the time and memory cost, although the search cost of *Pol-NAS-1* is much higher than that of other autodesign models, the mean values of OA and Kappa of *Pol-NAS-1* are 16.85% and 23.90% higher at most and 6.46% and 8.77% higher at least. The Mem. for Search of *Pol-NAS-1* is 3.78 GB higher than that of CV-PDAS, but the mean values of OA and Kappa are 6.78% and 9.21% higher than those of CV-PDAS. The training time, predict time, Mem. for Train, Params, and FLOPs of *Pol-NAS-1* are all between the best and the worst one, respectively, but *Pol-NAS-1* has the best mean value of OA and Kappa with a low standard deviation, which is also between the best and the worst one.

For San Francisco, *Pol-NAS-1* has the best mean values of OA and Kappa and obtains best mean results in most classes. CV-FCN [20] obtains the second best mean values of OA and Kappa. The mean values of OA and Kappa of *Pol-NAS-1* are 0.31% and 0.45% higher than those of CV-FCN [20], respectively. For ocean, the mean result of *Pol-NAS-1* is only 0.02% lower than that of SAE-MOEA/D. For high density urban, the mean result of *Pol-NAS-1* is only 0.04% lower than that of CV-FCN. As for the time and memory cost, although the search cost and Mem. for Search of *Pol-NAS-1* are much higher than those of other autodesign models, the mean values of OA and Kappa of *Pol-NAS-1* are 9.55% and 13.74% higher at most and 2.07% and 2.96% higher at least. The training time, predict time, Mem. for Train, Params, and FLOPs of *Pol-NAS-1* are all between the best and the worst one, respectively, but *Pol-NAS-1* has the best mean value of OA and Kappa with a low standard deviation, which is also between the best and the worst one.

In summary, we can draw the conclusions as follows.

1) The architectures searched by *Pol-NAS-1* are always the best on all datasets. But the second best ones on these datasets are not the same (i.e., FCN on Oberpfaffenhofen, FCN* on Flevoland, and CV-FCN on San Francisco), which means we still need to select a model by ourselves when we need to cope with a new dataset if we do not use *Pol-NAS-1*.
2) The search costs of *Pol-NAS-1* on these datasets are much higher than those of other autodesign models, but the mean values of OA and Kappa of *Pol-NAS-1* are 16.85% and 23.90% higher at most and 2.07% and 2.96% higher at least on these datasets. (In fact, when we need to cope with a new dataset, we can use the searched architecture of *Pol-NAS-1* on Oberpfaffenhofen directly to avoid the high search cost, because the searched architecture of *Pol-NAS-1* has a good generalization ability, which can be seen from

TABLE VI
TIME AND SPACE COMPLEXITY OF DIFFERENT METHODS

		Handcrafted Models									Auto-design Models					
		SSAE-LSI	CV-CNN	PFDCN*	PFDCN	DSNet	PolMPCNN	FCN*	FCN	CV-FCN	SAE-MOEA/D	PDAS	CV-PDAS	Pol-NAS-1	Pol-NAS-2	Pol-NAS-3
Time Complexity	Search	–						–			$O(HW*TV\_ratio)$			$O(\lceil HW/(ST)^2 \rceil)$		
	Train	$O(HW*TV\_ratio)$						$O(\lceil HW/(ST)^2 \rceil)$			$O(HW*TV\_ratio)$			$O(\lceil HW/(ST)^2 \rceil)$		
	Predict	$O(HW)$						$O(\lceil HW/(ST)^2 \rceil)$			$O(HW)$			$O(\lceil HW/(ST)^2 \rceil)$		
Space Complexity	Search	–									$O(BS*(PS)^2)$					
	Train	$O(BS*(PS)^2)$									$O(BS*(PS)^2)$					
	Predict	$O(BS*(PS)^2)$									$O(BS*(PS)^2)$					

* means the input features of the corresponding model are not $\mathbf{T}$ matrix in the original paper. We change its features to $\mathbf{T}$ matrix.
H: The height of a whole image. W: The width of a whole image. TV_ratio: The ratio of training set and validation set. ST: The stride between adjacent patches. PS: The patch size of an input patch. BS: The batch size.

Table VIII. If you want to find the optimal architecture, you can use *Pol-NAS-1* to search it on the new dataset.)

3) The Training time, predict time, Mem. for Train, Params, and FLOPs of *Pol-NAS-1* on all the datasets are neither the best nor the worst one, which means the time and memory costs of *Pol-NAS-1* are on par with existing classification models.

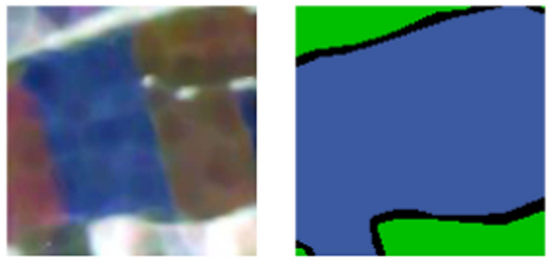
2) Time and Space Complexity of Different Methods: We provide the time and space complexity of different methods in Table VI. We first introduce how to obtain the time and space complexity of *Pol-NAS* and then compare the complexity of *Pol-NAS* with that of other methods.

In all the stages, *Pol-NAS* crops the whole image into many patches. Therefore, the number of patches *Pol-NAS* needs to handle can be calculated as follows:

$$\begin{aligned} patch\_num &= \left(\left\lceil \frac{H-PS+pad}{ST} \right\rceil + 1\right) \\ &\quad \times \left(\left\lceil \frac{W-PS+pad}{ST} \right\rceil + 1\right) \\ &\approx \left(\left\lceil \frac{H}{ST} \right\rceil + 1\right)\left(\left\lceil \frac{W}{ST} \right\rceil + 1\right) \\ &= \left\lceil \frac{H}{ST} \right\rceil \left\lceil \frac{W}{ST} \right\rceil + \left\lceil \frac{H}{ST} \right\rceil + \left\lceil \frac{W}{ST} \right\rceil + 1. \quad (20) \end{aligned}$$

H and W are the height and width of a whole image, respectively. PS denotes the patch size of an input patch. pad represents the number of pixels with zero padding. ST is the stride between adjacent patches. Therefore, the time complexity of *Pol-NAS-1* in all stages can be denoted as $O(\lceil HW/(ST)^2 \rceil$. Equation (20) also applies to *Pol-NAS-2* and *Pol-NAS-3*. As for the space complexity of *Pol-NAS-1*, in all stages, only BS patches with the size of $PS * PS$ are input to the model. (BS denotes batch size.) Therefore, the space complexity of *Pol-NAS-1* in all the stages can be denoted as $O(BS*(PS)^2)$. *Pol-NAS-2* and *Pol-NAS-3* have the same space complexity.

As for the comparison of complexity of *Pol-NAS* and other methods, we can conclude that *Pol-NAS and other methods have the same time and space complexity*. TV_ratio, ST, BS, and PS are predefined hyperparameters and can be regarded as constants. (TV_ratio is the ratio of training set and validation set. ST denotes the stride between adjacent patches. PS is the patch size.) Therefore, with regard to different datasets, the time and space complexity of *Pol-NAS* and other methods can be approximately represented as $O(HW)$ and $O(1)$, respectively.



(a) (b)

Fig. 13. (a) PauliRGB of input patch for the feature visualization of each model. (b) Ground truth of input patch for the feature visualization of each model.

3) Visualization of Feature Maps: To give a more intuitive comparison between *Pol-NAS-1* and other state-of-the-art methods, we provide the results of feature visualization of these methods. A patch from the Oberpfaffenhofen dataset is selected as the input patch of each model. The PauliRGB and ground truth of the input patch are shown in Fig. 13.

We only visualize the feature maps of PFDCN*, DSNet, PolMPCNN, FCN*, CV-FCN, PDAS, CV-PDAS, and *Pol-NAS-1* for two reasons. First, the input features of these methods are all $\mathbf{T}$ matrix. Second, consider the fact that the first three layers in *Pol-NAS-1* are stem layers (manually designed), and the results of these layers cannot prove the effectiveness of *Pol-NAS-1*. Therefore, we choose to visualize the features of searched cell in layer 1 of *Pol-NAS-1*, i.e., the fourth layer of *Pol-NAS-1* (including the three stem layers) and other methods. Therefore, comparison methods with less than four layers in Table V (i.e., SSAE-LSI, CV-FCN and SAE-MOEA/D) are not selected for feature visualization for the sake of fairness. The activations of the fourth layer of each model are not only shown separately by channel, but also shown after elementwise addition (i.e., sum feature). The number below each visualized feature denotes the channel index of the activations.

We only provide the results of FCN*, CV-FCN, and *Pol-NAS-1*, as shown in Figs. 14–16, in the main body of this article due to space limitation. (Complete results can be found at https://github.com/guangyuanLiu/Pol-NAS.) From the visualization results, we can find that the features of *Pol-NAS-1* have a clearer outline of the input patch than those of other methods. Besides, *Pol-NAS-1* does not have dead activations (i.e., a whole activation having a value of 0, as shown in Fig. 14). Therefore, we can draw the conclusion that *Pol-NAS-1* has a better ability of feature extraction, which proves that the architecture of *Pol-NAS-1* is better than that of other methods.

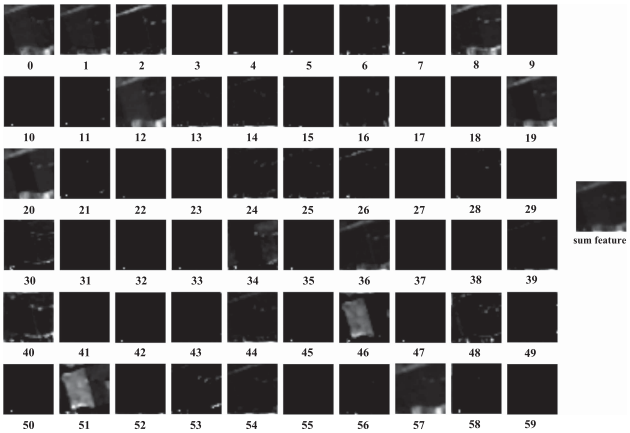


Fig. 14. Features for the input patch of FCN*.

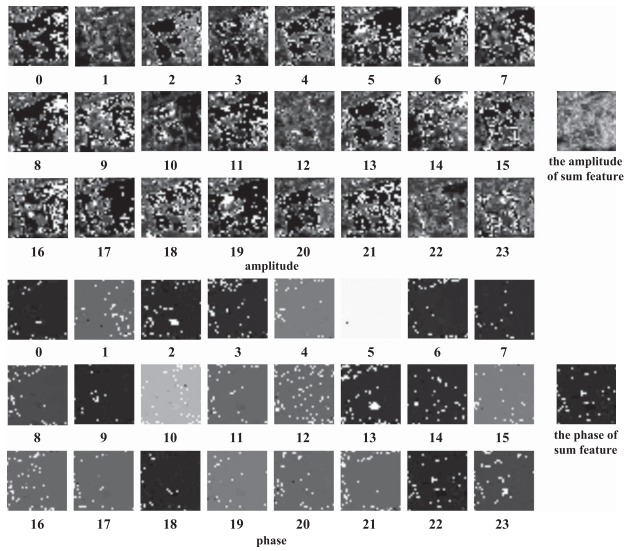


Fig. 15. Features for the input patch of CV-FCN.

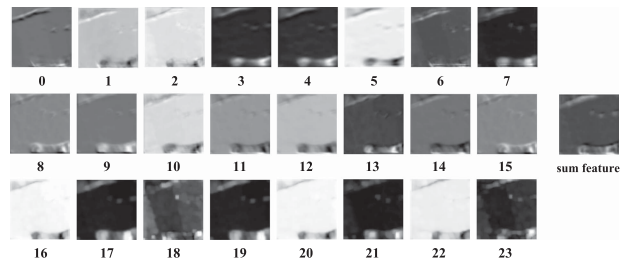


Fig. 16. Features for the input patch of Pol-NAS-1.

4) Ablation Studies: Here, we conduct ablation experiments to prove the effectiveness of layer-num search and feature selection (the effectiveness of FA block/redesigned stem layers of searching the importance of input features) in *Pol-NAS*. We provide the results of the baseline and three editions of our proposed method denoted as *Pol-NAS-1*, *Pol-NAS-2*, and *Pol-NAS-3* on three datasets in Table VII. Comparing the results of the baseline and *Pol-NAS-1*, we can conclude that the layer-num search is effective because the mean values of OA and Kappa of *Pol-NAS-1* are higher than those of the baseline on all datasets. Besides, the search costs of them are exactly the same. The training time, predict time, Mem. for Search, Mem. for Train, Params, and FLOPs of *Pol-NAS-1* and *Pol-NAS-2* are almost the same. As for the fact that the gaps of OA and Kappa between them are not big, it is because the space for improvement is limited. We want to emphasize that *improving the classification results is one of our goals. The most important thing is that layer-num search can help us automatically find the optimal number of layers rather than set it manually by trial and error.*

TABLE VII
RESULTS OF THE BASELINE AND THREE EDITIONS OF POL-NAS

		Baseline	Pol-NAS-1	Pol-NAS-2	Pol-NAS-3
Oberpfaffenhofen	OA	98.61 ± 0.38	99.22 ± 0.16	99.54 ± 0.06	**99.56 ± 0.03**
	Kappa	97.62 ± 0.66	98.66 ± 0.27	99.22 ± 0.1	**99.26 ± 0.05**
	Search Cost (h)	17.04	17.04	17.04	17.04
	Training Time (h)	2.98	**2.64**	2.92	2.86
	Predict Time (s)	41.46	**33.6**	35.40	35.70
	Mem. for Search (GB)	**19.71**	19.89	19.90	19.90
	Mem. for Train (GB)	**4.71**	5.64	9.50	10.73
	Params (M)	2.42	2.14	**1.93**	1.95
	FLOPs (G)	**4063.45**	5105.99	12636.08	13218.17
Flevoland	OA	98.94 ± 0.1	98.96 ± 0.03	99.09 ± 0.02	**99.11 ± 0.02**
	Kappa	98.56 ± 0.13	98.59 ± 0.04	98.77 ± 0.03	**98.8 ± 0.03**
	Search Cost (h)	13.92	13.92	13.92	13.92
	Training Time (h)	2.28	**1.89**	2.19	2.28
	Predict Time (s)	29.69	**29.04**	29.58	30.35
	Mem. for Search (GB)	**19.71**	19.89	19.90	19.90
	Mem. for Train (GB)	**4.30**	4.49	7.12	8.16
	Params (M)	2.47	3.15	**2.43**	2.45
	FLOPs (G)	5455.21	**3003.81**	6700.09	7182.40
San Francisco	OA	99.4 ± 0.08	99.47 ± 0.09	99.48 ± 0.11	**99.52 ± 0.03**
	Kappa	99.13 ± 0.11	99.24 ± 0.14	99.25 ± 0.16	**99.31 ± 0.04**
	Search Cost (h)	27.12	27.12	27.12	27.12
	Training Time (h)	4.53	**4.39**	4.58	4.63
	Predict Time (s)	57.74	**56.97**	58.22	57.88
	Mem. for Search (GB)	**19.71**	19.90	19.91	19.91
	Mem. for Train (GB)	**4.53**	4.58	7.08	8.11
	Params (M)	**2.07**	2.38	**2.07**	2.09
	FLOPs (G)	12610.97	**10783.07**	15189.23	16158.22

From the results of *Pol-NAS-2* and *Pol-NAS-3*, we can conclude that the proposed redesigned stem layers (i.e., FA block) are effective and can improve the performance of *Pol-NAS*. Besides, the search costs of *Pol-NAS-2* and *Pol-NAS-3* on each dataset are exactly the same. The training time, predict time, Mem. for Search, Mem. for Train, Params, and FLOPs of *Pol-NAS-2* and *Pol-NAS-3* are almost the same. As for the fact that the gaps of OA and Kappa between them are not big, it is because the space for improvement is limited. We want to emphasize that *improving the classification results is one of our goals. The most important thing is that the FA block can help us automatically find the optimal input features (the importance of each feature) rather than set them manually by trial and error.*

We do not compare *Pol-NAS* with other feature selection algorithms in PolSAR image classification, because feature selection algorithms select a feature subset as the input of a classification model. But *Pol-NAS* with feature selection (i.e., *Pol-NAS-3*) takes all features as the input. Experimental results under the condition of different input features and different architectures cannot prove anything.

In summary, we can draw the conclusions as follows.

1) The layer-num search and feature selection (i.e., FA block) are effective and can further improve the classification results. *But the most important thing is that we can avoid setting the layer-num and finding the optimal input features by trial and error with the help of them.*
2) The baseline, *Pol-NAS-1*, *Pol-NAS-2*, and *Pol-NAS-3* have almost the same costs of search cost, training time, predict time, Mem. for Search, and Params. Only *Pol-NAS-2* and

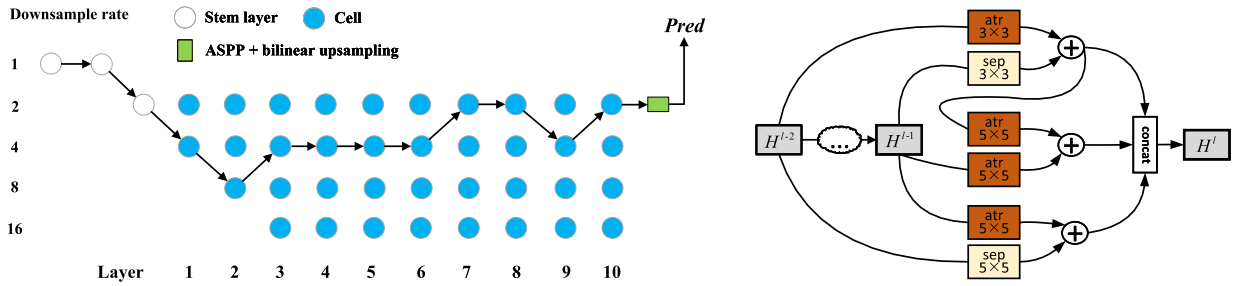


Fig. 17. Searched architecture of Pol-NAS on the Oberpfaffenhofen dataset. "atr" means atrous separable convolution. "sep" is the depthwise-separable convolution.

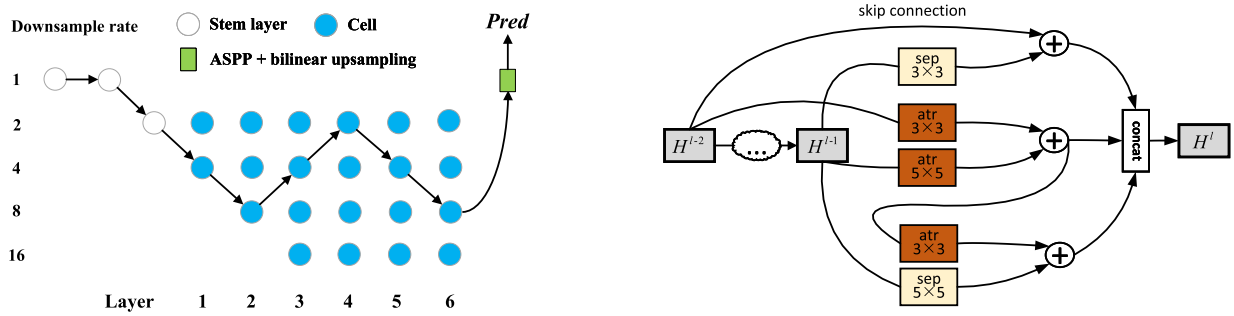


Fig. 18. Searched architecture of Pol-NAS on the Flevoland dataset. "atr" means atrous separable convolution. "sep" is the depthwise-separable convolution.

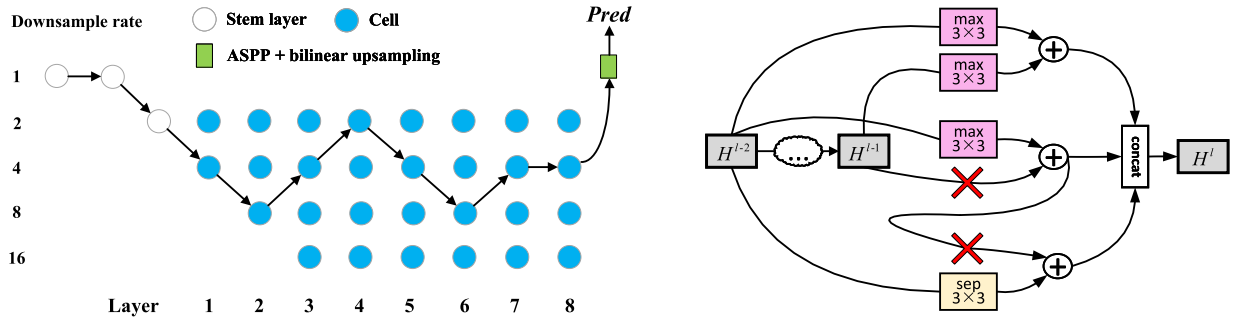


Fig. 19. Searched architecture of Pol-NAS on the San Francisco dataset. "max" denotes the max pooling. "sep" is the depthwise-separable convolution. "×" means zero connection.

Pol-NAS-3 have extra but acceptable costs of Mem. for Train and FLOPs.

5) Searched Architectures of Pol-NAS on Three Datasets: The searched architectures of *Pol-NAS* on three datasets are presented in Figs. 17–19. (We only provide the architectures of *Pol-NAS-3* here.) We find that the best layer-num for three datasets is 10, 6, and 8, respectively. It proves that the layer-num level search is necessary.

6) Generalization Ability of Searched Architectures of Pol-NAS: In order to test the generalization ability of searched architectures of *Pol-NAS*, we use the architectures searched on the Oberpfaffenhofen dataset by *Pol-NAS-1* and *Pol-NAS-3* and train the model using the $trainA$ set of Flevoland and San Francisco datasets, respectively. (*Pol-NAS-2* is only used for ablation studies. The only difference between *Pol-NAS-2* and *Pol-NAS-3* is the type of stem layers adopted in the training stage.) The results are presented in Table VIII. We can conclude that the searched architecture has a good generalization ability, because the mean values of OA and Kappa of them are still higher than those of handcrafted and autodesign models. Thus, if you do not have enough GPU resources, you can directly use the searched architecture of *Pol-NAS* on the Oberpfaffenhofen instead of searching to avoid using a lot of GPU resources.

TABLE VIII
RESULTS OF USING SEARCHED ARCHITECTURE ON THE OBERPFAFFENHOFEN DATASET

		Flevoland	San Francisco
Pol-NAS-1	OA	98.94 ± 0.09	99.42 ± 0.15
	Kappa	98.56 ± 0.12	99.17 ± 0.22
Pol-NAS-3	OA	99.04 ± 0.07	99.5 ± 0.07
	Kappa	98.7 ± 0.09	99.28 ± 0.1

7) Results of Different Methods on the Dataset With Various Degradation Variabilities:

TABLE IX
RESULTS OF DIFFERENT METHODS ON THE OBERPFAFFENHOFEN DATASET WITH MORE NOISE

	Handcrafted Models			Auto-design Models				
	FCN*	FCN	CV-FCN	PDAS	CV-PDAS	Pol-NAS-1	Pol-NAS-2	Pol-NAS-3
OA	98.41 ± 0.27	98.37 ± 0.23	97.44 ± 0.11	96.04 ± 0.20	96.20 ± 0.14	98.63 ± 0.41	99.59 ± 0.04	**99.61 ± 0.02**
Kappa	97.29 ± 0.46	97.23 ± 0.39	95.64 ± 0.19	93.25 ± 0.35	93.52 ± 0.24	97.66 ± 0.70	99.30 ± 0.06	**99.34 ± 0.03**

* means the input features of the corresponding model are not $\mathbf{T}$ matrix in the original paper. We change its features to $\mathbf{T}$ matrix.

1) When *Pol-NAS* meets data with the noise effect: In order to see what will happen if *Pol-NAS* meets data with the noise effect, we prepare an additional Oberpfaffenhofen dataset with more noise by not using refined Lee filter. Then, we select some models from handcrafted and autodesign ones with good performances in Table V and test them on this noisy dataset. The results of them are shown in Table IX. We can find that most methods suffer from the noise effect such as FCN*, FCN, PDAS, and CV-PDAS. The mean values of OA and Kappa of them on the noisy dataset are lower than those of them on dataset with less noise. *Pol-NAS-1* also suffers from the noise effect but is still better than other state-of-the-art methods. Surprisingly, *Pol-NAS-2* and *Pol-NAS-3* obtain higher OA and Kappa on the noisy dataset than corresponding models on the dataset with less noise. This may be because using more features as input can improve the robustness of a model.
2) When *Pol-NAS* meets data with other variabilities: As shown in Table II, the three datasets for testing the performances of *Pol-NAS* and other state-of-the-art methods are very different. First, these datasets are acquired with different platforms and different bands. E-SAR is an airborne platform, while Radarsat-2 is a spaceborne platform. Second, they are acquired in different years and have different resolutions. Therefore, the three datasets already contain many variabilities. The results in Tables V, VII, and VIII can prove that *Pol-NAS* is still effective when meeting many variabilities.

V. CONCLUSION

In this article, we proposed a NAS method with feature selection called *Pol-NAS* for PolSAR image classification, which could automatically find the optimal network architecture and the importance of each input feature. With the help of *Pol-NAS*, we only need to prepare the data and wait for the result, which can avoid manually designing network structures and searching the optimal input features by trial and error, because the manual designing process is time consuming and disgusting. Experiments on three real PolSAR datasets were conducted to prove the effectiveness and the good generalization ability of *Pol-NAS*. Ablation studies were performed to test the effectiveness of the proposed layer-num search and our redesigned stem layers (i.e., FA block), which make *Pol-NAS* have the ability of automatically determining the optimal number of layers and searching the importance of input features. Besides, *Pol-NAS* is also effective when meeting various degradation variabilities, which can prove the robustness of *Pol-NAS*.

For our future work, we hope to design a NAS method considering not only classification accuracy but also model size and inference latency for PolSAR image classification.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments of improving the quality of this article. The authors would also like to thank MindSpore for the partial support of this work, which is a new deep learning computing framework (https://www.mindspore.cn/).

REFERENCES

[1] F. Liu, L. Jiao, B. Hou, and S. Yang, "POL-SAR image classification based on wishart DBN and local spatial information," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3292–3308, Jun. 2016.
[2] F. Liu, L. Jiao, X. Tang, S. Yang, W. Ma, and B. Hou, "Local restricted convolutional neural network for change detection in polarimetric SAR images," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 818–833, Mar. 2019.
[3] F. Liu, L. Jiao, and X. Tang, "Task-oriented GAN for PoLSAR image classification and clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2707–2719, Sep. 2019.
[4] R. Touzi, "Target scattering decomposition of one-look and multi-look SAR data using a new coherent scattering model: The TSVM," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2004, vol. 4, pp. 2491–2494.
[5] L. Zhang, B. Zou, H. Cai, and Y. Zhang, "Multiple-component scattering model for polarimetric SAR image decomposition," *IEEE Geosci. Remote Sens. Lett.*, vol. 5, no. 4, pp. 603–607, Oct. 2008.
[6] J.-S. Lee, K. W. Hoppel, S. A. Mango, and A. R. Miller, "Intensity and phase statistics of multilook polarimetric and interferometric SAR imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, no. 5, pp. 1017–1028, Sep. 1994.
[7] A. Richardson, D. G. Goodenough, H. Chen, B. Moa, G. Hobart, and W. Myrvold, "Unsupervised nonparametric classification of polarimetric SAR data using the K-nearest neighbor graph," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2010, pp. 1867–1870.
[8] C. Lardeux et al., "Support vector machine for multifrequency SAR polarimetric data classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 12, pp. 4143–4152, Dec. 2009.
[9] L. Zhang, W. Ma, and D. Zhang, "Stacked sparse autoencoder in PoLSAR data classification using local spatial information," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 9, pp. 1359–1363, Sep. 2016.
[10] X. Sun et al., "FAIR1M: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery," *ISPRS J. Photogrammetry Remote Sens.*, vol. 184, pp. 116–130, 2022.
[11] Y. Zhou, H. Wang, F. Xu, and Y.-Q. Jin, "Polarimetric SAR image classification using deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1935–1939, Dec. 2016.
[12] Z. Zhang, H. Wang, F. Xu, and Y.-Q. Jin, "Complex-valued convolutional neural network and its application in polarimetric SAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 12, pp. 7177–7188, Dec. 2017.
[13] S.-W. Chen and C.-S. Tao, "PoLSAR image classification using polarimetric-feature-driven deep convolutional neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 4, pp. 627–631, Apr. 2018.
[14] R. Shang, J. He, J. Wang, K. Xu, L. Jiao, and R. Stolkin, "Dense connection and depthwise separable convolution based CNN for polarimetric SAR image classification," *Knowl.-Based Syst.*, vol. 194, 2020, Art. no. 105542.
[15] Y. Cui et al., "Polarimetric multipath convolutional neural network for PoLSAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–18, 2022, Art. no. 5207118.
[16] Q. Liu, L. Xiao, J. Yang, and Z. Wei, "Multilevel superpixel structured graph U-Nets for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2021, Art. no. 5516115.
[17] F. Liu, J. Wang, X. Tang, J. Liu, X. Zhang, and L. Xiao, "Adaptive graph convolutional network for PoLSAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5208114.
[18] Q. He, X. Sun, Z. Yan, and K. Fu, "DABNet: Deformable contextual and boundary-weighted network for cloud detection in remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2021, Art. no. 5601216.
[19] Y. Li, Y. Chen, G. Liu, and L. Jiao, "A novel deep fully convolutional network for PoLSAR image classification," *Remote Sens.*, vol. 10, no. 12, 2018, Art. no. 1984.

[20] Y. Cao, Y. Wu, P. Zhang, W. Liang, and M. Li, "Pixel-wise PoLSAR image classification via a novel complex-valued deep fully convolutional network," *Remote Sens.*, vol. 11, no. 22, 2019, Art. no. 2653.

[21] Q. He, X. Sun, Z. Yan, B. Li, and K. Fu, "Multi-object tracking in satellite videos with graph-based multitask modeling," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5619513.

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

[23] G. Liu, Y. Li, L. Jiao, Y. Chen, and R. Shang, "Multiobjective evolutionary algorithm assisted stacked autoencoder for PoLSAR image classification," *Swarm Evol. Comput.*, vol. 60, 2021, Art. no. 100794.

[24] H. Dong, B. Zou, L. Zhang, and S. Zhang, "Automatic design of CNNs via differentiable neural architecture search for PoLSAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 9, pp. 6362–6375, Sep. 2020.

[25] C. Hou, F. Nie, H. Tao, and D. Yi, "Multi-view unsupervised feature selection with adaptive similarity and view weight," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 9, pp. 1998–2011, Sep. 2017.

[26] X. Huang and X. Nie, "Multi-view feature selection for PoLSAR image classification via $l_{2,1}$ sparsity regularization and manifold regularization," *IEEE Trans. Image Process.*, vol. 30, pp. 8607–8618, 2021.

[27] A. Haddadi, G. M. Reza Sahebi, and A. Mansourian, "Polarimetric SAR feature selection using a genetic algorithm," *Can. J. Remote Sens.*, vol. 37, no. 1, pp. 27–36, 2011.

[28] Y. Bai, D. Peng, X. Yang, L. Chen, and W. Yang, "Supervised feature selection for polarimetric SAR classification," in *Proc. 12th Int. Conf. Signal Process.*, 2014, pp. 1006–1010.

[29] C. Yang, B. Hou, B. Ren, Y. Hu, and L. Jiao, "CNN-based polarimetric decomposition feature selection for PoLSAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8796–8812, Nov. 2019.

[30] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[31] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–16.

[32] E. Real et al., "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2902–2911.

[33] 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, pp. 4780–4789.

[34] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, doi: 10.1109/TNNLS.2021.3100554.

[35] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[36] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–13.

[37] H. Liang et al., "DARTS+: Improved differentiable architecture search with early stopping," 2019, *arXiv:1909.06035*.

[38] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive DARTS: Bridging the optimization gap for NAS in the wild," *Int. J. Comput. Vis.*, vol. 129, no. 3, pp. 638–655, 2021.

[39] C. Liu et al., "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 82–92.

[40] X. Zhang et al., "DCNAS: Densely connected neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13956–13967.

[41] Y. Bengio, *Learning Deep Architectures for AI*. Delft, The Netherlands: Now Publishers, 2009.

[42] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[43] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[44] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[46] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.

[47] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

Guangyuan Liu received the B.S. degree in intelligent science and technology from Xidian University, Xi'an, China, in 2016.

Since then, he has been taking successive postgraduate and doctoral programs with the Key Laboratory of Intelligent Perception and Image Understanding of the Ministry of Education of China, International Research Center of Intelligent Perception and Computation, School of Artificial Intelligence, Xidian University. His current research interests include deep learning, evolutionary computation, and polarimetric synthetic aperture radar image classification.

Yangyang Li (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science and technology in 2001 and 2004, respectively, and the Ph.D. degree in pattern recognition and intelligent system in 2007, all from Xidian University, Xi'an, China.

She is currently a Professor with the School of Artificial Intelligence, Xidian University. Her research interests include quantum-inspired evolutionary computation, artificial immune systems, and deep learning.

Yanqiao Chen was born in 1991. He received the Ph.D. degree in pattern recognition and intelligent system from Xidian University, Xi'an, China, in 2019.

He is currently an Engineer with the 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, China. His research interests include computer vision and telemeter and telecontrol of unmanned aerial vehicles.

Ronghua Shang (Member, IEEE) received the B.S. degree in information and computation science and the Ph.D. degree in pattern recognition and intelligent systems from Xidian University, Xi'an, China, in 2003 and 2008, respectively.

She is currently a Professor with Xidian University. Her research interests include machine learning, pattern recognition evolutionary computation, image processing, and data mining.

Licheng Jiao (Fellow, IEEE) received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, in 1982, and the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively, all in electronic engineering.

From 1990 to 1991, he was a Postdoctoral Fellow with the National Key Laboratory for Radar Signal Processing, Xidian University, Xi'an. Since 1992, he has been a Professor with the School of Electronic Engineering, Xidian University. He is currently the Director of the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University. He is in charge of about 40 important scientific research projects and authored or coauthored more than 20 monographs and 100 papers in international journals and conferences. His research interests include image processing, natural computation, machine learning, and intelligent information processing.

Dr. Jiao is a Member of the IEEE Xi'an Section Execution Committee and the Chairman of Awards and Recognition Committee, the Vice Board Chairperson of the Chinese Association of Artificial Intelligence, the Councilor of the Chinese Institute of Electronics, the Committee Member of the Chinese Committee of Neural Networks, and an expert of Academic Degrees Committee of the State Council.