Lightweight Convolutional Neural Networks for Surface Defect Segmentation based on Neural Architecture Search

Biao Chen[1], Tongzhi Niu[1*], Yuchen Lin[1], Hang Zhang[1], Baohui Liu[1] and Miao Wang[1]

[1]School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Street, Wuhan, 430074, Hubei, China.

*Corresponding author(s). E-mail(s): tzniu@hust.edu.cn;
Contributing authors: u202010899@hust.edu.cn;
yuchenlin@hust.edu.cn; u202010924@hust.edu.cn;
lbhui@hust.edu.cn; m202170809@hust.edu.cn ;

Abstract

Convolutional Neural Networks have achieved impressive performance in many surface defect inspection tasks over the past years. However, state-of-art networks become increasingly heavy and expensive, which limits their deployments in edge or embedded devices in industrial scene. In this paper, we proposed a new solution that automatically designs lightweight convolutional neural networks for surface defect inspection via neural architecture search (Light SDI-NAS). At first, a search space suitable for industrial applications is proposed by combining experience in neural network architecture design and preliminary experimental results. Secondly, we design a new loss to balance the model accuracy and computational efficiency. Finally, the lightweight network obtained through Light SDI-NAS performs well on three industrial datasets, showing comparable or even better results than state-of-art handcraft networks on these datasets, with the number of parameters greatly being reduced and 1.8 times faster.

Keywords: lightweight neural networks; real-time neural network; surface defect segmentation; neural architecture search.

1 Introduction

In the field of surface defect inspection, deep convolutional neural networks (DCNNs) have been one of the research hotspots because of its powerful feature extraction ability and robustness [1] [2] [3] [4] [5]. In general, the impressive performance of DCNNs depends on the increase of model parameters and the improvement of computing hardware [6]. However, due to low latency requirements, the DCNNs is usually deployed at the edge or embedded devices in industrial scene [7]. Therefore, realizing high-precision and real-time detection on limited computing platforms has become one of the most valuable topics in surface defect detection.

The existing methods of lightweight DCNNs can be mainly divided into the following categories: 1) model compression. After the networks structure deigning or training are completed, the model parameters are compressed by pruning [8] [9] [10], quantization [11] and knowledge distillation [12] [13]. But model compression is wasteful of computing resources during training. And it is difficult to balance network parameters and computational efficiency with generalization and accuracy. Besides, some model compression methods need the support of special devices to improve inference speed because of the limitations from sparse matrix. 2) lightweight neural architecture design. Recently, many lightweight network structures have been designed, such as MobileNet series [14] [15] [16], ShuffleNet [17], SENet [18], ICNet [19] etc. Nevertheless, most of these lightweight architectures are designed for natural images rather than industrial images. These classical network architectures are usually designed by relevant domain experts for several months or even several years, which is laborious and time-consuming .

Therefore, we proposed a new solution that automatically designs lightweight convolutional neural networks for surface defect inspection via neural architecture search (SDD-NAS). It is non-trivial because the following two aspect. 1) There are some challenges in surface defect detection. Firstly, the number of industrial defect images is very small. Secondly, the difference between the defect area and the normal area is unobvious. Finally, defects vary in size and have irregular outlines. 2) SDD-NAS must not only achieve high-precision, but also realize real-time detection on limited computing platforms. To address above challenges, we focus on search space, search strategy, and performance estimation strategy.

For search space, the current search space is mainly constructed by a number of manually selected convolution operations. However, such a construction method may not be able to select the most suitable convolution operations to complete the corresponding visual tasks. According to the existing experience, we initially selected the candidate convolution operation blocks to form the search space, and then considered the lightweight and other factors for further selection. Finally, in order to ensure that the candidate convolution operation blocks can overcome the above challenges of surface defect detection on the target task, we designed the exploration experiment in Section 3.2.2 and combined the experimental results to determine the final search space.

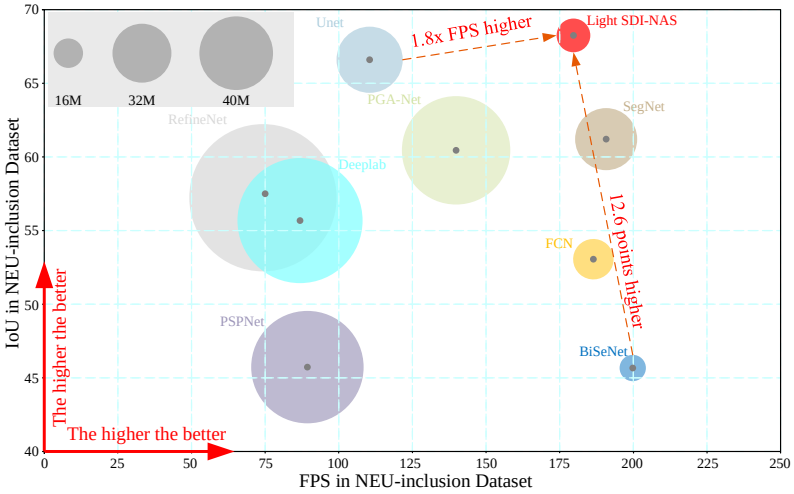

Fig. 1 Comparisons with state-of-the-art networks in terms of IoU performance, model inference FPS and parameters on NEU-inclusion dataset.

For search strategy, the current network structure search methods are mainly based on reinforcement learning, neuro-evolutionary algorithm, Bayesian optimization algorithm and gradient. Our goal is to quickly search for a lightweight network architecture suitable for the task of surface defect segmentation on the corresponding dataset, so we choose gradient-based search method, which is efficient and easy to implement.

For performance estimation strategy, the existing model performance evaluation strategy is mainly realized by testing the performance of the model on datasets. We choose the gradient-based search method, which can directly guide the search network through the loss function to get the desired neural network model architecture. At present, the common loss function mainly aims at performance improvement. However, in order to search for the neural network model with the best balance between performance and reasoning speed, we modify the loss function to consider the lightweight of network structure while optimizing the model performance (See section 3.3 for details).

In this paper, we proposed a lightweight networks design method, called SDD-NAS, which can search corresponding lightweight networks for different datasets. Our main contribution is as fallows.

Firstly, combing the experience of neural network architecture design and preliminary experimental results, we build a suitable search space for surface defect detection.

Secondly, a loss function that comprehensively considers model accuracy and lightweight is proposed.

Finally, the lightweight network obtained through operation search performs well on several industrial datasets, showing comparable or even better results than state-of-art handcraft networks on these datasets. Meanwhile,

the network obtained in the end has significantly fewer parameters, and the experimental results show that its computation delay is also lower.

2 Related Work

2.1 Lightweight Neural Network Design

While big models can achieve impressive performance, they often come with a significant amount of computing overhead and memory usage. As a result, an increasing amount of research is being focused on designing lightweight network architectures that can be used in real-world scenarios. The first work considering the efficiency of image segmentation was ENet [20], which designed a feature extraction block with few channels and used the point convolution in the residual connection layer to reduce the computational consumption. ICNet [19] was designed for real-time segmentation of high-resolution images, adopting multi-scale image input and proposing the strategy of cascaded feature fusion unit and cascaded tag guidance to introduce medium and high-resolution feature maps and gradually improve accuracy. Another notable lightweight model for semantic segmentation is BiSeNet [21], which uses a double branch architecture, proposes the ARM attention mechanism, and builds an FFM feature fusion module. These methods have achieved competitive performance on some datasets under the premise of greatly reduced computation. However, due to the unique characteristics of industrial datasets, many of these lightweight networks may be untrainable or ineffective on these types of datasets.

In conclusion, the research in the field of industrial image segmentation has yet to strike a good balance between model performance and computation. Further exploration and development of lightweight network architectures that are effective on industrial datasets could lead to significant advancements in this field.

2.2 Neural Architecture Search

Designing a high-performing neural network can be a time-consuming and laborious process for researchers. Neural Architecture Search (NAS) is a promising technique for automating the design of neural networks. It involves defining the search space, search strategy, and performance estimation strategy to efficiently explore a vast space of potential network architectures. The search space defines which architectures can be represented in principle, while the search strategy details how to explore the search space, along with that the Performance Estimation refers to the process of estimating this performance [22]. Early NAS approaches relied on reinforcement learning algorithms [23] or evolutionary algorithms, which required extensive computational resources to train thousands of candidate networks from scratch. However, recent approaches such as Differentiable Architecture Search (DARTs) [24] and ProxyLessNAS [25] have improved the efficiency of the search process by relaxing

the discrete search space into a continuous spatial structure and introducing structural parameters to learn the redundancy of each path. Other works have utilized weight sharing strategies to share the search cost by training a single over-parameterized hypernetwork and sharing the weight across subnets. NAS has primarily been used for image classification tasks but has since been extended to downstream tasks such as semantic segmentation. For instance, Chen et al. [26] applied NAS to search for a small-sized ASPP called DPC and fixed the pretrained backbone as an encoder for semantic segmentation. Nekrasov et al. [27] improved the speed of searching decoders for reinforcement learning strategies by using knowledge distillation and Polyak Averaging method, enabling the discovery of corresponding image segmentation networks.

2.3 Defect Segmentation

At present, most defect segmentation networks are based on Fully Convolutional Network (FCN) architecture [28]. For example, Dung et al. [29] used an FCN network based on VGG16 encoder to segment cracks on concrete surface. Li et al. [30] proposed a surface defect segmentation method for concrete structures using a Unet network with Dense Block modules in the encoder and pixel-by-pixel summation instead of concatenation for the jump layer connection. Dong et al. [31] proposed a pyramid feature fusion and global context attention network (PGA-Net) for pixel-wise detection of surface defect. Although these methods achieved good performance on different defect segmentation datasets, they do not consider network lightweight, resulting in architectures that are too large to be deployed on mobile devices for practical applications.

3 Methods

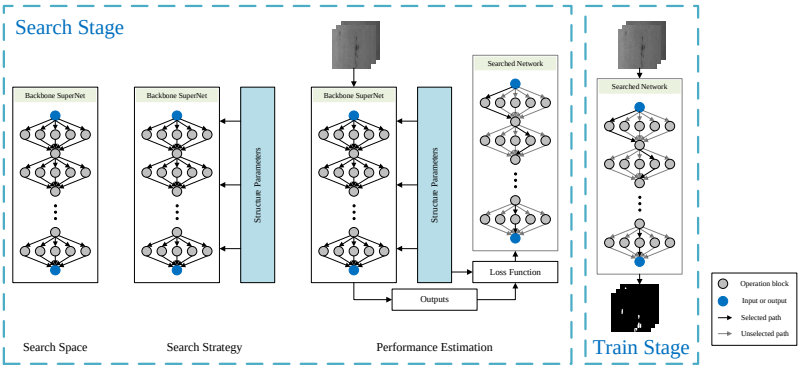

Fig. 2 Search pipeline of the proposed Lightweight Neural Architecture Search. There are three parts in search stage: search space, search strategy, and performance estimation. After search stage, the searched network is going to be trained.

3.1 Overview

As shown in Figure 2, our method is divided into two stages. The first stage is the search stage, and the second stage is the training stage. In the search stage, we first build a suitable search space according to the characteristics of surface defect detection data set, network design experience and exploration experiment (see Section 3.2.3 for details). Then, we introduce network structure parameters to characterize the importance of every convolutional operation block in the search space, and use a gradient-based search method to search the network architecture. Finally, we evaluate the performance of the model according to the output of the model and the structural parameters as the input of the loss function, and then search to get the final network architecture. In the training stage, we train and test the network model obtained from the search stage on the corresponding dataset.

3.2 Search Space

In this section, we will first outline the selection principle for the operation block used in constructing the search space. We will then describe our finalized method for selecting the operation block. Finally, we will present the various convolution operations that we have chosen to include in the search space.

3.2.1 Operation Block Selection Principle

The process of building the search space involves two main steps. First, we determine the principle for selecting candidate convolution operation blocks. This involves drawing on existing experience to establish a selection criterion and identify suitable candidate convolution operations. In the second step, we construct the search space by selecting the most appropriate convolution operations from the candidate set. This is done by leveraging our network design expertise, as well as conducting experimental verification to identify the most suitable convolution operations for inclusion in the search space. Based on our designing aims of searching for performance and lightweight optimal balancing network, we select candidate action blocks based on the following principles. The first criterion we use for selecting convolution operations is computational complexity. We prioritize convolution operations with low computational complexity, as there are a variety of lightweight convolution operations currently available that are highly parameter-efficient while still delivering comparable performance to ordinary convolution. Examples of such lightweight convolution operations include depth-wise convolution and depth-wise dilation convolution. By incorporating these existing lightweight convolution operations into the search space, we can explore the search network to build lightweight segmentation networks that are well-suited for industrial images.

The second criterion we use for selecting convolution operations is the ability to handle multiple receptive fields. Receptive fields are essential for feature extraction in the network, as different receptive fields enable the network to extract different features. Given that defects in industrial images vary in size

and have irregular contours, we include convolution operations with different receptive fields in the search space. Through the search network, we finalize the convolution operations with different receptive fields at corresponding positions, ensuring that the resulting network has better feature extraction and segmentation abilities for features of multiple scales and is more sensitive to boundary information. This enables the network to more effectively identify and classify defects in industrial images, so it can better deal with the challenge in surface defect detection mentioned in the introduction.

3.2.2 Introduction of Candidate Convolution Blocks and Test Experiment

Based on the aforementioned selection principles, we have identified seven candidate convolution operation blocks for inclusion in the search space. These are: Single Conv, Double Conv, Single Dilation Conv, Double Dilation Conv, Double DWconv [14], Double DW-D-Conv [32], and Cutconnect Block. Single Conv and Double Conv represent a single normal convolution operation block and a double normal convolution operation block, respectively. Single Dilation Conv and Double Dilation Conv correspond to single and double dilation convolution operation blocks, respectively. Additionally, we have included Double DWconv, a double depth-wise convolution operation block, and Double DW-D-Conv, a double depth-wise dilation convolution operation block. Finally, we have designed a new convolution operation block called Cutconnect Block. The structure of each convolution operation block is illustrated in Figure 3.

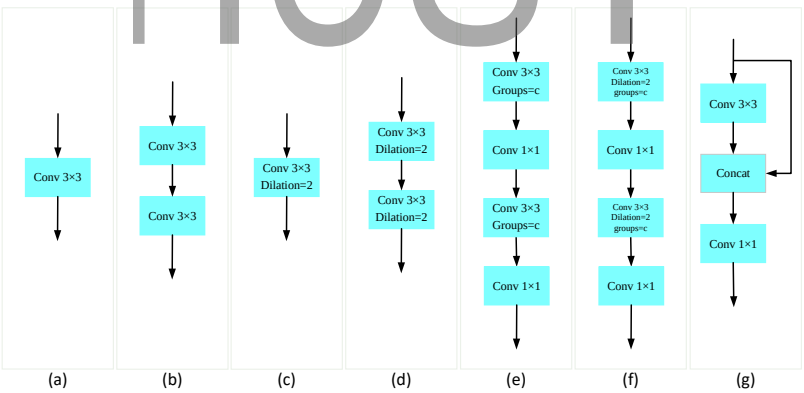

Fig. 3 Structure of convolution operation blocks ((a), (b), (c), (d), (e), (f) and (g) stand for Single conv, Double conv, Single Dilation conv, Double Dilation conv, Double DWconv, Double DW-D-Conv and Connect Block respectively.)

Once the candidate operation blocks are selected, we standardize the input and output dimensions of each convolution operation block to 3x96x96 and

64x96x96, respectively. Then the ptflops [33] library in Python is used to calculate the number of parameters and FLOPs for each operation block. The results of these experiments are presented in Table 1.

Table 1 Test experimental results of candidate operation blocks. The column called IOU is the result of the exploration experiment on the NEU-inclusion dataset of every convolution operation block.

Operation Blocks	Name	Parameters	FLOPs	IOU
1	Single Conv	0.01MB	18.28M	65.98
2	Double Conv	0.15MB	359.79M	66.60
3	Single Dilation Conv	0.01MB	18.28M	64.07
4	Double Dilation Conv	0.15MB	359.79M	65.20
5	Double DWconv	0.01MB	11.54M	-
6	Double DW-D-Conv	0.02MB	49.82M	-
7	Cutconnect Block	0.02MB	59.57M	66.53

To evaluate the performance of the dilation convolution operation block on industrial image datasets, we conduct experiments on the NEU-inclusion industrial image segmentation dataset. Specifically, we replace the original double convolution operation blocks in UNet with the Single Dilation Conv block and Double Dilation Conv block, respectively. The results of these experiments are also included in Table 1.

3.2.3 Finalize Search Space

As shown in Table 1, the Double Dilation Conv block do not demonstrate significantly better performance than the Single Dilation Conv block, but it has much higher parameters count. Therefore, we choose the Single Dilation Conv block as a part of the search space and abandon the Double Dilation Conv block. The double ordinary convolution operation is also excluded due to its high parameter count and computational complexity. To increase the diversity of search space, considering that Single Conv block does not bring high parameter number and computational complexity, we retain this block as a part of the search space. At the same time, in order to limit the number of network parameters and computational complexity and obtain a large sensitivity field, two kinds of convolution operation blocks, called Double DWconv block [14] and Double DW-D-Conv block [32], are retained according to the experimental results of parameter number and FLOPs.

Finally, the Cutconnect Block operation block, which we designed to better retain the input feature information and improve the adaptability of the search network to different datasets, is included in our search space. Therefore,

the final search space consists of Single Conv, Single Dilation Conv, Double DWconv, Double DW-D-Conv, and Cutconnect Block (Figure 4).

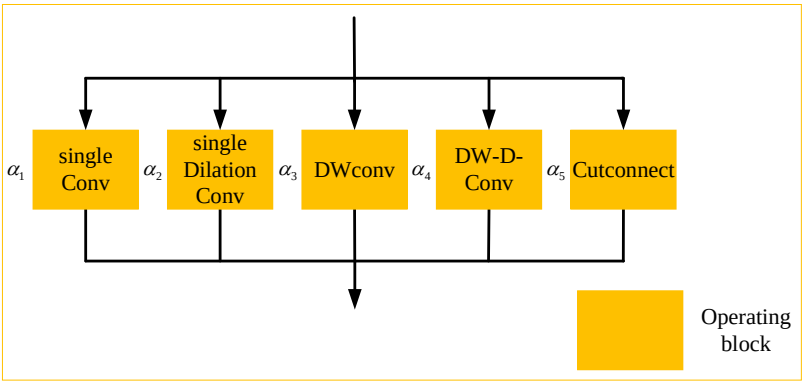

Fig. 4 Structure of search block. The final output is determined by all operation blocks (see in Search Strategy).

Our task is to search for convolution operation blocks that are suitable to replace the double convolution operation blocks on the original encoder and decoder of UNet [34]. We focus on operations with low computational cost and did not search for replacements for upsampling, maximum pooling, jump join, and the point convolution operation before final output. Finally, the search network structure has been designed and shown in Figure 5.

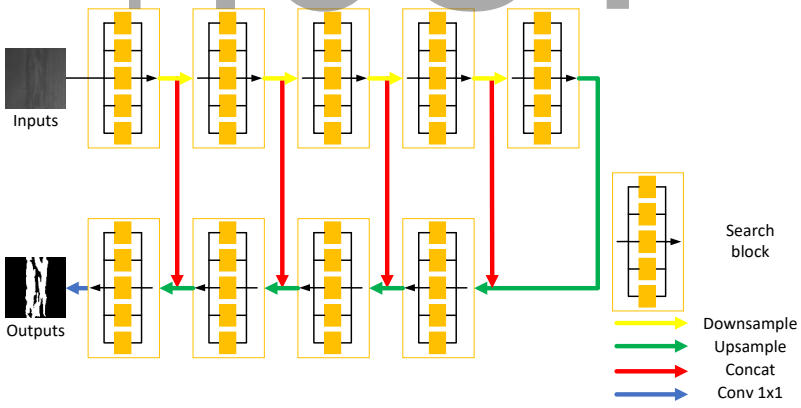

Fig. 5 Structure of the whole search network.

3.3 Search Strategy

In order to search for the most appropriate network structure, structural parameters are introduced to represent the importance of different operation

blocks at every position (as shown in Figure 4). At each position of the encoder and decoder, every operation block is multiplied by a coefficient α, the value of which represents the importance of the corresponding operation block. The coefficient α form a matrix A of size 9 by 5 which is the structural parameter matrix. Therefore, the output of each operation block at the same position is multiplied by its corresponding α and summed up to obtain the output of the search block.

During training, we utilize different learning rates for the structural parameters and network parameters, and separate the training processes of the two through gradient descent. A detailed description of this approach can be found in Algorithm 1.

Algorithm 1: Hierarchical Gradient Descent Architecture Search

Create architectural parameters $A=\{\alpha_{(i,j)}\}$ and search network weights W

for i in range(epoch):

1. Update search network weights W by SGD
2. **if** (i+1) % 10 == 0:

Update architectural parameters A by SGD

end if

end

Derive the final architecture based on optimized architectural parameters A

3.4 Performance Estimation Strategy

In addition to network performance, the other goal is to control the number of parameters and computational complexity of the searched network. To achieve this, we modify the loss function according to the parameter number and computational complexity of every operation block in the search space (as shown in Table 1) . As the parameter number and computational complexity are positively correlated, we only considered the parameter number when optimizing the loss function. The modified calculation formula of the loss function is as follows:

$$L= (1-\beta)L_{weight} + \beta L_{arch} \tag{1}$$

where β is the lightweight coefficient. As can be seen from formula (1), by adjusting the size of the lightweight coefficient β, we can control the search network to search for different degrees of lightweight network. The greater the β is, the greater the influence of the number of network parameters will be when searching, and we finally selected β=0.3. The specific formula of L_{weight} is as follows:

$$L_{weight}(p_d, g_d)=L_{dice}(p_d, g_d) + 0.5 \times L_{bce}(p_d, g_d) \tag{2}$$

where $p_d \in {}^{H\times W}$ denotes the predicted pixel and the $g_d \in {}^{H\times W}$ denotes the corresponding pixel of ground-truth. Besides, L_{bce} denotes the binary cross-entropy loss while L_{dice} denotes the dice loss, which is given as follows:

$$L_{dice}(p_d, g_d) = 1 - \frac{2\sum_i^{H\times W} p_d^i g_d^i + \varepsilon}{\sum_i^{H\times W} \left(p_d^i\right)^2 + \sum_i^{H\times W} \left(g_d^i\right)^2 + \varepsilon} \tag{3}$$

The specific formula of L_{arch} is as follows:

$$L_{arch} = \sum_{i=1}^{9} \frac{P_i}{\sum_{j=1}^{5} P_{ij}} \tag{4}$$

where, the P_i represents the parameter number of the operation block corresponding to the maximum structural parameter $\alpha_{i\,\max}$ at the ith search block, and P_{ij} represents the parameter number of the jth operation block at the ith operation block.

By training the search network on the industrial image segmentation dataset, we obtain the final structural parameter matrix. According to the value of the corresponding structural parameters of different operation blocks at every position, the operation block with the largest structural parameters at every position is selected as the final searched operation block at this position. Finally, the network structure suitable for the corresponding industrial image segmentation data set is constructed.

4 Experiment and Results

In this part, we first introduce the industrial image segmentation data set, the implementation details, and evaluation indicators. After that, we demonstrate the performance of our method and other semantic segmentation networks on different data sets, and carry out in-depth analysis.

4.1 Datasets

In this article, three surface defect datasets are selected to prove and evaluate the applicability and generality of our method, including NEU-DET defect dataset, DAGM defect dataset and Wafer Defect dataset.

NEU-Seg Dataset: the NEU data set is a standard data set collected by [35] to solve the problem of automatic recognition for hot-rolled steel strips. There are six types of strip steel plates in the data set, including patch, crazing, pitted-surface, inclusion, scratches and rolled-in scale, and every surface defect contains 300 images. The original resolution of images in the data set is 200x200 and all have corresponding defect type labels. We chose three surface defects (inclusion, patches and scratches) for pixel level marking. Then we changed their resolution to 96x96 and then divided them into training sets and test sets, which contain 250 and 50 images respectively to allow them to be applied to our industrial defect image segmentation.

DAGM Dataset: The DAGM dataset [36] is manually generated and contains multiple types of industrial surface defect images with an original resolution of 512x512. We chose four of these categories, first converting their resolutions to 256x256, then dividing them into training sets and test sets. For the data sets of tile, cement and fabric, the training set and test set contain 125 and 25 images respectively, while the training set and test set of wallpaper category contain 250 and 50 images respectively.

Wafer Defect Dataset: Light4new data set is a surface defect image of industrial wafer. The defect area in the image is small. The image resolution of the data set is 256x256, and contains 645 images in total. We divided it into a training set and a test set, which contains 545 and 100 images respectively.

4.2 Implementation Details and Evaluation Indicators

Search. Firstly, we search for a lightweight segmentation network suitable for each data set. The search process used a batch size of 4 and employed the stochastic gradient descent (SGD) algorithm with a learning rate of 0.0003 for network parameters. The learning rate for structural parameters is set at 0.1, and the structural parameters are updated every ten iterations. We conducted 400,400, and 600 iterations on the NEU-Seg, DAGM 2007, and Light4new datasets, respectively. Since the convolution operations in our searching network consisted of lightweight convolution operation blocks, the search process does not take too much time. At the end of the search, we obtained lightweight image segmentation networks suitable for each dataset and then proceeded to network training and testing.

Training. To ensure fairness, all of our models are trained from scratch using the stochastic gradient descent (SGD) algorithm with a learning rate 0.0003 and momentum of 0.9. A batch size of 16 and weight decay of 0.0001 are adopted for all datasets. The models are trained for 600, 600, 800 iterations for the NEU-Seg, DAGM 2007 and Wafer Defect dataset, respectively. However, due to the small amount of data, lightweight models such as BiSeNet struggled to converge, so we conducted 2000 iteration training for each dataset. Unfortunately, the lightweight models ESNet and ERFNet failed to converge during training on the above datasets, so we only used BiSeNet for comparison. To expand the training set and prevent overfitting, we randomly rotated images by 90° and reversed them during training for data augmentation. Evaluation. When evaluating network performance, we adopted a simple and efficient method by directly loading the test data to test the performance of each network after training. However, when evaluating lightweight networks, we also considered factors such as parameter number and inference time in addition to network performance. Therefore, we tested the parameter number and inference time of different networks on each dataset. Evaluation indicators. In order to comprehensively evaluate the performance of various networks on different datasets, we use three evaluation indexes, namely IoU, network parameters and inference FPS (Frames Per Second). The IoU is calculated using the formula

given as follows:

$$IoU = \frac{TP}{FP + TP + FN} \tag{5}$$

where, TP denotes true positive. FP denotes false positive. FN denotes false negative . The ptflops[] module in python is utilized to calculate the number of parameters for different networks. In addition, we measure the inference time using a single GPU and repeat the process 2500 times to minimize error fluctuation. After that, the result are then converted into inference FPS. While loading data may have introduced some errors in the inference time test, these errors are consistent across all networks and are therefore unlikely to have a significant impact on our evaluation of network speed.

Setup. Our experiments are conducted using PyTorch 1.4.0 and we obtain the inference time by running on a single Tesla P100 with CUDA10.2, which is used for comparison with other methods.

4.3 Results

We ended up choosing six classic segmentation networks (FCN [28], RefineNet [37], PSPNet [38], Deeplab [39], UNet [34]), a network designed for industrial image segmentation (PGA-Net [31]), and a lightweight and real time network (BiSeNet [21]) that performs well in natural images as the baseline network to compare with our network.

4.3.1 NEU-Seg Dataset

We perform a search on all three NEU-Seg datasets [35] and obtain the lightweight segmentation network structure that is suitable for each dataset (the network structure shows in the appendix). Based on the above experiments, we have obtained the results as shown in the Table 2.

Table 2 Performance of big networks and our method on the NEU-Seg Dataset.

	NEU-inclusion			NEU-patches			NEU-scratches		
CNN	IoU	params	fps	IoU	params	fps	IoU	params	fps
FCN	53.06	19.18M	181.43	80.74	19.18M	181.55	60.32	19.18M	180.38
RefineNet	57.23	80.22M	74.25	80.85	80.22M	76.76	66.36	80.22M	73.19
PSPNet	45.73	53.32M	89.34	76.54	53.32M	89.93	58.18	53.32M	89.67
Deeplab	55.68	59.34M	86.84	80.55	59.34M	90.78	56.03	59.34M	86.51
PGA-Net	60.45	51.41M	139.82	80.70	51.41M	164.91	61.96	51.41M	165.02
UNet	66.60	31.39M	110.47	81.51	31.39M	109.31	75.00	31.39M	105.44
Ours	**68.25**	**16.75M**	**186.60**	**82.20**	**18.67M**	**184.69**	**76.26**	**15.78M**	**195.77**

As shown in Table 2, our method outperform all other methods in three datasets. Compared to the classic segmentation networks and the PGA-Net

[31] designed for industrial image segmentation, our method achieves higher performance while significantly reducing the number of parameters and inference time. In terms of speed, our method also ranked at the top of the table.

To further demonstrate the effectiveness of our method, we replaced the double convolutional operation blocks in UNet [34] with feature extraction modules in our search space. This resulted in the construction of three networks - Single-U, Cut-U and KD-U (DW-U and DW-D-U are not shown due to poor performance). In addition, we also compare our method with BiSeNet [21] and all the results in Table 3. Compared to the lightweight network BiSeNet [21], our methods did not increase the number of parameters or decrease the speed by much, but our network performance was much better (up to 83 % higher) and the generalization is significantly better.

Table 3 Performance of small networks and our method on the NEU-Seg Dataset.

	NEU-inclusion			NEU-patches			NEU-scratches		
CNN	IoU	params	fps	IoU	params	fps	IoU	params	fps
Single-U	65.98	15.68M	198.85	80.38	15.68M	**198.16**	75.77	15.68M	194.42
Cut-U	66.53	19.17M	163.72	**82.68**	19.17M	174.09	75.89	19.17M	174.95
KD-U	64.07	15.68M	167.67	80.63	15.68M	165.34	75.08	15.68M	164.37
BiSeNet	45.67	**12.40M**	**199.84**	75.04	**12.40M**	197.94	41.62	**12.40M**	**197.16**
Ours	**68.25**	16.75M	186.60	82.20	18.67M	184.69	**76.26**	15.78M	195.77

As shown in Table 3, the model obtained through our search method performs well on the dataset. It ensured a light model structure and fast reasoning speed while achieving superior performance.

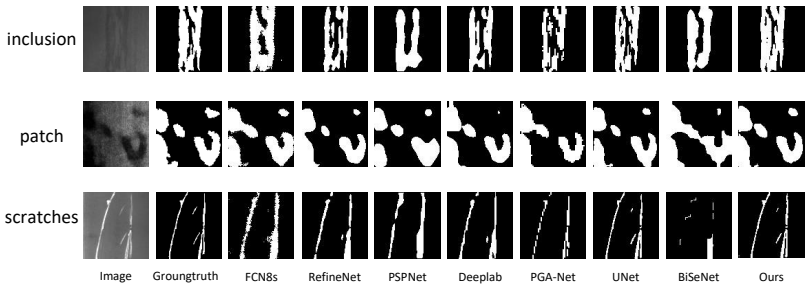

Fig. 6 The visual display of the results of each network on the NEU-Seg Dataset.

Figure 4 shows the visual display of the results of each network on the NEU-Seg Dataset. As shown in figure 4, our model achieves higher precision in the overall defect profile segmentation and is more sensitive to defect details. This

is mainly due to our well-designed search space, which includes convolution operations with different field sizes, and the search network identifies the most suitable convolution operation for each location of the network. This enables the final network to better extract both long-distance and short-distance features.

4.3.2 DAGM Dataset

Table 4 Performance of each network on DAGM Dataset.

	DAGM-tile			DAGM-cement			DAGM-fabric			DAGM-wallpaper		
CNN	IoU	params	fps	IoU	params	fps	IoU	params	fps	IoU	params	fps
FCN	47.06	19.18M	51.77	70.26	19.18M	51.81	77.91	19.18M	49.77	77.08	19.18M	73.66
RefineNet	63.58	80.22M	19.64	63.09	80.22M	19.94	25.58	80.22M	19.91	50.78	80.22M	21.31
PSPNet	64.25	53.32M	31.39	72.54	53.32M	30.09	**79.01**	53.32M	31.13	76.74	53.32M	35.16
Deeplab	59.62	59.34M	47.93	69.06	59.34M	48.37	76.98	59.34M	48.32	56.49	59.34M	57.80
UNet	67.33	31.39M	33.55	**74.47**	31.39M	36.03	78.87	31.39M	34.58	**79.09**	31.39M	49.59
BiSeNet	58.68	**12.4M**	**62.41**	70.95	12.4M	**69.27**	53.47	**12.4M**	**63.21**	41.25	12.4M	**91.26**
Ours	**67.35**	17.63M	49.22	74.31	**11.53M**	49.43	78.55	15.17M	49.31	78.89	**11.41M**	64.38

Similar to the previous experiment, we conducted a search on the four datasets of DAGM [36] to obtain the lightweight segmentation network structure suitable for each datasets. The network structure for each dataset are provided in the appendix. We trained various networks on different datasets and obtained the results presented in Table 4. Based on the data presented in Table 4, our model has achieved comparable performance to the strongest network, UNet [34]. However, our model has significantly reduced the number of parameters (up to 63.7% lower) and improved running speed (up to 47% faster) compared to UNet [34]. Our model also outperforms other classical segmentation models in terms of both performance and parameter quantity, with little reduction in speed.

While BiSeNet [21] has a great advantage in running speed, its generalization performance, convergence speed, performance, and parameter numbers on the datasets are significantly different from our model. Overall, our model achieves the best precision-speed balance among all the models in the table.

4.3.3 Wafer Defect Dataset

Similar to the previous experiment, we conducted a search on the Wafer Defect dataset to obtain lightweight segmentation network structures suitable for this dataset. The network structures are provided in the appendix. We trained various networks on the Wafer Defect dataset and obtained the results presented in Table 5. Compared to classical split networks, our model achieves excellent

Table 5 Performance of each network on Wafer Defect Dataset.

	Wafer Defect Dataset		
CNN	IoU	params	fps
FCN	44.56	19.18M	97.05
RefineNet	47.68	80.22M	22.71
PSPNet	**48.61**	53.32M	38.91
Deeplab	38.89	59.34	71.96
UNet	46.42	31.39M	53.69
BiSeNet	21.29	12.4M	**102.71**
Ours	46.63	**11.23M**	77.18

performance with significantly fewer parameters. It has less than 2% lower Intersection over Union (IoU) compared to the best-performing PSPNet [38], while reducing the number of parameters by more than 40% compared to all classical models (up to 86% lower, compared to RefineNet [37]).

Compared to BiSeNet [21], our model shows a 37% reduction in speed, but a 9.4% reduction in parameter count, and a 119% improvement in performance. Overall, our method achieves the best precision-speed balance on the Wafer Defect dataset.

5 Conclusion

We believe that industrial image segmentation requires both low-level details and high-level semantics. To achieve this, we constructed a search space containing lightweight convolutional operation blocks with different sizes of sensitivity fields. Using network architecture search, we obtained a lightweight network suitable for industrial image segmentation, which we call the lightweight operation search UNet (LOS-UNet).

Our LOS-UNet achieves an excellent precision-speed balance on industrial image segmentation datasets. It has significantly fewer parameters than classical segmentation networks and, in some datasets, even fewer parameters than the lightweight network BiSeNet [21]. Despite having fewer parameters, LOS-UNet maintains competitive network performance. We hope that our method will facilitate further research in the field of industrial image segmentation.

References

[1] Zheng, J., Wang, L., Liu, J., Wang, H., Wang, S., Wang, L., Zhang, J.: An inspection method of rail head surface defect via bimodal structured light sensors. International Journal of Machine Learning and Cybernetics, 1–18 (2022)

[2] Banharnsakun, A.: Hybrid abc-ann for pavement surface distress detection and classification. International Journal of Machine Learning and Cybernetics **8**, 699–710 (2017)

[3] Tao, Y., Jun, Z., Zhi-hao, Z., Yi, Z., Fu-qiang, Z., Xiao-zhi, G.: Fault detection of train mechanical parts using multi-mode aggregation feature enhanced convolution neural network. International Journal of Machine Learning and Cybernetics **13**(6), 1781–1794 (2022)

[4] Dou, Y., Huang, Y., Li, Q., Luo, S.: A fast template matching-based algorithm for railway bolts detection. International Journal of Machine Learning and Cybernetics **5**, 835–844 (2014)

[5] Zhu, X., Liu, J., Zhou, X., Qian, S., Yu, J.: Enhanced feature fusion structure of yolo v5 for detecting small defects on metal surfaces. International Journal of Machine Learning and Cybernetics, 1–11 (2023)

[6] Zhou, Y., Chen, S., Wang, Y., Huan, W.: Review of research on lightweight convolutional neural networks. In: 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), pp. 1713–1720 (2020). IEEE

[7] Tulbure, A.-A., Tulbure, A.-A., Dulf, E.-H.: A review on modern defect detection models using dcnns–deep convolutional neural networks. Journal of Advanced Research **35**, 33–48 (2022)

[8] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)

[9] Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440 (2016)

[10] He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1389–1397 (2017)

[11] Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., Van Baalen, M., Blankevoort, T.: A white paper on neural network quantization. arXiv preprint arXiv:2106.08295 (2021)

[12] Bashivan, P., Tensen, M., DiCarlo, J.J.: Teacher guided architecture search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5320–5329 (2019)

[13] Aguilar, G., Ling, Y., Zhang, Y., Yao, B., Fan, X., Guo, C.: Knowledge distillation from internal representations. In: Proceedings of the AAAI

Conference on Artificial Intelligence, vol. 34, pp. 7350–7357 (2020)

[14] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

[15] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)

[16] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., *et al.*: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1314–1324 (2019)

[17] Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6848–6856 (2018)

[18] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)

[19] Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: Icnet for real-time semantic segmentation on high-resolution images. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 405–420 (2018)

[20] Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)

[21] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 325–341 (2018)

[22] Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. The Journal of Machine Learning Research **20**(1), 1997–2017 (2019)

[23] Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)

[24] Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)

[25] Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018)

[26] Chen, L.-C., Collins, M., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., Shlens, J.: Searching for efficient multi-scale architectures for dense image prediction. Advances in neural information processing systems **31** (2018)

[27] Nekrasov, V., Chen, H., Shen, C., Reid, I.: Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9126–9135 (2019)

[28] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)

[29] Dung, C.V., *et al.*: Autonomous concrete crack detection using deep fully convolutional neural network. Automation in Construction **99**, 52–58 (2019)

[30] Li, S., Zhao, X., Zhou, G.: Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. Computer-Aided Civil and Infrastructure Engineering **34**(7), 616–634 (2019)

[31] Dong, H., Song, K., He, Y., Xu, J., Yan, Y., Meng, Q.: Pga-net: Pyramid feature fusion and global context attention network for automated surface defect detection. IEEE Transactions on Industrial Informatics **16**(12), 7448–7458 (2019)

[32] Guo, M.-H., Lu, C.-Z., Liu, Z.-N., Cheng, M.-M., Hu, S.-M.: Visual attention network. arXiv preprint arXiv:2202.09741 (2022)

[33] Sovrasov, V.: Ptflops: a Flops Counting Tool for Neural Networks in Pytorch Framework. https://github.com/sovrasov/flops-counter.pytorch

[34] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp. 234–241 (2015). Springer

[35] Song, K., Yan, Y.: A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. Applied Surface Science **285**, 858–864 (2013)

[36] Wieler, M., Hahn, T.: Weakly supervised learning for industrial optical

inspection. In: DAGM Symposium In (2007)

[37] Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1925–1934 (2017)

[38] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6230–6239 (2016)

[39] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)

Appendix A

The following table A1 shows the LOS-UNet network architecture obtained by searching the network.

Table A1 Details of the searched LOS-UNet network architecture.

	Operation block								
Dataset	Encoder-1	Encoder-2	Encoder-3	Encoder-4	Encoder-5	Decoder-1	Decoder-2	Decoder-3	Decoder-4
inclusion	Double DW-D-Conv	Single Conv	Single Conv	Single Conv	Single Dilation Conv	Cutconnect Block	Single Conv	Single Conv	Cutconnect Block
patches	Single Conv	Single Conv	Cutconnect Block	Single Conv	Cutconnect Block	Cutconnect Block	Cutconnect Block	Single Conv	Single Conv
scratches	Cutconnect Block	Single Conv	Cutconnect Block	Single Conv	Single Conv	Single Conv	Single Conv	Single Conv	Single Conv
tile	Double DWconv	Single Conv	Cutconnect Block	Single Conv	Cutconnect Block	Single Dilation Conv	Cutconnect Block	Single Conv	Cutconnect Block
cement	Cutconnect Block	Single Conv	Single Conv	Single Dilation Conv	Double DWconv	Single Dilation Conv	Single Conv	Single Dilation Conv	Cutconnect Block
fabric	Cutconnect Block	Single Conv	Single Conv	Double DW-D-Conv	Single Dilation Conv	Single Dilation Conv	Cutconnect Block	Single Conv	Single Dilation Conv
wallpaper	Double DWconv	Cutconnect Block	Single Conv	Cutconnect Block	Cutconnect Block	Double DWconv	Single Dilation Conv	Single Conv	Single Conv
Wafer Defect	Double DWconv	Single Conv	Cutconnect Block	Single Conv	Double DWconv	Cutconnect Block	Double DW-D-Conv	Cutconnect Block	Single Conv