Improving Deep Learning based Optical Character Recognition via Neural Architecture Search

Zhenyao Zhao, Min Jiang*, *Senior Member, IEEE*, Shihui Guo*, *Member, IEEE*
Zhenzhong Wang, Fei Chao and Kay Chen Tan, *Fellow, IEEE*

***Abstract*—Optical character rcecognition (OCR) is a process of converting images of typed, handwritten or printed text into machine-encoded one. In recent years, the methods represented by deep learning have greatly improved the performance of OCR systems, but the main challenges of such systems are 1) to accurately perform text detection in complex scenes and 2) to identify and set the optimal parameters to optimize the performance of the system. In this paper, we propose an OCR method based on Neural Architecture Search technique, called AutOCR. The characteristic of the proposed method is the automatic design of text detection framework using an evolutionary computation neural architecture search method. This design can not only accurately recognize the text in a complex environment, but also avoid the process of experts participating in parameter adjustment. We compared it with different methods, and the experimental results proved the effectiveness of our method.**

I. Introduction

Optical character recognition (OCR) is a process of converting images of typed, handwritten or printed text into machine-encoded one. Because OCR has a wide range of applications, it has been attracting the research interest of different scholars [1]–[4]. OCR system includes two sub frameworks: text detection and text recognition. For a specific task, these two sub frameworks need to be designed according to the task requirements. For example, the mobile OCR system is more sensitive to the speed of operation, and the document OCR system requires higher recognition accuracy. Therefore, once the target task requirements change, the experts need to redesign the OCR system, which is time-consuming, labor-intensive and inefficient. The automatic design of OCR system by machines can effectively alleviate this problem. However, there are very few existing works on automating the design of the OCR system.

The main purpose of this work is to propose an automated design of an OCR system framework. Since there are already many reliable frameworks for text recognition, and most of them can be used for additional training on the text of the target task, for many OCR tasks [5]–[7], text recognition frameworks often do not require human involvement in design. Hence the challenge of OCR system automation lies in the automation of the text detection framework. With the application and rapid development of deep neural networks in object detection [8]–[10], the performance of text detection has also been greatly improved. Network architecture search (NAS) automates the architecture design of the deep neural network, and has made great achievements in image classification, language models [11]–[14] and object detection [15]–[18] in recent years. Architectures designed by many state-of-the-art NAS methods have even achieved better performance than hand-crafted ones. The automated architecture search process not only lowers the demanding requirements of expertise of designing networks, but also reduces a lot of design time, labor costs and can easily search for a suitable architecture based on design requirements.

This work proposed AutOCR, an automatic solution to design the architecture of deep neural network for a specific OCR task. More specifically, we introduced a NAS-based method called DetNAS [16] in the text detection framework. This is a novel method that searches for the backbone architecture in the object detection framework. It first pre-trains a one-shot supernet [19] on the image classification dataset and then fine-tunes this one-shot supernet on target detection datasets. Finally, we search for a high-performance backbone network on this trained supernet using evolutionary algorithm (EA). Many object detection framework backbone networks use existing high-performance image classification networks, such as ResNet [20]. But the performance of the backbone network on image classification and object detection performance is not positively correlated. Literature [16] shows that using the ClsNASNet, the best architecture searched on ImageNet [21], is not the optimal backbone network. Therefore, in order to achieve better performance, it is necessary to search for a suitable text detection framework for OCR target tasks. In addition, some OCR tasks have different requirements on system latency or accuracy, the NAS-based method can simply adjust the size of the NAS search space and search strategy to find an architecture that meets the requirements, which is a capability not available in OCR systems using fixed network architectures.

In our AutOCR framework, text recognition framework uses the currently excellent tesseract engine [5], which can be trained for the special font of the target task. Combining the training steps in [16], the design process of the AutOCR framework can be summarized in 4 steps:

1) Adjust the one-shot supernet search space according

Z. Zhao, M. Jiang, Z. Wang and F. Chao are with the Department of Artificial Intelligence, Xiamen University, China, Fujian, 361005. S. Guo is with the Department of Software Engineering of Xiamen University. Min Jiang and Shihui Guo are the corresponding authors and email: minjiang@xmu.edu.cn, guoshihui@xmu.edu.cn.

KC TAN is with the Department of Computer Science, City University of Hong Kong.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

to the target task's requirements. Pre-train the supernet on image classification datasets such as ImageNet (text image classification dataset is better).
2) Fine-tune the supernet on the text detection dataset of the target task.
3) Perform architecture search using EA on the trained supernet.
4) Train special fonts of target task on the tesseract engine.

In our experiments, we perform AutOCR on the street view dataset SVHN, a credit card dataset published by China Merchants Bank, and a vehicle dataset we collected from the Internet. We follow the above 4 steps to create our automatic OCR system AutOCR. We use a small search space size with 20 ShuffleNetv2 blocks set in the literature [16]. For comparison, we combine several mainstream object detection frameworks and tesseract engine to form different OCR systems. In comparison with OCR system using Yolov3-tiny [22], our automatic search framework has a stronger detection ability and higher recognition accuracy. Compared with different OCR systems using Faster R-CNN [23], Mask R-CNN [8] or Yolo v3 [22], AutOCR achieves a comparable performance.

Our main contributions are summarized as below:

- By combining the current high-performance architecture search method, we propose a complete automated OCR system AutOCR. This system effectively avoids the difficulty that needs to manually design networks and reduces the barrier for designing OCR systems for specific tasks. This automated approach has great application potential. As far as we know, the research on automated OCR systems is still under-explored.
- This proposed framework AutOCR not only works well but also can achieve comparable performance as existing OCR systems using advanced object detection frameworks.

II. Background and Related Work

A. Optical Character Recognition System

OCR has been an interesting and hot topic for many years. It can be used for the identification of specific objects, such as license plate recognition and certificate recognition. It also can be used for wide recognition. For example, autopilot uses machine vision [24] to identify signs to help path planning [25]. OCR process can be divided into two phases: 1) Detect position coordinates containing text in input image. 2) Recognize text based on position coordinates. Compared to text recognition, text detection is often more challenging. Recently, CNN-based object detection and segmentation frameworks have been widely used in text detection problems. One type of solution [1]–[4], [6], [26] for text detection is to treat text in an image as a specific object and then detect it with an object detection framework. Some of the latest state-of-the-art object detection frameworks [3], [6], [26] are used to detect text.

B. Object detection

Object detection aims to locate and classify each object instance in an image. With the rapid development of deep convolutional network, the performance of object detectors has been greatly improved. At present, CNN-based object detection can be divided into two major methods: two-step method based on R-CNN [23], [27] and one-step method based on YOLO [10], [22].

R-CNN based object detection: R-CNN uses the ability of convolutional neural networks (CNN) to extract image features. It views a detection problem as a classification problem leveraging the development of classification. It uses CNN to extract deep features of proposals generated by selective search [28] and then uses Support Vector Machine (SVM) to classify these features.

YOLO based object detection: YOLO's approach is to extract feature maps on the entire image and then directly regresses the bounding boxes on the feature maps. SSD [10] is based on YOLO, which uses different aspect ratio boxes at different stages to predict the bounding box and further improve YOLO's performance.

Generally, the two-step method is slower than the one-step method, but has higher accuracy. The DetNAS used in our framework is a two-step method.

C. Neural Architecture Search

The goal of NAS is to automatically find a network with the best performance on a specific task, it can be expressed by the formula 1.

$$a^* = \underset{a \in S}{argmax}\, \mathrm{Per}(a) \tag{1}$$

S represents a predefined search space, a represents an architecture. Per function is used to evaluate the performance of an architecture on the given task. NAS offers a variety of advantages over manual design: 1) it not only effectively alleviates the difficulty of manual design and the demand of expert knowledge, and 2) the performance of architectures searched by the latest NAS methods even surpasses the human-designed ones. Mainstream methods are three types: reinforcement learning (RL) based approach, evolutionary algorithms (EA) and gradient-based approach. Literatures NAS [29] and NASNet [30] designed neural architecture using RL-based controller. AmoebaNet [31] used EA to search and prove that EA can also achieve comparable results with RL. Typical gradient-based approach are [32] and [33], which relax the discrete architectural space into a continuous space through a mixture model. In order to reduce the search speed, some acceleration methods have been proposed such as one-shot NAS [19]. At present, NAS methods for object detection are also attracting more and more researchers' interest. For example, NAS-FPN [15], Auto-fpn [17] and MnasFPN [18] explored the object detection NAS method for searching FPN. Auto-DeepLab [34] and DetNAS [16] explored the way to search for backbone network. In this work, we apply the NAS method to develop our automated OCR system framework.

III. Methods

A. The execution process of our OCR system

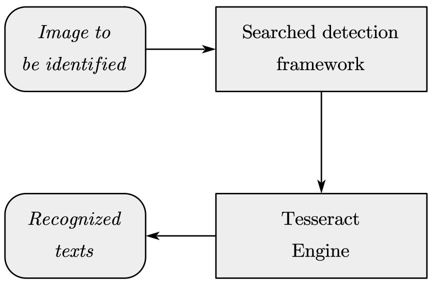

Fig. 1. The execution process of our OCR system. The detection framework is searched by NAS method. The tesseract engine is a high performance text recognition framework maintained by Google.

As shown in Figure 1, the execution process of our OCR system is divided into two steps. Searched detection framework detects position coordinates of texts in an input image, then outputs images containing only text areas. This framework is searched by the object detection NAS method. Specifically, we used DetNAS [16] to search a suitable backbone network for detection framework. Specific details are shown in Section III-B. Tesseract engine [5] is a high performance text recognition framework maintained by Google. In our system, it is responsible for recognizing text images transmitted by the detection framework and outputting recognized texts. It has a rich font library and the advantage of being able to customize font library.

B. The pipeline of AutOCR

As in Figure 2, AutOCR contains 4 steps. Steps 1 to 3 create a text detection framework automatically using the NAS method, and Step 4 creates a text recognition framework. The main process of creating a text detection framework is to search for a suitable backbone network. The process of creating a text recognition framework is mainly training on the target task font to improve the recognition accuracy.

Step 1. We first create and pre-train a one-shot supernet. The supernet including all paths [35] represents the search space of the entire backbone. An architecture can be obtained on this supernet by sampling one single path. As defined in [16], the supernet is mainly composed of multiple ShuffleNetv2 block. The complexity and performance of the detection framework model can be customized by changing the number of blocks. That is, we can determine the size of the search space based on the requirements of the target task and the complexity of the target data. For example, OCR systems that work on mobile devices like mobile phones are more sensitive to computational complexity, document OCR systems running on high-performance devices like servers may require the highest possible recognition accuracy. In order to meet the different requirements of the task, the network inference time can be reduced or the network performance can be enhanced by reducing or increasing the number of blocks. After determining the search space size, the supernet performs pre-training on ImageNet or other image classification datasets. Pre-training is the basis for the next fine-tuning step.

Step 2. Pre-trained supernet already has general image classification capabilities. In this step, the supernet needs to be fine-tuned on the training set of the target task. Specifically, the supernet with the feature pyramid network (FPN) [36] is continuously trained on the task detection training set.

Step 3. The third step is to use EA to generate the final architecture. The search space contains 4 operations that change the original ShuffleNetv2 block: changing the kernel size with 3×3, 5×5, 7×7 or replacing the right branch with an Xception block which contains 3 repeated separable depthwise 3×3 convolutions. Determine the type of each block from sampling the above 4 operations (i.e. one single path), a complete architecture is sampled from the supernet. The parameters of each architecture are directly inherited from supernet, so there is no need to train from scratch. To run the evolutionary process, multiple architectures sampled from the trained supernet as the initial population, then repeat the following steps until the iteration stop condition is met.

- First, each architecture in the population together with FPN forms a complete detection framework. We then verify the performance of each framework on the validation set of the target task. Some of the better performing architectures are selected and left behind.
- Second, those leftover architectures perform crossover and mutation to generate new architectures. These new architectures constitute a new population.

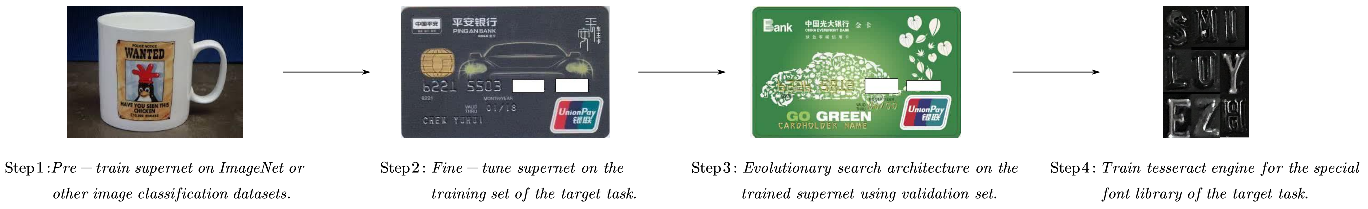
Step1: Pre-train supernet on ImageNet or other image classification datasets.
Step2: Fine-tune supernet on the training set of the target task.
Step3: Evolutionary search architecture on the trained supernet using validation set.
Step4: Train tesseract engine for the special font library of the target task.

Fig. 2. The pipeline of AutOCR that uses the NAS method to automatically create a task-oriented OCR systems. The first step is to pre-train the supernet on ImageNet or other image datasets to obtain general image classification capabilities. The second step is to build a complete detection architecture with pre-trained supernet and FPN, then fine-tune the supernet on the training set of the target task. The third step is to use EA on the trained supernet to search for the best backbone network on the validation set of the target task. The fourth step is to train tesseract engine on the font library of the target task to improve the recognition accuracy. The first 3 steps created the text detection framework automatically, and the last step created the text recognition framework.

After completing the iteration, we choose the one with the best performance and train it from scratch as the final backbone architecture. The specific evolution process is shown in Algorithm 1.

Algorithm 1 Evolutionary search algorithm

```
Input:
    Population size: $Pop$;
    Offspring size: $Pos$;
    Number of iterations: $NI$;
    Trained supernet: $S$;
    Validation set of target task: $D_V$;
Output:
    The backbone architecture with the best performance;
1: $Pop$ architectures are randomly sampled from the $S$ by sampling type of each block, these architectures as the initial population $P^{(1)}$;
2: for $i = 1 \to NI$ do
3:     Evaluate the performance of each architecture in $P^{(i)}$ on $D_V$;
4:     Select $Pos$ best performing architectures from $P$ as parent $Par$;
5:     Cross and mutate architectures in $Par$ according to the method in [16] to generate $P^{(i+1)}$.
6: end for
7: Choose the best architecture $\alpha$ from $P^{(NI)}$;
8: return The best performance backbone architecture $\alpha$.
```

Step 4. The last step is to train the text recognition framework, i.e. tesseract engine, on the special font library of the target task. The trained text recognition framework can effectively recognize the special font of the target task. For example, the text on the credit card may use special fonts. It is difficult to accurately recognize the information on the credit card without training the text recognition framework with the specific task set-up.

The creation framework of AutOCR is depicted in Algorithm 2. During the creation process, no expert is needed to design the structure of the network, and only requires little involvement in setup.

IV. Experiments

We test the effectiveness of our method on three datasets. One is the street view house numbers (SVHN) Dataset dataset [37], one is a credit card dataset published by China Merchants Bank [1], and one is a vehicle dataset we collected from the Internet. For comparison, we use Yolov3-tiny, Yolov3, Mask R-CNN and Faster R-CNN as the text detection framework respectively and combine the tesseract recognition framework to build different manual OCR systems. Faster R-CNN, Mask R-CNN and Yolo v3 are several advanced target detection algorithms in the past few years. Yolov3-tiny is a simplified version of Yolo v3, with a lower but faster performance than Yolo v3.

[1]Available download at: https://www.kesci.com/home/dataset/5954cf1372ea054a5e25870

Algorithm 2 Creation framework of AutOCR

```
Input:
    Training set of target task, $D_T$;
    Validation set of target task, $D_V$;
    Font library of the target task, $F$;
    ImageNet dataset $D_I$;
    Number of fine-tune iterations, $I_F$;
    Number of evolutionary iterations, $I_E$;
Output:
    OCR system AutOCR;
1: Construct one-shot supernet $S$;
2: Pre-train $S$ on $D_I$;
3: for $i = 1 \to I_F$ do
4:     Fine-tune $S$ on $D_T$;
5: end for
6: for $i = 1 \to I_E$ do
7:     Search for architecture $\alpha$ in the $S$ using evolutionary algorithm 1 on $D_V$;
8: end for
9: Fine-tune $\alpha$ from scratch on $D_T$;
10: Train *Tesseract engine* on $F$;
11: return OCR system AutOCR.
```

A. Experiment Settings

1) Datasets: SVHN is an image dataset used to identify street view house numbers. It contains 3.3k training images and 1.3k test images. We use the full numbers format, that is, the uncropped images. The text detection framework only recognizes the position of the numbers and does not classify it. The credit card dataset contains a total of 235 images with different card face patterns. We use 235 images as training set and 60 images as validation set. 4 categories are labeled on 4 areas of the card surface: card number, period of validity, type and name of card owner. The vehicle dataset contains 3665 images, 3000 images as the training set, and 665 images as the validation set. The position of the license plate is labeled. In the experiments, OCR systems detect and recognize the user name on the face of the credit card, the license plate numbers of the vehicle, and the numbers on the street view house.

2) Implementation Details: We use a search space containing 20 ShuffleNetv2 blocks in both three datasets. The other settings are consistent on these three datasets except for 30,000 iterations of the credit card dataset and 70,000 iterations of the vehicle and SVHN dataset during the fine-tuning phase with 4 GPUs. The setting of pre-training follows [16], the commonly used 1.28M training images are used for supernet pre-training. In the fine-tuning phase, the initial learning rate is set to 0.02, the weight decay is 1×10^{-4} and the momentum is 0.9. In the evolutionary search phase, the population size is set to 50, the number selected is 10, and the total iteration is 20 times. The final searched architectures are retrained in the pre-training and fine-tuning phase. For comparison, the Yolo v3 and Yolov3-tiny detection frameworks are trained according to the officially recommended settings. Faster R-CNN and Mask

R-CNN use ResNet-50 as backbone, FPN as feature extractor. Faster R-CNN and Mask R-CNN iterate over 36,000 times on the credit card dataset and vehicle dataset, and 24,000 times on the SVHN dataset with 2 GPUs. The initial learning rate is 0.005 with the weight decay of 1×10^{-4} and the momentum of 0.9. The recognition framework of each OCR system uses the same trained tesseract engine.

B. *Experiment Results*

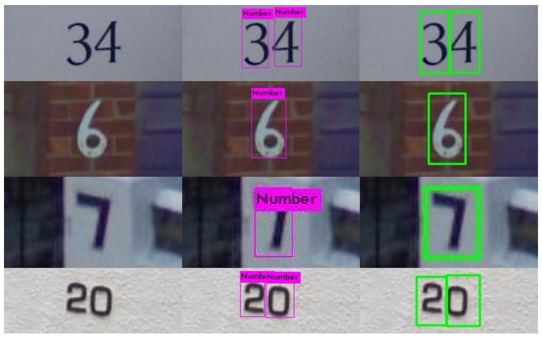

Fig. 3. Detection results on 4 SVHN images. The left column is the original images, the middle column is the Yolov3-tiny detection result, and the right column is the AutoCR detection result. Images from top to bottom correspond to ID 1 to 4.

1) Results on SVHN Dataset: On SVHN dataset, we test the accurate detection accuracy of AutOCR and 2 manual OCR systems using Faster R-CNN or Mask R-CNN on 100 images containing 199 numbers. The backbone FLOPs and accuracy results are shown in Table I. AutOCR's accuracy reaches 92.5%, which is the best performing one. It is 0.5% higher than the OCR system using Mask R-CNN and 2% higher than the system using Faster R-CNN. We selected 4 test images with different colors and backgrounds to show the detection and recognition results of the text. In addition to Faster R-CNN and Mask R-CNN, we also test on Yolo v3 and Yolov3-tiny. Figure 3 shows the text detection results of Yolov3-tiny and AutOCR on these 4 images. The left column shows the original images, the middle and right columns are the text detection results of Yolov3-tiny and AutOCR respectively. It can be seen that AutOCR correctly detects the numbers. Yolov3-tiny detects correctly on the 2nd to 4th images, but lost the left part of the information when it detects the "4" in the 1st image. The recognition results are shown in Table II. Image ID 1 to 4 correspond to the 4 images from top to bottom in Figure 3. Manual OCR system with Yolov3-tiny misidentified the 1st image as 3A. AutOCR and other manual OCR systems correctly recognize all images. This shows that AutOCR has a stronger performance than OCR system using Yolov3-tiny and has a comparable performance compared to manual OCR systems using object detection frameworks Faster R-CNN, Mask R-CNN or Yolo v3.

2) Results on Credit Card Dataset: On credit card Dataset, systems are tested on 60 images. AutOCR still performs best with 96.7% accuracy. It is 3.4% higher than the OCR system using Mask R-CNN and 6.7% higher than the system using

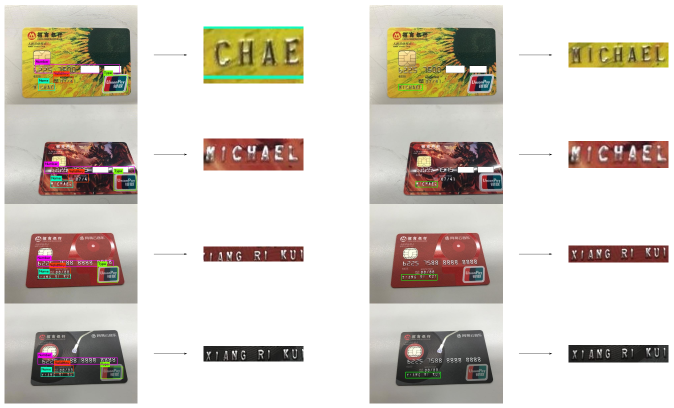
Fig. 4. Detection results on 4 credit card images. The left two columns are the Yolov3-tiny detection results, and the right two columns are the AutOCR detection results (we only draw the boxes that detect the names, and hide the boxes of the other 3 categories).

Faster R-CNN. Same as the previous experiment, we select 4 card images with different surfaces and colors to show the results of text detection and recognition. As shown in Figure 4, the first column on the left is the name detection results of Yolov3-tiny, the third column is the detection result of AutOCR. The second and fourth columns show the corresponding enlarged images. It can be seen that the detection result of AutOCR is more accurate. The Yolov3-tiny detection framework basically fails to detect the 1st image, and missed some text parts of the 3th and 4th images. ID from 5 to 8 in Table II shows the corresponding recognition results. The OCR system with Yolov3-tiny is partially unrecognizable on the 1st, 3th, and 4th images. AutOCR and manual OCR systems using Faster R-CNN, Mask R-CNN or Yolo v3 all accurately recognize these 4 images. This shows that AutOCR's automated search is effective and has a powerful performance.

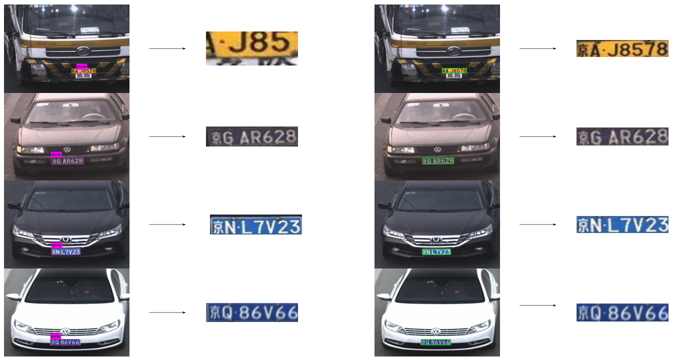
Fig. 5. Detection results on 4 vehicle images. The left two columns are the Yolov3-tiny detection results, and the right two columns are the AutOCR detection results.

3) Results on Vehicle Dataset : On vehicle dataset, systems are tested on 100 images. AutOCR is still the highest performing one. Compared to OCR systems using Mask R-CNN and Faster R-CNN, AutOCR is 4% and 1% higher, respectively. We show the results of 4 test images including different models

TABLE I
THE ACCURACY RESULT OVER THREE DATASETS.

	Backbone FLOPs	SVHN Acc	Credit Card Acc	Vehicle Acc
Faster R-CNN + Tesseract	380M	90.5%	90.0%	93.0%
Mask R-CNN + Tesseract	380M	92.0%	93.3	96.0%
AutOCR	300M	92.5%	96.7%	97.0%

TABLE II
RECOGNITION RESULTS ON 12 IMAGES IN 3 DATASETS.

Image ID	Faster R-CNN + Tesseract	Mask R-CNN + Tesseract	Yolo v3 + Tesseract	Yolov3-tiny + Tesseract	AutOCR
1	34	34	34	3A	34
2	6	6	6	6	6
3	7	7	7	7	7
4	20	20	20	20	20
5	MICHAEL	MICHAEL	MICHAEL	CHAE!	MICHAEL
6	MICHAEL	MICHAEL	MICHAEL	MICHAEL	MICHAEL
7	XIANG RI KUI	XIANG RI KUI	XIANG RI KUI	tANG RI KUI	XIANG RI KUl
8	XIANG RI KUI	XIANG RI KUI	XIANG RI KUI	XIANG RI KU!	XIANG RI KUl
9	J85 /¢	J8578	J8578	JOO,	J8578
10	AR628	AR628	AR628	AR628	AR628
11	L7V23	L7V23	L7V23	L7V23	L7V23
12	86V66	86V66	86V66	86V66	86V66

and colors. As shown in Figure 5, the left two columns are the detection results of Yolov3-tiny, and the right two columns are the results of AutOCR. ID 9 to 12 in Table II show the corresponding recognition results. It can be seen that AutOCR still performs excellent. The OCR system with Yolov3-tiny fails to recognize the 1st image, and some areas in the license plate are not detected. The OCR system with Faster R-CNN fails to recognize ”7” and ”8” in the 1st image. AutOCR and system with Mask R-CNN or Yolo v3 all correctly recognize 4 images. This further illustrates the advanced performance of AutOCR, which can achieve higher performance with manual OCR systems using advanced object detection frameworks.

C. Discussions

It can be seen from the Table I that the automated AutOCR has better performance and fewer backbone FLOPs than the manually designed OCR system used for comparison. Because AutOCR can search for a dedicated backbone network for the target task. This shows that the automated design OCR system not only has higher convenience but also can achieve better performance.

V. CONCLUSION

Although OCR has achieved excellent performance, most OCR systems are manually designed, which is very time-consuming and labor-intensive. In this paper, by introducing the NAS method, we propose a framework, AutOCR, for automatically designing OCR systems. AutOCR's high degree of automation can not only effectively reduce the workload of experts in designing OCR systems, but also can design different OCR systems according to the requirements of the target task. Our experiments show that AutOCR performs well and verify the effectiveness of our method. In addition, we also hope to combine this technique with dynamic multi-objective optimization [38]–[43] in future research.

REFERENCES

[1] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, “East: an efficient and accurate scene text detector,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 5551–5560.
[2] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu, “Deep direct regression for multi-oriented scene text detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 745–753.
[3] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, “Textboxes: A fast text detector with a single deep neural network,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
[4] Y. Liu and L. Jin, “Deep matching prior network: Toward tighter multi-oriented text detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1962–1969.
[5] R. Smith, “An overview of the tesseract ocr engine,” in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.
[6] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading text in the wild with convolutional neural networks,” *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.
[7] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.
[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[11] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," 2017.

[12] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," 2017.

[13] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017.

[14] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," 2018.

[15] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7036–7045.

[16] Y. Chen, T. Yang, X. Zhang, G. Meng, C. Pan, and J. Sun, "Detnas: Neural architecture search on object detection," *arXiv preprint arXiv:1903.10979*, 2019.

[17] H. Xu, L. Yao, W. Zhang, X. Liang, and Z. Li, "Auto-fpn: Automatic network architecture adaptation for object detection beyond classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6649–6658.

[18] B. Chen, G. Ghiasi, H. Liu, T.-Y. Lin, D. Kalenichenko, H. Adams, and Q. V. Le, "Mnasfpn: Learning latency-aware pyramid architecture for object detection on mobile devices," *arXiv preprint arXiv:1912.01106*, 2019.

[19] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," 2018.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[24] M. Jiang, Y. Ding, B. Goertzel, Z. Huang, C. Zhou, and F. Chao, "Improving machine vision via incorporating expectation-maximization into deep spatio-temporal learning," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 1804–1811.

[25] M. Jiang, Y. Yu, X. Liu, F. Zhang, and Q. Hong, "Fuzzy neural network based dynamic path planning," in *2012 International Conference on Machine Learning and Cybernetics*, vol. 1. IEEE, 2012, pp. 326–330.

[26] Z. Huang, Z. Zhong, L. Sun, and Q. Huo, "Mask r-cnn with pyramid attention network for scene text detection," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 764–772.

[27] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[28] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[29] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016.

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

[31] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[32] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," 2018.

[33] R. Luo, F. Tian, T. Qin, and T. Y. Liu, "Neural architecture optimization," 2018.

[34] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 82–92.

[35] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," *arXiv preprint arXiv:1904.00420*, 2019.

[36] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[37] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[38] W. Zhenzhong, M. JIANG, G. Xing, F. Liang, H. Weizhen, and K. C. TAN, "Evolutionary dynamic multi-objective optimization via regression transfer learning," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 2375–2381.

[39] C. Wang, G. G. Yen, and M. Jiang, "A grey prediction-based evolutionary algorithm for dynamic multiobjective optimization," *Swarm and Evolutionary Computation*, p. 100695, 2020.

[40] M. Jiang, W. Huang, Z. Huang, and G. G. Yen, "Integration of global and local metrics for domain adaptation learning via dimensionality reduction," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 38–51, 2015.

[41] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 501–514, 2017.

[42] C. Guokun, M. JIANG, G. Xing, H. Weizhen, G. Shihui, and K. C. TAN, "Online bagging for anytime transfer learning," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 941–947.

[43] H. Weizhen, M. Jiang, X. Gao, K. C. Tan, and Y.-m. Cheung, "Solving dynamic multi-objective optimization problems using incremental support vector machine," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 2794–2799.