applied sciences

MDPI

Article

Collaborative Encoding Method for Scene Text Recognition in Low Linguistic Resources: The Uyghur Language Case Study

Miaomiao Xu [1,2,3], Jiang Zhang [1], Lianghui Xu [1], Wushour Silamu [1,2,3,*] and Yanbing Li [1,2,3]

[1] College of Computer Science and Technology, Xinjiang University, Urumqi 830017, China; xmm@stu.xju.edu.cn (M.X.); zhangjiang@stu.xju.edu.cn (J.Z.); xlh@stu.xju.edu.cn (L.X.); liyb@xju.edu.cn (Y.L.)
[2] Xinjiang Laboratory of Multi-Language Information Technology, Xinjiang University, Urumqi 830017, China
[3] Xinjiang Multilingual Information Technology Research Center, Xinjiang University, Urumqi 830017, China
* Correspondence: wushour@xju.edu.cn

Abstract: Current research on scene text recognition primarily focuses on languages with abundant linguistic resources, such as English and Chinese. In contrast, there is relatively limited research dedicated to low-resource languages. Advanced methods for scene text recognition often employ Transformer-based architectures. However, the performance of Transformer architectures is suboptimal when dealing with low-resource datasets. This paper proposes a Collaborative Encoding Method for Scene Text Recognition in the low-resource Uyghur language. The encoding framework comprises three main modules: the Filter module, the Dual-Branch Feature Extraction module, and the Dynamic Fusion module. The Filter module, consisting of a series of upsampling and downsampling operations, performs coarse-grained filtering on input images to reduce the impact of scene noise on the model, thereby obtaining more accurate feature information. The Dual-Branch Feature Extraction module adopts a parallel structure combining Transformer encoding and Convolutional Neural Network (CNN) encoding to capture local and global information. The Dynamic Fusion module employs an attention mechanism to dynamically merge the feature information obtained from the Transformer and CNN branches. To address the scarcity of real data for natural scene Uyghur text recognition, this paper conducted two rounds of data augmentation on a dataset of 7267 real images, resulting in 254,345 and 3,052,140 scene images, respectively. This process partially mitigated the issue of insufficient Uyghur language data, making low-resource scene text recognition research feasible. Experimental results demonstrate that the proposed collaborative encoding approach achieves outstanding performance. Compared to baseline methods, our collaborative encoding approach improves accuracy by 14.1%.

Keywords: scene text recognition; low-resource languages; collaborative encoding; data augmentation

check for updates

Citation: Xu, M.; Zhang, J.; Xu, L.; Silamu, W.; Li, Y. Collaborative Encoding Method for Scene Text Recognition in Low Linguistic Resources: The Uyghur Language Case Study. *Appl. Sci.* **2024**, *14*, 1707. https://doi.org/10.3390/app14051707

Academic Editor: Eleonora Iotti

Received: 17 December 2023
Revised: 14 February 2024
Accepted: 14 February 2024
Published: 20 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Scene text recognition (STR) [1–5] is a computer vision technology designed to extract and convert text from natural images into a computer-understandable character sequence. This technology finds widespread applications in various fields [6,7] including but not limited to autonomous driving, intelligent surveillance, image search, and document recognition. Unlike traditional Optical Character Recognition (OCR) [8–12], scene text recognition needs to address text diversity in natural scenes and tackle challenges such as complex backgrounds, scale variations, lighting differences, and variable fonts.

Traditional scene text recognition typically involves identifying text through the detection of individual characters [13–15]. In contrast, contemporary methods based on deep learning transform the recognition problem into sequence prediction [16], thereby bypassing the intricate feature extraction and classification processes in traditional approaches. Currently, deep learning-based scene text recognition methods have achieved certain achievements. CLIP4STR [17] has excelled in scene text recognition by introducing

the CLIP [18] large language pre-trained model. TrOCR [19] adopts pre-trained image Transformer and text Transformer models for image understanding and word-level text generation. CPDD [20] proposes a context-aware parallel decoder, while MGP-STR [21] predicts characters at different granularities. However, most research currently focuses on languages with abundant resources such as English and Chinese. Yet, there is relatively little research on low-resource languages like Uyghur, Mongolian, Kazakh, and Kyrgyz, which lack real-world scene text recognition data. Mainstream methods primarily rely on Transformer-based frameworks, which are often characterized by more parameters. While this grants the models enhanced representational capabilities, it also intensifies their reliance on substantial amounts of annotated data. As a result, when dealing with limited resources, models based on the Transformer architecture encounter challenges in achieving effective convergence compared to methods based on CNN, leading to relatively poorer performance [22].

Currently, research on Uyghur language text recognition primarily focuses on printed and handwritten text recognition [23–26]. However, there is limited research on scene text recognition for the Uyghur language. Uyghur language characters possess distinctive features, such as varying forms at different positions within words, with some characters having three or more forms. The Uyghur language is written from right to left, and characters can form ligatures during writing, meaning they can connect and transform, creating a continuous character sequence that increases the difficulty of recognizing individual characters. Additionally, the differences between some characters are extremely subtle. All these characteristics pose certain challenges to existing scene text recognition methods.

To address the scarcity of Uyghur scene text data, we have constructed a real-world Uyghur language dataset named the Raw dataset. Using this dataset, we applied data augmentation techniques to generate two extended scene datasets, Aug1 and Aug2. The Raw dataset comprises 9082 street scene images, including road signs, billboards, book titles, and other common street text images. Of these, 7267 images were used for training, and 1815 were reserved for testing. The Aug1 dataset was created by augmenting the 7267 training images from the Raw dataset, resulting in 254,345 images. The Aug2 dataset, generated through a second round of data augmentation on the Aug1 dataset, consists of 3,052,140 scene images. Data augmentation techniques can help somewhat alleviate the problem of insufficient training data for low-resource languages.

In response to the characteristics of Uyghur language characters, this paper adopts a collaborative encoding method using both CNN and Transformer. Firstly, in the convolutional operation of CNN, each kernel slides on the image, focusing on a small region with its receptive field size. This enables CNN to detect similar local features at different positions, such as edges and textures, better capture the differential features between similar characters in Uyghur text and reduce reliance on a large amount of training data. Secondly, the self-attention mechanism of the Transformer establishes associations between different positions, enhancing the model's perceptual ability for contextual information. This contributes to a better understanding of the relationships between Uyghur characters in our research.

The contributions of our article are as follows:

(1) We created a real-world Uyghur text recognition dataset comprising 7267 training images and 1815 test images. Additionally, we employed data augmentation techniques to generate two enhanced datasets, Aug1 and Aug2, containing 254,345 and 3,052,140 scene images, respectively.

(2) In the context of the low-resource Uyghur language, we propose a Collaborative Encoding Method (CEM) for scene text recognition. The encoding consists of three key modules: a Filter module, a Dual-Branch Feature Extraction (DBFE) module, and a Dynamic Fusion (DF) module. The Filter module aims to obtain multi-level low-level features, emphasizing textual information in the images and reducing irrelevant background noise. The DBFE module enables the model to effectively capture global and local details, thereby improving its ability to discern intricate features. The DF

module provides the flexibility to choose the most suitable Transformer and CNN encoding weights, allowing optimal performance.
(3) Our method achieves state-of-the-art results in Uyghur scene text recognition.
(4) This article provides reference examples for other low-resource languages lacking real-world scene text recognition data, such as Mongolian, Kazakh, and Kyrgyz.

The paper's structure is outlined as follows: Section 2 discusses the current research status of scene text recognition. Section 3 provides a detailed introduction to the text recognition method proposed in this paper and its various components. Section 4 offers a thorough presentation and analysis of the data and experiments used in this article. Section 5 summarizes the main achievements of this paper.

2. Related Work

In recent years, large-scale pre-trained language models have demonstrated powerful language perception capabilities across various computer vision tasks [27–32]. Scholars have also applied these models to the field of scene text recognition [17,33–35]. We categorize scene text recognition methods based on their reliance on language models, distinguishing between those with and without such dependencies [19]. Additionally, we classify the usage of language models into two types: Scene Text Recognition Methods Guided by External Language Models and Scene Text Recognition Methods Employing Joint Learning of Internal Language Models.

Scene Text Recognition Methods Without Language Models: These methods primarily rely on visual information for text recognition, disregarding language relationships between characters and typically predicting characters independently. One classical method in this category is based on Connectionist Temporal Classification (CTC) [36]. The Rosetta [37] adopts a two-stage text recognition framework utilizing ResNet and CTC loss. This model accommodates inputs of variable sizes and predicts texts of arbitrary lengths. However, the reliance on monotonic alignment and conditional independence assumptions in CTC-based methods [37–39] results in the neglect of semantic information between characters in the text. With the widespread application of Transformers in the field of computer vision, some approaches [21,40,41] have started to leverage Vision Transformers (ViTs) [22] for scene text recognition. Among them, ViTSTR [40] introduces ViT methods to achieve a straightforward scene text recognition model architecture. SVTR [41] builds upon ViT by incorporating local and global fusion blocks dedicated to extracting stroke-like features and capturing dependencies between characters. Additionally, it integrates a multi-scale backbone network, enabling a multi-granular feature description. Similarly, MGP-STR [21] directly employs ViT for feature extraction but introduces multi-granular character predictions. These methods utilize ViT encoders for feature extraction and then map these features to category labels through fully connected layers, achieving a certain degree of success. However, their exclusive reliance on visual features for prediction results in poorer performance on low-quality images. Given the relatively small size of the Uyghur language real-world scene dataset, methods relying solely on visual information exhibit relatively weaker performance on small datasets. This paper incorporates co-encoding in the encoding phase to address this issue and simultaneously adopts internal language models for joint learning.

Scene Text Recognition Methods Guided by External Language Models: These methods utilize externally trained language models that do not participate in the overall model's backpropagation process. Some external pre-trained methods pre-train all branches of the model or only the language branch [42,43]. ABiNet [42] has separately pre-trained the visual branch and the language branch, employing an iterative correction method to alleviate errors in the visual branch predictions. TrOCR [43] accomplishes image understanding and word-level text generation through pre-trained Image Transformer and Text Transformer models. Recent trends lean toward utilizing large-scale multi-modal models such as CLIP [18], ALIGN [44], Florence [45], and ChatGPT4 for external guidance [17,33,35]. CLIP4STR [35] leverages the pre-trained CLIP model to guide the training of its visual

model. CLIP-OCR [33] introduces a symmetric language feature distillation framework designed to leverage both visual and language knowledge within CLIP to the fullest extent. Despite their outstanding results, training external language models often requires extensive data, and these models are typically designed for resource-rich languages.

Scene Text Recognition Methods Employing Joint Learning of Internal Language Models: Unlike external guided language models, internal language models typically participate in the entire model training process and learn contextual semantic information during training. Methods based on RNN [1,38,46–50] capture the temporal information of characters in the text through RNN, thereby understanding the dependency relationships between characters in the text. The TRBA [1] model adopts BiLSTM for sequence modeling, aiming to better comprehend the contextual information between characters. Existing internal language joint learning methods typically employ the Transformer architecture and integrate corresponding language branches. SRN [51] employs a semantic reasoning network to assist in text recognition, while CDistNet [52] introduces positional query vectors to align visual and semantic features. PARSeq [53] achieves decoding for arbitrary methods by using Permutation Language Modeling, creating connections between arbitrary characters. Despite the excellent results achieved by these methods, they often come with a higher number of parameters, making the model more powerful in terms of representation but also more dependent on a large amount of annotated data. This paper adopts a parallel encoding method with CNN and Transformer to address this issue. By introducing a CNN encoding branch, we reduce the number of layers in the Transformer encoding branch, thereby decreasing the model's reliance on a large amount of annotated data.

3. Methodology

In this section, we present a detailed overview of the overall framework of the Collaborative Encoding Method (CEM). The encoding component of CEM adopts a dual-branch collaboration involving CNN and Transformer. The CNN branch employs ResNet for feature extraction, while the ViT [22] is used for feature extraction. The decoder follows the architecture of Parseq, a baseline model, incorporating two multi-head self-attention (MHA) modules, as illustrated in Figure 1.

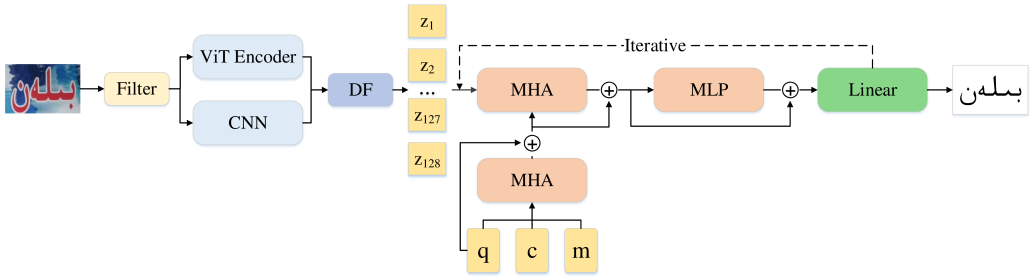


Figure 1. The overall framework of the Collaborative Encoding Method.

3.1. Encoder

The encoder primarily comprises the Filter module, the DBFE module, and the DF module. The DBFE module includes a CNN feature extraction branch and a Transformer feature extraction branch.

3.1.1. Filter Module

We introduce the Filter module to improve the model's learning from scarce training data. It consists of 4 downsampling and 4 upsampling modules, integrating feature information from various levels via skip connections. This module is utilized to extract multi-level low-level features, focusing on textual details in images while minimizing background noise. Figure 2 depicts the structure of the Filter module.

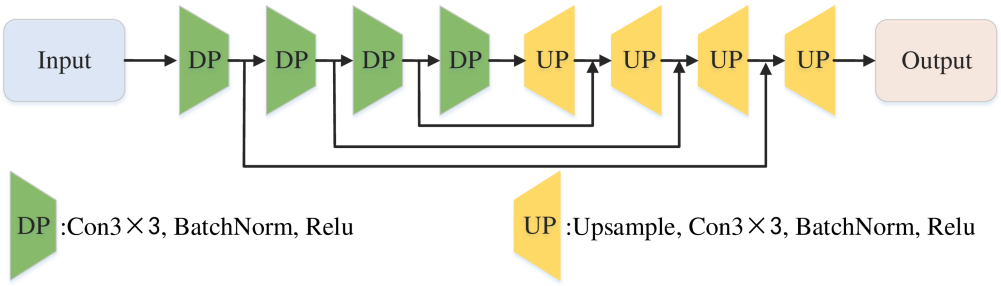


Figure 2. The structure of the Filter module.

3.1.2. Dual-Branch Feature Extraction Module

CNN Feature Extraction Branch: We utilize a ResNet network as the backbone for our CNN feature extraction branch, comprising four blocks. The channel dimensions for these blocks are [64, 128, 256, 384], and the repetition stack counts are [2, 2, 2, 2]. Each residual block comprises one 1 × 1 convolutional kernel and one 3 × 3 convolutional kernel. We introduce the Convolutional Block Attention Module (CBAM) [54] after the ResNet network to obtain more effective features. CBAM applies attention to each channel and spatial position, allowing the network to select and focus on crucial features. This helps reduce sensitivity to irrelevant or redundant information, particularly when training data are limited, enabling better capture of key information.

The CBAM consists of channel and spatial attention modules. The channel attention mechanism preserves the channel dimension while compressing the spatial dimension. By weighting the feature maps of each channel, the network can learn the importance of different channels, thereby enhancing its sensitivity to specific features. The mathematical expression for this mechanism is shown in Equation (1).

$$\begin{aligned} \mathrm{A}_c(F_x) &= \sigma(\mathrm{MLP}(\mathrm{AvgPool}(F_x)) + \mathrm{MLP}(\mathrm{MaxPool}(F_x))) \\ &= \sigma(W_1(W_0(F^c_{avg})) + W_1(W_0(F^c_{max}))) \end{aligned} \tag{1}$$

F_x represents the feature map output of the Filter module, represented as $C \times H \times W$, where C is the number of channels, H is the height, and W is the width. AvgPool and MaxPool denote average pooling and max pooling operations conducted in the spatial dimension to obtain each channel's maximum and average values. F^c_{avg} represents the average value on channel c, and F^c_{max} represents the maximum value on channel c, where $c \in C$. MLP is a multi-layer perceptron utilized for learning the weights and relationships between channels. α represents the Sigmoid function, normalizing the attention weights of each channel to the range (0, 1) used in generating channel attention feature maps. W_0 and W_1 are the learned weight parameters.

The spatial attention mechanism preserves spatial dimensions while compressing channel dimensions. By weighting the spatial dimensions of the feature map, the network can allocate different weights to different spatial locations, thus focusing more on crucial areas in the image. This enhances the network's ability to perceive local spatial information. The mathematical expression is shown in Equation (2).

$$\begin{aligned} \mathrm{A}_s(\mathrm{A}_c F_x) &= \sigma(\mathrm{f}^{7\times7}(\mathrm{Cat}[\mathrm{AvgPool}(\mathrm{A}_c F_x); \mathrm{MaxPool}(\mathrm{A}_c F_x)])) \\ &= \sigma(\mathrm{f}^{7\times7}(\mathrm{Cat}[F^s_{avg}; F^s_{max}]) \end{aligned} \tag{2}$$

$\mathrm{A_c}$ represents the output of the channel attention mechanism. F_x represents the feature map output of the Filter module. The AvgPool and MaxPool indicate the operations of average and max pooling. In MaxPool, the maximum value is extracted along the channels with the extraction times being the product of height and width. Similarly, AvgPool extracts the average value along the channels with extraction times also being the product of height

and width. The Cat operation combines the feature maps extracted by MaxPool and AvgPool (with a single channel), resulting in a 2-channel feature map. $f^{7\times7}$ represents a convolution operation with a kernel size of 7×7, reducing the number of channels from 2 to 1. α represents the Sigmoid function, normalizing the attention weights of the regions to the range (0, 1) used for generating spatial attention feature maps.

Transformer Feature Extraction Branch: In this article, we employed the ViT [22] encoder as the feature extraction component of the Transformer, comprising 6 ViT layers, excluding classification heads and the [CLS] token. The input image F_x is consistent with the input to the CNN branch, representing the feature map output of the filtering module with a width of W, a height of H, and a channel count of C. The input image F_x is divided into N patches, denoted as x_p, where $N = WH/wh$, with w and h representing the length and width of each patch. In this paper, $w = 8$, $h = 4$. Subsequently, each patch is fed into the embedding layer, producing N tokens. Using the "Add" operation, we incorporate position encoding into each token. Finally, all tokens are input into Transformer encoding blocks, each encoding block consists of a multi-head self-attention (MSA) mechanism, layer normalization (LN), multi-layer perceptron (MLP), and residual connections. We represent the feature extraction process of the Transformer branch through Equation (3).

$$\begin{aligned} z_0 &= \left[x_p^1 E; x_p^2 E; \cdots; x_p^N E\right] + E_{pos}, \\ z_l' &= \mathrm{MSA}(LN(z_{l-1})) + z_{l-1}, \quad l = 1, \cdots, L \\ z_l &= \mathrm{MLP}(LN(z'_l)) + z'_l, \qquad l = 1, \cdots, L \\ F_t &= z_L \end{aligned} \tag{3}$$

where $E \in R^{(wh\cdot C)\times D}$ represents the linear projection matrix, and $E_{pos} \in R^{(N+1)\times D}$ corresponds to positional encoding. The variable z_0 denotes the input to Transformer encoding blocks, where L represents the number of encoder blocks set to $L = 6$ in this article. Additionally, F_t represents the output of the final encoding module in Transformer.

3.1.3. DF Module

We have introduced a dynamic fusion module to facilitate the integration of features from the CNN and Transformer branches. This module concatenates features from different branches using the Cat operation and activates the concatenated features with the Sigmoid activation function to obtain attention weights. Subsequently, we multiply the respective weights with features from different branches. This design enables the model to dynamically adjust the fusion degree during the training, determining the relative importance of the CNN and Transformer branches. This process is specifically represented by Equation (4).

$$\begin{aligned} F_u &= \mathrm{Cat}(F_c + F_t) \\ F_a &= \sigma(F_u) \\ F_o &= F_a \times F_c + (1 - F_a) F_t \end{aligned} \tag{4}$$

where F_c and F_t represent features from the CNN and Transformer branches, respectively. α denotes the Sigmoid activation function. F_u is the feature obtained after fusion using the concatenation operation. F_a is the attention score matrix. F_o is the output feature after dynamic fusion.

3.2. Decoder

The decoder adopts the same architecture as the Parseq [53] decoder and employs two MHA modules. The inputs include the query vector q (trainable vector), context information c (utilized with real labels during training), and image features z from the encoder. The query vector q and context information c in the first MHA module are inputs. The primary objective of this module is to determine the most relevant context

information for the current decoding step through the self-attention mechanism. This lets the decoder effectively utilize previous context information during decoding, enhancing the model's performance.

$$h_c = q + \mathrm{MHA}(q, c, c, m) \in R^{(T+1)\times d_{model}} \tag{5}$$

T represents the length of contextual information, and m denotes an optional attention mask. During the training process, we drew inspiration from PARSeq's Permutation Language Modeling [53] approach to enhance the correlation between characters at different positions. We employed a strategy of random masking, achieving different decoding sequences through various masking methods. During training, masks were generated from random permutations. However, we followed the conventional approach during the inference and testing phases, masking in a left-to-right sequence.

The input to the second MHA includes the output from the previous MHA and image features z from the encoder. Notably, the second MHA module does not use a mask when processing inputs, ensuring comprehensive information consideration.

$$h_i = h_c + \mathrm{MHA}(h_c, z, z) \in R^{(T+1)\times d_{model}} \tag{6}$$

Subsequently, we take the output of the second MHA as input, and it undergoes computation through an MLP, following Equation (7).

$$h_{dec} = h_i + \mathrm{MLP}(h_i) \in R^{(T+1)\times d_{model}} \tag{7}$$

In the end, we obtained the final output through a linear model.

$$y = \mathrm{Linear}(h_{dec}) \in R^{(T+1)\times(S+1)} \tag{8}$$

The size of the character set used for training is denoted as S. Due to the inclusion of a character end marker, the character length and category length are processed with an additional increment of 1.

4. Experiments

4.1. Datasets

Our experimental data were collected in the Hotan region of Xinjiang by the Multilingual Information Technology Laboratory for Scene Text Detection and Recognition at Xinjiang University. Based on this, we constructed a dataset (referred to as the Raw dataset) to support the experiments and results of this article. The Raw dataset comprises images of street texts, including road signs, billboards, book titles, museum exhibit labels, architectural signage, and other common textual elements. In the data processing stage, we conducted a series of meticulous and systematic operations, including screening, annotation, cropping, calibration, etc., to ensure the quality and accuracy of the data. We excluded heavily occluded, extremely distorted, and repetitive data in this article during the screening process. Subsequently, we annotated the screened data. We performed re-cropping in cases of unreasonable cropping, such as images with excessive blank spaces or cropped to other interfering words. We conducted a cross-check of the annotated data to ensure the consistency between labels and image content.

To further enhance the data's credibility, this paper's authors conducted three rounds of calibration work on the scene recognition images and their labels.

We obtained a dataset of 9082 scene images with 7267 for training and 1815 for testing. To ensure the rationality of the training and testing datasets during the dataset partitioning process, we allocated the majority of compliant categories in a ratio of 8:2. For categories that did not meet the requirements, such as those with fewer than two data samples, we randomly retained data from these categories in either the training or testing set. Figure 3 illustrates some samples from the training and testing datasets.



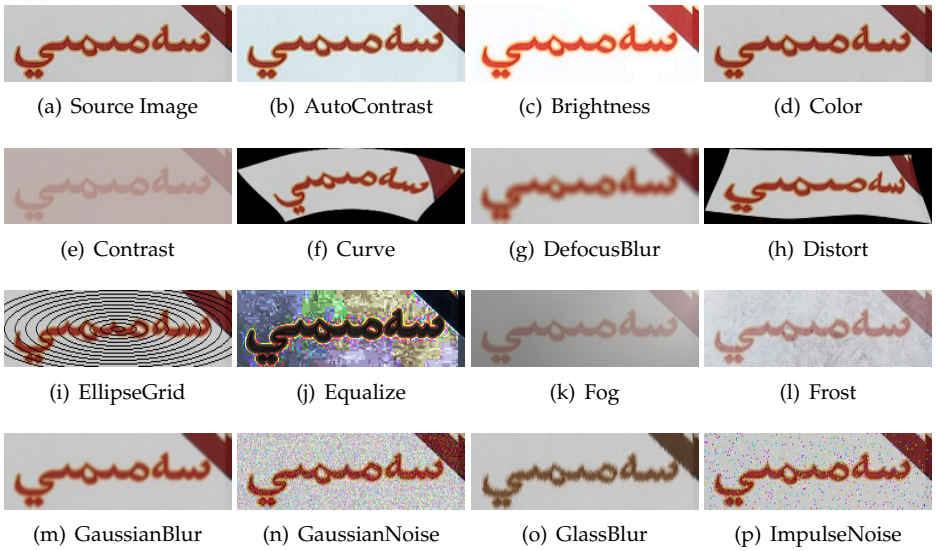
(a) Train (b) Test

Figure 3. Samples of training and testing datasets.

4.2. Data Augmentation

The dataset's quality is crucial for achieving satisfactory model performance. However, the quantity of the real dataset Raw used in this paper was insufficient to meet the experimental requirements. Therefore, we performed two rounds of data augmentation on the experimental data. We employed the STR Augmentation method proposed by Google, which covers eight categories of augmentation methods: distortion, geometry, pattern, blur, camera, processing, noise, and weather. Each category includes multiple specific augmentation functions, totaling 38 functions used for augmentation.

Our data augmentation strategy is grounded in our prior research [55]. The approach utilized in earlier research involved employing Google's default set of 38 augmentation methods in the initial round of data augmentation. Following that, in the second round, a refined selection was made, consisting of 12 methods, namely Curve, Distort, Stretch, Rotate, Perspective, Shrink, TranslateX, TranslateY, Contrast, Brightness, JpegCompression, and Pixelate. We conducted the initial round of data augmentation on the original dataset, consisting of 7267 scene images. In contrast to previous research methods, we omitted the Grid, VGrid, and HGrid methods, as we observed that these three methods produced entirely black images after augmentation. We opted for the remaining 35 augmentation functions from STRAug, including Curve, Rotate, Perspective, Distort, Stretch, Shrink, TranslateX, TranslateY, Contrast, Brightness, JpegCompression, Pixelate, RectGrid, EllipseGrid, GaussianNoise, ShotNoise, ImpulseNoise, SpeckleNoise, GaussianBlur, DefocusBlur, MotionBlur, GlassBlur, ZoomBlur, Fog, Snow, Frost, Rain, Shadow, Posterize, Solarize, Invert, Equalize, AutoContrast, Sharpness, and Color. This led to the creation of the Aug1 dataset, which encompasses 254,345 scene images. Examples of augmented images are depicted in Figure 4.



(a) Source Image (b) AutoContrast (c) Brightness (d) Color

(e) Contrast (f) Curve (g) DefocusBlur (h) Distort

(i) EllipseGrid (j) Equalize (k) Fog (l) Frost

(m) GaussianBlur (n) GaussianNoise (o) GlassBlur (p) ImpulseNoise

Figure 4. *Cont.*

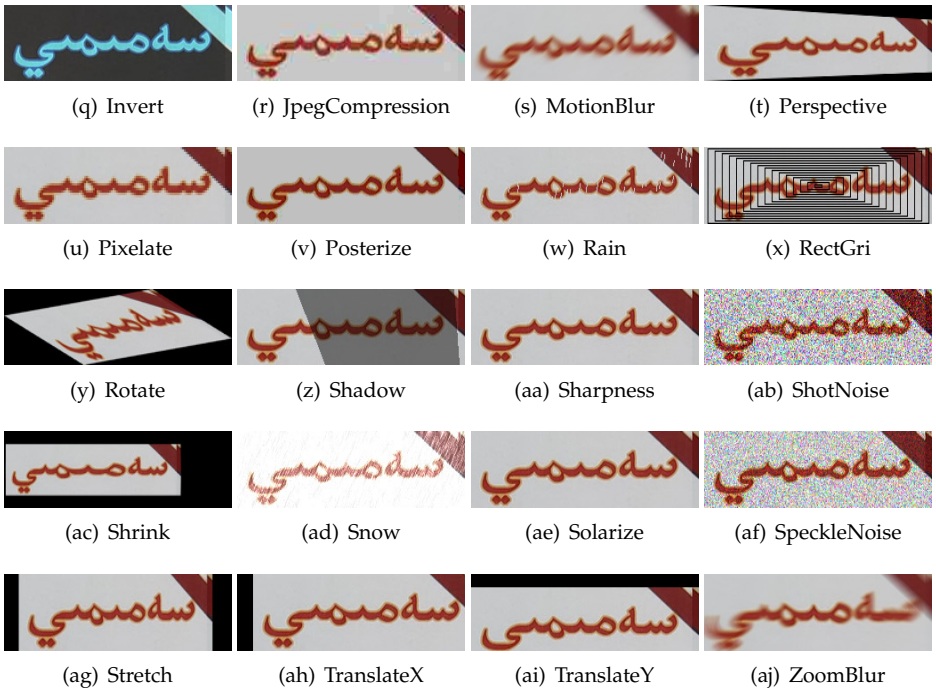


Figure 4. Examples of augmented images.

Subsequently, we applied the second round of augmentation to the Aug1 dataset, selecting 12 augmentation functions: Curve, Distort, Stretch, Rotate, Perspective, Shrink, TranslateX, TranslateY, Contrast, Brightness, JpegCompression, and Pixelate. This led to the creation of the Aug2 dataset, comprising 3,052,140 scene images.

4.3. Experimental Setup

We set the resolution of the image input to 32 × 128, with a batch size of 224, and conducted 20 training epochs. In the initial stage, we chose Adam [56] as the optimizer and used OneCycleLR to adjust the learning rate. After completing the training for the first 14 batches, we replaced OneCycleLR with Stochastic Weight Average (SWA). The batch size for all methods in the experiment is fixed at 224 with a training batch count of 20. All experiments were conducted in an environment with 6 NVIDIA Tesla V100 GPUs, utilizing Python 3.9.18 and Torch 1.13.1. The parameter settings for different methods in the experiment are outlined in Table 1.

Table 1. Experimental parameter configuration. En_layer represents the number of encoder layers, and De_layer represents the number of decoder layers.

Models	Optimizer	lr	Image_Size	Patch_Size	En_Layer	De_Layer
CRNN	Adam	5.1×10^{-4}	[32, 128]	–	–	–
TRBA	Adam	6.9×10^{-4}	[32, 128]	–	–	–
ViTSTR	Adam	8.9×10^{-4}	[224, 224]	[16, 16]	12	–
PARSeq	Adam	7.0×10^{-4}	[32, 128]	[4, 8]	12	1
CDistNet	Adam	4.0×10^{-4}	[32, 128]	–	3	3
MGP-tiny-char	Adadelta	1	[32, 128]	[4, 4]	12	–
CEM(ours)	Adam	7.0×10^{-4}	[32, 128]	[4, 8]	6	1

This article considers word accuracy the primary metric for measuring scene text recognition performance. The character set used in this paper includes 33 Uyghur characters, as shown in Figure 5.



Figure 5. Character set of the Uyghur.

4.4. Comparative Analysis with Existing Methods

To demonstrate the effectiveness of our approach, we conducted a series of experiments on the Raw, Aug1, and Aug2 datasets, employing both the most current and classic methods. To ensure fairness in our comparisons, our selected methods encompass various aspects, including visual Transformers like ViTSTR [40] and MGP-tiny-char [21]. Additionally, classic methods for feature extraction using CNN, such as CRNN [38] and TRBA [1], were also included. Additionally, we incorporated the CDistNet method, which combines CNN and Transformer for joint feature extraction [52]. To serve as the baseline for this article, we introduced the PARSeq method [53], involving the joint learning of internal language models.

When training exclusively on the original dataset, the method described in Table 2 failed to converge. Consequently, we opted not to present the data in Table 2. Therefore, data augmentation is considered a crucial prerequisite for text recognition in low-resource scenarios. As depicted in Table 2, our approach achieved outstanding results by training solely on the Raw + Aug1 dataset, achieving an accuracy of 94.1%, significantly outperforming other methods. Compared to the baseline model PARSeq, our method demonstrated a 48.4% accuracy improvement on the Raw+Aug1 dataset and an 11.7% improvement on the Raw + Aug1 + Aug2 dataset. Overall, irrespective of the training set conditions, our method outperforms the baseline model by 14.1% in terms of accuracy.

Table 2. Comparison experiment with other methods. CEM-tiny is a model in the CNN branch that employs only three convolutional operations. At the same time, the CEM-base is a model in the CNN branch utilizing the architecture of ResNet17, as detailed in Section 4.5. A1 and A2 represent the datasets Aug1 and Aug2, respectively.

Models	**Test Accuracy (%) / Train Dataset**		**Params (M)**
	Raw + A1	**Raw + A1 + A2**	
CRNN	78.7	80.2	8.3
TRBA	87.2	90.3	49.8
ViTSTR	-	73.1	21.7
PARSeq	45.7	80.0	23.8
CDistNet	64.7	82.5	65.5
MGP-tiny-char	27.2	33.6	5.4
CEM-tiny(ours)	91.7	90.4	24.1
CEM-base(ours)	94.1	91.7	26.2

The accuracy of our approach on the Raw + Aug1 + Aug2 dataset was lower than that on the Raw + Aug1 dataset. This is attributed to our method's ability to obtain sufficiently stable features with Raw + Aug1, whereas Aug2, an additional augmentation on Aug1, introduced more unstable features, adversely affecting the feature extraction process. Therefore, achieving excellent results can be accomplished using the Raw + Aug1 dataset alone. This also indirectly substantiates that our approach can, to some extent, mitigate reliance on training data. Although our method's accuracy on Raw + Aug1 + Aug2 was lower than the model trained only on Raw + Aug1, it still outperformed other models trained on Raw + Aug1 + Aug2.

We also observe that the training results of other models on Raw + Aug1 + Aug2 are superior to their training results on the Raw + Aug1 dataset. This is because other methods have higher requirements for data volume. When the training data are limited, they cannot obtain sufficient information, further demonstrating the effectiveness of the method proposed in this paper.

In Table 2, significant differences among the data are evident, particularly with ViTSTR and MGP exhibiting poorer performance. This discrepancy can be attributed to both methods utilizing ViT's pre-training data during training, leading to subpar results in Uyghur scene text recognition due to language variations. The use of a pure Transformer architecture, coupled with the relatively small size of our dataset, contributes to this experimental situation. We opted for the MGP method's tiny version after initially validating its base version, which failed to converge.

CRNN and TRBA, employing CNN-based feature extraction methods, excel in scenarios with limited training data. TRBA, in particular, demonstrates outstanding performance. CDistNet, a technique combining CNN and Transformer sequential encoding, also yields better outcomes in scenarios with more scarce training data than other self-attention encoding methods. PARSeq, utilizing a Permutation Language Modeling strategy, achieves favorable results on the Raw + A1 + A2 dataset by establishing associations between arbitrary characters. However, based on the Transformer architecture, its performance is slightly inferior to that of CRNN, TRBA, and CDistNet.

4.5. Ablation Experiments

We conducted a series of ablation experiments to validate the effectiveness of the proposed collaborative encoding method for text recognition in low-resource Uyghur language scenarios. Firstly, we performed detailed ablation experiments on the filter, CNN feature extraction branches, and dynamic fusion modules. The main experimental groups include the following:

(1) Using the filter module before the dual-branch feature extraction module and not using the filter module.
(2) Utilizing three consecutive convolutional layers, ResNet17, and ResNet33 for the CNN branch.
(3) Applying the CBAM module and not applying the CBAM module in the CNN branch.
(4) Employing the dynamic fusion module and not employing the dynamic fusion module.
(5) Investigating the impact of different encoding layers in the Transformer branch on model performance.

To present the experimental results more clearly, we organized experiments (1) to (4) into groups in Table 3, providing detailed descriptions of the specific settings in the table. In Table 3, the Transformer branch in the experiments uniformly adopted a six-layer ViT encoder. The results of the experiment group (5) are presented in detail in Table 4.

Observing the data in the 1st and 2nd rows of Table 3, when only three convolutional operations are employed in the CNN branch and a filtering module is incorporated before the dual-branch feature extraction module, the model achieves an 8.2% increase in accuracy compared to the model without the filtering module. This is because the CNN branch employs only three consecutive convolution operations. Without the filtering module, the CNN branch's contribution is limited, leading to excessive reliance on the encoding module on the Transformer branch and hindering model convergence. By examining the data in the 5th and 6th rows of Table 3, it can be observed that when using ResNet17 for feature extraction in the CNN branch, incorporating a filtering module before the dual-branch feature extraction module led to a 0.6% increase in model accuracy compared to the model without the filtering module. Based on the data in the 1st, 2nd, 5th, 6th, 9th, and 10th rows, it can be observed that the filter module has the greatest impact when the CNN branch employs only three convolution operations. When the CNN branch uses ResNet17, its effect is reduced, but the accuracy is still 0.6% higher than models that do not use the filter module. However, data from the 9th and 10th rows indicate that enhancing the CNN

branch cannot fully replace the role of the filter module. In this case, even though the CNN branch adopts the architecture of ResNet33, models using the filter module still achieve an accuracy improvement of 0.8%. This further demonstrates that the filter module can help the model better extract text features and reduce the influence of background noise.

Table 3. Ablation experiments of the Filter module, CNN Feature Extraction Branch, and DF module. In the Filter module, the symbol ✗ indicates that the Filter module is not used, and the symbol ✓indicates that the Transformer branch and the CNN branch share a common Filter module. For 3-Con, it denotes three consecutive convolution operations. In the CBAM and dynamic fusion modules, ✓and ✗, respectively, represent the utilization and non-utilization of the module.

Filter	CNN	CBAM	DF	Test Accuracy (%) / Train Dataset
				Raw + A1
✓	3-Con	✓	✓	91.7
✗	3-Con	✓	✓	83.5
✓	3-Con	✗	✓	91.5
✓	3-Con	✓	✗	91.6
✓	ResNet17	✓	✓	94.1
✗	ResNet17	✓	✓	93.5
✓	ResNet17	✗	✓	93.2
✓	ResNet17	✓	✗	92.3
✓	ResNet33	✓	✓	93.4
✗	ResNet33	✓	✓	92.6
✓	✗	✓	✓	79.2

The analysis of the data in the 1st and 3rd rows found that the accuracy of the model improved by 0.2% when the CBAM module was incorporated into the CNN branch using a lightweight architecture. Similarly, based on the data in the 5th and 7th rows, it can be inferred that the accuracy of the model increased by 0.9% when the CBAM module was integrated into the CNN branch using the ResNet17 architecture. This further validates that CBAM can better assist the model in extracting key features, thereby improving the model's accuracy.

The experimental data in the 1st and 4th rows, and the 5th and 8th rows, demonstrate that when the CNN branch adopts a lightweight architecture, the model's accuracy increases by 0.1% with the use of the dynamic fusion module. When the CNN branch adopts the ResNet17 architecture, the model's accuracy improves by 1.8% after employing the dynamic fusion module. This is because with the dynamic fusion module, the model can dynamically select the most suitable proportion of CNN and Transformer branches during the training process, maximizing both advantages. Without the dynamic fusion module, directly adding the features of the two branches somewhat diminishes their respective strengths.

Combining the experimental results from the 1st, 5th, and 9th rows, it is evident that the CNN branch module plays a vital role in the collaborative coding method for Uyghur text recognition in low-resource scenarios. When using a 17-layer ResNet architecture, the accuracy improves by 2.4% compared to the model using only three convolutional modules. However, when using a 33-layer ResNet architecture, although the model surpasses the one using only three convolutional operations, the accuracy is lower than that of the 17-layer ResNet architecture. This may be due to the excessive complexity of the CNN branch structure, causing the model parameters to be too many for the limited dataset, leading to incomplete model convergence.

Data from the experiment's 1st, 5th, and final rows indicate that the model performs poorly when the CNN branch is not utilized. This further supports that the CNN-Transformer dual-branch cooperative encoding can obtain more comprehensive feature representations and encoding.

To verify the impact of different encoding layer numbers in the Transformer branch on the model in the collaborative coding method proposed in this paper for Uyghur text recognition in low-resource scenarios, experiments were designed with Transformer branch encoding layer numbers of 6, 8, and 12. Meanwhile, a lightweight model with three convolutional operations was adopted in the CNN branch, as shown in Table 4. All experiments in Table 4 used the Filter module, CBAM, and DF methods.

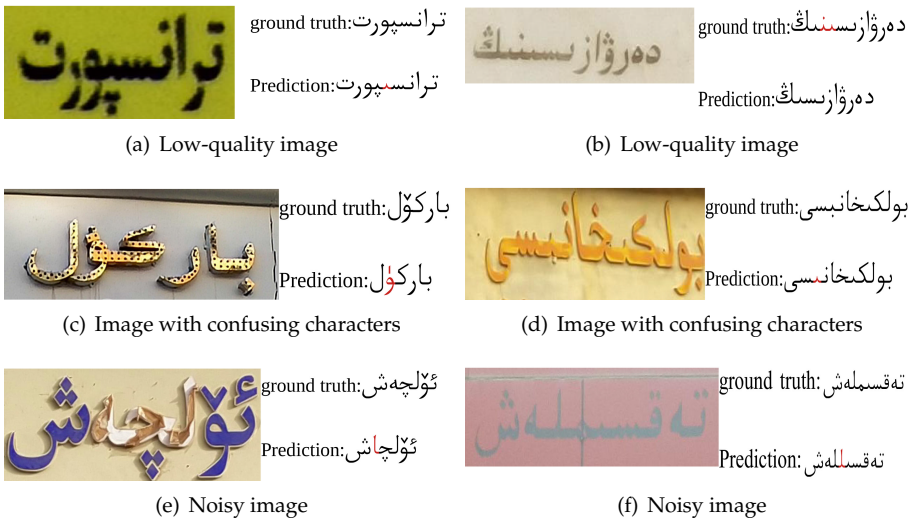
Table 4. Ablation experiments of the number of encoding layers in the Transformer branch.

Transformer Encoder Layers	Test Accuracy (%) / Train Dataset		Params (M)
	Raw + A1	Raw + A1 + A2	
6	91.7	90.4	24.1 M
8	91.7	90.2	27.7 M
12	91.1	90.4	34.8 M

Table 4 shows that the model performs best when the Transformer branch adopts a 6-layer ViT layer. This is because in situations with limited training data, an excessive number of encoder layers can lead to an increase in parameters, adversely affecting the convergence of the model. Therefore, in this article, the collaborative coding method chose a 6-layer ViT layer for the Transformer branch. This research result emphasizes the importance of selecting an appropriate number of encoder layers for model performance in text recognition tasks in low-resource contexts. Our experimental results provide valuable references for designing more effective text recognition models tailored to low-resource contexts.

4.6. Failure Cases

Figure 6 illustrates some instances of prediction errors. From the figure, it can be observed that the reasons for prediction failures include poor image quality, as in Figure 6a,b, and situations where character features in the images are prone to confusion, as in Figure 6c,d. Additionally, noise interference is also a contributing factor to failures, especially in examples like Figure 6e,f.



(a) Low-quality image (b) Low-quality image

(c) Image with confusing characters (d) Image with confusing characters

(e) Noisy image (f) Noisy image

Figure 6. Examples of incorrect predictions. We employ red markings to distinguish segments where predicted characters deviate from the ground truth.

5. Conclusions

Research methods for text recognition in low-resource scenarios are relatively limited. Existing studies primarily focus on languages with abundant resources, and most state-

of-the-art methods are based on the Transformer architecture, showing poor adaptability to low-resource datasets. This article specifically targets low-resource Uyghur text and introduces a collaborative encoding approach. The encoder comprises a filtering module, a dual-branch feature extraction module, and a dynamic fusion module. Our method achieves state-of-the-art results in Uyghur text scene recognition. Additionally, we have curated a genuine natural scene Uyghur text recognition dataset and two synthetic datasets, Aug1 and Aug2. These datasets provide robust data support for Uyghur text scene recognition research. Our method can somewhat mitigate the reliance on training data while also providing reference examples for other low-resource languages that lack real-world scene text recognition data, such as Mongolian, Kazakh, and Kyrgyz.

In our future research, we will pursue our work in two main aspects. Firstly, we intend to expand our existing real-world Uyghur language dataset. Secondly, we are preparing to shift our next research objectives to other low-resource languages, such as Mongolian and Kazakh.

Author Contributions: M.X.: Methodology, Software, Writing—original draft. L.X.: Visualization, Investigation. J.Z.: Writing—review and editing. Y.L.: Supervision. W.S.: Conceptualization. All authors have read and agreed to the published version of the manuscript.

Funding: The funds to support this work come from the Joint Funds of the National Natural Science Foundation of China Joint Fund Project under Grant U1911401 hosted by Wushour Silamu, and the Shenzhen Municipal Science and Technology Innovation Committee Project under Grant GJGJZD20210408092806017 hosted by Wushour Silamu.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting this study's findings are available on request from the corresponding author or first author, upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. 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 Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 4715–4723.
2. Liu, X.; Meng, G.; Pan, C. Scene text detection and recognition with advances in deep learning: A survey. *Int. J. Doc. Anal. Recognit.* **2019**, *22*, 143–162. [CrossRef]
3. Long, S.; He, X.; Yao, C. Scene text detection and recognition: The deep learning era. *Int. J. Comput. Vis.* **2021**, *129*, 161–184. [CrossRef]
4. Ye, Q.; Doermann, D. Text detection and recognition in imagery: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 1480–1500. [CrossRef]
5. Zhu, Y.; Yao, C.; Bai, X. Scene text detection and recognition: Recent advances and future trends. *Front. Comput. Sci.* **2016**, *10*, 19–36. [CrossRef]
6. Zhang, C.; Tao, Y.; Du, K.; Ding, W.; Wang, B.; Liu, J.; Wang, W. Character-level street view text spotting based on deep multisegmentation network for smarter autonomous driving. *IEEE Trans. Artif. Intell.* **2021**, *3*, 297–308. [CrossRef]
7. Tsai, S.S.; Chen, H.; Chen, D.; Schroth, G.; Grzeszczuk, R.; Girod, B. Mobile visual search on printed documents using text and low bit-rate features. In Proceedings of the 2011 18th IEEE International Conference on Image Processing, Brussels, Belgium, 11–14 September 2011; pp. 2601–2604.
8. Islam, N.; Islam, Z.; Noor, N. A survey on optical character recognition system. *arXiv* **2017**, arXiv:1710.05703.
9. Sabu, A.M.; Das, A.S. A survey on various optical character recognition techniques. In Proceedings of the 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), Namakkal, India, 2–3 March 2018; pp. 152–155.
10. Nagy, G. Twenty years of document image analysis in PAMI. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 38–62. [CrossRef]
11. Chaudhuri, A.; Mandaviya, K.; Badelia, P.; Ghosh, S.K. *Optical Character Recognition Systems for Different Languages with Soft Computing*; Springer: Berlin/Heidelberg, Germany, 2019.
12. Somerville, P.J. Method and Apparatus for Barcode Recognition in a Digital Image. U.S. Patent 4,992,650, 12 February 1991.
13. Wang, K.; Babenko, B.; Belongie, S. End-to-end scene text recognition. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 1457–1464.

14. Wang, K.; Belongie, S. Word spotting in the wild. In Proceedings of the Computer Vision—ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Greece, 5–11 September 2010; Proceedings, Part I 11; Springer: Berlin/Heidelberg, Germany, 2010; pp. 591–604.
15. Yao, C.; Bai, X.; Shi, B.; Liu, W. Strokelets: A learned multi-scale representation for scene text recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 4042–4049.
16. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *arXiv* **2014**, arXiv:1409.3215 .
17. Rang, M.; Bi, Z.; Liu, C.; Wang, Y.; Han, K. Large OCR Model: An Empirical Study of Scaling Law for OCR. *arXiv* **2023**, arXiv:2401.00028.
18. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 8748–8763.
19. Fujitake, M. Dtrocr: Decoder-only transformer for optical character recognition. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 4–8 January 2024; pp. 8025–8035.
20. Du, Y.; Chen, Z.; Jia, C.; Yin, X.; Li, C.; Du, Y.; Jiang, Y.G. Context Perception Parallel Decoder for Scene Text Recognition. *arXiv* **2023**, arXiv:2307.12270.
21. Wang, P.; Da, C.; Yao, C. Multi-granularity prediction for scene text recognition. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Cham, Switzerland, 2022; pp. 339–355.
22. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16 × 16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
23. Li, P.; Zhu, J.; Peng, L.; Guo, Y. RNN based Uyghur text line recognition and its training strategy. In Proceedings of the 2016 12th IAPR Workshop on Document Analysis Systems (DAS), Santorini, Greece, 11–14 April 2016; pp. 19–24.
24. Ibrayim, M.; Simayi, W.; Hamdulla, A. Unconstrained online handwritten Uyghur word recognition based on recurrent neural networks and connectionist temporal classification. *Int. J. Biom.* **2021**, *13*, 51–63. [CrossRef]
25. Li, W.; Mahpirat; Kang, W.; Aysa, A.; Ubul, K. Multi-lingual Hybrid Handwritten Signature Recognition Based on Deep Residual Attention Network. In Proceedings of the Biometric Recognition: 15th Chinese Conference, CCBR 2021, Shanghai, China, 10–12 September 2021; Proceedings 15; Springer: Cham, Switzerland, 2021; pp. 148–156.
26. Xamxidin, N.; Mahpirat.; Yao, Z.; Aysa, A.; Ubul, K. Multilingual Offline Signature Verification Based on Improved Inverse Discriminator Network. *Information* **2022**, *13*, 293. [CrossRef]
27. Song, H.; Dong, L.; Zhang, W.N.; Liu, T.; Wei, F. Clip models are few-shot learners: Empirical studies on vqa and visual entailment. *arXiv* **2022**, arXiv:2203.07190.
28. Patashnik, O.; Wu, Z.; Shechtman, E.; Cohen-Or, D.; Lischinski, D. Styleclip: Text-driven manipulation of stylegan imagery. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2085–2094.
29. Kim, G.; Kwon, T.; Ye, J.C. Diffusionclip: Text-guided diffusion models for robust image manipulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 2426–2435.
30. Luo, H.; Ji, L.; Zhong, M.; Chen, Y.; Lei, W.; Duan, N.; Li, T. Clip4clip: An empirical study of clip for end to end video clip retrieval and captioning. *Neurocomputing* **2022**, *508*, 293–304. [CrossRef]
31. Subramanian, S.; Merrill, W.; Darrell, T.; Gardner, M.; Singh, S.; Rohrbach, A. Reclip: A strong zero-shot baseline for referring expression comprehension. *arXiv* **2022**, arXiv:2204.05991.
32. Hessel, J.; Holtzman, A.; Forbes, M.; Bras, R.L.; Choi, Y. Clipscore: A reference-free evaluation metric for image captioning. *arXiv* **2021**, arXiv:2104.08718.
33. Wang, Z.; Xie, H.; Wang, Y.; Xu, J.; Zhang, B.; Zhang, Y. Symmetrical Linguistic Feature Distillation with CLIP for Scene Text Recognition. In Proceedings of the 31st ACM International Conference on Multimedia, Ottawa, ON, Canada, 29 October–3 November 2023; pp. 509–518.
34. Yu, H.; Wang, X.; Li, B.; Xue, X. Chinese Text Recognition with A Pre-Trained CLIP-Like Model through Image-IDS Aligning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023; pp. 11943–11952.
35. Zhao, S.; Wang, X.; Zhu, L.; Yang, Y. CLIP4STR: A Simple Baseline for Scene Text Recognition with Pre-trained Vision-Language Model. *arXiv* **2023**, arXiv:2305.14014.
36. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 369–376.
37. Borisyuk, F.; Gordo, A.; Sivakumar, V. Rosetta: Large scale system for text detection and recognition in images. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 71–79.
38. 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 Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 2298–2304. [CrossRef]
39. He, P.; Huang, W.; Qiao, Y.; Loy, C.; Tang, X. Reading scene text in deep convolutional sequences. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.

40. Atienza, R. Vision transformer for fast and efficient scene text recognition. In Proceedings of the International Conference on Document Analysis and Recognition, Lausanne, Switzerland, 5–10 September 2021; Springer: Cham, Switzerland, 2021; pp. 319–334.
41. Du, Y.; Chen, Z.; Jia, C.; Yin, X.; Zheng, T.; Li, C.; Du, Y.; Jiang, Y.G. Svtr: Scene text recognition with a single visual model. *arXiv* **2022**, arXiv:2205.00159.
42. Fang, S.; Xie, H.; Wang, Y.; Mao, Z.; Zhang, Y. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 7098–7107.
43. Li, M.; Lv, T.; Chen, J.; Cui, L.; Lu, Y.; Florencio, D.; Zhang, C.; Li, Z.; Wei, F. Trocr: Transformer-based optical character recognition with pre-trained models. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 13094–13102.
44. Jia, C.; Yang, Y.; Xia, Y.; Chen, Y.T.; Parekh, Z.; Pham, H.; Le, Q.; Sung, Y.H.; Li, Z.; Duerig, T. Scaling up visual and vision-language representation learning with noisy text supervision. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 4904–4916.
45. Yuan, L.; Chen, D.; Chen, Y.L.; Codella, N.; Dai, X.; Gao, J.; Hu, H.; Huang, X.; Li, B.; Li, C.; et al. Florence: A new foundation model for computer vision. *arXiv* **2021**, arXiv:2111.11432.
46. Li, W.; Zhang, Y.; Huang, Y.; Shen, Y.; Wang, Z. A Three-Stage Uyghur Recognition Model Combining the Attention Mechanism and Different Convolutional Recurrent Networks. *Appl. Sci.* **2023**, *13*, 9539. [CrossRef]
47. Ibrayim, M.; Mattohti, A.; Hamdulla, A. An effective method for detection and recognition of uyghur texts in images with backgrounds. *Information* **2022**, *13*, 332. [CrossRef]
48. Wang, Y.; Ao, N.; Guo, R.; Mamat, H.; Ubul, K. Scene Uyghur Recognition with Embedded Coordinate Attention. In Proceedings of the 2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML), Chengdu, China, 22–24 July 2022; pp. 253–260.
49. Liu, Y.; Kong, F.; Xu, M.; Silamu, W.; Li, Y. Scene Uyghur Recognition Based on Visual Prediction Enhancement. *Sensors* **2023**, *23*, 8610. [CrossRef]
50. Lee, C.Y.; Osindero, S. Recursive recurrent nets with attention modeling for ocr in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2231–2239.
51. Yu, D.; Li, X.; Zhang, C.; Liu, T.; Han, J.; Liu, J.; Ding, E. Towards accurate scene text recognition with semantic reasoning networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 12113–12122.
52. Zheng, T.; Chen, Z.; Fang, S.; Xie, H.; Jiang, Y.G. Cdistnet: Perceiving multi-domain character distance for robust text recognition. *arXiv* **2023**, arXiv:2111.11011.
53. Bautista, D.; Atienza, R. Scene text recognition with permuted autoregressive sequence models. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Cham, Switzerland, 2022; pp. 178–196.
54. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
55. Luan, X.; Zhang, J.; Xu, M.; Silamu, W.; Li, Y. A data augmentation strategy for scene text recognition. In Proceedings of the 2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT), Jilin, China, 28–30 April 2023; pp. 968–973.
56. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.