AutoSTR: Efficient Backbone Search for Scene Text Recognition

Hui Zhang[1,2], Quanming Yao[2], Mingkun Yang[1], Yongchao Xu[1], Xiang Bai[1]

[1]Department of Electronics and Information Engineering,
Huazhong University of Science and Technology
[2]4Paradigm Inc.

Abstract. Scene text recognition (STR) is challenging due to the diversity of text instances and the complexity of scenes. However, no STR methods can adapt backbones to different diversities and complexities. In this work, inspired by the success of neural architecture search (NAS), we propose automated STR (AutoSTR), which can address the above issue by searching data-dependent backbones. Specifically, we show both choices on operations and the downsampling path are very important in the search space design of NAS. Besides, since no existing NAS algorithms can handle the spatial constraint on the path, we propose a two-step search algorithm, which decouples operations and downsampling path, for an efficient search in the given space. Experiments demonstrate that, by searching data-dependent backbones, AutoSTR can outperform the state-of-the-art approaches on standard benchmarks with much fewer FLOPS and model parameters. [1]

Keywords: Scene Text Recognition, Neural Architecture Search, Convolutional Neural Network, Automated Machine Learning

1 Introduction

Scene text recognition (STR) [28, 49], which targets at recognizing text from natural images, has attracted great interest from both industry and academia due to its huge commercial values in a wide range of applications, such as identity authentication, digital financial system, and vehicle license plate recognition [1, 9, 41], etc. However, natural images are diverse, the large variations in size, fonts, background and layout all make STR still a very challenge problem [38]. A STR pipeline (Fig.1) [38, 48] usually consists of three modules: a rectification module, which aims at rectifying irregular text image to a canonical form before recognition; a feature sequence extractor, which employs a stack of convolutional layers to convert the input text image to a feature sequence; and a feature translator module, which is adopted to translate the feature sequence into a character sequence.

[1] Correspondence authors are Quanming Yao and Xiang Bai, and code is available at https://github.com/AutoML-4Paradigm/AutoSTR.git.

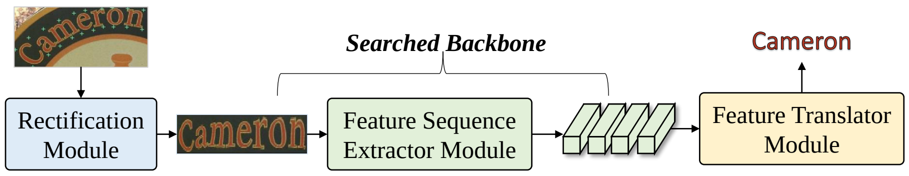


Fig. 1. Illustration of general structure of text recognition pipeline. Feature sequence extractor is searched in this paper.

In recent years, numerous methods [38, 43, 48] have successfully improved the text recognition accuracy via enhancing the performance of the rectification module. As for feature translator, inspired by some other sequence-to-sequence tasks, such as speech recognition [11] and machine translation [3], the translation module is also elaborately explored with both Connectionist Temporal Classification (CTC) based [36] and attention based methods [4, 7, 8, 38, 43]. In the contrast, the design of feature sequence extractor is relatively fewer explored. How to design a better feature sequence extractor has not been well discussed in the STR literature. However, text recognition performance can be greatly affected by feature sequence extractor. For example, the authors [38] can obtain significant performance gains by simply replacing feature extractor from vgg [39] to ResNet [14]. Furthermore, the feature sequence extractor bears heavy calculation and storage burden [21,43]. Thus, no matter for effectiveness or efficiency, the architecture of feature sequence extractor should be paid more attention to.

Besides, neural architecture search (NAS) [10,46] has also made a great success in designing data-dependent network architectures, of which the performances exceed the architectures crafted by human experts in many computer vision tasks, e.g., image classification [25, 32], semantic segmentation [24] and object detection [6]. Thus, rather than adopt an off-the-shelf feature extractor (like ResNet [14]) from other tasks, a data-dependent architecture should be redesigned for a better text recognition performance.

In this paper, we present the first work, i.e., AutoSTR, on searching feature sequence extractor (backbone) for STR. First we design a domain-specific search space for STR, which contains both choices on operation for every convolution layer and constraints feature downsampling path. Since no existing NAS algorithms can handle the path constraint efficiently, we propose a novel two-step search pipeline, which decouples operation and downsampling path search. By optimizing the recognition loss with complexity regularization, we achieve a trade off between model complexity and recognition accuracy. Elaborate experiments demonstrate that, given a general text recognition pipeline, the searched sequence feature extractor can achieve state-of-the-art results with fewer FLOPS and parameters. The main contributions of this paper are summarized as follows:

– We discover that the architecture of the feature extractor, which is of great importance to STR, has not been well explored in the literature. This mo-

tivates us to design a data-dependent backbone to boost text recognition performance, which is also the first attempt to introduce NAS into STR.
– We introduce a domain-specific search space for STR, which contains choices for downsampling path and operations. Then we propose a new search algorithm, which decouples operations and downsampling path for an efficient search in the space. We further incorporate an extra regularizer into the searching process, which helps effectively trade off the recognition accuracy with the model size.
– Finally, extensive experiments are carried on various benchmark datasets. Results demonstrate that, AutoSTR can discover data-dependent backbones, and outperform the state-of-the-art approaches with much fewer FLOPS and model parameters.

2 Related Works

2.1 Scene Text Recognition (STR)

As in Sec 1, the general pipline (Fig.1) of sequence-based STR methods [38,48], where a rectification module, a feature sequence extractor module and a feature translator module are involved. Currently, most works focus on improving rectification module or feature translator. Shi *et al.* [37,38] first introduce spatial transform network (STN) [17] for rectifying irregular text to a canonical form before recognition. Since then, [30,43,48] further push it forward and make the rectification module become a plug-in part. As for feature translator, CTC based and attention based decoder dominate this landscape for a long time [4,36,38]. Nevertheless, as another indispensable part, the feature sequence extractor has not been well discussed in the recent STR literature. As shown in Tab.1, although different feature extractors are used in [36,43], they just follow the architecture proposed for other fundamental tasks, like image classification. But recently, some authors observe that the architectures of feature extractor have gaps between different tasks, like semantic segmentation [24] and object detection [6]. Therefore, these popular feature extractor might not be perfectly suitable for STR, and it is important to search a data-dependent architecture.

2.2 Neural Architecture Search (NAS)

Generally, there are two important aspects in neural architecture search (NAS) [10,46], i.e., the *search space* and *search algorithm*. The search space defines possibilities of all candidate architectures, and the search algorithm attempts to find suitable architectures from the designed space. Specifically, a proper space should explore domain information and cover good candidates, which are designed by humans. Besides, it cannot be too large, otherwise, no algorithms can efficiently find good architectures. Thus, usually designing a better search space by domain knowledge can make the searching process easier.

Table 1. Comparison with example text recognition and NAS methods that contain a downsampling path search. "feat.seq.extractor", "DS", "seq. rec.", "cls.","seg." and "grad. desc." means feature sequence extractor, downsampling, sequence recognition, classification,segmentation and gradient descent algorithm respectively.

	model	**feat. seq. extractor**		**task**	**search**
		operation	**DS path**		algorithm
hand-designed	CRNN [36]	vgg	fixed	seq. rec.	—
	ASTER [38]	residual	fixed	seq. rec.	—
	SCRN [43]	residual	fixed	seq. rec.	—
NAS	DARTS [25]	searched	fixed	cls.	grad. desc.
	AutoDeepLab [24]	searched	one-dim	seg.	grad. desc.
	AutoSTR	searched	two-dim	seq. rec.	two-step

Classically, network architectures are treated as hyper-parameters, which are optimized by an algorithm like reinforcement learning [50] and evolution algorithms [42]. These methods are expensive since they need to train each sampled architecture fully. Currently one-shot neural architecture search (OAS) [5, 25, 32, 45] have significantly reduced search time by sharing the network weights during the search progress. Specifically, these methods encode the whole search space into a supernet, which consists of all candidate architectures. Then, instead of training independent weights for each architecture, distinctive architecture inherits weights from the supernet. In this way, architectures can be searched by training the supernet once, which makes NAS much faster. However, as the supernet is a representation of the search space, a proper design of it is a non-trivial task. Without careful exploitation of domain information, OAS methods may not be even better than random search [22, 35].

3 Methodology

3.1 Problem Formulation

As the input of STR is natural images, feature sequence extractor is constructed with convolutional layers. A convolutional layer $\mathcal{C}$ can be defined as $\mathcal{C}(X; o, s^h, s^w)$ (hereinafter referred to as $\mathcal{C}(X)$), where X is input tensor, o denotes convolution type, (e.g., 3×3 convolution, 5×5 depth-wise separable convolution), s^h and s^w represent its stride in height and width direction respectively. Therefore, a backbone $\mathcal{N}$ can be regarded as a stack of L convolution layers, i.e., $\mathcal{C}_L(\dots \mathcal{C}_2(\mathcal{C}_1(X)))$. After been processed by $\mathcal{N}$, X with input size (H, W) will be mapped into a feature map with a fixed size output to the feature translator module.

To determine a data-dependent backbone for STR, we need to search proper architectures, which is controlled by $\mathcal{S} \equiv \{(s_i^h, s_i^w)\}_{i=1}^L$ (for strides) and $\mathcal{O} \equiv \{o_i\}_{i=1}^L$ (for convolution type). Let $\mathcal{L}_{\text{tra}}$ measure the loss of the network on training dataset and $\mathcal{A}_{\text{val}}$ measure the quality of architecture on the validation set. We formulate the AutoSTR problem as:

$$\min_{\mathcal{S},\mathcal{O}} \mathcal{A}_{\text{val}}(\mathcal{N}(\boldsymbol{w}^*,\mathcal{S},\mathcal{O})), \quad \text{s.t.} \quad \begin{cases} \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}_{\text{tra}}(\mathcal{N}(\boldsymbol{w},\mathcal{S},\mathcal{O})) \\ \mathcal{S} \in \mathcal{P} \end{cases}, \tag{1}$$

where $\mathcal{S}$ and $\mathcal{O}$ are upper-level variables representing architectures, and $\boldsymbol{w}$ as lower-level variable and $\mathcal{P}$ as constraint, i.e.,

$$\mathcal{P} \equiv \{\{(s_i^h, s_i^w)\}_{i=1}^{L} \mid {}^{H}/\prod_{i=1}^{L} s_i^h = c_1, {}^{W}/\prod_{i=1}^{L} s_i^w = c_2\}.$$

Specifically, c_1 and c_2 are two application dependent constants; and the constraint $\mathcal{P}$ is to ensure the output size of the searched backbone is aligned with the input size of the subsequent feature translator module.

3.2 Search Space

As explained in Sec 2.2, the search space design is a key for NAS. Here, we design a two-level hierarchical search space for the AutoSTR problem in (1), i.e., the downsampling-path level and operation level, to represent the selection range of $\mathcal{S}$ and $\mathcal{O}$ respectively.

Downsampling-path level search space. Since the characters are horizontally placed in the rectified text image, following [36], we use CNN to exact a feature sequence to represent them. To reserve more discriminable features of the characters in compact text or in narrow shapes, a common way is to keep collapsing along the height axis until it reduces to 1, but compress less along the width axis to ensure that the length of final sequence is greater than the length of characters [36,38,48]. Specifically, the current mainstream methods are using the feature extractor proposed in ASTER [38]. The height of the input text image is unified to a fixed size, like 32. And to reserve more resolution along the horizontal axis in order to distinguish neighbor characters [4,36,38,43,48], the strides hyper-parameter s only selected from $\{(2,2),(2,1),(1,1)\}$, where $(2,2)$ appears twice and $(2,1)$ appears three times in the whole downsampling path to satisfy $\mathcal{P}$ with $\prod_{i=1}^{L} s_i^h = 32$ and $\prod_{i=1}^{L} s_i^w = 4$. Finally, a text image with size $(32,W)$ is mapped into a feature sequence with a length of ${}^{W}/4$.

The downsampling-path level search space is illustrated in Fig.2. Our goal is to find an optimal path in this 3D-mesh, and the downsampling strategy in [38] is a spatial case in this search space. To the best of our knowledge, there are no suitable methods to search in such a constrained search space.

Operation level search space. The convolutional layers of current text recognition networks usually share the same operation [7,38,48], such as 3×3 residual convolution. Instead of setting each o_i to be a fixed operation, we select a operator for each convolutional layer from a *choice block* with C parallel operators, as illustrated in the bottom of Fig.2. Then, we can obtain a deep convolutional network by stacking these *choice blocks*. Our choices on operations are inspired by

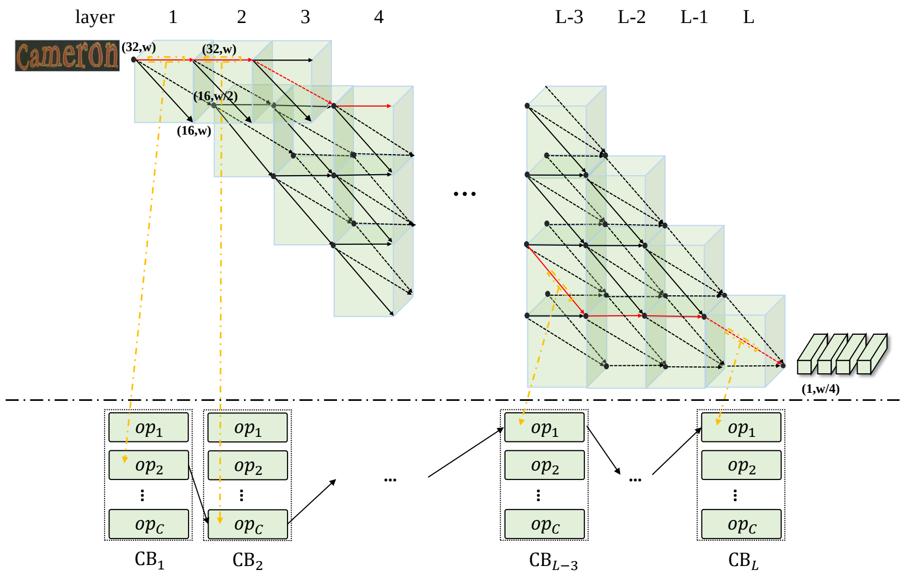


Fig. 2. Search space illustration. Top: a 3D-mesh representation of the downsampling-path level search space. Bottom: operation level search space, where "CB_i" donates the ith *choice block* and each block allows C choices for operations.

MobileNetV2 [34], which uses lightweight depthwise convolutions to save FLOPS and the number of parameters. Thus, we build a set of mobile inverted bottleneck convolution layers (MBConv) with various kernel sizes $k \in \{3, 5\}$, expansion factors $e \in \{1, 3, 6\}$ and a skip-connect layer.

Table 2. Basic operation (i.e., op_i's) in the choice block (the bottom of Fig.2), where "k" denotes kernel size and "e" denotes expansion factors.

MBConv(k:3×3,e:1)	MBConv(k:3×3,e:3)	MBConv(k:3×3,e:6)	MBConv(k:5×5,e:1)
MBConv(k:5×5,e:3)	MBConv(k:5×5,e:6)	Skip-Connect	

Complexity of the search space. When $L = 15$, there are 30030 possible downsampling paths in search space illustrated in Fig.2. On operation level, if we allow it to be one of the seven operations in Tab.2, then it leads to a total number $30030 \times 7^{15} \simeq 1.43 \times 10^{17}$ possible architectures for the backbone, which is prohibitively large. In the sequel, we show a two-step algorithm which can efficiently search through the space.

3.3 Search Algorithm

Since the combination of $\mathcal{S}$ and $\mathcal{O}$ generates a very space, directly optimizing the problem in (1) is a huge challenge. Motivated by the empirical observation that choices of the downsampling path and the operations are almost orthogonal with each other (see Sec 4.6), in this section, we decouple the process of searching $\mathcal{S}$ and $\mathcal{O}$ into two steps for the backbone search. Specifically, in the first step, we fix the operation $\mathcal{O}$ as the 3×3 residual convolution (denote as $\hat{\mathcal{O}}$) and search for the downsampling path. In the second step, we search a convolution operation for every layer in the path.

Step 1: search downsampling path. Let $\mathcal{L}_{\text{rec}}(\mathcal{N};\mathcal{D})$ measure the sequence cross-entropy loss [38] with predictions from a network $\mathcal{N}$ on a dataset $\mathcal{D}$, and $\mathcal{D}_{\text{tra}}$ (resp. $\mathcal{D}_{\text{val}}$) denotes the training (resp. validation) dataset. In this step, we search downsampling path when operations are fixed, and (1) becomes

$$\mathcal{S}^* = \arg\min_{\mathcal{S}\in\mathcal{P}} \mathcal{L}_{\text{rec}}(\mathcal{N}(\boldsymbol{w}^*, \mathcal{S}, \hat{\mathcal{O}}); \mathcal{D}_{\text{val}}), \tag{2}$$
$$\text{s.t. } \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}_{\text{rec}}(\mathcal{N}(\boldsymbol{w}, \mathcal{S}, \hat{\mathcal{O}}); \mathcal{D}_{\text{tra}}).$$

As in Sec 3.2, downsampling path can only have two $(2,2)$ and three $(2,1)$ to satisfy $\mathcal{P}$, which are denoted as downsampling type A and B respectively. Note that some current NAS methods [25, 45] use the same number of layers per convolution stage and have achieved good results. Using this reasonable prior knowledge, for a network with $L = 15$, we set downsampling at layers 1, 4, 7, 10, 13, and equally separate the network into five stages. Then the downsampling strategies can be divided into 10 types of typical paths: AABBB, ABABB, ABBAB, ABBBA, BAABB, BABAB, BABBA, BBAAB, BBABA, and BBBAA. We can do a small grid search in these typical paths to find a good path that is close to $\mathcal{S}^*$. Then by learning the skip-connect in searching step 2, we can reduce the number of layers for each convolutional stage.

Step 2: search operations. First, inspired by recent NAS methods [5, 12] we associate the operation op_i^l at the lth layer with a hyperparameter α_i^l, which relaxes the categorical choice of a particular operation in the *choice block* (Fig.2) to be continuous. Since op_i^l's influence both complexity and performance of the backbone [5, 34, 45], we introduce a regularizer, i.e.,

$$r(\boldsymbol{\alpha}) = [\log(\sum_{l=1}^{L}\sum_{j=1}^{C} \text{FLOPS}(\text{op}_j^l)) \cdot \alpha_j^l / \log \mathcal{G}]^{\beta}, \tag{3}$$

on the architecture $\boldsymbol{\alpha}$, where $\beta > 0$ and $\mathcal{G} > 1$ be application-specific constants. Then, (1) is transformed as

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} r(\boldsymbol{\alpha}) \cdot \mathcal{L}_{\text{rec}}(\mathcal{N}(\boldsymbol{w}^*, \mathcal{S}^*, \boldsymbol{\alpha}), \mathcal{D}_{\text{val}}), \tag{4}$$
$$\text{s.t. } \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}_{\text{rec}}(\mathcal{N}(\boldsymbol{w}, \mathcal{S}^*, \boldsymbol{\alpha}); \mathcal{D}_{\text{tra}}).$$

We can see that when $\mathcal{L}_{\text{rec}}$ is the same, $r(\boldsymbol{\alpha})$ make architectures with less FLOPS favored. Thus, the regularizer can effectively trade off the accuracy with the model size. Finally, (4) can be solved by many existing NAS algorithms, such as DARTS [25], ProxylessNAS [5] and NASP [45]. Here, we adopt ProxylessNAS [5] as it consumes the GPU memory less.

3.4 Comparison with other NAS works

The search constraint $\mathcal{P}$ on the downsampling path for is new to NAS, which is specific to STR. Unfortunately, none of existing NAS algorithms can effectively deal with the AutoSTR problem in (1) here. It is the proposed search space (Sec 3.2) and two-stage search algorithm (Sec 3.3) that make NAS for STR possible (see Tab.1). We notice that AutoDeepLab [24] also considers the downsampling path. However, it targets at in segmentation task and its unique decoding method is not suitable for our search space.

4 Experiments

4.1 Datasets

We evaluate our searched architecture on the following benchmarks that are designed for general STR. Note that the images in the first four datasets are regular while others are irregular.

– IIIT 5K-Words (**IIIT5K**) [31]: it contains 5,000 cropped word images for STR, 2,000 for validation and other 3,000 images for testing; all images are collected from the web.
– Street View Text (**SVT**) [40]: It is harvested from Google Street View. Its test set contains 647 word images, which exhibits high variability and often has low resolution. Its validation set contains 257 word images.
– ICDAR 2003 (**IC03**) [29]: it contains 251 full scene text images. Following [40], we discard the test images containing non-alphanumeric characters or have less than three characters. The resulting dataset contains 867 cropped images for testing and 1,327 images as validation dataset.
– ICDAR 2013 (**IC13**) [19]: it contains 1,015 cropped text images as test set and 844 images as validation set.
– ICDAR 2015 (**IC15**): it is the 4th Challenge in the ICDAR 2015 Robust Reading Competition [18], which is collected via Google Glasses without careful positioning and focusing. As a result, the dataset consists of a large proportion of blurred and multi-oriented images. It contains 1811 cropped images for testing and 4468 cropped text images as validation set.
– SVT-Perspective (**SVTP**): it is proposed in [33] which targets for evaluating the performance of recognizing perspective text and contains 645 test images. Samples in the dataset are selected from side-view images in Google Street View. Consequently, a large proportion of images in the datasets are heavily deformed by perspective distortion.

4.2 Implementation Details

The proposed method is implemented in PyTorch. We adopt ADADELTA [47] with default hyper-parameters (rho=0.9, eps=1e-6, weight decay=0) to minimize the objective function. When searching the downsampling path, we train 5 epochs for each typical path where the convolutional layers are equipped with default 3×3 convolution. In the operation searching step, we warm-up weights of each choice block by uniformly selecting operations for one epoch. And then use the method proposed in Sec 3.3 to jointly train architecture parameters and weight parameters for two epochs. In the evaluation step, the searched architectures are trained on Synth90K [15] and SynthText [13] from scratch without finetuning on any real datasets. All models are trained on 8 NVIDIA 2080 graphics cards. Details of each module in Fig.1 are as follows:

Rectification module. Following [38], we use Spatial Transformer Network (STN) to rectify the input text image. The STN contains three parts: (1) Localization Network, which consists of six 3×3 convolutional layers and two fully connected layers. In this phase, 20 control points are predicted. Before fed into the localization network, the input image is resized to 32×64. (2) Grid Generator, which yields a sampling grid according to the control points. (3) Sampler, which generates a rectified text image with the sampling grid. The sampler produces a rectified image of size 32×100 from the input image of size 64×256 and sends it to the subsequent sequence feature extractor.

Feature sequence extractor module. The feature sequence extractor dynamically changes during the search phase. For every test dataset, an architecture is searched. During the search phase, Synth90K [15] (90k) and SynthText [13] (ST) are used for training. The validation set of each target dataset is considered as the search validation set and is used to optimize the network structure. In order to prevent overfitting caused by too small validation set, we add extra COCO-Text training set to the search validation set. To control the complexity of the backbone, we only search those with less than 450M FLOPS which is close to that of the backbone used in ASTER. The maximum depth of our search network is 16 blocks, including a stem with 3×3 residual convolution and 15 choice blocks. It has a total of 5 convolutional stages, each stage has 3 choice blocks, the number filters of each stage are respectively 32, 64, 128, 256, 512 respectively.

Feature translator module. A common attention based decoder [38] is employed to translate the feature sequence to a character sequence. For simplicity, only left-to-right decoder is used. The module contains two layers of Bidirectional LSTM (BiLSTM) encoder (512 input units, 256 hidden units) and an attention based GRU Cell decoder (1 layer, 512 input units, 512 hidden units, 512 attention units) to model variable-length character sequence. The decoder yields 95 character categories at each step, including digits, upper-case and lower-case letters, 32 ASCII punctuation marks and an end-of-sequence symbol (EOS).

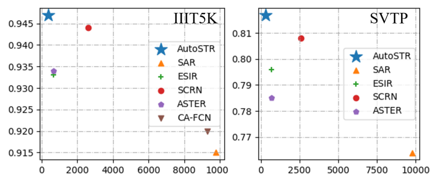
Table 3. Performance comparison on regular and irregular scene text datasets. "ST", "90k" are the training data of SynthText [27], Synth90k [15], and "extra" means extra real or synthetic data, respectively. The methods marked with "†" use the character box annotations. "ASTER (ours)" is the reproduced ASTER baseline method, whose difference is that only left-to-right translator is equipped.

Methods	**Data**	**Regular**				**Irregular**	
		IIIT5K	**SVT**	**IC03**	**IC13**	**SVTP**	**IC15**
Jaderberg et al. [16]	90k	-	80.7	93.1	90.8	-	-
CRNN [36]	90k	81.2	82.7	91.9	89.6	-	-
RARE [37]	90k	81.9	81.9	90.1	88.6	71.8	-
R^2AM [20]	90k	78.4	80.7	88.7	90.0	-	-
Yang et al. [44]	90k	-	-	-	-	75.8	-
Char-net [26]	90k	83.6	84.4	91.5	90.8	73.5	-
Liu et al [27]	90k	89.4	87.1	<u>94.7</u>	94.0	73.9	-
AON [8]	ST+90k	87.0	82.8	91.5	-	73.0	68.2
FAN† [7]	ST+90k	87.4	85.9	94.2	93.3	-	70.6
EP [4]	ST+90k	88.3	87.5	94.6	**94.4**	-	73.9
SAR [21]	ST+90k	91.5	84.5	-	91.0	76.4	69.2
CA-FCN† [23]	ST+extra	92.0	86.4	-	91.5	-	-
ESIR [48]	ST+90k	93.3	<u>90.2</u>	-	91.3	79.6	76.9
SCRN† [43]	ST+90k	<u>94.4</u>	88.9	**95.0**	93.9	<u>80.8</u>	<u>78.7</u>
ASTER [38]	ST+90k	93.4	89.5	94.5	91.8	78.5	76.1
ASTER (ours)	ST+90k	93.3	89.0	92.4	91.5	79.7	78.5
AutoSTR	ST+90k	**94.7**	**90.9**	93.3	<u>94.2</u>	**81.7**	**81.8**

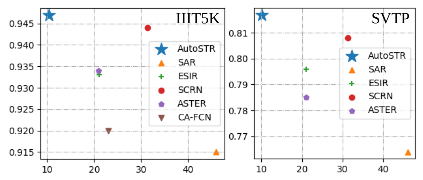
4.3 Comparison with State of the Art

Recognition accuracy. Following [2], all related works are compared in the unconstrained-lexicon setting. Equipped with the searched backbone, the whole framework is compared with other state-of-the-art methods, as shown in Tab.3. AutoSTR achieves the best performance in IIIT5K, SVT, IC15, SVTP and get comparable results in IC03, IC13. It is worth noting that AutoSTR outperforms ASTER (ours) on IIIT5K, SVT, IC03, IC13, SVTP, IC15 by 1.4%, 1.9%, 0.9%, 2.7%, 2%, 3.3%, which domonstrate the effectiveness of AutoSTR. Although SCRN can achieve comparable performance with AutoSTR, its rectification module requires extra character-level annotations for more precise rectification. As a plug-in part, AutoSTR is expected to further improve the performance while been equipped with the rectification module of SCRN.

Memory and FLOPS. The comparison on FLOPS and memory size are in Fig.3. We can see that, compared with the state-of-the-art methods, like SAR [21], CA-FCN [23], ESIR [48], SCRN [43], ASTER [38], the searched architecture cost much less in FLOPS and memory size. Thus, AutoSTR is much more effective in mobile setting, where FLOPS and model size is limited.



(a) FLOPS (million).



(b) Number of parameters (million).

Fig. 3. Accuracy versus computational complexity and memory on IIIT5K and SVTP. Points closer to the top-left are better. Only methods with good performance, i.e., with accuracy greater than 90% on IIIT5K, are plotted.

4.4 Case Study: Searched Backbones and Discussion

Dataset-dependency. In Fig.4, we illustrate architectures of searched feature extractors on each test dataset to give some insights about the design of backbones. We observe some interesting patterns. The shallower convolutional stages (e.g., 1, 2) of the network, prefer larger MBConv operations (e.g., MBConv(k:5,e:6)) and do not have skip-connect layer. But in the deeper convolutional stages (e.g., 3, 4, 5), smaller MBConvs are employed and skip connections are learned to reduce the number of convolutional layers. Especially in the last convolutional stage, only one convolutional layer exists. The observed phenomenon is consistent with some manually designed network architecture, such as SCRN [43]. Specifically, in its first two stages, ResNet50 is used to extract features, and in the later stages, only a few convolutional layers are attached to quickly downsample feature map to generate feature sequences in the horizontal direction. This phenomenon may inspire us to design better text image feature extractor in the future.

Compactness. We compare our searched architectures with All MBConv(k:5,e:6) baseline model which choices blocks with the maximum number of parameters in each layer and uses ABABB downsampling strategy as shown in the right of Fig.4. Comparing architectures in Fig.4, we can see that our searched structure has less FLOPS and parameters, while maintain better accuracy, as shown in Tab.4. Our searched architectures use less FLOPS and parameters, but exceeds the accuracy of the baseline model, which explains that the maximum number of parameters model (All MBConv(k:5,e:6) baseline) have lots of redundancy parameters, AutoSTR can remove some redundant layers and optimize the network structure.

Table 4. Accuracies compared with the baseline of All MBConv(k:5,e:6).

Methods	IIIT5K	SVT	IC13	IC15
All MBConv(k:5,e:6)	94.5	90.4	92.3	81.1
AutoSTR	94.7	90.9	94.2	81.8

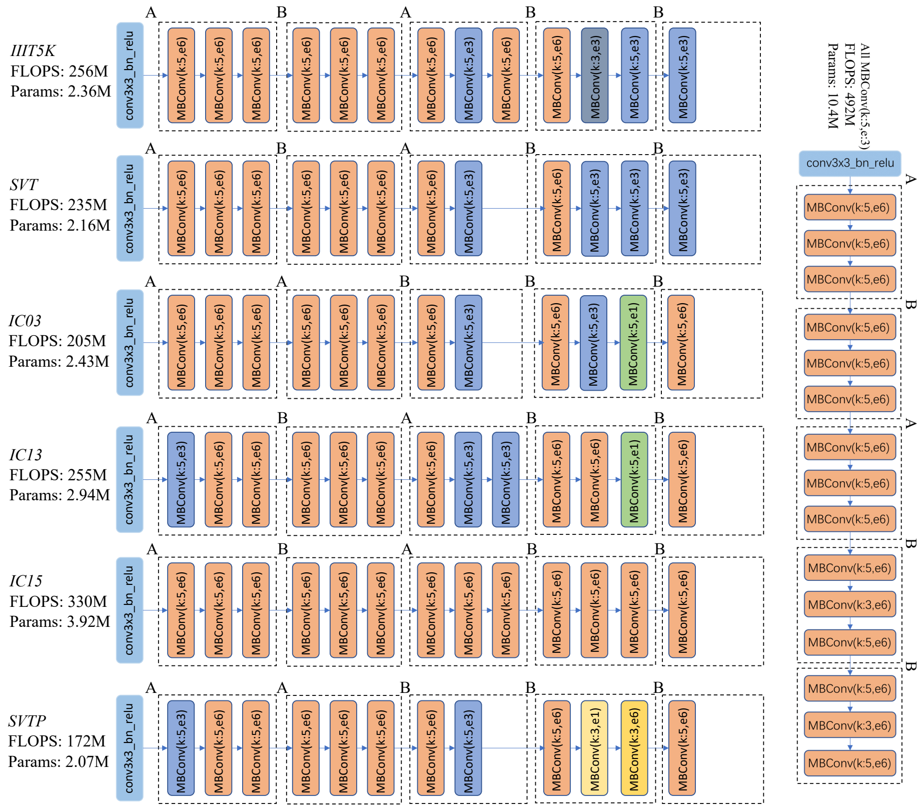


Fig. 4. Left: the searched architectures for AutoSTR in Tab.3. Right: all MBConv(k:5,e:6) baseline architecture.

4.5 Comparison with Other NAS Approaches

Search algorithm comparison. From recent benchmarks and surveys [6, 22, 35], the random search algorithm is a very strong baseline, we perform a random architecture search from the proposed search space. We choice 10 random architectures, train them from scratch, then test these random architectures on IIIT5K dataset. Random search takes about 15 × 4GPU days, while AutoSTR only costs 1.7 × 4GPU days in downsampling-path search step and 0.5 × 4GPU days in operation search step. The discovered architecture outperforms random architectures in IIIT5K dataset by 0.5%-1.4% as in Fig.5, which demonstrates AutoSTR is more effectiveness and efficiency.

Reusing other searched architectures. NAS has been extensive researched on image classification task [5, 25, 45] and segmentation task [24], et al. We study whether those searched architectures are applicable here or not. Since the

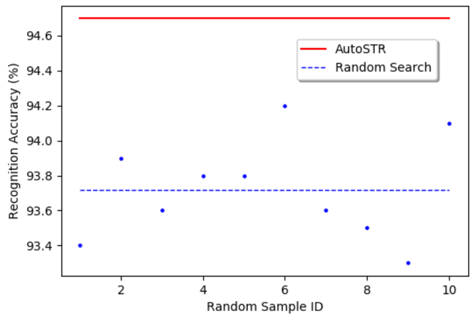


Fig. 5. Comparison to random search on IIIT5K dataset.

constraint in (1) cannot be directly satisfied, we manually tune the searched cell (from DARTS [25] and AutoDeepLab [24]) in ASTER. As can be seen from Tab.5, the performance of the backbone from DARTS and AutoDeepLab is much worse. This further demonstrates direct reusing architecture searched from other tasks is not good.

Table 5. Comparison with DARTS and AutoDeepLab.

backbones	**IIIT5K**	**SVT**	**SVT**	**IC13**	**SVTP**	**IC15**
ASTER [38]	93.3	89.0	92.4	91.5	79.7	78.5
DARTS [25]	90.6	83.9	91.3	88.3	76.1	73.5
AutoDeepLab [24]	93.0	87.2	91.8	91.0	77.5	76.6
AutoSTR	**94.7**	**90.9**	**93.3**	**94.2**	**81.7**	**81.8**

4.6 Ablation Study

Downsampling path. In our proposed method, we decouple the searching problem in (1) into a two-step optimization problem as (2) and (4). This is based on an empirical assumption that a better feature downsampling path can provide a better startup for the operation searching problem, thus can get better architectures easier. We use two typical strategies in our downsampling path search space, i.e., AABBB and ABABB to search operations on IIIT5K datasets. As shown in Tab.6, the optimal downsampling path will not be affected by the default operation (i.e. 3×3 residual convolution, MBConv(k:3,e:1)). Besides, a better downsampling strategy (i.e. ABABB) helps AutoSTR to find a better architecture in the operation search step, which confirms our assumption.

Table 6. Comparison of different downsampling path on IIIT5K dataset.

Downsample Path	**Default Conv**	**Search Step 1**	**Search Step 2**
AABBB	3x3 residual conv	92.5	93.9
	MBConv(k:3,e:1)	91.3	
ABABB	3x3 residual onv	**93.1**	**94.7**
	MBConv(k:3,e:1)	**92.0**	

Impact of the regularizer. In (3), we introduce FLOPS into objective function as a regularization term. By adjusting β, we can achieve the trade off between the calculation complexity and accuracy, as shown in Tab.7.

Table 7. Impact of the regularization on IIIT5K dataset.

β	0.0	0.3	0.6	0.9
Accuracy (%)	94.6	94.5	94.7	93.5
FLOPS (M)	319	298	256	149
Params (M)	3.82	3.40	2.36	1.32

5 Conclusion

In this paper, we propose to use neural architecture search technology finding data-dependent sequence feature extraction, i.e., the backbone, for the scene text recognition (STR) task. We first design a novel search space for the STR problem, which fully explore the prior from such a domain. Then, we propose a new two-step algorithm, which can efficiently search the feature downsampling path and operations separately. Experiments demonstrate that our searched backbone can greatly improve the capability of the text recognition pipeline and achieve the state-of-the-art results on STR benchmarks. As for the future work, we would like to extend the search algorithm to the feature translator.

Acknowledgments

The work is performed when H. Zhang was an intern in 4Paradigm Inc. mentored by Dr. Q. Yao. This work was partially supported by National Key R&D Program of China (No. 2018YFB1004600), to Dr. Xiang Bai by the National Program for Support of Top-notch Young Professionals and the Program for HUST Academic Frontier Youth Team 2017QYTD08.

References

1. Almazán, J., Gordo, A., Fornés, A., Valveny, E.: Word spotting and recognition with embedded attributes. IEEE Transactions on Pattern Analysis and Machine Intelligence (2014)
2. Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S.J., Lee, H.: What is wrong with scene text recognition model comparisons? dataset and model analysis. In: International Conference on Computer Vision (2019)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. Tech. rep., arXiv preprint arXiv:1409.0473 (2014)
4. Bai, F., Cheng, Z., Niu, Y., Pu, S., Zhou, S.: Edit probability for scene text recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
5. Cai, H., Zhu, L., Han, S.: ProxylessNAS: Direct neural architecture search on target task and hardware. In: International Conference on Learning Representations (2019)
6. Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., Sun, J.: DetNAS: Backbone search for object detection. In: Advances in Neural Information Processing Systems (2019)
7. Cheng, Z., Bai, F., Xu, Y., Zheng, G., Pu, S., Zhou, S.: Focusing attention: Towards accurate text recognition in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
8. Cheng, Z., Xu, Y., Bai, F., Niu, Y., Pu, S., Zhou, S.: Aon: Towards arbitrarily-oriented text recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
9. Dutta, K., Mathew, M., Krishnan, P., Jawahar, C.: Localizing and recognizing text in lecture videos. In: International Conference on Frontiers in Handwriting Recognition (2018)
10. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. Journal of Machine Learning Research (2019)
11. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: International Conference on Machine learning (2006)
12. Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J.: Single path one-shot neural architecture search with uniform sampling. Tech. rep., arXiv preprint arXiv:1904.00420 (2019)
13. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2315–2324 (2016)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
15. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. Tech. rep., arXiv preprint arXiv:1406.2227 (2014)
16. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. International Journal of Computer Vision (2016)
17. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in Neural Information Processing Systems (2015)
18. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., et al.: Icdar 2015

competition on robust reading. In: International Conference on Document Analysis and Recognition (2015)
19. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., i Bigorda, L.G., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., De Las Heras, L.P.: Icdar 2013 robust reading competition. In: International Conference on Document Analysis and Recognition (2013)
20. Lee, C.Y., Osindero, S.: Recursive recurrent nets with attention modeling for OCR in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
21. Li, H., Wang, P., Shen, C., Zhang, G.: Show, attend and read: A simple and strong baseline for irregular text recognition. In: AAAI Conference on Artificial Intelligence (2019)
22. Li, L., Talwalkar, A.: Random search and reproducibility for neural architecture search. In: Uncertainty in Artificial Intelligence (2019)
23. Liao, M., Zhang, J., Wan, Z., Xie, F., Liang, J., Lyu, P., Yao, C., Bai, X.: Scene text recognition from two-dimensional perspective. In: AAAI Conference on Artificial Intelligence (2019)
24. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
25. Liu, H., Simonyan, K., Yang, Y.: DARTS: differentiable architecture search. In: International Conference on Learning Representations (2019)
26. Liu, W., Chen, C., Wong, K.Y.K.: Char-net: A character-aware neural network for distorted scene text recognition. In: AAAI Conference on Artificial Intelligence (2018)
27. Liu, Y., Wang, Z., Jin, H., Wassell, I.: Synthetically supervised feature learning for scene text recognition. In: European Conference on Computer Vision (2018)
28. Long, S., He, X., Yao, C.: Scene text detection and recognition: The deep learning era. Tech. rep., arXiv preprint arXiv:1811.04256 (2018)
29. Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R., Ashida, K., Nagai, H., Okamoto, M., Yamamoto, H.: Icdar 2003 robust reading competitions: entries, results, and future directions. International Journal of Document Analysis and Recognition (2005)
30. Luo, C., Jin, L., Sun, Z.: MORAN: A multi-object rectified attention network for scene text recognition. Pattern Recognition (2019)
31. Mishra, A., Alahari, K., Jawahar, C.: Top-down and bottom-up cues for scene text recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
32. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. In: International Conference on Machine Learning (2018)
33. Quy Phan, T., Shivakumara, P., Tian, S., Lim Tan, C.: Recognizing text with perspective distortion in natural scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (2013)
34. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
35. Sciuto, C., Yu, K., Jaggi, M., Musat, C., Salzmann, M.: Evaluating the search phase of neural architecture search. In: International Conference on Learning Representations (2020)

36. Shi, B., Bai, X., Yao, C.: 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 (2017)
37. Shi, B., Wang, X., Lyu, P., Yao, C., Bai, X.: Robust scene text recognition with automatic rectification. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
38. Shi, B., Yang, M., Wang, X., Lyu, P., Yao, C., Bai, X.: Aster: An attentional scene text recognizer with flexible rectification. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
39. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Tech. rep., arXiv preprint arXiv:1409.1556 (2014)
40. Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: International Conference on Computer Vision (2011)
41. Xie, F., Zhang, M., Zhao, J., Yang, J., Liu, Y., Yuan, X.: A robust license plate detection and character recognition algorithm based on a combined feature extraction model and bpnn. Journal of Advanced Transportation (2018)
42. Xie, L., Yuille, A.: Genetic CNN. In: IEEE International Conference on Computer Vision (2017)
43. Yang, M., Guan, Y., Liao, M., He, X., Bian, K., Bai, S., Yao, C., Bai, X.: Symmetry-constrained rectification network for scene text recognition. In: IEEE International Conference on Computer Vision (2019)
44. Yang, X., He, D., Zhou, Z., Kifer, D., Giles, C.L.: Learning to read irregular text with attention mechanisms. In: International Joint Conferences on Artificial Intelligence (2017)
45. Yao, Q., Xu, J., Tu, W.W., Zhu, Z.: Efficient neural architecture search via proximal iterations. In: AAAI Conference on Artificial Intelligence (2020)
46. Yao, Q., Wang, M.: Taking human out of learning applications: A survey on automated machine learning. Tech. rep., arXiv preprint arXiv:1810.13306 (2018)
47. Zeiler, M.: Adadelta: an adaptive learning rate method. Tech. rep., arXiv preprint arXiv:1212.5701 (2012)
48. Zhan, F., Lu, S.: ESIR: End-to-end scene text recognition via iterative image rectification. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
49. Zhu, Y., Yao, C., Bai, X.: Scene text detection and recognition: Recent advances and future trends. Frontiers of Computer Science (2016)
50. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: International Conference on Learning Representations (2017)