Memory-Efficient Models for Scene Text Recognition via Neural Architecture Search

SeulGi Hong*
hutom, KAIST
seulgihong@kaist.ac.kr

DongHyun Kim*
Clova AI Research, NAVER Corp.
dong.hyun@navercorp.com

Min-Kook Choi
hutom
mkchoi@hutom.io

Abstract

Meta-learning techniques based on neural architecture search (NAS) show excellent performance in the design of learning models used in deep neural networks. In particular, when NAS is applied to design a convolutional neural network (CNN) for image recognition, the performance of the network when evaluating public benchmark datasets such as CIFAR10 and ImageNet exceeds that of hand-designed models. Nevertheless, there are very few cases wherein NAS has been applied to real-world problems, i.e. recognition problems with a limited dataset. We proposed a method in which the NAS technique does not require a proxy task for the scene text recognition (STR) framework to apply the NAS method to a new image recognition field. Therefore, we proposed an architecture space for CNN-based modules in the STR framework and applied the ProxylessNAS method, enabling end-to-end training while meta learners design a new model that requires only a single commonly used GPU (approximately 100 GPU hours). To evaluate the STR model obtained by the proposed NAS method, seven STR benchmark datasets were used. Finally, the obtained model could achieve a performance similar to that of the ideal model in terms of accuracy and number of parameters. We thus confirm that the model design based on NAS can be effectively applied to STR scenarios.

1. Introduction

Recently, owing to the neural architecture search (NAS), there has been a significant change in the design procedure of recognition models based on deep neural networks. The NAS defines an *architecture space*, which is a set of operations and parameters, that can operate efficiently for a specific task, instead of designing building blocks of the network or creating its architectural structures based on intuition and experience. NAS then automatically builds the model according to architectural space, specific search strategies, and evaluation metrics. In the early stages of NAS research, it was mainly limited to a few tasks such as image classification and language modeling with small datasets [37, 38]. The application field of the method is, however, expanding to object detection, semantic segmentation model design, and training with large-scale data [17, 8].

*indicates equal contribution

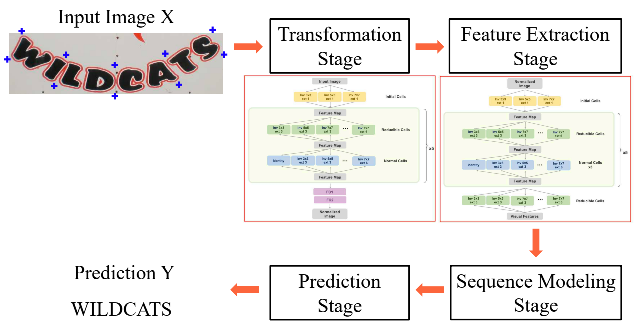


Figure 1. **STR framework proposed by neural architecture search (NAS).** Red boxes present the architecture space for CNNs to apply NAS without any proxy task.

The architecture search techniques proposed at the beginning of NAS were mainly based on reinforcement learning or evolutionary algorithms, requiring a large computing power to search all spaces [37]. The final model obtained with the NAS was able to achieve a performance comparable to the network designed by humans; however, it could not be applied to large-scale data as the training time to evaluate one candidate model was too extensive. After the optimal structure was generated for proxy tasks, the architecture search for large-scale datasets was achieved by repeatedly stacking some structures of the proxy model in a dataset of similar nature. Nevertheless, the architecture search, which requires a long processing time to be completed, still faces problems when being applied to practical application.

Recent algorithms, such as ENAS [24], One-Shot [3], DARTS [19], and ProxlessNAS [5] however, have been proposed to dramatically reduce the search cost of the architecture space or optimize the search without any proxy dataset.

Based on these methods, the computing power of the search is reduced, enabling real-world applications, and achieving the high accuracy for various public datasets. Despite these developments, NAS-based models target applications in only limited domains. As a result, the performance of most of them is evaluated merely on representative datasets of image classification and language modeling. To overcome this problem, NAS has been studied in various applications, such as object detection [8], semantic segmentation [17], and medical image segmentation [36, 35]; however, there is still a need to utilize NAS techniques in more real-world scenarios.

To extend the field of application of the NAS-based model design, we propose a NAS methodology for a scene text recognition (STR) case, one of the typical applications of computer vision. STR is an extension of optical character recognition (OCR) and is used to recognize text sequences in natural images. The framework based on deep learning networks for STR consists of four steps. The procedure consists of a transformation stage, in which the deformation of the text in the input image is minimized, feature extraction stage, for extracting features associated with characters from a rectified image, a sequence modeling stage, for obtaining contextual information from the extracted image features, and prediction stage, to recognize the sequence features.

We define the architecture spaces for CNN-based modules to be applied to the transformation and feature extraction stages in the design of the deep neural network modules for STR, which are capable of end-to-end training, to apply NAS techniques. Figure 1 shows the pipeline of our NAS-based STR framework. For each CNN module, specific search, in which each architecture space is defined, and a gradient-based search are used. Both methods do not depend on any proxy dataset. Finally, the selected model was able to achieve the accuracy and memory efficiency that is comparable to state-of-the-art (SOTA) models, enabling effective meta learner training merely on a single, commonly used GPU hardware with approximately 100 GPU hours of inexpensive computing resources.

The key contributions of this study are as follows. 1) Expand the application field of NAS in real-world scenarios, by applying the architecture search for the STR framework. 2) Design an automated STR model capable of end-to-end training, and an architecture space based on the inverted residual block of the CNN module was designed to create a model with high memory efficiency and accuracy. 3) Apply meta-learner training technique without a proxy task, enabling meta-learner to directly find well-adapted modules on STR scenario, which is a new type of vision application. As a result, we obtained an optimized model within 100 GPU hours on a single commonly used GPU. Finally, the selected model achieved a similar accuracy and memory efficiency as the SOTA models on seven STR benchmark datasets.

2. Related Research

Neural Architecture Search. The search algorithms proposed at the startup stage of the NAS technique were based on reinforcement learning or evolutionary algorithms [37, 38, 4, 24, 18]. In [37, 38], the authors proposed a method of training a meta controller that defines the search space of the cell for basic operations and selects the appropriate operation for each cell, allowing an efficient meta-learning process. In this case, the meta controller updates the parameters of a candidate model at a specific cycle consisting of learning, evaluation, and updating. This method executes a dense search of the search space; however, a very large computing resource is required. Furthermore, extensive time is needed to obtain sufficient information to update the meta controller. To compensate these drawbacks, the authors in [38] searched for submodules of architectures using proxy tasks in similar domains and applied them to a large-scale dataset through an iterative structure of submodules. This type of NAS, however, generates a design structure that depends on the proxy task. As a result, the proxy dataset needs to be redefined every time the task type is changed.

We applied the architecture search technique using the ProxylessNAS method [5], of which training with a proxy task is not required, to apply NAS directly to the target task. In doing so, the meta learner can find the model considering the entire pipeline of STR. ProxylessNAS generates an over-parameterized network with all candidate paths for the architecture search. Thereafter, it performs a binarized architecture parameter search and training by path-level pruning, achieving an efficient training and accurate model search regardless of any proxy task. In this study, we trained the network based on the architecture space with various topologies of STR module. The training that did not require any proxy task and was conducted on a commonly used GPU through binarized architecture training and path-level pruning.

Scene Text Recognition. Initially, recognizing the characters was the main objective of optical character recognition (OCR). The method started from a simple classification problem, in which an edited number or letter was recognized [15, 23], and has advanced to be able to recognize all characters on cleaned documents [26]. Traditional OCR algorithms, however, do not work well for STR in the real world, as the character recognition is assumed to be conducted in preprocessed clean documents. To solve this problem, the STR framework for recognizing text in a natural scene has been actively studied [29, 33, 30, 7, 20, 2, 6, 31, 21]. As deep neural networks be-

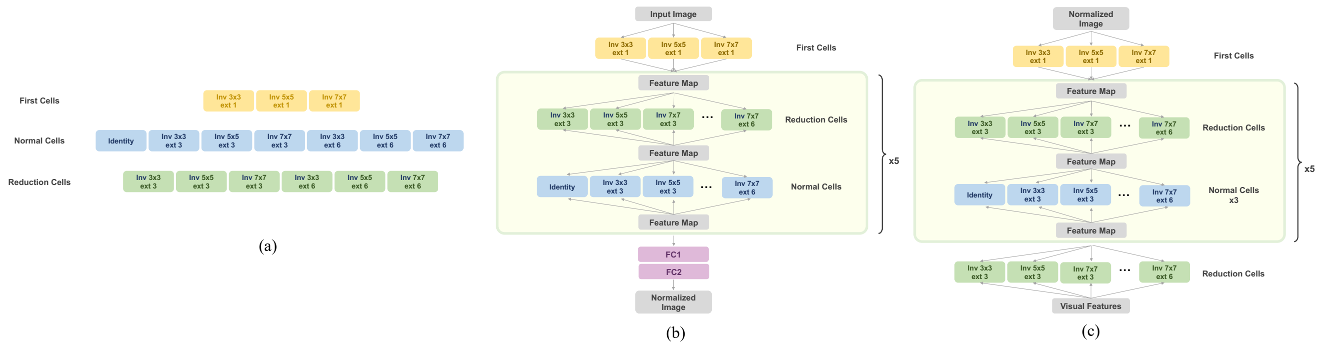


Figure 2. **Architecture space of NAS in the STR framework.** (a) Candidate operations for architecture space. Architecture space of over-parameterized architecture for (b) transformation module and (c) feature extraction module.

gan to be used in the STR framework, various geometric deformations and image distortions problems of text in natural scenes started to be solved. Models that enable end-to-end training by applying neural network modules at each stage, from transformation to prediction, and adequate structure of the input and output of each module were, therefore, proposed. AON [7], MORAN [21], and ASTER [31] presented a CNN model that enables efficient rectification at the transformation stage, improving the recognition performance of STR. However, an inconsistency problem in the verification of the proposed STR frameworks occurred.

To solve the problem, [1] analyzed the contribution of the proposed STR modules. They performed experiments on the possible combinations of each STR module, proposing, in the end, a model that corresponds to the best combination of the modules proposed in the previous study. We evaluated the STR model obtained by our NAS algorithm on three criteria(accuracy, memory[1] and inference time) using the benchmark tool proposed in [1]. Among the STR modules, an optimized STR model was obtained by defining the architectural spaces of the rectifiers and feature extractors composed of CNNs while performing architecture search. This search has shown great benefits regarding memory and accuracy when it was applied to the feature extraction module, which has a large number of parameters. Finally, the STR models obtained from the proposed NAS achieved similar performance results when compared to the SOTA. As far as we are aware of, we are the pioneers of the adaptation of NAS on STR application with results suggesting positive potential development.

3. Method

To obtain an optimized STR model, we defined an architecture space for the transformation and feature extraction stages consisting of CNNs among the four stages of the STR framework. This architecture search did not depend on proxy tasks, i.e. NAS exactly reflects every stage of STR with end-to-end training. In order to obtain an STR model with good memory efficiency and accuracy, the candidate operation is composed of an inverted residual block, and a gradient-based learning technique using binarized path level pruning.

3.1. Architecture Space for CNNs of STR Framework

To apply network search strategy based on the Proxyless-NAS algorithm, the architecture space of the target module should be configured. Representative methods that defines the architecture space are the directed acyclic graph (DAG) [24] or tree-structure [4] schemes. We constructed a tree-structured architecture space to execute an efficient architecture search and defined a backbone structure suitable to each stage module and set of candidate operations. To design a network considering the trade-off between the accuracy and memory efficiency, we defined an structure based on inverted residual block [28] that produces high computational accuracy results depending on the number of parameters of one cell of the convolution computation. In this case, the structure of the backbone CNN, for the architecture search in the rectifier and the feature extractor modules, was depended on the type of each module. The candidate operations for the cell were similarly applied. Figure 2 shows the architecture space and the set of candidate operations of each module.

3.1.1 Candidate Operations

We narrow down the candidate operator of the convolution operations to an inverted residual block [28] to efficiently search for a memory-efficient model in the architecture space development. The candidate operator was divided into first, normal, and reduction cells for input features, according to the stage and role of the feature extrac-

[1]Note that even ’memory’ not directly equals number of parameter size, we follow concept of ’memory-efficient’ from former study. [1]

tion in the backbone CNN. In case of a resize operation, due to the feature size adjustment, the operation is replaced by applying the stride parameter of the reduction cells. In case of an first cell, the low-level feature extraction was the main purpose of the convolutional operation of the first layer of the CNN input, therefore, the number of candidate operators was minimized to $\{3 \times 3, 5 \times 5, 7 \times 7\}$, and the appropriate kernel of the convolution operations was selected by fixing the expansion ratio of the inverted residual block to 1. The normal cell was a repetitive convolutional block that was applied to the layered architecture inside the CNN. The candidate cells for the normal ones were composed of a combination of the expansion ratio $\{3 \text{ or } 6\}$ and the kernel parameters $\{3 \times 3, 5 \times 5, 7 \times 7\}$, as well as the identity mapping. Reduction cells simultaneously extract and resize a feature, resulting in a CNN architecture with appropriate structure. Moreover, they are composed of the same set of candidate operations as the normal cells, except for the identity map. Figure 2 (a) shows the candidate operations of the architecture space separated by the role of each cell.

3.1.2 Architecture Space for Rectifier

The rectifier of the STR framework is a module that is applied at the transformation stage. The rectifier is used to correct an input deformation with an offset grid [21] or to *transform* parameters of the control points using thin-plate-spline (TPS) [30, 31]. We chose the TPS-based rectifier as the basic module to execute a relatively stable search with low degree of freedom in the training procedure. For TPS-based modules, transformation parameters, which results in the localization of the control points, are learned through convolution operations of the localization network. We designed an architecture space for the convolutional operation of the localization network to apply NAS to the rectifier module. The backbone structure of rectifier is composed of a shallow CNN with a small number of layers than that of the feature extractor, similar to the baseline localization network.

The commonly used structure of a rectifier is as follows: Following one first cell, a total of five normal cells and five reduction cells are added. A reduction cell is always preceded by each normal cell and they are responsible to adjust the image size based on the stride parameter, ensuring that the structure of the CNN required for input and output. To prevent the architecture space to increase more than the ideal scenario, therefore, avoiding performance loss, and to create the structure of the CNN, 2×2 rectangular stride is used. After the feature map extraction, two fully connected layers would be able to create a normalized image. Figure 2 (b) shows the over-parameterized network for the architecture search of the rectifier.

3.1.3 Architecture Space for Feature Extractor

Until now, the structure of the feature extractor that is mainly used in the STR framework such as ResNet, VGG, and RCNN [11, 32, 16], has been a CNN model, which focuses on the accuracy improvement. We have defined an over-parameterized architecture, similar to MobileNetV2 [28] using lightweight building blocks to design a model with high accuracy and low memory usage, unlike the existing network architectures. As in the rectifier, this network is optimized through a combination of kernel size and expansion ratio. As a result, an inverted bottleneck structure can be generated according to the type of candidate operation.

The architecture of the space for the tree-structured architecture search is as follows. Excluding the first cell, the entire network is configured assuming that one reduction cell is added to three normal cells. The total network consists, therefore, of 15 normal cells and 6 reduction cells. Reduction cells are composed of a combination of $\{2 \times 2, 2 \times 1\}$ rectangular stride parameters; and they have more layers than the rectifiers to effectively extract the feature information required for text recognition and to have a fully convolutional CNN structure. In Figure 2 (c) the shape of over-parameterized network for architecture search of the feature extractor is shown.

3.2. Architecture Search with ProxylessNAS

We performed a module-specific architecture search based on ProxylessNAS and the architecture space defined in 3.1. [5] propose to use a reinforcement learning and gradient-based learning method to efficiently analyze the binarized architecture. We used a gradient-based learning to train the proposed architecture space and estimated the expected time, F, by measuring the operation time for each learnable block to account for the expected latency.

In the training process, the gradient descent scheme is performed on the input layout of a randomly selected active path of an over-parameterized architecture defined in the architecture space. The architecture parameter is updated based on the result of the softmax output to calculate the weight for each operation among the candidate cells. Thereafter, only two paths are left through binarization, and the remaining paths are excluded to minimize the computational cost during the training. The update rule to obtain the parameters can be approximated as follows.

$$\frac{\partial L}{\partial \alpha_i} \approx \sum_{j=1}^{N} \frac{\partial L}{\partial g_j} \frac{\partial \frac{exp(\alpha_j)}{\sum_k exp(\alpha_k)}}{\partial \alpha_i} = \sum_{j=1}^{N} \frac{\partial L}{\partial g_j} p_j (\delta_{ij} - p_i) \quad (1)$$

where α is defined as N real-valued architecture parameters to be updated and p is the softmax output of an architecture parameter. Moreover, g is a binary gate, which is approximated to update only two paths. If $i = j$, $\delta_{ij} = 1$ or else $\delta_{ij} = 0$. The total loss function, L, used to update the architecture parameters during the training procedure is defined as a linear summation of the cross-entropy loss and expected latency. $L_{latency}$ is a weighted sum of the latency of candidate operations which is measured multiple times in advance. λ_1 is a scaling term of latency loss.

$$L_{total} = L_{CE} + \lambda_1 * L_{latency} \tag{2}$$

3.3. Other components of STR model

To train and test the STR model, it is necessary to design modules for the sequence modeling and prediction stages. For sequence modeling, modules based on recurrent neural network (RNNs) are widely used as they are able to reproduce the behavior of the continuity of characters in a word and extract the correlation between each character in the CNN feature obtained from the rectified image. In the prediction stage, the feature obtained from sequence modeling is used to obtain predictions through a post-processing technique, such as CTC [9] or attention-based techniques. We used a bidirectional sequence modeling module and a prediction module based on the LSTM [29, 30, 6] and the attention [30, 6] techniques to create an STR model capable of end-to-end training.

4. Experiments and Discussions

We performed extensive tests with the STR benchmark [1] dataset to evaluate the performance of the proposed NAS technique. In this case, the training process of the architecture was tested on a single GPU which is NVIDIA GeForce GTX 1080Ti, as it is a commonly used GPU. The evaluation environment was the same experimental environment as described in [1]. All evaluations of the STR models were conducted on the same as [1] with NVIDIA P40 GPU and Intel Xeon E5-2630 v4@2.20 GHz. To compare the results between the models, we measured the recognition accuracy, inference time, and memory usage of each model.

4.1. Datasets

Training. We used the MJSynth (MJ) [12] and SynthText dataset (ST) [10] synthetic datasets to train the STR model. **MJ** is a dataset of images that considers various fonts, borders, shadows, background images, projective distortions, and noise of a word box. It has a total of 8.9M images. Meanwhile, **ST** is a dataset originally designed for scene text detection. It consists of 8000 background images obtained from Google image search and is synthesized by applying a perspective transformation to the target text using the normal of the local surface. In our tests, we used images that focus on the text area of the ST dataset (5.5M cropped images). All 14.4M images, including those in the MJ and ST, were used to train the STR models.

Testing. For evaluating the STR models, we used all data classified into regular and irregular datasets for real-world text images as defined in [1]. Regular text datasets include IIIT5K-Words (IIIT) [22], Street View Text (SVT) [34], ICDAR 2003 (IC03) [14]. **IIIT** is a dataset composed of 3,000 images from Google image search, including billboards, signboard, house numbers, house nameplates, and movie posters. **SVT** consists of outdoor street images collected from Google street view. It consists of 257 images for training and 647 for evaluation, some of which present a lot of noise or low resolution. **IC03** is a dataset created for the ICDAR 2003 robust reading competition, consisting of 1,156 images for training and 1,110 images for evaluation. Too short or non-alphanumeric characters are removed from the dataset in our tests. A total of 860 or 867 processed images are used from this dataset. **IC13** is a dataset created for the ICDAR 2013 robust reading competition, which contains most of IC03 data and also provides 848 training and 1095 evaluation images. As in IC13, 1015 or 857 evaluation images were used, and alphanumeric or short texts were removed.

Irregular datasets include ICDAR2015 (IC15) [13], SVT Perspective (SP) [25], and CUTE80 (CT) [27]. **IC15** was created for the ICDAR 2015 robust reading challenge and consists of 4,468 images for training and 2,077 images for evaluation. All images were collected with Google glass, and 1,811 images were removed in the pre-processing stage to exclude non-alphanumeric or excessively deformed images. The **SP** dataset consists of 645 images collected from Google street view and provides other perspective projection images, such as side images. The **CT** consists of 288 clipped images collected from a normal image, and most of the text is curved.

We evaluated the training model for a total of 10 benchmark datasets using some or all of the data from each dataset. The total evaluation data consists of 8,539 entities, as in [1] (3000 from IIIT, 647 from SVT, 860 or 867 from IC03, 857 or 1015 from IC13, 1811 or 2,077 from IC15, 645 from SP, and 288 from CT). In addition, the averaged total accuracy was calculated by performing five tests with different initialization techniques on one STR model. The average clock time (ms) per image was measured to evaluate the speed, and the number of parameters for learnable floating-point operations in the entire STR framework was calculated to evaluate the memory usage.

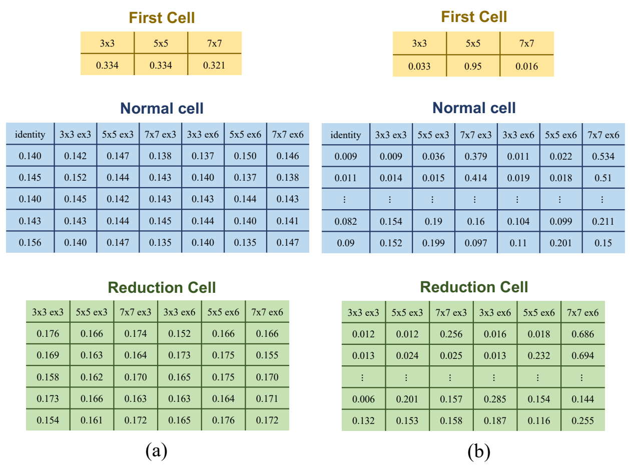
4.2. Architecture Search

Training parameters for architecture search Training details about architecture search based on ProxylessNAS are as follows. The Adam was used as the optimizer of the architecture parameters. The learning rate was set to 1e-5 and the weight decay to 3e-6. Meanwhile, AdaDelta was used as the optimizer for STR model training. In this case, the learning rate was set to 1.0, ρ to 0.95, and ϵ to 1e-8. The size of the batch used in the learning stage was 192. The algorithm performed $300k$ iterations, and the gradient clipping value was 5. Less than 100 GPU hours are needed to attain the optimal STR model. After the model search, it takes 2 more days for STR model parameter tuning on our searched model.

To evaluate the performance of the meta learner among these models, we evaluated the STR model obtained from three different architecture training settings. 1) Model obtained with the same hyperparameter as ProxylessNAS and using the complete total loss function (STRNAS1). 2) Due to the size of the proposed architecture space and the behavior of the network during the calculation, the latency term was removed from the total loss (STR-NAS2). 3) The STR-NAS2 model was used to perform more iterations with lower learning rates for updating architecture parameters than those from the other cases (STR-NAS3). Table 1 shows the training performance of the STR models obtained with these training settings. According to Table 1, the results from the STR-NAS3 model show the best accuracy and number of parameters.

Search results. We conducted a comparative experiment with limited architecture space to verify the utility of the STR model obtained with the proposed architecture search technique. For this purpose, we compared the performance of models by excluding search effects within the limited search space. The same operation parameters were applied to all inverted residual blocks of the over-parameterized network structure. The STR model designed with limited architecture space, presented four cases for the convolutional parameter set {kernel_size, expansion_ratio} of the inverted residual block: $\{3 \times 3, 3\}, \{5 \times 5, 3\}, \{5 \times 5, 6\}, \{7 \times 7, 6\}$. Table 1 shows the performance difference between the STR model obtained from the proposed NAS and the one with limited architecture space. Our comparative experiment shows that NAS has proven to be effective on generating STR module.

In addition, we analyzed the architecture parameter search results of the rectifier and feature extractor for each STR, module to evaluate the search results. Fig. 3 shows the softmax output after the architecture parameters of the rectifier and feature extractor had fully converged. As shown in Fig 4, the output of the feature extractor shows a prominent training result in a specific cell. In the rectifier, however, the definition of convolutional parameters did not significantly affect the transformation parameter optimization of the control point. as the feature extractor had more parameters than the rectifier, and the definition of convolutional parameters related to search directly affects the CNN performance during feature extraction. The total time spent in the learning stage was 100 GPU hours on a NVIDIA GeForce GTX 1080Ti. Fig. 4 shows the feature extraction module for the STR-NAS3.



(a)

First Cell

3x3	5x5	7x7
0.334	0.334	0.321

Normal cell

identity	3x3 ex3	5x5 ex3	7x7 ex3	3x3 ex6	5x5 ex6	7x7 ex6
0.140	0.142	0.147	0.138	0.137	0.150	0.146
0.145	0.152	0.144	0.143	0.140	0.137	0.138
0.140	0.145	0.142	0.143	0.143	0.144	0.143
0.143	0.143	0.144	0.145	0.144	0.140	0.141
0.156	0.140	0.147	0.135	0.140	0.135	0.147

Reduction Cell

3x3 ex3	5x5 ex3	7x7 ex3	3x3 ex6	5x5 ex6	7x7 ex6
0.176	0.166	0.174	0.152	0.166	0.166
0.169	0.163	0.164	0.173	0.175	0.155
0.158	0.162	0.170	0.165	0.175	0.170
0.173	0.166	0.163	0.163	0.164	0.171
0.154	0.161	0.172	0.165	0.176	0.172

(b)

First Cell

3x3	5x5	7x7
0.033	0.95	0.016

Normal cell

identity	3x3 ex3	5x5 ex3	7x7 ex3	3x3 ex6	5x5 ex6	7x7 ex6
0.009	0.009	0.036	0.379	0.011	0.022	0.534
0.011	0.014	0.015	0.414	0.019	0.018	0.51
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.082	0.154	0.19	0.16	0.104	0.099	0.211
0.09	0.152	0.199	0.097	0.11	0.201	0.15

Reduction Cell

3x3 ex3	5x5 ex3	7x7 ex3	3x3 ex6	5x5 ex6	7x7 ex6
0.012	0.012	0.256	0.016	0.018	0.686
0.013	0.024	0.025	0.013	0.232	0.694
⋮	⋮	⋮	⋮	⋮	⋮
0.006	0.201	0.157	0.285	0.154	0.144
0.132	0.153	0.158	0.187	0.116	0.255

Figure 3. **Softmax output of the architecture parameters.** These tables are obtained after sufficient convergence of the architecture space for (a) the rectifier and (b) the feature extractor. The first, reduction, and normal cells are included from the top. As shown in the architecture parameter the output distribution, the performance gain from the NAS is significant during the feature extractor.

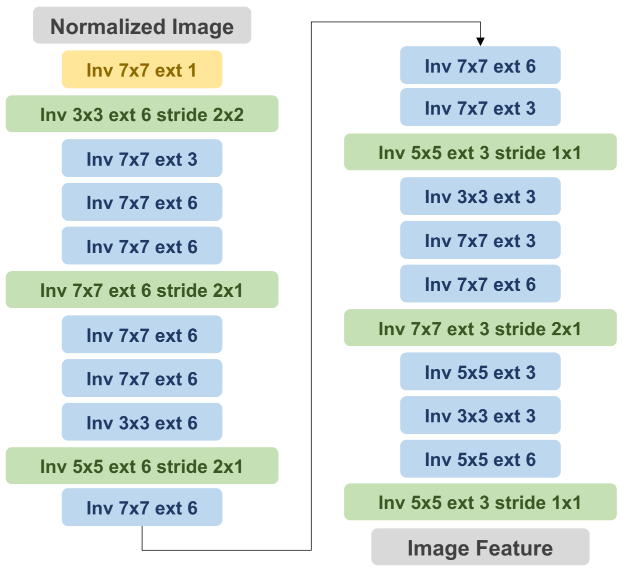


Figure 4. **Feature extraction module obtained from the overparameterized network with the proposed NAS technique.** This model is applied to STR-NAS3, which achieved the highest performance among the STR models.

Comparison with state-of-the-art models. We compared the accuracy, number of parameters, and inference time of

Table 1. Performance difference between the model trained from the proposed NAS (STR-NAS) and that with limited architecture space. The model trained with a limited architecture space consists of all layers of the same convolutional operations.

Model	IIIT 3000	SVT 647	IC03 860	IC03 867	IC13 857	IC13 1015	IC15 1811	IC15 2077	SVTP	CUTE80	Time ms/image	params $\times 10^6$
3*3 ex3	84.433	82.380	92.442	92.272	92.065	90.148	72.170	66.731	74.109	69.097	**30.361**	**14.101**
5*5 ex3	83.667	83.308	92.791	92.964	91.715	90.443	71.121	65.527	75.039	69.097	31.251	14.292
5*5 ex6	84.333	81.917	93.140	93.541	91.599	90.049	72.612	66.827	75.194	67.361	32.994	23.283
7*7 ex6	84.567	83.153	92.907	92.388	91.249	89.655	73.771	67.646	75.194	68.056	30.415	23.857
STR-NAS1	84.967	83.771	92.907	92.849	92.182	91.034	73.771	67.935	74.729	70.139	32.238	21.332
STR-NAS2	85.000	83.771	94.070	93.772	92.532	91.034	73.440	67.935	73.488	68.750	30.738	18.923
STR-NAS3	**85.667**	**85.935**	**93.488**	**93.080**	**91.599**	**90.542**	**75.594**	**69.860**	**76.899**	**72.222**	33.017	16.821

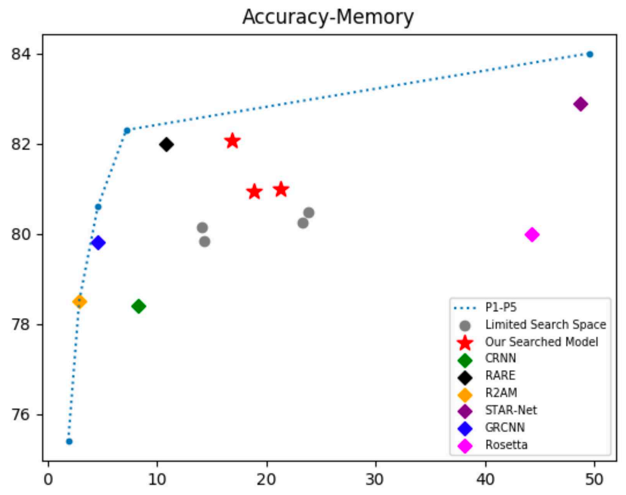


Figure 5. **Accuracy versus memory trade-off curve.** P1-P5 curve corresponds to the top 5 performance model. The area under this curve is maximum for the accuracy–memory tradeoff. The limited search space model is trained with the experimental setup described in Section 4.2. The x and y-axis represent the number of parameters ($\times 10^6$) and the y and total accuracy for all datasets, respectively.

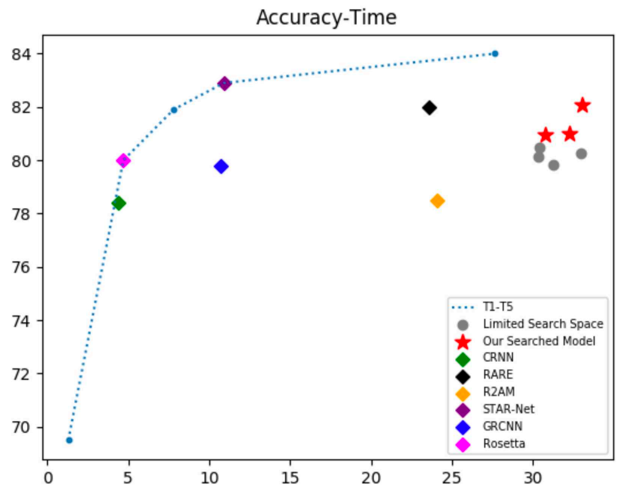


Figure 6. **Accuracy versus time trade-off curve.** The T1-T5 curve corresponds to the models that had the 5 highest performance. The area under this curve is maximum for accuracy-time tradeoff. The limited search space model is trained with the experimental setup described in Section 4.2. The x and y-axis represent the inference speed (ms) and total accuracy for all datasets, respectively.

our model with those of various SOTA models fed with 10 different test datasets. In order to reproduce the performance of the other models, we used the evaluation metric of the SOTA models suggested in [1]. Furthermore, we used SOTA model, which had the highest performance, for the comparisons. Table 2 shows the comparison results of STR-NAS3, SOTA, and best combination models. The models corresponding to the six widest curve area for accuracy-memory are shown in Fig. 5.

Fig. 5 shows the accuracy versus memory curve of the inference results for each STR model. The figure shows the total accuracy on 10 test benchmarks (y-axis) and the number of parameters that can be used for floating-point operations (x-axis). The models obtained from the proposed NAS technique show similar performance to the SOTA models regarding accuracy and memory efficiency. In particular, STR-NAS3 shows similar or higher efficiencies than those from the previously proposed SOTA models.

Fig. 6 shows the accuracy-time graph of the results obtained for each STR model. The inference time of the proposed STR models differs greatly depending on the module utilized in the prediction stage. Models based on attention technique are located on the right side of the graph as they require a lot of computation resources, however, their results are highly accurate. Meanwhile, models such as CTC-based models whose results are located on the left side of the graph indicate relatively fast inference speeds. The proposed STR-NAS models show a relatively slow inference rate due to the attention-based prediction module. In addition, the convolution operation of the inverted residual block focused on the parameter weight reduction, which has good efficiency in reducing the parameter. The bottleneck is, however, attributed to the computational speed.

Table 2. Comparison between the performance of the model obtained with the proposed NAS (STR-NAS3), SOTA, and best combination model. In the last case, the STR model obtained from the combination of the STR modules for the SOTA models can be regarded as the upper limit of accuracy.

Model	IIIT 3000	SVT 647	IC03 860	IC03 867	IC13 857	IC13 1015	IC15 1811	IC15 2077	SVTP	CUTE80	Time ms/image	params $\times 10^6$
CRNN [29]	82.9	82.380	81.6	93.1	92.6	91.1	89.2	69.4	70.0	65.5	**4.4**	8.3
RARE [30]	86.2	85.8	93.9	93.7	92.6	91.1	74.5	68.9	76.2	70.4	23.6	10.8
R2AM [16]	83.4	82.4	92.2	92.0	90.2	88.1	68.9	63.6	72.1	64.9	24.1	2.9
STAR-Net [20]	**87.0**	**86.9**	**94.4**	**94.0**	**92.8**	**91.5**	**76.1**	**70.3**	**77.5**	71.7	10.9	48.7
GRCNN [33]	84.2	83.7	93.5	93.0	90.9	88.8	71.4	65.8	73.6	68.1	10.7	**4.6**
Rosetta [2]	84.3	84.7	93.4	92.9	90.9	89.0	71.2	66.0	73.8	69.2	4.7	44.3
STR-NAS3	85.7	85.9	93.5	93.1	91.6	90.5	75.6	69.9	76.9	**72.2**	33.0	16.8
Best combination [1]	87.9	87.5	94.9	94.4	93.6	92.3	77.6	71.8	79.2	74.0	27.6	49.6

5. Conclusion and Future Works

We applied a NAS method, regardless of proxy task, to an STR field to design a model with high memory efficiency versus accuracy. With our best knowledge, our work is the first trial of adapting NAS on STR application, and suggests the potential development of STR. By applying the proposed technique to the STR framework, we could automatically generate high-performance models without extensive experiments, such as module combination [1]. In addition, on a single commonly used GPU, the performance of the SOTA models was achieved with only approximately 100 GPU hours. To apply NAS to more advanced STR frameworks, the methodology needs to be adapted for designing an architecture space for a CNN that serves a special purpose, such as a rectifier. In addition, by applying architecture search to the sequence modeling and prediction stages, an improvement in the performance is expected for all modules of the STR framework.

References

[1] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *ICCV*, 2019.

[2] F. Borisyuk, A. Gordo, and V. Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *KDD*, 2018.

[3] A. Brock, T. Lim, J. Ritchie, and N. Weston. SMASH: One-shot model architecture search through hypernetworks. In *ICLR*, 2018.

[4] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu. Path-level network transformation for efficient architecture search. In *ICML*, 2018.

[5] H. Cai, L. Zhu, and S. Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

[6] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou. Focusing attention: Towards accurate text recognition in natural images. In *ICCV*, 2017.

[7] Z. Cheng, Y. Xu, F. Bai, Y. Niu, S. Pu, and S. Zhou. Aon: Towards arbitrarily-oriented text recognition. In *CVPR*, 2018.

[8] G. Ghiasi, T.-Y. Lin, R. Pang, and Q. V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019.

[9] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. In *ICML*, 2006.

[10] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[12] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *NeurIPS*, 2014.

[13] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. Icdar 2015 competition on robust reading. In *13th IAPR International Conference on Document Analysis and Recognition, ICDAR 2015 - Conference Proceedings*, Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pages 1156–1160, United States, 11 2015. IEEE Computer Society.

[14] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Bigorda, S. Mestre, J. Mas, D. Mota, J. Almazan, and L. De Las Heras. Icdar 2013 robust reading competition. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 1484–1493, 12 2013.

[15] Y. LeCun, L. Bottou, Y. Bengio, Y. Niu, S. Py, and P. Haner. Gradientbased learning applied to document recognition. In *Proc. of The IEEE*, 1998.

[16] C.-Y. Lee and S. Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *CVPR*, pages 2231–2239, 06 2016.

[17] 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 *CVPR*, 2019.

[18] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

[19] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

[20] W. Liu, C. Chen, K.-Y. Wong, Z. Su, and J. Han. Star-net: A spatial attention residue network for scene text recognition. In *BMVC*, 2016.

[21] C. Luo, L. Jin, and Z. Sun. A multi-object rectified attention network for scene text recognition. *Pattern Recognition*, 90:109–118, 2019.

[22] A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2009.

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, Y. Reading digits in natural images with unsupervised feature learning. In *Proc. of The IEEE*, 2011.

[24] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018.

[25] T. Q. Phan, P. Shivakumara, S. Tian, and C. L. Tan. Recognizing text with perspective distortion in natural scenes. *2013 IEEE International Conference on Computer Vision*, pages 569–576, 2013.

[26] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, Jan. 2000.

[27] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan. A robust arbitrary text detection system for natural scene images. *Expert Syst. Appl.*, 41:8027–8048, 2014.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[29] 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*, 39(11):2298–2304, Nov 2017.

[30] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai. Robust scene text recognition with automatic rectification. In *CVPR*, 2016.

[31] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai. Aster: An attentional scene text recognizer with flexible rectification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2035–2048, Sep. 2019.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[33] J. Wang and X. Hu. Gated recurrent convolution neural network for ocr. In *NeurIPS*. 2017.

[34] K. Wang, B. Babenko, and S. J. Belongie. End-to-end scene text recognition. *2011 International Conference on Computer Vision*, pages 1457–1464, 2011.

[35] Y. Weng, T. Zhou, Y. Li, and X. Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

[36] Z. Zhu, C. Liu, D. Yang, A. Yuille, and D. Xu. V-nas: Neural architecture search for volumetric medical image segmentation. In *3DV*, pages 240–248, 09 2019.

[37] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, volume abs/1611.01578, 2017.

[38] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.