Building A Mobile Text Recognizer via Truncated SVD-based Knowledge Distillation-Guided NAS

Weifeng Lin[1], Canyu Xie[1], Dezhi Peng[1], Jiapeng Wang[1], Lianwen Jin[1], Wei Ding[2], Cong Yao[2], Mengchao He[2]

[1]South China University of Technology, [2]Alibaba Group

BMVC 2023

Demo Code

Background & Motivation

- Text plays a crucial role in human information acquisition and retention. Some popular intelligent mobile applications, such as digitalizing paper bills and scan translation, require the deployment of deep learning-based systems on terminal devices with limited computing resources.
- Most of the current methods aimed at improving text recognition accuracy increase model complexity and parameter count, making them unsuitable for operation on resource-constrained devices.

Overall Architecture

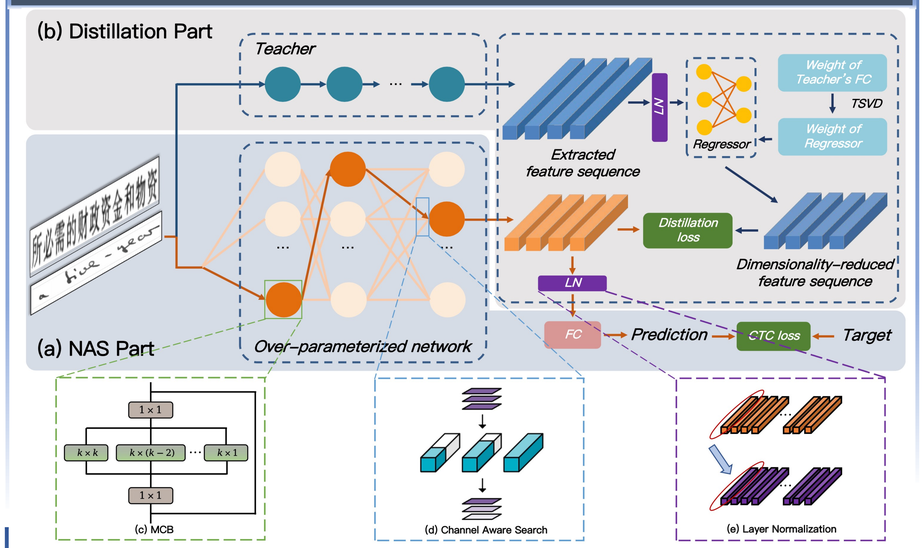


Figure 1. (a) An over-parameterized network with learnable blocks and channel configurations for neural architecture search. (b) The TSVD-based knowledge distillation is introduced to the NAS process to guide the student searching. (c) Mobile Char Block (MCB). (d) Channel-Aware Search. (e) Layer Normalization.

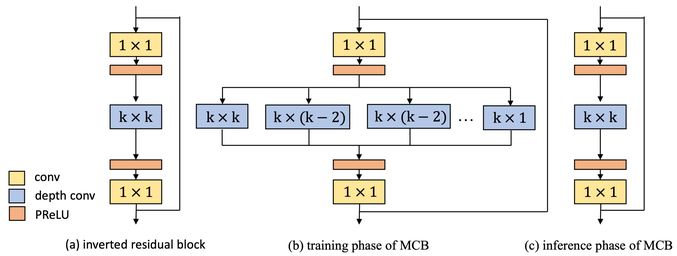
For the whole pipeline for text recognition, we firstly search for a lightweight student model from the over-parameterized network and subsequently train the search model with the proposed distillation method to get an enhanced performance.The search problem can be formulated as :

$$\min_{\alpha} L_{val}(\mathcal{N}(w^*, \alpha))$$

$$\text{s.t.} w^* = \arg\min_{w} L_{train}(\mathcal{N}(w, \alpha)),$$

Methodology

☐ Mobile Char Block



MCB builds upon the Inverted Residual Block by replacing the single-branch Depth-wise convolution with a multi-branch Depth-wise convolution. The convolution kernels in multiple branches have decreasing widths from K to 1 while maintaining consistent height.

Furthermore, during the inference stage, a reparameterization technique is employed to merge the linear multi-branch structure into a single-branch structure, reducing inference computational cost and parameter count.

☐ Channel-Aware Search

We adopt a channel-masking mechanism to enable the NAS algorithm to search with different channel configurations.

$$\mathcal{O}_{masked-i} = \beta_1 M_1(\mathcal{O}_i) + \beta_2 M_2(\mathcal{O}_i) + \ldots + \beta_n M_n(\mathcal{O}_i)$$
$$= \sum_j^n \beta_j M_j(\mathcal{O}_i),$$

☐ Layer Normalization

More than half of the CTC-based models in the feature sequence belong to the blank cate- gory.Therefore, we propose to use the LN layer to normalize the feature sequence

$$\mu^j = \frac{1}{C}\sum_{i=1}^{C} a_i^j, \quad \sigma^j = \frac{1}{C}\sum_{i=1}^{C}(a_i^j - \mu^j)^2,$$

☐ TSVD-based Knowledge Distillation

- Knowledge distillation can effectively reduce the gap between complex models and lightweight models. The loss in traditional feature-based knowledge distillation methods

$$L_{KD} = \frac{1}{2}||reg(\mathcal{F}_{CNN}^T, W_{reg}) - \mathcal{F}_{CNN}^S||^2,$$

- We proposes a TSVD method, with the core idea being to perform an SVD decomposition on the teacher's fully connected (FC) layer weights and truncate them to obtain the weights Wreg for the regressor reg

$$W_{cls}^T = U_{m\times m}\Sigma_{m\times c^T}V^*_{c^T\times c^T} \approx U_{m\times c^S}\Sigma_{c^S\times c^S}V^*_{c^S\times c^T},$$

- Based on this, we can obtain the form of the loss for TSVD as follows:

$$L_{TSVD-KD} = \frac{1}{2}||\mathcal{F}_{CNN}^T V^*_{c^S\times c^T} - \mathcal{F}_{CNN}^S||^2,$$

$$L = \alpha L_{CTC} + \beta L_{TSVD-KD}$$

Experiments

Dataset	Method	AR	Storage(MB)	FLOPs(G)	Speed per line(ms)
IAM	Ingle et al. [16]	85.90%	42.4	-	-
	Chaudhary and Bali [6]	90.20%	112	-	-
	Kang et al. [20]	92.38%	400	-	-
	DAN [42]	**93.60%**	-	-	-
	ResNet24LN (Teacher)	92.00%	71.3	62.62	189(x86) / 246(ARM) / 12.8(GPU)
	Ours (Search)	92.60%	**8.7**	**1.92**	**18(x86) / 24(ARM) / 2.0(GPU)**
ICDAR2013	Xie et al. [43](Full model)	91.55%	61	16.57	318 (x86)
	Xie et al. [43](Compact model)	90.50%	**2.8**	4.46	146 (x86)
	Huang et al. [15]	91.82%	45.6	-	-
	Liu et al. [23]	93.62%	203	-	-
	Peng et al. [31]	**94.50%**	119	-	164 (x86)
	ResNet24LN (Teacher)	92.91%	80.10	63.42	141(x86) / 272(ARM) / 12.5(GPU)
	Ours (Search)	91.86%	8.34	**1.26**	**14(x86) / 18(ARM) / 1.9(GPU)**
SCUT-HCCDoc	Zhang et al. [49]	87.46%	59	16.40	312 (x86)
	Peng et al. [31]	**90.71%**	116.5	-	152 (x86)
	Liu et al. [23]	89.06%	200.5	-	-
	ResNet24LN (Teacher)	89.09%	77.80	63.26	142(x86) / 273(ARM) / 11.2(GPU)
	Ours (Search)	88.10%	**5.74**	**0.95**	**11(x86) / 15(ARM) / 1.9(GPU)**
JS-Printed	ResNet24LN (Teacher)	**99.24%**	120.00	33.96	98(x86) / 186(ARM) / 9.4(GPU)
	Ours (Search)	98.25%	**12.3**	**1.03**	**15(x86) / 18(ARM) / 1.9(GPU)**

Table 1: Comparison with existing Text-line recognition methods

Model	ICDAR2013			IAM			SCUT-HCCDoc			JS-Printed		
	AR	Storage (MB)	FLOPs (G)	AR	Storage (MB)	FLOPs (G)	AR	Storage (MB)	FLOPs (G)	AR	Storage (MB)	FLOPs (G)
ShuffleNetV2 [27]	84.92%	10.5	0.98	89.52%	10.52	1.85	81.09%	9.72	0.96	89.61%	15.3	1.03
MobileNetV2 [35]	85.35%	8.23	**0.97**	89.22%	9.32	**1.14**	81.14%	8.1	0.96	93.71%	14.2	0.99
MobileNetV3 [14]	86.45%	10.7	1.00	90.00%	11.08	1.17	81.96%	14.7	1.34	93.60%	16.1	1.00
DARTS [25]	84.21%	11.0	0.99	89.37%	11.72	1.35	80.85%	10.9	1.04	86.61%	18.2	1.21
FBNetV2-F1 [41]	85.83%	12.1	0.99	89.77%	13.84	1.70	81.36%	11.6	0.97	92.95%	15.7	1.00
EfficientNet-B0 [39]	85.52%	12.5	1.00	89.52%	12.84	1.25	73.21%	19.2	1.02	88.36%	18.5	1.20
AutoSTR [50]	88.33%	**6.8**	0.99	90.80%	11.51	2.01	84.82%	6.33	0.97	95.37%	12.4	**0.97**
Ours	**91.86%**	8.34	1.26	**92.60%**	**8.7**	1.92	**88.10%**	**5.74**	**0.95**	**98.25%**	**12.3**	1.03

Table 2: Comparison with existing lightweight models