C3-STISR: Scene Text Image Super-resolution with Triple Clues
Minyi Zhao*, Miao Wang1, Fan Bai1
Bingjia Li1,*, Jie Wang2, Shuigeng Zhou†
1Shanghai Key Lab of Intelligent Information Processing, and School of Computer Science, Fudan University, Shanghai 200438, China
2ByteDance, China
{zhaomy20, fbai19, bjli20, sgzhou}@fudan.edu.cn
{wangmiao.01, wangjie.bernard}@bytedance.com

Abstract
Scene text image super-resolution (STISR) has been regarded as an important pre-processing task for text recognition from low-resolution scene text images. Most recent approaches use the recognizer’s feedback as clue to guide super-resolution. However, directly using recognition clue has two problems: 1) Compatibility. It is in the form of probability distribution, has an obvious modal gap with STISR — a pixel-level task; 2) Inaccuracy. It usually contains wrong information, thus will mislead the main task and degrade super-resolution performance. In this paper, we present a novel method C3-STISR that jointly exploits the recognizer’s feedback, visual and linguistical information as clues to guide super-resolution. Here, visual clue is from the images of texts predicted by the recognizer, which is informative and more compatible with the STISR task; while linguistical clue is generated by a pre-trained character-level language model, which is able to correct the predicted texts. We design effective extraction and fusion mechanisms for the triple cross-modal clues to generate a comprehensive and unified guidance for super-resolution. Extensive experiments on TextZoom show that C3-STISR outperforms the SOTA methods in fidelity and recognition performance. Code is available in https://github.com/zhaominyiz/C3-STISR.

1 Introduction
Scene text recognition (STR), which aims to recognize texts from input scene images has wide applications such as auto-driving [Zhang et al., 2020] and scene-text-based image understanding [Singh et al., 2019]. Although great progress has been made in STR due to the development of deep learning, recognition performance on low-resolution (LR) text images is still unsatisfactory. Ergo, scene text image super-resolution (STISR) [Wang et al., 2020] is gaining popularity as a pre-processing technique to recover the missing details in LR images for boosting text recognition performance.

Existing STISR works roughly fall into two categories: generic high-resolution (HR) methods and clue-guided solutions. As shown in Fig. 1, the generic methods [Xu et al., 2017; Pandey et al., 2018] usually try to learn missed details through HR-LR image pairs with pixel loss functions (e.g. L1 or L2 loss). They treat text images as normal images and disregard their text-specific characteristics, usually cannot achieve satisfied recognition performance. Recently, more and more works attempt to take text-specific characteristics as clues to guide super-resolution, which leads to better performance in terms of image quality and recognition accuracy. For example, [Chen et al., 2021a] takes the attention map and recognition result of the recognizer as clues to compute text-focused loss. [Ma et al., 2021] uses the recognition result as text-prior clue to iteratively conduct super-resolution. [Chen et al., 2021b] introduces stroke-level recognition clue to generate more distinguishable images.

Although these methods have definitely improved the recognition accuracy, their designs have some obvious shortcomings: 1) They mostly use the recognizer’s feedback as clue to guide super-resolution, ignore other potentially useful information such as visual and linguistical information. 2)
The widely used recognition clue is in the form of probability distribution (PD), which has an obvious modal gap with STISR — a low-level vision task, so there is a modal compatibility issue. 3) The recognizer’s feedback is usually inaccurate (the recognition accuracy on LR/HR images is only 26.8%/72.4%, see Sec. 4.3), thus will mislead the following super-resolution, especially in some tough scenarios. For example, in Fig. 1(c), the recognizer’s feedback is a PD of “blrd”, but the ground truth is “bird”. Such error in the feedback will inevitably impact super-resolution.

Imagine how humans will repair LR text images in practice. In addition to the information directly from the images, they may also exploit character compositional/structural information and linguistic knowledge to guess the blurred characters and words. With this in mind, in this paper we present a novel method C3-STISR that jointly exploits the recognizer’s feedback, visual and linguistical information as clues to guide super-resolution, as shown in Fig. 1(c). Concretely, the visual clue is extracted from the painted images of texts predicted by the recognizer, which is informative and more compatible with the STISR task, and thus will lead to better recovery (in Fig. 1(c), a clearer and better ‘B’ is gotten due to the usage of visual clue), while the linguistic clue is generated by a pre-trained character-level language model, which is able to correct the predicted text (in Fig. 1(c), “blrd” is corrected to “bird”). Furthermore, regarding that these clues are in different modalities, we first extract them in a divide-and-conquer way, and then aggregate them. We develop effective clue extractors and a unified gated fusion module that integrates the triple clues as a comprehensive guidance signal for super-resolution.

Main contributions of this paper are summarized as follows: 1) We propose a novel method C3-STISR to jointly utilize recognition, visual, and linguistical clues to guide super-resolution. Comparing with existing methods, C3-STISR can generate higher quality text images with the help of newly introduced visual and linguistical clues. 2) We design a powerful clue generator that extracts the triple cross-modal clues in a divide-and-conquer manner, and then fuse them to a comprehensive and unified one. 3) We conduct extensive experiments over the TextZoom dataset, which show that C3-STISR significantly outperforms the state-of-the-art approaches.

## 2 Related Work

Here we review the related works that roughly fall into two groups: generic approaches and clue-guided approaches, according to whether they use text-specific clues.

### 2.1 Generic Approaches

These methods treat STISR as a general SR problem and recover LR images via pixel information captured by pixel loss functions. In particular, SRCNN [Dong et al., 2015] designs a three-layer convolutional neural network for the SR task. [Xu et al., 2017] and SRRResNet [Ledig et al., 2017] adopt generative adversarial networks to generate distinguishable images. [Pandey et al., 2018] combines convolutional layers, transposed convolution, and sub-pixel convolution layers to extract and upscale features. RCAN [Zhang et al., 2018] and SAN [Dai et al., 2019] introduce attention mechanisms to boost the recovery. Nevertheless, such approaches ignore text-specific characteristics, cannot achieve optimal performance.

### 2.2 Clue-guided Approaches

Recent approaches focus on text-specific characteristics of the images and utilize them as clues to boost the recovery. They usually use an additional recognizer to boost the super-resolution. Specifically, [Wang et al., 2019; Nakaune et al., 2021; Fung et al., 2021a] calculate text-specific losses to enhance text recognition. [Wang et al., 2020] introduces TSRN and gradient profile loss to capture sequential and text-specific information of text images. PCAN [Zhao et al., 2021a] is proposed to learn sequence-dependent and high-frequency information of the reconstruction. STT [Chen et al., 2021a] makes use of character-level clue from a pre-trained transformer recognizer to conduct text-focused super-resolution. TPGSR [Ma et al., 2021] and [Ma et al., 2022] extract predicted probability distribution or semantic feature as clues to recover low quality images. TG [Chen et al., 2021b] uses stroke-level clue to generate more distinguishable images. Although these methods have definitely improved recognition accuracy, the clue from the recognizer is mainly in a probability distribution modality incompatible with the STISR task, and usually inaccurate, which limits the improvement of recognition performance.

## 3 Method

Here we first give an overview of our method C3-STISR (meaning triple clues for STISR), then present the triple-clue guided super-resolution backbone. Subsequently, we introduce the extraction and fusion components of the triple clues, followed by the design of loss function.

### 3.1 Overview

Given a low-resolution image \( I_{LR} \in \mathbb{R}^{C \times N} \). Here, \( C \) is the number of channels of each image, \( N = H \times W \) is the collapsed spatial dimension, \( H \) and \( W \) are the height and width of image \( I_{LR} \). Our aim is to produce a super-resolution (SR) image \( I_{SR} \in \mathbb{R}^{C \times (4 \times N)} \) based on the input LR image \( I_{LR} \) and some text-specific clue \( h_t \). Fig. 2 shows the architecture of our method C3-STISR, which is composed of two major components: the triple-clue guided super-resolution backbone \( f_{SR} \) that takes \( I_{LR} \) and \( h_t \) as input to generate a super-resolution image \( I_{SR} = f_{SR}(I_{LR}, h_t) \), and the clue generator \( f_{CG} \) that generates the clue \( h_t \) to guide super-resolution. Specifically, \( f_{CG} \) consists of two subcomponents: the clue extraction branch \( f_{CE} \) and the clue fusion branch \( f_{CF} \). The former generates the triple clues: recognition clue \( h_{rec} \), visual clue \( h_{vis} \) and linguistic clue \( h_{ling} \) based on the feedback of a recognizer \( R \) with \( I_{LR} \) as input, i.e., \( \{h_{rec}, h_{vis}, h_{ling}\} = f_{CE}(R(I_{LR})) \). Then, the latter fuses the triple clues to generate the comprehensive clue \( h_t \) for super-resolution, i.e., \( h_t = f_{CF}(h_{rec}, h_{vis}, h_{ling}) \). During model training, the HR image \( I_{HR} \) (ground truth) of each training LR image is taken as supervision to evaluate the pixel and text-specific losses.
3.2 Triple-clue Guided Super-Resolution Backbone

We design the backbone in the following way: 1) Notice that in the TextZoom dataset [Wang et al., 2020], the HR-LR pairs are manually cropped and matched by humans, which may incur several pixel-level offsets. Following previous works, the backbone starts with a Spatial Transformer Network (STN) [Jaderberg et al., 2015]. 2) Five modified TSRN blocks are employed to recover \( I_{LR} \) with the guidance of \( h_t \). The clue \( h_t \) is concatenated with the feature map extracted by the convolution layers of TSRN blocks at channel dimension. 3) A pixel shuffle module is applied to reshaping the super-resolution image. 4) Two different losses \( L_{pix} \) and \( L_{txt} \) are used to provide pixel and text-specific supervision, respectively. In particular, the \( L_2 \) pixel loss \( (L_{pix}) \) and the text-focused loss \( (L_{txt}) \) [Chen et al., 2021a] are separately adopted to trade-off fidelity and recognition performance:

\[
L_{pix} = || I_{HR} - I_{SR} ||_2, \\
L_{txt} = \lambda_1 || A_{HR} - A_{SR} ||_1 + \lambda_2 KL(p_{SR}, p_{HR}),
\]

where \( A \) and \( p \) are the attention map and probability distribution predicted by a fixed transformer-based recognizer, respectively. KL denotes the Kulback-Leibler divergence, and \( \lambda_1 \) and \( \lambda_2 \) are two hyper-parameters.

3.3 Clue Generator

The clue generator aims to generate a comprehensive clue \( h_t \) to guide the super-resolution backbone. To this end, we first extract triple cross-modal clues: recognition clue \( h_{rec} \), visual clue \( h_{vis} \) and linguistical clue \( h_{ling} \) in a divide-and-conquer manner. Then, we fuse them to output \( h_t \). Now, we start with the introduction of the clue extraction branch.

**Clue Extraction Branch.** Clue extraction can be divided into two steps: first extracting the initial cross-modal clues, and then transforming them into corresponding pixel-level ones for fusion.

\( h_{rec} \) **Extraction.** The recognition clue \( h_{rec} \) is computed from the probability distribution predicted by the recognizer \( R \): \( h_{rec} = f_{rec}(\tilde{R}(I_{LR})) \), and \( \tilde{R}(I_{LR}) \in \mathbb{R}^{L \times |A|} \), \( h_{rec} \in \mathbb{R}^{C' \times N} \). Here, \( C' \), \( L \) and \( |A| \) denote the channel number of hidden state, the max predicted length and the length of alphabet \( A \), respectively. \( f_{rec} := \mathbb{R}^{L \times |A|} \rightarrow \mathbb{R}^{C' \times N} \) is a processing network that transforms the probability distribution \( R(I_{LR}) \) to a pixel feature map and performs error reduction via masking uncertain information. Here, the processing network is implemented by a projection network and a deformable spatiotemporal attention (DSTA) block [Zhao et al., 2021b]. In particular, the projection network consists of four transposed convolution layers followed by batch normalization and a bilinear interpolation; while the DSTA block utilizes the powerful deformable convolution [Dai et al., 2017] to compute a spatial attention map for masking uncertain information. Considering that the performance of the recognizer can heavily influence \( h_{rec} \), we adopt the distillation loss [Ma et al., 2021] to finetune the recognizer \( R \):

\[
L_{rec} = k_1 || R(I_{LR}) - \tilde{R}(I_{LR}) ||_1 + k_2 KL(R(I_{LR}), \tilde{R}(I_{LR})),
\]

where \( k_1, k_2 \) are two hyper-parameters.

\( h_{vis} \) **Extraction.** Given the predicted probability distribution \( \tilde{R}(I_{LR}) \), the goal of the visual clue extractor is to generate the visual information of the text image derived from the recognition result of \( I_{LR} \). To this end, we first introduce a decoding function \( f_{de} := \mathbb{R}^{L \times |A|} \rightarrow \mathbb{N}^L \) to decode the probability distribution to a text string, and then utilize a skeleton painter \( f_{sp} := \mathbb{N}^L \rightarrow \mathbb{R}^{C' \times N} \) to draw the text image. The drawn text image presents the skeleton of the text to be recognized, and provides useful structural information.
for STISR. Here, we use Python Image Library (PIL) as \( f_{sp} \) to draw black-white text images. Nevertheless, the generated text image is in pixel level and has two shortcomings, which makes it fail to directly guide super-resolution. First, the prediction confidence is lost during decoding, which may exacerbate the propagation of errors. Second, the text image is generated in horizontal direction with fixed font, while the recognition clue is interpolated to the pixel level, which may incur motion and shape misalignment. Ergo, we also design a processing network \( f_{vis} := \mathbb{R}^{C \times N} \rightarrow \mathbb{R}^{C \times N} \) to handle these problems. Specifically, \( f_{vis} \) consists of a deformable convolution [Dai et al., 2017] that uses \( h_{vis} \) to align and compensate the text image and a DSTA block for error reduction. Finally, \( h_{vis} \) is extracted as follows:

\[
h_{vis} = f_{vis}(f_{sp}(f_{de}(R(I_{LR}))), h_{rec}). \tag{4}
\]

**htling Extraction.** Given \( R(I_{LR}) \), the linguistic clue extractor is to correct \( R(I_{LR}) \) via a language model \( f_{LM} \) and output the corrected probability distribution \( p_{LM} \), i.e., \( p_{LM} = f_{LM}(R(I_{LR})) \). To achieve this, we employ a pre-trained bidirectional clone network [Fang et al., 2021b] as the language model (LM) to perform character-level correction. The LM is first pre-trained via spelling mutation and recovery with a corpus [Merity et al., 2016], and then finetuned via the distillation loss to adapt to the super-resolution task. That is, we finetune the LM as follows:

\[
\mathcal{L}_{ling} = k_1||p_{LM} - R(I_{HR})||_1 + k_2KL(p_{LM}, R(I_{HR})). \tag{5}
\]

We also design a processing network \( f_{ling} := \mathbb{R}^{L \times |A|} \rightarrow \mathbb{R}^{C \times N} \) for the linguistic clue. Similar to \( f_{rec} \), \( f_{ling} \) consists of a projection network and a DSTA block for error reduction as the correction operation may also be inaccurate.

**Clue Fusion Branch.** With the clue extraction branch, the triple clues are transformed into unified pixel feature maps of \( C' \times N \) size. Here, we employ a modified gated fusion [Xu et al., 2021] to fuse the clues softly. Specifically, given the three pixel-level clues \( h_{rec}, h_{ting} \) and \( h_{vis} \), we first adopt several dilated convolution layers to extract their features. Then, we stack these features with the LR image \( I_{LR} \) in the channel dimension, and utilize a group of convolution layers to generate a mask \( M \in \mathbb{R}^{3 \times C' \times N} \). After performing softmax along the first dimension of \( M \), we get the fused clue \( h_t \) as follows:

\[
h_t = M[0,:] \otimes h_{rec} + M[1,:] \otimes h_{ting} + M[2,:] \otimes h_{vis}, \tag{6}
\]

where \( \otimes \) and \( \oplus \) indicate pixel multiplication and pixel addition, respectively.

### 3.4 Overall Loss Function

There are four types of loss functions used in our method: the first is a pixel loss (Eq. (1)), the second is for recognition performance (Eq. (2)), the third is for finetuning the recognizer (Eq. (3)), and the last is for finetuning the LM (Eq. (5)). Thus, the overall loss function is

\[
\mathcal{L} = \alpha_1 \mathcal{L}_{pix} + \alpha_2 \mathcal{L}_{txt} + \alpha_3 \mathcal{L}_{rec} + \alpha_4 \mathcal{L}_{ling}, \tag{7}
\]

where \( \alpha_1, \alpha_2, \alpha_3, \alpha_4 \) are four hyper-parameters.

### 3.5 Multi-stage Training

To exploit the triple clues of different modalities to the greatest extent, the training process of our method is split into three steps: first, we pre-train the LM via spelling mutation and recovery. Second, we pre-train the recognition clue and visual clue extraction modules. Finally, integrating the pretrained LM with the other modules, we finetune the whole model. Such a training scheme can ensure that the model does not forget the pre-trained linguistic knowledge.

### 4 Performance Evaluation

In this section, we first introduce the dataset and metrics used in the experiments and the implementation details. Then we compare our method with the state-of-the-art approaches. Finally, we conduct extensive ablation studies to validate the design of our method.

### 4.1 Dataset and Metrics

The TextZoom [Wang et al., 2020] dataset consists of 21,740 LR-HR text image pairs collected by lens zooming of the camera in real-world scenarios. The training set has 17,367 pairs, while the test set is divided into three settings based on the camera focal length, namely easy (1,619 samples), medium (1,411 samples) and hard (1,343 samples).

We utilize recognition accuracy to evaluate the recognition performance of the method. We remove all the punctuations and convert uppercase letters to lowercase letters for calculating recognition accuracy, by following the settings of previous works [Chen et al., 2021a]. In addition, we use Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) to evaluate fidelity.

### 4.2 Implementation Details

Our model is implemented in PyTorch1.8. All experiments are conducted on 8 NVIDIA Tesla V100 GPUs with 32GB memory. The model is trained using Adam [Kingma and Ba, 2014] optimizer with a learning rate of 0.0005. The batch size is set to 48. The recognizer \( R \) used in our method is CRNN [Shi et al., 2016]. The hyper-parameters in our method are set as follows: \( \lambda_1 = 10, \lambda_2 = 0.0005, k_1 = 1.0, k_2 = 1.0, \alpha_1 = 20, \alpha_2 = 20, \alpha_3 = 1, \alpha_4 = 0.2, C' = 32 \), which are recommended in [Chen et al., 2021a; Ma et al., 2021]. The font used by the skeleton painter is ubuntu bold. Two text images (one uppercase, one lowercase) are generated by the skeleton painter for each LR image. Our training and evaluation are based on the following protocol: save the averagely best model during training with CRNN as the recognizer, and use this model to evaluate the other recognizers (MORAN, ASTER) and the three settings (Easy, Medium, Hard).

### 4.3 Comparing with the SOTA Approaches

Here we evaluate our method on TextZoom, and compare it with existing super-resolution models on three recognition models, including CRNN [Shi et al., 2016], MORAN [Luo et al., 2019] and ASTER [Shi et al., 2018]. The results are presented in Tab. 1. We can see that our method significantly
improves the recognition accuracy. Taking CRNN as an example, comparing with the SOTA method TG [Chen et al., 2021b] that boosts the performance from 48.1% to 48.9% (increasing 0.8%), our method lifts the accuracy from 48.9% to 53.7% (increasing 4.8%). This demonstrates the effectiveness and advantage of our method.

We also present the results of fidelity (PSNR and SSIM) comparison with major existing methods in Tab. 2. Our method is advantageous over or comparable to the SOTA in fidelity, while significantly outperforms the others in recognition performance. Furthermore, we visualize some examples in Fig. 3. Compared with the other methods, C3-STISR can recover the blurry pixels better. Experimental results on more

Table 1: Performance (recognition accuracy) comparison on TextZoom.

<table>
<thead>
<tr>
<th>Method</th>
<th>CRNN [Shi et al., 2016]</th>
<th>MORAN [Luo et al., 2019]</th>
<th>ASTER [Shi et al., 2018]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BICUBIC</td>
<td>56.4% 21.1% 21.1% 26.8%</td>
<td>60.6% 37.9% 30.8% 44.1%</td>
<td>67.4% 42.4% 31.2% 48.2%</td>
</tr>
<tr>
<td>HR</td>
<td>76.4% 75.1% 64.6% 72.4%</td>
<td>91.2% 85.3% 74.2% 84.1%</td>
<td>94.2% 87.7% 76.2% 86.6%</td>
</tr>
<tr>
<td>SRCNN</td>
<td>41.1% 22.3% 22.0% 29.2%</td>
<td>63.9% 40.0% 29.4% 45.6%</td>
<td>70.6% 44.0% 31.5% 50.0%</td>
</tr>
<tr>
<td>SRResNet</td>
<td>45.2% 32.6% 25.5% 35.1%</td>
<td>66.0% 47.1% 33.4% 49.9%</td>
<td>69.4% 50.5% 35.7% 53.0%</td>
</tr>
<tr>
<td>RCAN</td>
<td>46.8% 27.9% 26.5% 34.5%</td>
<td>63.1% 42.9% 33.6% 47.5%</td>
<td>67.3% 46.6% 35.1% 50.7%</td>
</tr>
<tr>
<td>SAN</td>
<td>50.1% 31.2% 28.1% 37.2%</td>
<td>65.6% 44.4% 35.2% 49.4%</td>
<td>68.1% 48.7% 36.2% 52.0%</td>
</tr>
<tr>
<td>TSRN</td>
<td>92.5% 38.2% 31.4% 41.4%</td>
<td>70.1% 55.3% 37.9% 55.4%</td>
<td>75.1% 56.5% 40.1% 58.3%</td>
</tr>
<tr>
<td>STT</td>
<td>59.6% 47.1% 35.3% 48.1%</td>
<td>74.1% 57.0% 40.8% 58.4%</td>
<td>75.7% 59.9% 41.6% 60.1%</td>
</tr>
<tr>
<td>PCAN</td>
<td>59.6% 45.4% 34.8% 47.4%</td>
<td>73.7% 57.6% 41.0% 58.5%</td>
<td>77.5% 60.7% 43.1% 61.5%</td>
</tr>
<tr>
<td>TG</td>
<td>61.2% 47.6% 35.5% 48.9%</td>
<td>75.8% 57.8% 41.4% 59.4%</td>
<td>77.9% 60.2% 42.4% 61.3%</td>
</tr>
<tr>
<td>Baseline (w/o clue)</td>
<td>54.8% 42.9% 32.7% 44.2%</td>
<td>67.5% 52.7% 37.1% 53.4%</td>
<td>72.3% 56.1% 38.5% 56.8%</td>
</tr>
<tr>
<td>Ours (C3-STISR)</td>
<td>65.2% 53.6% 39.8% 53.7%</td>
<td>74.2% 61.0% 43.2% 60.5%</td>
<td>79.1% 63.3% 46.8% 64.1%</td>
</tr>
</tbody>
</table>

Table 2: Fidelity and recognition performance comparison with major existing methods. The results are obtained by averaging that of three settings (Easy, Medium and Hard).

<table>
<thead>
<tr>
<th>Method</th>
<th>Metric</th>
<th>PSNR</th>
<th>SSIM (×10^-2)</th>
<th>Avg Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>BICUBIC</td>
<td></td>
<td>20.35</td>
<td>69.61</td>
<td>26.8</td>
</tr>
<tr>
<td>TSRN</td>
<td></td>
<td>21.42</td>
<td>76.91</td>
<td>41.4</td>
</tr>
<tr>
<td>STT</td>
<td></td>
<td>21.05</td>
<td>76.14</td>
<td>48.1</td>
</tr>
<tr>
<td>PCAN</td>
<td></td>
<td><strong>21.49</strong></td>
<td><strong>77.53</strong></td>
<td><strong>47.4</strong></td>
</tr>
<tr>
<td>TG</td>
<td></td>
<td>21.40</td>
<td>74.56</td>
<td><strong>48.9</strong></td>
</tr>
<tr>
<td>Ours (C3-STISR)</td>
<td></td>
<td><strong>21.51</strong></td>
<td><strong>77.21</strong></td>
<td><strong>53.7</strong></td>
</tr>
</tbody>
</table>

Table 3: Ablation study on the design of the clue extraction branch. Here, “ft” and “pt” denote finetuning and pre-training, respectively.

<table>
<thead>
<tr>
<th>Variant</th>
<th>Metric</th>
<th>PSNR</th>
<th>SSIM (×10^-2)</th>
<th>Avg Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o ft</td>
<td></td>
<td>21.09</td>
<td>75.48</td>
<td>30.3</td>
</tr>
<tr>
<td>with ft</td>
<td></td>
<td><strong>21.14</strong></td>
<td><strong>75.98</strong></td>
<td><strong>52.2</strong></td>
</tr>
<tr>
<td>w/o compensation</td>
<td></td>
<td>20.80</td>
<td>74.25</td>
<td>49.4</td>
</tr>
<tr>
<td>with compensation</td>
<td></td>
<td><strong>21.21</strong></td>
<td><strong>76.38</strong></td>
<td><strong>51.7</strong></td>
</tr>
<tr>
<td>w/o pt</td>
<td></td>
<td><strong>21.07</strong></td>
<td><strong>75.37</strong></td>
<td>49.3</td>
</tr>
<tr>
<td>with pt &amp; ft</td>
<td></td>
<td>20.84</td>
<td><strong>76.06</strong></td>
<td>50.4</td>
</tr>
<tr>
<td>with pt &amp; ft</td>
<td></td>
<td>20.94</td>
<td>75.78</td>
<td><strong>51.0</strong></td>
</tr>
</tbody>
</table>

Figure 3: Examples of generated SR images and recognition results from the SR images by different methods. Red characters are incorrectly recognized, and black characters are correctly recognized.
convolutions to align them. For comparison, we also implement a variant that does not use compensation. As can be seen in the second part (Rows 5-6) of Tab. 3, our design with compensation significantly boosts the performance.

**Linguistical Clue Extraction.** In C3-STISR, we apply pre-training and distillation loss $L_{ling}$ (Eq. (5)) to boost the knowledge learning from and the adaption to the linguistical domain. To check the effect of our design, we provide the performance of the variants that do not use pre-training or $L_{ling}$. As shown in the third part (Rows 7-9) of Tab. 3, such variants are inferior to that using both pre-training and $L_{ling}$ in recognition accuracy.

**Design of Clue Fusion Branch.** There are many techniques to fuse multiple signals (e.g. multi-head attention and deformable fusion [Zhao et al., 2021b]). In our method, we fuse three clues via a modified gated fusion. The reason for our design lies in that after the projection network and deformable convolutions, there is no more modal gap. Ergo, taking aligned clues as input, simple gated fusion is enough to fuse the triple clues via aggregating the pixels that are considered being correct. The experimental results are presented in Tab. 4, from which we can see that the proposed gated fusion achieves the best performance among all the three fusion techniques.

**Different Combinations of the Triple Clues.** Here, we check the performance of different combinations of the triple clues. The results are shown in Tab. 5. The baseline without any clues tends to repair each pixel in the image, which leads to good fidelity but low recognition accuracy. When clues are applied, the recognition accuracy is obviously improved. Among them, the recognition clue achieves the best recognition performance, and the visual clue outperforms the others in fidelity. The linguistical clue is inferior to the other two clues since STISR is a vision task. When two clues are combined, recognition-linguistical achieves the best accuracy. What is more, when recognition clue combines with visual clue, the fidelity is better than that of either single clue.

This shows the effectiveness of the linguistical and visual clues. Finally, the combination of all the triple clues achieves the best performance in both fidelity (PSNR) and recognition performance, which shows that the proposed triple clues are complementary and all are required for better performance.

**Effect of Multi-stage Training.** To exploit the potential of each cue to the greatest extent, we design a multi-stage training procedure. To check the effect of multi-stage training scheme, we compare the performance with and without the scheme. As shown in Tab. 6, without the proposed multi-stage training, performance is degraded.

**Effect of DSTA.** We stack three DSTA [Zhao et al., 2021b] blocks in our clue extraction branch to mask uncertain information. To check the effect of such design, we present the results without stacking DSTA blocks in Tab. 6. Obviously, without DSTA, the performance is degraded.

**Hyper-parameter Study.** We have some hyper-parameters to balance different losses. Here, $\lambda_1, \lambda_2$ are set as recommended in [Chen et al., 2021a], while $k_1, k_2, \alpha_1, \alpha_2, \alpha_3$ are set as suggested in [Ma et al., 2021]. The remaining hyper-parameter to set is $\alpha_4$, which controls the language model. Here, we set $\alpha_4$ to relatively small values, aiming at retaining the linguistic knowledge as much as possible. We use grid search to determine $\alpha_4$. As shown in Tab. 7, when $\alpha_4 = 0.2$, the best performance is achieved. Ergo, $\alpha_4$ is set to 0.2 in our experiments.

**5 Conclusion**

In this paper, we present a novel method called C3-STISR that jointly utilizes recognition, visual, and linguistical clues to guide super-resolution. Comparing with the recognition clue used in existing works, the proposed visual clue is informative and more compatible, and the linguistical clue is able to correct error information in the recognition feedback. We develop an effective clue generator that first generates the triple cross-modal clues in a divide-and-conquer manner, and then aggregates them. Extensive experiments demonstrate the effectiveness and superiority of the proposed method.

**Acknowledgments**

The work was supported in part by a ByteDance Research Collaboration Project.
References


