Towards More Powerful Multi-column Convolutional Network for Crowd Counting

Jiabin Zhang[1,2], Qi Chu[1,2], Weihai Li[1,2(✉)], Bin Liu[1,2], Weiming Zhang[1,2], and Nenghai Yu[1,2]

[1] School of Cyberspace Security, University of Science and Technology of China, Hefei, China
munian@mail.ustc.edu.cn, whli@ustc.edu.cn
[2] Key Laboratory of Electromagnetic Space Information, Chinese Academy of Science, Hefei, China

Abstract. Scale variation has always been one of the most challenging problems for crowd counting. By using multi-column convolutions with different receptive fields to deal with different scales in the scene, the multi-column convolutional networks have achieved good performance. However, there is still great potential waiting to be explored for multi-column convolutional networks. To this end, we propose to design a multi-column neural network that can more effectively adapt to scene scale variations automatically, by applying Neural Architecture Search technology. First, we combine Progressive Neural Architecture Search scheme with crowd counting to construct our Progressive Multi-column Architecture Serach (PMAS) framework. Furthermore, to reduce the bias caused by the weight-share scheme, which is widely adopted in efficient Neural Architecture Search, we propose a novel pre-architecture-based weight-share scheme. Experiments on several challenging datasets demonstrate the effectiveness of our method.

Keywords: Crowd counting · Neural architecture search · Multi-column convolutional network

1 Introduction

Crowd counting is one of the most important tasks of crowd scene understanding and has attracted the interest of many researchers due to its practical applications, such as traffic monitoring, crowd flows analysis and other public safety field. One of the most challenging difficulties of crowd counting is the extreme variations in the size of people in the scene, as shown in Fig. 1. To obtain multi-scale features that encode different scale information, previous works [1–3] attempt to design their network as a multi-column form. However, [4–6] pointed out that different columns of these architectures tend to generate similar features, which contraries to the intention of the multi-column architecture design. In other words, there are huge redundant parameters among columns. To solve

Y. Peng et al. (Eds.): ICIG 2021, LNCS 12888, pp. 381–392, 2021.
https://doi.org/10.1007/978-3-030-87355-4_32

Fig. 1. Examples from ShanghaiTech Part A dataset [1]. The extreme scale variation caused by perspective distortion is one of the most challenging difficulties for crowd counting.

this problem, McML [4] proposed a novel training strategy, which uses an auxiliary network to estimate the mutual information among columns. The estimated mutual information measures the correlation between features learned by different columns, so the learning target is to minimize counting errors and mutual information between columns, which achieved by an iterative and mutual learning scheme. But what if we improve the multi-column architecture from the architecture itself? In other words, we want to design a multi-column architecture that can learn richer multi-scale features easier and each column can learn the real characteristics of scenes of different scales. And with the development of Neural Architecture Search (NAS) [8–10,18,19], we can design the ideal network in an automatic and learnable way.

In this paper, we develop a novel Progressive Multi-column Architecture Search (PMAS) framework. First, we develop a Progressive Multi-column Architecture Search framework with a novel multi-column search space and multi-target search scheme. Furthermore, inspired by previous works [9,10,13–16], we propose a novel weight-share strategy, pre-architecture-based weight-share. The proposed strategy can improve the search result without increasing the cost of the search process. The proposed framework is illustrated in Fig. 2.

The main contributions of this paper include:

1. A novel, NAS-based framework to improve the multi-column architecture performance in crowd counting.
2. A novel weight-share strategy that achieves better search result than previous weight-share strategy.

2 Related Work

2.1 Density Map Estimation-Based Crowd Counting

There are mainly three types of approaches in previous literature on crowd counting: detection-based methods [21,22], regression-based methods [23,24] and density map estimation-based methods [1–6,12,17]. In recent years, density map

estimation-based methods are dominant in related research, a lot of novel methods are proposed: MCNN [1] proposed to learn multi-scale features by multi-column network for the first time. But CSRNet [5] criticized that the multi-column architecture in MCNN cannot effectively learn different feature representations for different scale, and then designed a deeper single-column network. McML [4] turned back to multi-column architecture again, they started with the discovery that there are lots of redundancy in the learned parameters in different columns of the multi-column architecture, and proposed mutual learning scheme assisted by a mutual information estimation network to reduce the redundancy.

Our work is inspired by McML [4], but we focus on the multi-column architecture itself.

2.2 Neural Architecture Search

In the past, because it would cost hundreds or thousands of GPU hours [13,18,19], the research and application of NAS progressed slowly. But since ENAS [12] adopted the weight-share scheme, such efficient NAS [9,10] only needs to cost a few GPU hours to complete the search, which brings great convenience. With the rapid development of efficient NAS technology, the application of NAS has expanded from classification to other fields of deep learning. When talking about the design and application of a NAS algorithm, three points are mainly considered:

Search Space. Existing methods can be categorized into searching the macro space [13,19], which performs a global search and the search result is just the network itself, the micro space [8–10], which presets the network to be constructed by several cells, typically regular-cell and reduction-cell, and the search is performed on the cell. The search space of our framework belongs to the macro space, which means when the search is completed, we can directly get the searched network.

Search Algorithm. It is the key to designing an efficient and effective NAS framework. So far, various search strategies based on different theories are proposed, such as: evolutionary algorithm based [13], reinforcement learning based [10,18,19], gradient-based [20], and the method we adopt: performance predictor based [8,9].

Search Target. It is task-specific, for example, in classification, the accuracy rate may be adopted as the search target [9,10,13,18,19]; In detection, mean Average Precision (mAP) may be the search target [20]. As for crowd counting, one of the most intuitive idea is taking Mean Absolute Error (MAE) as the search target, but we additionally proposed using the weighted sum of MAE and Multi-scale Structural SIMilarity (MSSIM) as the search target, and experiments demonstrate that it performs better than only MAE used.

3 Progressive Multi-column Architecture Search

In this section, we present the proposed Progressive Multi-column Architecture Search framework, as shown in Fig. 2. The definition of our search space is intro-

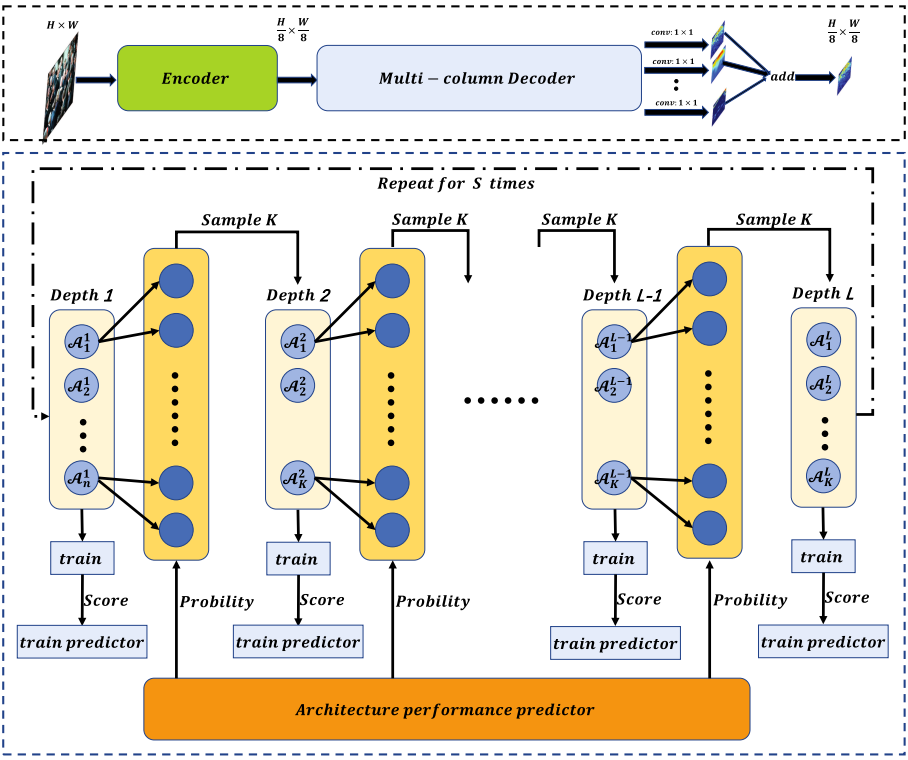

Fig. 2. An illustration of our Progressive Multi-column Architecture Search framework. *Top*: The paradigm of the multi-column network. The encoder is the first ten layers of VGG-16 [9], and the decoder is searched by algorithm. *Bottom*: The sample → train → predict → sample circulation search progress, and we repeat it for S times then select the top-5 to train from scratch. The blue circle nodes represent network architectures, $\mathcal{A}_j^i$ represents the j-th network sampled for depth i, the search space for depth i + 1 is derived by stacking convolutional layers on $\mathcal{A}_1^i$ to $\mathcal{A}_K^i$. More details on Algorithm 1 (Color figure online)

duced in Sect. 3.1. Then the search target is defined in Sect. 3.2. The search algorithm and the overview of the framework is described in Sect. 3.3.

3.1 Multi-column Decoder Search Space

Similar to CSRNet [5], all networks in our search space are constructed by a front-end encoder and a back-end decoder. Regarding the front-end, we adapt the first ten layers of VGG-16 network [7] and only use 3×3 kernels, as settings in CSRNet. The search process on the back-end decoder, which has maximum depth L and maximum number of columns K. Six operations are allowed when constructing a back-end decoder, and we identify them with the numbers {0, 1, 2, 3, 4, 5}:

– 0 : identity mapping;
– 1 : 3×3 convolution;
– 2 : 3×3 dilated convolution with rate 2;
– 3 : 3×3 dilated convolution with rate 4;
– 4 : 5×5 depth-wise separable convolution;
– 5 : 7×7 depth-wise separable convolution;

There have been three down-sampling in the front-end encoder [5], so all operations above used in back-end decoder will preserve the spatial size of the feature maps.

Then, we can encode a back-end decoder into a numerical matrix, which has maximum column L and maximum row K, corresponding to the maximum depth and maximum number of columns. In our experiments, the counting baseline is constructed as described in McML [4], we treat the four configurations of back-end in CSRNet [6] as four columns, and we discard the column with mixed dilation rate.

It is worth noting that in most previous NAS literature, the minimum unit during the search process is a single convolution, which may not be suitable in our multi-column search. The intention of the multi-column architecture is that different columns deal with different scales, and they are complementary and cooperating. But taking a single convolution as the minimum unit weakens the correlation among columns and cannot explicitly meet the above requirements. So, we take the K convolutions in the same depth l as the minimum search unit, in which way, the expansion of network and weight-share will not process on a single convolution, but K convolutions in the same depth.

3.2 Multi-objective Search

Typically, the so-called search target is the metric that evaluates how a network performs on the target-task. Mean Absolute Error (MAE) is widely used in crowd counting:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} \left| C_i - C_i^{gt} \right| \tag{1}$$

Here N is the number of images in test-set, C_i is the estimated count and C_i^{gt} is the ground truth count of the i-th image.

In addition to MAE, there are many other metrics to evaluate the quality of estimated density map, such as PSNR, SSIM and so on. In these metrics, Multi-scale Structural Similarity (MS-SSIM) [11] is easy to implement and can effectively measure the similarity between the estimated density map and the ground truth one. Here we use the ameliorated Dilated MS-SSIM [12]:

$$DMS \cdot SSIM(X_0, Y_0) = \prod_{i=0}^{m-1} \{SSIM(X_i, Y_i)\}^{\alpha_i} \tag{2}$$

Where X_0 is the estimated density map and Y_0 is the corresponding ground truth, and X_i, Y_i is the corresponding features in i-th level of DMS-SSIM network, which is a dilated convolutional neural network with fixed Gaussian kernel,

α^i is the importance weight of $SSIM(X_i, Y_i)$. For more details, we recommend to refer to [12].

Finally, our search target can be formulated as:

$$score = MAE + \gamma \frac{1}{N} \sum_{i=1}^{N} [1 - DMS \cdot SSIM(C_i, C_i^{gt})] \tag{3}$$

Here, γ is the importance weight of the mean DMS · SSIM on the test-set. It is worth noting that the lower the score, the better the network performance.

3.3 Efficient Progressive Multi-column Architecture Search Algorithm

Algorithm 1 shows the pseudocode, and Fig. 2 illustrates the whole search process.

Our search algorithm incorporates the core idea of PNAS [8] and the improved version EPNAS [9], there are two key points:

First, a progressive and sequential searching process. This means our search starts with the simplest situation, for example, the networks that are constructed by one convolutional layer, and expands the depth and complexity of networks by stacking convolutional layers or blocks behind the former ones.

Second, network performance predictor. It is a neural network trained to learn the mapping from network structure to performance on the target task [8,9]. If the network performance predictor has a good prediction of the performance of networks in search space, we can just simply give networks that are predicted good higher probability to be sampled in next stage, and constrain the number of networks to be sampled in each stage. In this way, we can control the search cost without missing good networks.

But there are still some deficiencies in Algorithm 1. The reasonableness of pruning the search space lies in the assumption that the predictor has a good prediction of the performance of the networks. But in the initial stage, without sufficient training, the predictor cannot predict well. Following EPNAS [9], we adopted a temperature-driven sampling procedure, in which the sampling probability π_i of architecture i is amended to $\pi_i^{1/\tau} / \sum_j \pi_j^{1/\tau}$ with temperature τ decaying quickly to 1 as the search iterations increases. In this way, the sampling is likely random at beginning, and we trust more the prediction results of the predictor as it gets more training. More details will be introduced in Sect. 5.

4 Pre-architecture-Based Weight-Share

Weight-share scheme is key to conducting an efficient Neural Architecture Search. This method was proposed in ENAS [10], but similar idea 'weight inheritance' first appeared in [13], and the core of these two methods is "share whenever possible", in other words, as long as the two networks have the same shape in a certain convolutional layer, the weights of this layer will share or inherit.

Algorithm 1. Efficient Progressive Multi-Column Architecture Search

Input: K(max columns), L(max depth), N(numbers of arch sampled each time), S(times the search repeats), P(the performance predictor), $\mathcal{A}_1$(all possible arch that depth of back-end equal 1), trainSet, valSet.

for s=1:S **do**
  for l=1:L **do**
    if l==1 **then**
      weight-share($\mathcal{A}_1$) ▷ Loads weight that saved before when s is not 1
      train($\mathcal{A}_1$,trainSet) ▷ Train all depth-1 arch one epoch on trainSet
      save-weight($\mathcal{A}_1$) ▷ Save network weights
      MAE,MSSIM=eval($\mathcal{A}_1$,valSet) ▷ Evaluate performance on valSet
      Calculate score according formula(3)
      train(P,$\mathcal{A}_1$,score) ▷ Train predictor with input-label pairs (arch, score)
    end if
    if l>1 **then**
      $\mathcal{A}_l$=create-all-arch($\mathcal{A}_{l-1}$) ▷ Expand $depth_l$ search space based on $depth_{l-1}$ sampled architectures
      $\mathcal{P}$=inference($\mathcal{A}_l$) ▷ Predictor infers score of all architectures in the $depth_l$ search space
      π=SoftMax($-\mathcal{P}$) ▷ Calculate sample probability distribution, higher score, lower probability
      $\hat{\mathcal{A}}_l$=sample(pi,N,$\mathcal{A}_l$) ▷ Sample N architectures from the $depth_l$ search space
      weight-share($\hat{\mathcal{A}}_l$) ▷ Loads weights according to weight-share strategy
      train($\hat{\mathcal{A}}_l$,trainSet) ▷ Train sampled networks one epoch on trainSet
      save-weight($\hat{\mathcal{A}}_l$) ▷ Save network weights
      MAE,MSSIM=eval($\hat{\mathcal{A}}_l$,valSet) ▷ Evaluate performance
      Calculate score according formula(3)
      train(P,$\hat{\mathcal{A}}_l$,score) ▷ Train predictor
    end if
  end for
end for
Get top-5 architectures during the whole search space.

There is no doubt that weight-share can greatly speed up the search process. But recently, some works [14–16] indicated that weight-share scheme may degrade the performance because it may degenerate search-evaluation correlation, for example, architectures with a better validation performance during the search phase may perform worse in final evaluation. Specific causes of this phenomenon are not clear, but it is obvious that the weights shared from one network may not be suitable for another network, and the latter may perform worse because of this. So, we can alleviate this negative impact of weight-share by decreasing the amount of sharing. The proposed pre-architecture-based weight-share scheme is illustrated in Fig. 3. If and only if the two networks have the same structure from the 1-th layer to the i-th layer, and the $(i+1)$-th layer is inconsistent, there is a sharing of convolution weights from the 1-th layer to the i-th layer between the two networks. The intention of the proposed method is

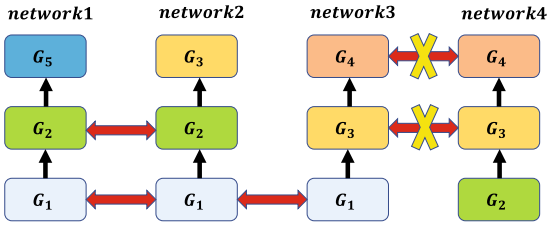

Fig. 3. An illustration of our pre-architecture-based weight-share scheme. G_i denotes index of convolutional layer type.

to strengthen the correlation among convolutional layers and thus alleviate the degraded search.

5 Experiments

We perform search on ShanghaiTech Part A dataset [1], and then train the searched model on other dataset [1,17] to demonstrate the robustness of our searched model.

5.1 Evaluation Metrics

We use mean absolute error (MAE) and mean square error (MSE) to evaluate the performance. MAE is defined as formula 1, and MSE is defined as:

$$MSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(C_i - C_i^{gt})^2} \tag{4}$$

These two metrics indicate the accuracy and robustness of the model. We also calculate Structural Similarity (SSIM) to evaluate the quality of estimated density map.

Table 1. Estimation errors on ShanghaiTech, UCF_CC_50 datasets.

Method	ShanghaiTech A		ShanghaiTech B		UCF_CC_50	
	MAE	MSE	MAE	MSE	MAE	MSE
MCNN [1]	110.2	173.2	26.4	41.3	377.6	509.1
ic-CNN [3]	72.5	118.2	13.6	21.1	291.4	349.4
CSRNet [7]	68.2	115.0	10.6	16.0	266.1	397.5
SANet [5]	67.0	104.5	8.4	13.6	258.4	334.9
McML [6]	59.1	104.3	8.1	10.6	246.1	367.7
Counting baseline	64.8	106.9	9.3	15.3	252.3	343.6
Best searched	61.2	98.3	8.1	13.0	246.6	351.2

Table 2. Ablation study results on ShanghaiTech Part A dataset.

Configurations		Performance	
		Top-5 mean MAE	Best model MAE
Weight-share scheme	Share-as-possible	64.01 ± 1.10	62.21
	Pre-architecture-based	63.30 ± 0.52	62.65
Search target	MAE	63.30 ± 0.52	62.65
	MSSIM	63.67 ± 0.87	62.26
	MAE+MSSIM	62.91 ± 1.35	61.19
vs. random search	Random search	64.65 ± 0.98	63.46
	Our full method	62.91 ± 1.35	61.19

Table 3. Comparison with other multi-column networks in Shanghai Part A dataset.

Method	SSIM among columns	SSIM between ET& GT
MCNN [1]	0.71	0.55
CSRNet [7]	0.84	0.71
ic-CNN [3]	0.72	0.68
McML [6]	0.70	0.82
Our best searched	0.60	0.76

5.2 Implementation Details

In architecture search experiments, the maximum column number is 3, and the maximum depth is 8. During the search, except for $depth_1$ in each stage, 25 architectures are sampled at each sampling step, and these sampled networks are trained with MSSIM loss [12] and Adam optimizer with learning rate 1e-5 for one epoch. As for the predictor, we implement it as a fully-connection network with two hidden layers. To deal with architecture with arbitrary columns and depth, we encode all possible permutations in a single convolutional layer (in the case 3 columns and 6 candidate-operations, there are $6^3 - 1 = 216$ for a single layer) as 100-dims vectors in a hash-embedding way, and save them as a lookup-table. Then, 8 embeddings of 8 layers are concatenated as a 800-dims vector, which is inputted into the predictor. For training the predictor, we use L2 loss and learning rate 1e-3. The whole search progress and training the searched networks from scratch cost approximately 24 TITAN Xp GPU hours.

5.3 Ablation Study

Effectiveness of Pre-architecture-Based Weight-Share. The result is shown in Table 2. Besides the best architectures searched by two weight-share schemes, which are somewhat random, we also calculate the mean MAE of the searched top-5 architectures, which can indicate the overall performance of the architecture

$$\begin{bmatrix} 1 & 2 & 4 & 4 & 4 & 4 & 0 & 0 \\ 3 & 5 & 4 & 3 & 0 & 4 & 0 & 4 \\ 5 & 3 & 2 & 4 & 1 & 3 & 1 & 4 \end{bmatrix}$$

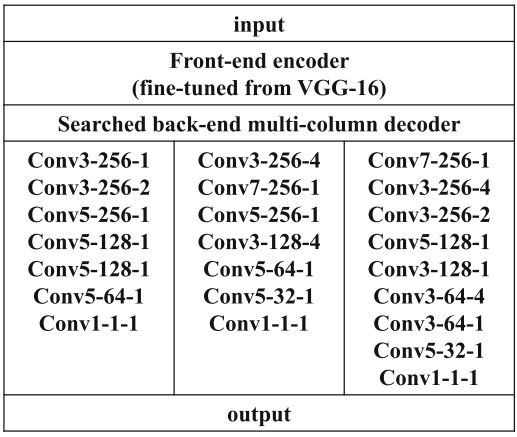

Fig. 4. The best architecture found by our progressive multi-column architecture search.

cluster and better reflect the ability of the search algorithm. As shown in Table 2, because our pre-architecture-based weight-share scheme strengthens the correlation between adjacent convolutional layers and the weights are shared in a more reasonable way, so it can improve the search result in a great margin.

Effectiveness of Multi-objective Search. As shown in Table 2, the result of taking MAE or MSSIM alone as search target are similar, and the combination of these two makes it possible to evaluate the performance of a candidate network during the search process from a more comprehensive metric, so that we can search for better architecture cluster.

Effectiveness of the Whole Search Framework. Previous works [14,15] has pointed out that the excellent performance shown by many NAS literature may be attributed to the design of the search space, and has nothing to do with the search algorithm. To validate the effectiveness of our search framework, we set up a random search baseline as the control. As a fair comparison, we constrain the random search costs the same GPU hours as our other search experiments, we randomly sample 1000 architectures from the search space, and train them for 1 epoch, then select the top-5 architectures and train them from scratch. The result is shown in Table 2. Our search framework outperforms the random search baseline by a great margin, and it indicates that our search framework can search excellent networks in the huge search space effectively.

5.4 Performance and Comparison

The best architecture we searched is shown in Fig. 4. Besides ShanghaiTech Part A, we also train the searched network on other public datasets, the result and comparison with other multi-column architectures are shown in Table 1 and Table 3. Notably, without complex training scheme, our searched architecture

is comparable with the state-of-the-art multi-column architecture not only in extremely crowd scenes (Shanghai Part A), but also in relatively sparse scenes (Shanghai Part B) and small datasets (UCF_CC_50). Besides, we evaluate the similarity among columns as McML [4] in Shanghai Part A dataset. We can observe that our searched architecture obtains lower SSIM among columns and higher SSIM between estimated density map and ground truth than other multi-column networks. Lower SSIM among columns indicates that different columns of our searched architecture adapt well to different scale in the regular end-to-end training manner. Higher SSIM between estimated density map and ground truth indicates the success of the MSSIM metric in our multi-objective search target, and our model can generate density map with higher quality.

6 Conclusion

In this paper, we integrate the Progressive Neural Architecture Search framework with multi-column architecture, and develop a novel Progressive Multi-column Architecture Search framework towards crowd counting task. To get a more powerful multi-column network, we not only construct special search space, but propose multi-objective search, which improve the search result and estimated density map of searched model. Besides, to deal with flaws of the weight-share scheme, we propose a novel pre-architecture-based weight-share scheme, and the experiments demonstrate its effectiveness. Finally, our searched model achieves comparable performance with the state-of-the-art multi-column architecture without a complex train scheme.

References

1. Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y.: Single-image crowd counting via multi-column convolutional neural network. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 589–597. IEEE (2016)
2. Cheng, Z.-Q., Li, J.-X., Dai, Q., Wu, X., Hauptmann, A.: Learning spatial awareness to improve crowd counting. In: 2019 IEEE/CVF International Conference on Computer Vision, pp. 6152–6161. IEEE (2019)
3. Ranjan, V., Le, H.M., Hoai, M.: Iterative crowd counting. In: Proceedings of the European Conference on Computer Vision, pp. 270–285. IEEE (2018)
4. Cheng, Z.-Q., Li, J.-X., Dai, Q., Wu, X., He, J.-Y., Hauptmann, A.G.: Improving the learning of multi-column convolutional neural network for crowd counting. In: Proceedings of the 27th ACM International Conference on Multimedia, pp. 1897–1906. ACM (2019)
5. Li, Y., Zhang, X., Chen, D.: CSRNet: dilated convolutional neural networks for understanding the highly congested scenes. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1091–1100. IEEE (2018)
6. Wang, Z., Xiao, Z., Xie, K., Qiu, Q., Zhen, X., Cao, X.: In defense of single-column networks for crowd counting. In BMVC, p. 78 (2018)
7. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)

8. Liu, C., et al.: Progressive neural architecture search. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11205, pp. 19–35. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01246-5_2
9. Perez-Rua, J.-M., Baccouche, M., Pateux, S.: Efficient progressive neural architecture search. In: BMVC, p. 150 (2018)
10. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning, pp. 4092–4101 (2018)
11. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, vol. 2, pp. 1398–1402 (2003)
12. Liu, L., Qiu, Z., Li, G., Liu, S., Ouyang, W., Lin, L.: Crowd counting with deep structured scale integration network. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1774–1783 (2019)
13. Real, E., et al.: Large-scale evolution of image classifiers. In: ICML 2017 Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2902–2911 (2017)
14. Yu, K., Sciuto, C., Jaggi, M., Musat, C.,Salzmann, M.: Evaluating the search phase of neural architecture search. In: Eighth International Conference on Learning Representations (2020)
15. Bender, G., et al.: Can weight sharing outperform random architecture search? An investigation with Tu-NAS. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14323–14332. IEEE (2020)
16. Li, G., Qian, G., Delgadillo, I.C., Muller, M., Thabet, A., Ghanem, B.: SGAS: sequential greedy architecture search. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1620–1630. IEEE (2020)
17. Idrees, H., Saleemi, I., Seibert, C., Shah, M.: Multi-source multi-scale counting in extremely dense crowd images. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2547–2554. IEEE (2013)
18. Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V.: Learning transferable architectures for scalable image recognition. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8697–8710. IEEE (2018)
19. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR (2016)
20. Ghiasi, G., Lin, T.-Y., Le, Q.V.: NAS-FPN: learning scalable feature pyramid architecture for object detection. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7036–7045. IEEE (2019)
21. Brostow, G.J., Cipolla, R.: Unsupervised Bayesian detection of independent motion in crowds. In: 2006 IEEE Computer Society Conference on Computer Vision and Pat-tern Recognition, vol. 1, pp. 594–601. IEEE (2016)
22. Lin, S.-F., Chen, J.-Y., Chao, H.-X.: Estimation of number of people in crowded scenes using perspective transformation. Syst. Man Cybern. **31**(6), 645–654 (2001)
23. Chan, A.B., Vasconcelos, N.: Counting people with low-level features and Bayesian regression. IEEE Trans. Image Process. **21**(4), 2160–2177 (2012)
24. Chen, K., Gong, S., Xiang, T., Loy, C.C.: Cumulative attribute space for age and crowd density estimation. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2467–2474. IEEE (2013)