Deep Learning-Based Text Detection in Natural Scenes

Lizhi Cui[1,a], Honglei Tian[1,b,*], Shumin Fei[1,c]

[1]Henan Polytechnic University, Jiaozuo, Henan, China
[a]clzh0308@hpu.cn, [b]vskyi@qq.com, [c]smfei@seu.edu.cn
[*]Corresponding author

Abstract: *Traditional text detection mainly relies on manual features, which are only applicable to simple environments and have limited generalisation capabilities. Although deep learning enhances the generalisation and robustness of detection, complex contexts still face challenges. Current CNN text detection algorithms are difficult to handle large-scale and long-distance text due to the limitation of receiving domain and spatial information extraction. This chapter proposes the GMSTNet model, which combines GhostNet V2, MobileNet V3, and Swin Transformer to enhance efficiency through segmented nonlinear activation, effectively handle small-size text and detailed features while enhancing global and local perception, and demonstrate good performance on multiple datasets.*

Keywords: *Deep Learning, Text Detection, Natural Scenes, Feature Fusion*

1. Introduction

Existing convolutional neural network (CNN) algorithms face the challenge of limited receptive domain and spatial information extraction capabilities when dealing with wide-ratio, widely-spaced, and irregularly shaped text, which affects their effectiveness in detecting large-scale and long-distance text. In this paper, we propose a scene text detection algorithm using the joint optimization of GhostNet V2[1], MobileNet V3[2] and Swin Transform[3] and we also propose the use of segmented nonlinear activation instead of continuous nonlinear activation to improve the efficiency of the model's execution on different hardware, which is hereinafter referred to as the model GMSTNet.

2. The overall architecture of GMSTNet

The GMSTNet proposed in this paper is primarily composed of five parts: Cheap Operation, Light Weight Se, Decoupled Fully Connection (DFC), Global Perception, and Light Weighted Se. Among these, the lightweight channel attention and cheap operations are connected through the decoupled fully connected module, integrating the five parts into a lightweight architecture. The network infrastructure is shown in Figure 1, and the overall architecture is presented in Table 1.

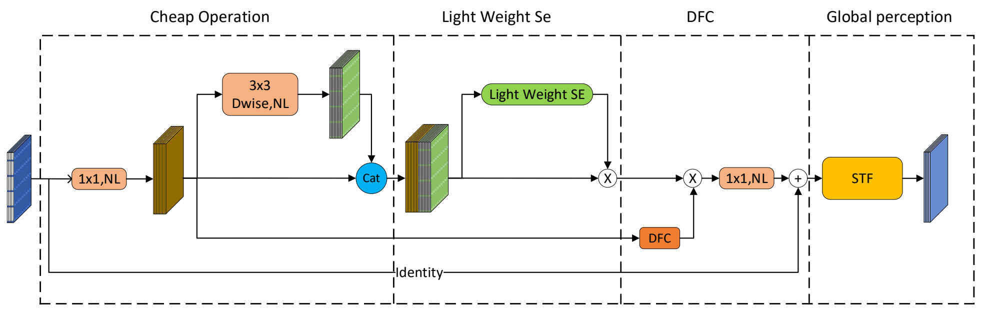

Figure 1: The structure of beck.

The model based on the MobileNet V3 Small architecture effectively captures local and global text features by cascading eight optimized neck and STF structures. The Se module, due to its lightweight nature, is enabled in all neck layers except the second one to dynamically extract features. The Identify branch forms pathways under specific conditions to facilitate the fusion of information flow. To address the detection of large-sized text, STF is activated in the first and last neck to enhance global perception and text pattern fusion. The model utilizes the Transp_Interp method for feature dimension expansion and channel reduction, with an expansion rate of 2 in each layer. Finally, scene text detection is completed using the Dbnet++ segmentation head, Dbhead. Subsequent sections will detail these technical specifics.

Table 1: Overall Architecture. Each row represents a layer. "exp size" indicates the expansion channel number of the first 1x1 convolution in Neck, "Se" indicates whether the lightweight channel attention is employed in the current layer, "STF" indicates whether global perception is activated, "NL" represents the type of activation function used in the current layer, "NxN" represents the size of the depthwise separable convolution kernel in neck, and "S" stands for stride.

Input	Operator	Exp Size	Out channels	Se	NL	STF	S
736x736x3	conv2d, 3x3	-	16	N	SES	N	2
368x368x16	neck, 3x3	16	16	Y	RE	√	2
368x368x24	neck, 3x3	88	24	N	RE	N	1
184x184x24	neck, 5x5	96	40	Y	SES	N	2
184x184x40	neck, 5x5	240	40	Y	SES	N	1
184x184x40	neck, 5x5	240	40	Y	SES	N	1
184x184x40	neck, 5x5	120	48	Y	SES	N	1
184x184x48	neck, 5x5	144	48	Y	SES	N	1
92x92x48	neck, 5x5	288	96	Y	SES	Y	1
92x92x48	Transp_Interp	-	-	-	-	-	-
736x736x1	Dbhead	-	-	-	-	-	-

2.1. Joint Optimization of DFC and Cheap Operation

DFC tokenizes feature maps conceptually, transforming the processing of two-dimensional features into a sequence-based approach. For example, tokenizing $Z \in \mathbb{R}^{H \times W \times C}$ into $z_{ij} \in \mathbb{R}^{C}$, at this point, the feature map is denoted as $Z = \{z_{11}, z_{12}, \cdots, z_{HW}\}$. The corresponding attention map is denoted as $A = \{a_{11}, a_{12}, \cdots, a_{HW}\}$. The fully connected attention can be represented by the following equation:

$$a_{xy} = \sum_{i,j} F_{xy,ij} \odot z_{ij} \tag{1}$$

Where F represents weights, $\odot$ denotes the dot product, and a_{xy} is the xy element in A. The computational complexity of tokenized fully connected attention is quadratic to the feature size (H^2W^2). Equation (1) is decomposed along the dimension of the feature map into horizontal and vertical directions as follows:

$$a'_{hw} = \sum_{h'=1}^{H} F^{H}_{h,h'w} \odot z_{h'w}, h = 1,2,\cdots,H, w = 1,2,\cdots,W \tag{2}$$

$$a_{hw} = \sum_{w'=1}^{W} F^{W}_{w,hw'} \odot a'_{hw'}, h = 1,2,\cdots,H, w = 1,2,\cdots,W \tag{3}$$

Where F^H and F^W are the fully connected weights along horizontal and vertical directions, respectively. Thus, the theoretical complexity of DFC is $o(K_H HW + K_W HW)$. Next, let's discuss Cheap Operation.

Traditional convolution can be represented by the following formula:

$$Y = X * f + b \tag{4}$$

Where $X \in \mathbb{R}^{c \times h \times w}$ represents the input feature map, and h ,w denote the height and width of the input feature map respectively, c is the number of channels, $Y \in \mathbb{R}^{C_{out} \times h_{out} \times w_{out}}$ represents the number of output feature map channels c_{out}, $f \in \mathbb{R}^{c \times k \times k \times c_{out}}$ are all convolution kernels, $k \times k$ represents the height and width of the kernel, and b is the bias.

In lightweight neural networks like MobileNet V3 Small, despite a streamlined design, the feature map channel count of 300 can increase computational costs. To optimize feature extraction, the Cheap

Operation method uses regular convolution to generate intrinsic feature maps, followed by linear operations to produce redundant feature maps, thus enhancing model performance without significantly increasing computational burden.

For generating an intrinsic feature map with m channels, ignoring bias and using the same parameters as in formula(4), the process can be defined by the following formula:

$$Y^{'} = X * f^{'} \tag{5}$$

Where $Y^{'} \in \mathbb{R}^{m \times h_{out} \times w_{out}}$ represents the intrinsic feature map, and $f^{'} \in \mathbb{R}^{c \times k \times k \times m}$ denotes all the convolution kernels.

2.2. Light Weighted Se

Similar to DFC and Cheap Operation, lightweight channel attention also aims to address the contradiction between device computation and efficiency, but it has its own characteristics. Lightweight channel attention optimizes the channel dimension, producing more focused feature aggregation attention maps, as shown in Figure 2

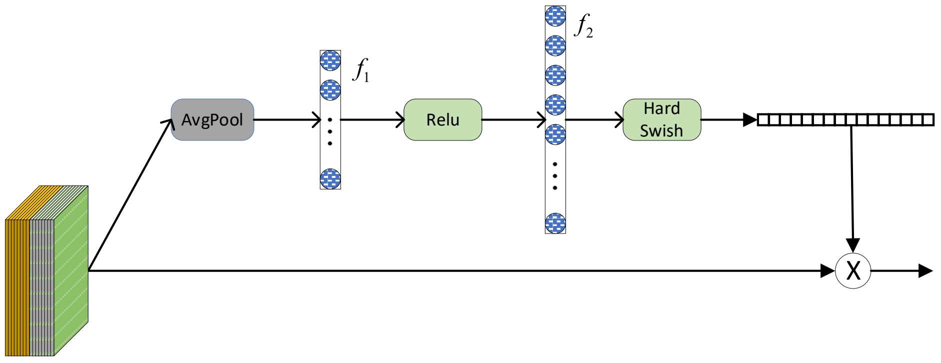

Figure 2: The structure of Light Weighted Se.

The algorithm process is summarized with the following formula:

$$Out = X \otimes handswish(f_2(Relu(f_1(X)))) \tag{6}$$

Here, $\otimes$ represents the element-wise multiplication operation. Following the literature[2], the Hard Swish function is used, which is an approximation of the Swish function. The Swish and Hard Swish functions are defined as follows:

$$\begin{gathered} h - swish = x \frac{\mathrm{ReLU}6(x+3)}{6} \\ swish = x\sigma(x) \end{gathered} \tag{7}$$

Here, σ represents the function Sigmoid. The GPU, a key hardware in the field of deep learning, has thousands of cores. However, for computations involving continuous nonlinear activations, the GPU requires numerical simulation to solve approximations, which cannot be optimized by traditional GPUs. To improve efficiency, this paper proposes using a piecewise activation function to approximate the Hard Swish function, detailed in formulas(8).

$$\mathrm{SegHardSwish}(x) = \begin{cases} 0 & x \in (-\infty, -3] \\ x \cdot \dfrac{x+3}{6} & x \in (-3, 3) \\ x & x \in [3, +\infty) \end{cases} \tag{8}$$

Firstly, for $(-\infty, -3]$, since the original function's operation $Min(\max(0,x),6)$ is simplified to an assignment operation, the CUDA cores do not perform complex, time-consuming mathematical operations, but instead carry out high-speed assignment operations. Secondly, for $[3, +\infty)$, the CUDA cores directly execute the assignment operation. Lastly, because the segmented function is simple, the gradient computation during backpropagation also benefits from the simplified Hard-Swish form.

2.3. Global feature enhancement

The STF module improves the feature detection capabilities of the Neck by integrating both WMSA and SWMSA[4]. This combination effectively captures the interrelationships among text features, background, and text elements, as illustrated in Figure 3.

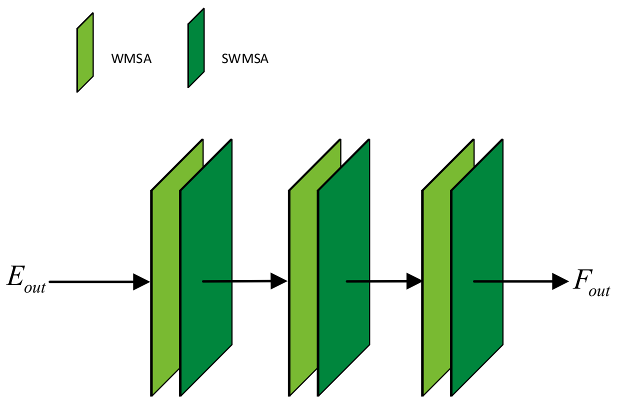

Figure 3: STF module, cascaded from 6 layers and 3 sets of Swin Transform base blocks.

The STF module includes six consecutive Swin Transformer base layers, with every two layers forming a basic block. These basic blocks have the same structure and use LN layers to normalize the inputs to the MSA and MLP layers. The WMSA layer captures local features by windowing feature maps and computing self-attention within independent windows. The SWMSA layer integrates local information from different windows.

2.4. Generating Training Labels

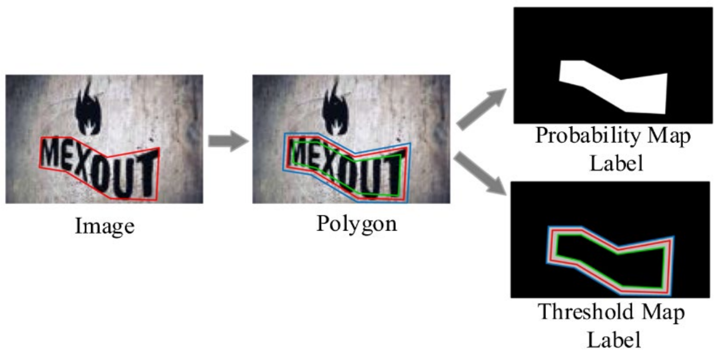

Figure 4: The process of generating training labels.

As illustrated in Figure 4, the training process utilizes actual labels to guide the network in the development of maps predicting probabilities, establishing thresholds, and approximating binaries. For this, the Vatti clipping algorithm is used to expand and contract real text polygon labels, obtaining polygons I_e and I_s, respectively. The expansion and contraction use the same pixel offset, which is calculated using the following formula:

$$m = \frac{\mathrm{Area}(b_o) \times (1 - r^2)}{\mathrm{Perimeter}(b_o)} \tag{9}$$

Here, $Area(\cdot)$ represents the area of the polygon, $Perimeter(\cdot)$ represents the perimeter of the polygon, and r represents the scaling factor, typically set at 0.4. The area between I_e and I_s serves as the text boundary. Similar labels are employed to guide the training of features of probability and threshold through measuring the spatial distance from boundary to actual text polygon labels.

2.5. Loss function

In the model, the formulation of the loss function L consists of three distinct components, as outlined by the equation below:

$$L = L_{bi} + \lambda_1 L_{pr} + \lambda_2 L_{th} \tag{10}$$

Here, L_{bi}, L_{pr} and L_{th} correspond to the losses associated with the maps for approximate binaries, probabilities, and thresholds, respectively. λ_1 and λ_2 are the weight coefficients balancing the importance of the three types of losses. In text images in the scene, where positive and negative samples are extremely unbalanced, the Dice loss is particularly suitable due to its focus on the overlap between predictions and real labels, commonly used in image segmentation. To address this issue, the Dice loss is used to compute L_{bi}, with the following formula:

$$L_{bi} = 1 - \frac{2\sum_{i=1}^{N}(p_i \bullet g_i)}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} g_i + \delta} \tag{11}$$

Here, p_i and g_i denote the predicted and actual values for the i-th pixel, respectively. To ensure numerical stability and minimize the risk of overfitting, δ serves as a regularization factor. We employ the binary cross-entropy method for calculating L_{pr}, which helps mitigate issues related to uneven sample distribution. The computation of L_{pr} is detailed by the following formula:

$$L_{pr} = \sum_{i \in S_P}\left[g_i^{'} \log p_i^{'} + (1 - g_i^{'}) \log (1 - p_i^{'})\right] \tag{12}$$

In this context, S_p denotes the set comprising both positive and negative samples. $p_i^{'}$ and $g_i^{'}$ correspond to the predicted and actual values of the i-th pixel, respectively. The L1 distance facilitates the computation of L_{th}, which quantifies the deviation of predictions from actual labels. The computation is governed by the following equation:

$$L_{th} = \sum_{i \in R_e} |\hat{g}_i - \hat{p}_i| \tag{13}$$

In this context, $\hat{p}_i$ denotes the anticipated probability map value, while R_e signifies the pixel group contained by the expanded text polygon. The actual threshold map label is represented by $\hat{g}_i$.

3. Experiment and Data Analysis

3.1. Datasets

The ICDAR 2017-MLT[5] dataset assembles a collection of textual visual content derived from natural settings in a polyglot context, comprising a total of 18,000 images divided into 7,200 for training, 1,800 for validation, and 9,000 for assessment. Each image annotates the text regions with precise quadrilateral vertex coordinates.

The ICDAR 2015[6] dataset emerges as a seminal resource for the detection of text oriented in diverse directions. It encompasses 1,500 images, split into 1,000 for training and 500 for testing. The dataset is distinguished by its assortment of image variables such as scale, orientation, and instances of diminished clarity, with meticulous annotations based on the coordinates of the text vertices.

The Total Text[7] dataset is specifically engineered for the detection of text with unconventional contours within scene contexts, containing 1,555 images—1,255 allocated for training and 300 for testing. This dataset challenges researchers with its varied backgrounds and the frequent occurrence of text in curved layouts.

3.2. Experimental Setup

In experiments, a dual-stage training protocol was adopted. Initial training was conducted using the training and validation collections from the ICDAR 2017-MLT dataset. This was followed by a refinement phase where models were fine-tuned using two subsequent datasets, specifically ICDAR 2015 and Total Text, across a total of 1,200 iterations. Throughout these stages, batches were processed in groups of eight, employing Stochastic Gradient Descent (SGD) as the optimization technique. An adaptive mechanism was implemented for the modulation of the learning rate, utilizing a Polynomial Learning Rate Policy(14), starting at a rate l_0 of 0.001 with a decay parameter p set at 0.9.

$$l_r = l_0 \times (1 - \frac{epoch}{\max\_epoch})^p \tag{14}$$

During the model's training phase, its capacity for generalization is enhanced through the implementation of three distinct data modification techniques: random rotations, random flips, and random crops adjusted to 640x640 pixels. In the subsequent inference phase, the procedure initiates with the segmentation predictions, subsequently generating a binary map. The experimental apparatus comprises a computing unit equipped with an Intel Core i7-8700K CPU, which features six cores and twelve threads, complemented by an NVIDIA GeForce RTX 3090 GPU, and bolstered by 32GB of DDR4 memory.

3.3. Experimental Analysis

In order to evaluate the stability and performance metrics of the newly introduced approach, datasets ICDAR 2015 and Total Text are used for comparison and analysis with recent methods. Hyperparameters Z are set to 5 and 6. Since different methods may use different hardware, the FPS listed in the tests are for reference only.

Table 2: Results in ICDAR 2015.

Model	Conference/Journal	ICDAR 2015			
		P%	R%	F%	FPS
PSENet[8]	CVPR2019	84.5	86.9	85.7	1.6
PAN[9]	ICCV2019	84.0	81.9	82.9	26.1
ContourNet[10]	CVPR2020	87.6	86.1	86.9	3.5
DBnet[11]	AAAI2020	91.8	83.2	87.3	12.0
FAST[12]	arXiv2021	89.7	84.6	87.1	15.7
DText[13]	PR2022	88.5	85.6	87.0	-
DBnet++[14]	TPAMI2022	90.9	83.9	87.3	10.0
LPAP[15]	TOMM2023	88.7	84.8	86.5	-
Leaf Text[16]	TMM2023	88.9	82.3	86.1	-
Ours	–	91.0	84.3	87.5	4.6

Multi-directional text detection assessment: As shown in Table 2, our method achieves good performance in the F-measure, scoring 87.5%. Compared to classic segmentation methods such as PSENet and PAN, our method shows an improvement of 1.8% and 4.6% in F-measure. Compared to DBnet and DBnet++, our method increases the F-measure by 0.2% through improved recall rates.

Table 3: Results in Total Text.

Model	Conference/Journal	Total Text			
		P	R	F	FPS
PSENet[8]	CVPR2019	84.0	80.0	80.9	3.9
PAN[9]	CVPR2019	89.3	81.0	85.0	39.6
ContourNet[17]	CVPR2020	86.9	83.9	85.4	3.8
DBnet[11]	AAAI2020	87.1	82.5	84.7	32.0
KPN[18]	TNNLS2022	88.0	82.3	85.1	22.7
DBnet++[14]	TPAMI2022	88.9	83.2	86.0	28
FS[19]	TIP2022	88.7	79.9	84.1	24.3
LPAP[15]	TOMM2023	87.3	79.8	83.4	-
Ours	–	89.4	83.6	86.4	15.8

Curved text detection assessment: Detecting curved text and complex scenes is more challenging than quadrilateral text. Test results on Total Text are shown in Table 3. Our proposed method achieves good results in F-measure, scoring 86.4%. Additionally, compared to DBnet and DBnet++, our method improves the F-measure by 1.7% and 0.4%, respectively. Our method also shows advantages over other methods.

4. Conclusion

This paper introduces the GMSTNet algorithm, optimized with GhostNet V2, MobileNet V3, and Swin Transform for scene text detection. The algorithm efficiently acquires features through Cheap Operation and dynamically calculates feature weights using Light Weight Se, ensuring effective utilization of all features. DFC technology is employed for precise localization and fusion of scene text features, reducing computation and parameter volume. The STF module extracts overall context information, optimizing processing of small-size and fine details. The effectiveness of the algorithm is validated through experiments on public datasets.

References

[1] Tang Y, Han K, Guo J, et al. GhostNetV2: Enhance cheap operation with long-range attention[J]. arXiv, 2022.
[2] Howard A, Sandler M, Chu G, et al. Searching for MobileNetV3[M/OL]. arXiv, 2019.
[3] Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows[EB/OL]//arXiv.org. (2021-03-25).
[4] Li X, Yao X, Liu Y. Combining swin transformer and attention-weighted fusion for scene text detection[J/OL]. Neural Processing Letters, 2024, 56(2): 52.
[5] Nayef N, Yin F, Bizid I, et al. ICDAR2017 robust reading challenge on multi-lingual scene text detection and script identification - RRC-MLT[C/OL]. 2017: 1454-1459.
[6] Karatzas D, Bigorda L G I, Nicolaou A, et al. ICDAR 2015 competition on Robust Reading[J/OL]. 2015 13th International Conference on Document Analysis and Recognition (ICDAR), 2015, null: 1156-1160.
[7] Ch'ng C K, Chan C S. Total-text: A comprehensive dataset for scene text detection and recognition[C/OL]//2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR): Vol. 01. 2017: 935-942.
[8] Wang W, Xie E, Li X, et al. Shape robust text detection with progressive scale expansion network[M/OL]. arXiv, 2019.
[9] Wang W, Xie E, Song X, et al. Efficient and Accurate Arbitrary-Shaped Text Detection With Pixel Aggregation Network[J/OL]. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, null: 8439-8448.
[10] Wang Y, Xie H, Zha Z, et al. ContourNet: Taking a Further Step Toward Accurate Arbitrary-Shaped Scene Text Detection[J/OL]. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, null: 11750-11759.
[11] Liao M, Wan Z, Yao C, et al. Real-time scene text detection with differentiable binarization[M/OL]. arXiv, 2019.
[12] Chen Z, Wang J, Wang W, et al. FAST: Faster Arbitrarily-Shaped Text Detector with Minimalist Kernel Representation[M/OL]. arXiv, 2023.
[13] Cai Y, Liu Y, Shen C, et al. Arbitrarily shaped scene text detection with dynamic convolution[J/OL]. Pattern Recognition, 2022, 127: 108608.
[14] Liao M, Zou Z, Wan Z, et al. Real-time scene text detection with differentiable binarization and adaptive scale fusion[M/OL]. arXiv, 2022.
[15] Fu Z, Xie H, Fang S, et al. Learning pixel affinity pyramid for arbitrary-shaped text detection[J/OL]. ACM Transactions on Multimedia Computing, Communications, and Applications, 2022, 19.
[16] Yang C, Chen M, Yuan Y, et al. Text Growing on Leaf[J/OL]. ArXiv, 2022, abs/2209.03016: null.
[17] Deng D, Liu H, Li X, et al. PixelLink: Detecting Scene Text via Instance Segmentation[J/OL]. Proceedings of the AAAI Conference on Artificial Intelligence, 2018, 32(1).
[18] Zhang S X, Zhu X, Hou J B, et al. Kernel Proposal Network for Arbitrary Shape Text Detection[J/OL]. IEEE transactions on neural networks and learning systems, 2022, PP: null.
[19] Wang F, Xu X, Chen Y, et al. Fuzzy semantics for arbitrary-shaped scene text detection[J/OL]. IEEE transactions on image processing, 2023, 32: 1-12.