Learning Texture Transformer Network for Image Super-Resolution

Fuzhi Yang¹*, Huan Yang², Jianlong Fu², Hongtao Lu¹, Baining Guo²
¹Department of Computer Science and Engineering, MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, ²Microsoft Research, Beijing, P.R. China, {yfzcopy0702, htlu}@sjtu.edu.cn, {huayan, jianf, bainguo}@microsoft.com

Abstract

We study on image super-resolution (SR), which aims to recover realistic textures from a low-resolution (LR) image. Recent progress has been made by taking high-resolution images as references (Ref), so that relevant textures can be transferred to LR images. However, existing SR approaches neglect to use attention mechanisms to transfer high-resolution (HR) textures from Ref images, which limits these approaches in challenging cases. In this paper, we propose a novel Texture Transformer Network for Image Super-Resolution (TTSR), in which the LR and Ref images are formulated as queries and keys in a transformer, respectively. TTSR consists of four closely-related modules optimized for image generation tasks, including a learnable texture extractor by DNN, a relevance embedding module, a hard-attention module for texture transfer, and a soft-attention module for texture synthesis. Such a design encourages joint feature learning across LR and Ref images, in which deep feature correspondences can be discovered by attention, and thus accurate texture features can be transferred. The proposed texture transformer can be further stacked in a cross-scale way, which enables texture recovery from different levels (e.g., from 1× to 4× magnification). Extensive experiments show that TTSR achieves significant improvements over state-of-the-art approaches on both quantitative and qualitative evaluations. The source code can be downloaded at https://github.com/researchmm/TTSR.

1. Introduction

Image super-resolution aims to recover natural and realistic textures for a high-resolution image from its degraded low-resolution counterpart [12]. The recent success of image SR can greatly enhance the quality of media content for a better user experiences. For example, the digital zoom algorithm for mobile cameras and image enhancement technology for digital televisions. Besides, this fundamental technology can benefit a broad range of computer vision tasks, like medical imaging [21] and satellite imaging [35].

The research on image SR is usually conducted on two paradigms, including single image super-resolution (SISR), and reference-based image super-resolution (RefSR). Traditional SISR often results in blurry effects, because the high-resolution (HR) textures have been excessively destructed in the degrading process which are unrecoverable. Although generative adversarial networks (GANs) [7] based image SR approaches are proposed to relieve the above problems, the resultant hallucinations and artifacts caused by GANs further pose grand challenges to image SR tasks.

*This work was performed when the first author was visiting Microsoft Research as a research intern.
Recent progress has been made by reference-based image super-resolution (RefSR), which transfers HR textures from a given Ref image to produce visually pleasing results [5, 6, 26, 29, 36]. However, state-of-the-art (SOTA) approaches usually adopt a straightforward way to transfer textures which may result in unsatisfied SR images (as shown in Figure 1). For example, Zheng et al. [43] adopts a flow-based approach which usually searches and transfers inaccurate textures (indicate by red) when facing large viewpoint changes between the LR and Ref image. Zhang et al. [41] adopts a feature space defined by a pre-trained classification model to search and transfer textures between the LR and Ref image. Nevertheless, such high-level semantic features can not effectively represent HR textures which remain to generate implausible results.

To address these problems, we propose a novel Texture Transformer Network for Image Super-Resolution (TTSR). Specifically, four closely-related modules optimized for image generation tasks are proposed. First, we propose a learnable texture extractor, in which parameters will be updated during end-to-end training. Such a design enables a joint feature embedding of LR and Ref images which creates a solid foundation for applying attention mechanism [19, 34, 31] in SR tasks. Second, we propose a relevance embedding module to compute the relevance between the LR and Ref image. More specifically, we formulate the extracted features from the LR and Ref image as the query and key in a transformer [31] to obtain a hard-attention map and a soft-attention map. Finally, we propose a hard-attention module and a soft-attention module to transfer and fuse HR features from the Ref image into LR features extracted from backbone through the attention maps. The design of TTSR encourages a more accurate way to search and transfer relevant textures from Ref to LR images.

Furthermore, we propose a cross-scale feature integration module to stack the texture transformer, in which the features are learnt across different scales (e.g, from 1× to 4×) to achieve a more powerful feature representation. As shown in Figure 1, the overall design enables our TTSR to search and transfer relevant textures from the Ref image (indicated by green) which achieves a better visual result compared with SOTA approaches. The main contributions of this paper are:

- To the best of our knowledge, we are one of the first to introduce the transformer architecture into image generation tasks. More specifically, we propose a texture transformer with four closely-related modules for image SR which achieves significant improvements over SOTA approaches.
- We propose a novel cross-scale feature integration module for image generation tasks which enables our approach to learn a more powerful feature representation by stacking multiple texture transformers.

2. Related Work

In this section, we review previous works of single image super-resolution (SISR) and reference-based image super-resolution (RefSR) which are the most relevant to our work.

2.1. Single Image Super-Resolution

In recent years, deep learning based SISR methods have achieved significant improvements over traditional non-learning based methods. Deep learning based methods in SISR treat this problem as a dense image regression task which learns an end-to-end image mapping function represented by a CNN between LR and HR images.

Dong et al. [3] proposed SRCNN that firstly adopted deep learning into SISR by using a three-layer CNN to represent the mapping function. Dong et al. [4] further sped up the SR process by replacing the interpolated LR image with the original LR image and using deconvolution at the very last layer to enlarge the feature map. Soon afterwards, Kim et al. proposed VDSR [14] and DRCN [15] with deeper networks on residual learning. Shi et al. [23] replaced deconvolution with the subpixel convolution layer to reduce the checkerboard artifact. Residual block [9] was introduced into SISR in SRResNet [16] and improved in EDSR [17]. With the help of residual block, a lot of works focused on designing deeper or wider networks [2, 27, 28]. Zhang et al. [40] and Tong et al. [30] adopted dense blocks [10] to combine features from different levels. Zhang et al. [39] improved residual block by adding channel attention. Liu et al. [18] proposed a non-local recurrent network for image restoration. Dai et al. [2] introduced second-order statistics for more discriminative feature representations.

The above methods use mean square error (MSE) or mean absolute error (MAE) as their objective function which ignores human perceptions. In recent years, more and more works aim to improve perceptual quality. Johnson et al. [13] introduced perceptual loss into SR tasks, while SRGAN [16] adopted generative adversarial networks (GANs) [7] showed visually satisfying results. Sajjadi et al. [22] used Gram matrix based texture matching loss to enforce local similar textures, while ESRGAN [32] enhanced SRGAN by introducing RRDB with relativistic adversarial loss. Recent proposed RSRGAN [38] trained a ranker and used rank-content loss to optimize the perceptual quality, which achieved state-of-the-art visual results.

2.2. Reference-based Image Super-Resolution

Different from SISR, RefSR can harvest more accurate details from the Ref image. This could be done by several approaches like image aligning or patch matching. Some existing RefSR approaches [33, 36, 43] choose to align the LR and Ref image. Landmark [36] aligned the Ref image to the LR image through a global registration to solve an
energy minimization problem. Wang et al. [33] enhanced
the Ref image by recurrently applying non-uniform warping
before feature synthesis. CrossNet [43] adopted optical
flow to align the LR and Ref image at different scales and
concatenated them into the corresponding layers of the de-
coder. However, the performance of these methods depends
largely on the aligning quality between the LR and Ref im-
age. Besides, the aligning approaches such as optical flow
are time-consuming, which is adverse to real applications.

Other RefSR approaches [1, 41, 42] adopt “patch
match” method to search proper reference information.
Boominathan et al. [1] matched the patches between gra-
dient features of the LR and down-sampled Ref image.
Zheng et al. [42] replaced the simple gradient features
with features in convolution neural networks to apply se-
matic matching and used a SISR method for feature syn-
thesis. Recent work SRNTT [41] applied patch matching
between VGG [24] features of the LR and Ref image to
swap similar texture features. However, SRNTT ignores the
relevance between original and swapped features and feeds
all the swapped features equally into the main network.

To address these problems, we propose a texture trans-
former network which enables our approach to search and
transfer relevant textures from Ref to LR images. Moreover,
the performance of our approach can be further improved
by stacking multiple texture transformers with a proposed
cross-scale feature integration module.

3. Approach

In this section, we introduce the proposed Texture
Transformer Network for Image Super-Resolution (TTSR).
On top of the texture transformer, we propose a cross-scale
feature integration module (CSFI) to further enhance model
performances. The texture transformer and CSFI will be
discussed in Section 3.1 and Section 3.2, respectively. A
group of loss functions for optimizing the proposed network
will be explained in Section 3.3.

3.1. Texture Transformer

The structure of the texture transformer is shown in Fig-
ure 2. LR, LR↑ and Ref represent the input image, the 4×
bicubic-upsampled input image and the reference image,
respectively. We sequentially apply bicubic down-sampling
and up-sampling with the same factor 4× on Ref to ob-
tain Ref↓↑ which is domain-consistent with LR↑. The tex-
ture transformer takes Ref, Ref↓↑, LR↑ and the LR features
produced by the backbone as input, and outputs a synthe-
sized feature map, which will be further used to generate
the HR prediction. There are four parts in the texture trans-
former: the learnable texture extractor (LTE), the relevance
embedding module (RE), the hard-attention module for fea-
ture transfer (HA) and the soft-attention module for feature
synthesis (SA). Details will be discussed below.

Learnable Texture Extractor. In RefSR tasks, texture ex-
traction for reference images is essential because accurate
and proper texture information will assist the generation of
SR images. Instead of using semantic features extracted by
a pre-trained classification model like VGG [24], we design
a learnable texture extractor whose parameters will be up-
dated during end-to-end training. Such a design encourages
a joint feature learning across the LR and Ref image, in
which more accurate texture features can be captured. The
process of texture extraction can be expressed as:

\[ Q = LTE(LR↑), \]
\[ K = LTE(Ref↓↑), \]
\[ V = LTE(Ref), \]

where LTE(·) denotes the output of our learnable texture
extractor. The extracted texture features, Q (query), K
(key), and V (value) indicate three basic elements of the at-
tention mechanism inside a transformer and will be further
used in our relevance embedding module.

Relevance Embedding. Relevance embedding aims to em-
bed the relevance between the LR and Ref image by esti-
mating the similarity between Q and K. We unfold both Q
and K into patches, denoted as \( q_i \) \( (i ∈ [1, H_{LR} × W_{LR}) \) and
\( k_j \) \( (j ∈ [1, H_{Ref} × W_{Ref}) \) Then for each patch \( q_i \) in Q

\[ E(q_i, k_j) = \frac{1}{d} \sum_{k=1}^{d} h(q_i[k], k_j[k]), \]

where \( h(·, ·) \) is a similarity function, and \( d \) is the num-
ber of channels. The relevance scores are then fed into the
hard-attention module to obtain the hard-attention map T.

Hard Attention. The hard-attention module is designed
to align the LR and Ref features by matching their corre-
sponding elements in the feature map. Let \( T \) be the hard
attention map, then the LR features are warped to align
with the Ref features, and the warped LR features are
fused with the Ref features to generate the SR output.

\[ F = H(F, T), \]

where \( H(·, ·) \) is a feature fusion function. The fused fea-
tures are then fed into the soft-attention module to ob-
tain the soft-attention map S.

Soft Attention. The soft-attention module is designed
to further refine the aligned features by matching their corre-
sponding elements in the feature map. Let \( S \) be the soft
attention map, then the fused features are warped to align
with the Ref features, and the warped fused features are
fused with the Ref features to generate the SR output.

\[ Output = S(Output, S), \]

where \( S(·, ·) \) is a feature fusion function. The final SR
output is generated by combining the aligned and refined
features.

Figure 2. The proposed texture transformer. Q, K and V are
the texture features extracted from an up-sampled LR image, a
sequentially down/up-sampled Ref image, and an original Ref im-
age, respectively. H and S indicate the hard/soft attention map,
calculated from relevance embedding. F is the LR features ex-
tracted from a DNN backbone, and is further fused with the trans-
ferred texture features T for generating the SR output.
and \( k_j \) in \( K \), we calculate the relevance \( r_{i,j} \) between these two patches by normalized inner product:

\[
r_{i,j} = \left\langle \frac{q_i}{\|q_i\|}, \frac{k_j}{\|k_j\|} \right\rangle. \tag{4}
\]

The relevance is further used to obtain the hard-attention map and the soft-attention map.

**Hard-Attention.** We propose a hard-attention module to transfer the HR texture features \( V \) from the Ref image. Traditional attention mechanism takes a weighted sum of \( V \) for each query \( q_i \). However, such an operation may cause blur effect which lacks the ability of transferring HR texture features. Therefore, in our hard-attention module, we only transfer features from the most relevant position in \( V \) for each query \( q_i \).

More specifically, we first calculate a hard-attention map \( H \) in which the \( i \)-th element \( h_i \) (\( i \in [1, H_{LR} \times W_{LR}] \)) is calculated from the relevance \( r_{i,j} \):

\[
h_i = \arg \max_j r_{i,j}. \tag{5}
\]

The value of \( h_i \) can be regarded as a hard index, which represents the most relevant position in the Ref image to the \( i \)-th position in the LR image. To obtain the transferred HR texture features \( T \) from the Ref image, we apply an index selection operation to the unfolded patches of \( V \) using the hard-attention map as the index:

\[
t_i = v_{h_i}, \tag{6}
\]

where \( t_i \) denotes the value of \( T \) in the \( i \)-th position, which is selected from the \( h_i \)-th position of \( V \).

As a result, we obtain a HR feature representation \( T \) for the LR image which will be further used in our soft-attention module.

**Soft-Attention.** We propose a soft-attention module to synthesize features from the transferred HR texture features \( T \) and the LR features \( F \) of the LR image from a DNN backbone. During the synthesis process, relevant texture transfer should be enhanced while the less relevant ones should be relived. To achieve that, a soft-attention map \( S \) is computed from \( r_{i,j} \) to represent the confidence of the transferred texture features for each position in \( T \):

\[
s_i = \max_j r_{i,j}, \tag{7}
\]

where \( s_i \) denotes the \( i \)-th position of the soft-attention map \( S \). Instead of directly applying the attention map \( S \) to \( T \), we first fuse the HR texture features \( T \) with the LR features \( F \) to leverage more information from the LR image. Such fused features are further element-wisely multiplied by the soft-attention map \( S \) and added back to \( F \) to get the final output of the texture transformer. This operation can be represented as:

\[
F_{out} = F + \text{Conv}(\text{Concat}(F,T)) \odot S, \tag{8}
\]

where \( F_{out} \) indicates the synthesized output features. \( \text{Conv} \) and \( \text{Concat} \) represent a convolutional layer and Concatenation operation, respectively. The operator \( \odot \) denotes element-wise multiplication between feature maps.

In summary, the texture transformer can effectively transfer relevant HR texture features from the Ref image into the LR features, which boosts a more accurate process of texture generation.

### 3.2. Cross-Scale Feature Integration

Our texture transformer can be further stacked in a cross-scale way with a cross-scale feature integration module. The architecture is shown in Figure 3. Stacked texture transformers output the synthesized features for three resolution scales (1×, 2× and 4×), such that the texture features of different scales can be fused into the LR image. To learn a better representation across different scales, inspired by [25, 37], we propose a cross-scale feature integration module (CSFI) to exchange information among the features of different scales. A CSFI module is applied each time the LR feature is up-sampled to the next scale. For each scale inside the CSFI module, it receives the exchanged features from other scales by up/down-sampling, followed by a concatenation operation in the channel dimension. Then a convolutional layer will map the features into the original number of channels. In such a design, the texture features transferred from the stacked texture transformers are exchanged across each scale, which achieves a more powerful feature representation. This cross-scale feature integration module further improves the performance of our approach.

### 3.3. Loss Function

There are 3 loss functions in our approach. The overall loss can be interpreted as:

\[
\mathcal{L}_{overall} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv} + \lambda_{per}\mathcal{L}_{per}. \tag{9}
\]
Reconstruction loss. The first loss is the reconstruction loss:
\[
L_{\text{rec}} = \frac{1}{C\cdot H \cdot W} \| I^{HR} - I^{SR} \|_1 ,
\]
where \((C, H, W)\) is the size of the HR. We utilize \(L_1\) loss which has been demonstrated to be sharper for performance and easier for convergence compared to \(L_2\) loss.

Adversarial loss. Generative adversarial networks [7] are proved effective in generating clear and visually favorable images. Here we adopt WGAN-GP [8], which proposes a penalization of gradient norm to replace weight clipping, resulting in more stable training and better performance. This loss can be interpreted as:
\[
\begin{align*}
L_D &= \mathop{\mathbb{E}}_{\hat{x} \sim P_g} [D(\hat{x})] - \mathop{\mathbb{E}}_{x \sim P_x} [D(x)] + \\
&\quad \lambda \mathop{\mathbb{E}}_{\hat{x} \sim P_g} [\| \nabla_{\hat{x}} D(\hat{x}) \|_2 - 1]^2 , \\
L_G &= - \mathop{\mathbb{E}}_{x \sim P_x} [D(x)].
\end{align*}
\]

Perceptual loss. Perceptual loss has been demonstrated useful to improve visual quality and has already been used in [13, 16, 22, 41]. The key idea of perceptual loss is to enhance the similarity in feature space between the prediction image and the target image. Here our perceptual loss contains two parts:
\[
L_{\text{per}} = \frac{1}{C_i \cdot H_i \cdot W_i} \| \phi_{i}^{vgg}(I^{SR}) - \phi_{i}^{vgg}(I^{HR}) \|_2^2 + \\
\frac{1}{C_j \cdot H_j \cdot W_j} \| \phi_{j}^{te}(I^{SR}) - T \|_2^2 ,
\]
where the first part is a traditional perceptual loss, in which \(\phi_i^{vgg}()\) denotes the \(i\)-th layer’s feature map of VGG19, and \((C_i, H_i, W_i)\) represents the shape of the feature map at that layer. \(I^{SR}\) is the predicted SR image. The second part in our perceptual loss is a transferal perceptual loss, in which \(\phi_j^{te}()\) denotes the texture feature map extracted from the \(j\)-th layer of the proposed LTE, and \((C_j, H_j, W_j)\) represents that layer’s shape. \(T\) is the transferred HR texture features in Figure 2. This transferal perceptual loss constrains the predicted SR image to have similar texture features to the transferred texture features \(T\), which makes our approach to transfer the Ref textures more effectively.

3.4. Implementation Details

The learnable texture extractor contains 5 convolutional layers and 2 pooling layers which outputs texture features in three different scales. To reduce the consumption of both time and GPU memory, the relevance embedding is only applied to the smallest scale and further propagated to other scales. For the discriminator, we adopt the same network used in SRNTT [41] and remove all BN layers. During training, we augment the training images by randomly horizontally and vertically flipping followed by randomly rotating \(90^\circ, 180^\circ\) and \(270^\circ\). Each mini-batch contains 9 LR patches with size \(40 \times 40\) along with 9 HR and Ref patches with size \(160 \times 160\). The weight coefficients for \(L_{\text{rec}}, L_{\text{adv}}\) and \(L_{\text{per}}\) are \(1, 1e^{-3}\) and \(1e^{-2}\), respectively. Adam optimizer with \(\beta_1 = 0.9, \beta_2 = 0.999\), and \(\epsilon = 1e^{-8}\) is used with learning rate of \(1e^{-4}\). We first warm up the network for 2 epochs where only \(L_{\text{rec}}\) is applied. After that, all losses are involved to train another 50 epochs.

4. Experiments

4.1. Datasets and Metrics

To evaluate our method, we train and test our model on the recently proposed RefSR dataset, CUFED5 [41]. The training set in CUFED5 contains 11,871 pairs, each pair consisting of an input image and a reference image. There are 126 testing images in CUFED5 testing set, each accompanied by 4 reference images with different similarity levels. In order to evaluate the generalization performance of TTSR trained on CUFED5, we additionally test TTSR on Sun80 [26], Urban100 [11], and Manga109 [20]. Sun80 contains 80 natural images, each paired with several reference images. For Urban100, we use the same setting as [41] to regard its LR images as the reference images. Such a design enables an explicit process of self-similar searching and transferring since Urban100 are all building images with strong self-similarity. For Manga109 which also lacks the reference images, we randomly sample HR images in this dataset as the reference images. Since this dataset is constructed with lines, curves and flat colored regions which are all common patterns. Even with a randomly picked HR Ref image, our method can still utilize these common patterns and achieve good results. The SR results are evaluated on PSNR and SSIM on Y channel of YCbCr space.

4.2. Evaluation

To evaluate the effectiveness of TTSR, we compare our model with other state-of-the-art SISR and RefSR methods. The SISR methods include SRCNN [3], MDSR [17], RDN [40], RCAN [39], SRGAN [16], ENet [22], ESRGAN [32], RSRGAN [38], among which RCAN has achieved state-of-the-art performance on both PSNR and SSIM in recent years. RSRGAN is considered to achieve the state-of-the-art visual quality. As for RefSR methods, CrossNet [43] and SRNTT [41] are two state-of-the-art methods recently, which significantly outperform previous RefSR methods. All experiments are performed with a scaling factor of \(4\times\) between LR and HR images.

Quantitative Evaluation. For fair comparison, we follow the setting in SRNTT [41] to train all the methods
Table 1. PSNR/SSIM comparison among different SR methods on four different datasets. Methods are grouped by SISR methods (top) and RefSR methods (down). Red numbers denote the highest scores while blue numbers denote the second highest scores.

| Method        | CUFED5 | Sun80 | Urban100 | Manga109 |
|---------------|--------|-------|-----------|-----------|
| SRCNN [3]     | 25.33 / .745 | 28.26 / .781 | 24.41 / .738 | 27.12 / .830 |
| MDSR [17]     | 25.93 / .777 | 28.52 / .792 | 25.51 / .783 | 28.93 / .891 |
| RDN [40]      | 25.95 / .769 | 29.63 / .806 | 25.38 / .768 | 29.24 / .894 |
| RCAN [39]     | 26.06 / .769 | 29.86 / .810 | 25.42 / .768 | 29.38 / .895 |
| SRGAN [16]    | 24.40 / .702 | 26.76 / .725 | 24.07 / .729 | 25.12 / .802 |
| ESRGAN [32]   | 24.24 / .695 | 26.24 / .702 | 23.63 / .711 | 25.25 / .802 |
| RSRGAN [38]   | 21.90 / .633 | 24.18 / .651 | 20.91 / .620 | 23.53 / .797 |
| CrossNet [43] | 22.31 / .635 | 25.60 / .667 | 21.47 / .624 | 25.04 / .803 |
| SRNTT [41]    | 25.48 / .764 | 28.52 / .793 | 25.11 / .764 | 23.36 / .741 |
| TTSR          | 27.09 / .804 | 30.02 / .814 | 25.87 / .784 | 30.09 / .907 |

Table 2. Ablation study on texture transformer.

| Method                  | HA | SA | LTE | PSNR/SSIM       |
|-------------------------|----|----|-----|-----------------|
| Base                    |    |    |     | 26.34 / .780    |
| Base+HA                 | ✓  |    |     | 26.59 / .786    |
| Base+HA+SA              | ✓  | ✓  |     | 26.81 / .795    |
| Base+HA+SA+LTE          | ✓  | ✓  | ✓   | 26.92 / .797    |

Table 3. Ablation study on CSFI.

| Method                  | CSFI | numC | param. | PSNR/SSIM       |
|-------------------------|------|------|--------|-----------------|
| Base+TT                 | 64   | 4.42M|        | 26.927 / .797   |
| Base+TT+CSFI            | ✓   | 64   | 4.42M  | 27.09 / .804    |
| Base+TT(C80)            |     | 80   | 6.53M  | 26.93 / .797    |
| Base+TT(C96)            |     | 96   | 9.10M  | 26.98 / .799    |

Figure 4. User study results. Values on Y-axis indicate the percentage of users that prefer TTSR over other approaches.

4.3. Ablation Study

In this section, we verify the effectiveness of different modules in our approach, including the texture transformer, the cross-scale feature integration, the adversarial loss and the transferal perceptual loss. In addition, we also discuss the influence of different reference similarity on TTSR.

Texture transformer. Our texture transformer contains mainly four parts: the learnable texture extractor (LTE), the relevance embedding module, the hard-attention module for feature transfer (HA) and the soft-attention module for feature synthesis (SA). Ablation results are shown in Table 2. We re-implement SRNTT [41] as our “Base” model by only removing all BN layers and Ref part. On top of the baseline model, we progressively add HA, SA, and LTE. Models without LTE use the VGG19 features to do relevance embedding. As we can see, when HA is added, the PSNR performance can be improved from 26.34 to 26.59, which verifies the effectiveness of the hard-attention module for feature transfer. When SA is involved, relevant texture features will be enhanced while the less relevant ones will be relieved during the feature synthesizing. This fur-
ther boosts the performance to 26.81. When replacing VGG with the proposed LTE, the PSNR is finally increased to 26.92, which proves the superiority of joint feature embedding in LTE.

To further verify the effectiveness of our LTE, we use the hard attention map to transfer the original image. It is expected that a better feature representation can transfer more accurate textures from the original images. Figure 6 shows the transferred original image by VGG19 in SRNTT and LTE in TTSR. In this figure, TTSR can transfer more accurate reference textures and generate a globally favorable result, which further proves the effectiveness of our LTE.

**Cross-scale feature integration.** On top of the texture transformer, CSFI can further enable texture recovery from different resolution scales (1x, 2x and 4x). We conduct an ablation study in Table 3. The first row shows the performance of our model with only TT, while the second row proves the effectiveness of CSFI, which brings 0.17
increase on PSNR metric. In order to verify that the performance improvement is not brought by the increase of parameter size, we increase the channel number of “Base+TT” model to 80 and 96. As we can see, there is almost no growth of “Base+TT(C80)” which has almost the same parameter number as “Base+TT+CSFI”. Even if we increase the parameter number to 9.10M to obtain “Base+TT(C96)” model, there is still a performance gap. This demonstrates that CSFI can efficiently utilize the reference texture information with a relatively smaller parameter size.

Adversarial loss. To make sure that the improvement of perceptual quality benefits from model design rather than the adversarial loss. We conduct an ablation among “Base-rec”, “Base”, TTSR-rec and TTSR, where TTSR can be interpreted as “Base+TT(HA+SA+LTE)+CSFI”.

Transferal perceptual loss. The transferal perceptual loss constraints the LTE’s features between the predicted SR image and the transferred image T to be similar. As shown in Figure 8, using this loss is able to transfer textures in a more effective way which achieves visually pleasing results. In addition, this loss also improves the quantitative metrics PSNR and SSIM of TTSR from 25.20/.757 to 25.53/.765.

Influence of different reference similarity. To study how relevance between LR and Ref images influences the results of TTSR, we conduct experiments on CUFED5 testing set, which has reference images of different relevance levels. Table 4 shows the results of five relevance levels, in which “L1” to “L4” represent the reference images provided by CUFED5 testing set where L1 is the most relevant level while L4 is the least relevant one. “LR” means using the input image itself as the reference image. As shown in Table 4, TTSR using L1 as the reference image achieves the best performance. When using LR as the reference image, TTSR still performs better than previous state-of-the-art RefSR approaches.

5. Conclusion

In this paper, we propose a novel Texture Transformer Network for Image Super-Resolution (TTSR) which transfers HR textures from the Ref to LR image. The proposed texture transformer consists of a learnable texture extractor which learns a jointly feature embedding for further attention computation and two attention based modules which transfer HR textures from the Ref image. Furthermore, the proposed texture transformer can be stacked in a cross-scale way with the proposed CSFI module to learn a more powerful feature representation. Extensive experiments demonstrate the superior performance of our TTSR over state-of-the-art approaches on both quantitative and qualitative evaluations. In the future, we will further extend the proposed texture transformer to general image generation tasks.

Acknowledgement This paper is partially supported by NSFC (No. 61772330, 61533012, 61876109), the pre-research project (No. 61403120201), Shanghai Key Laboratory of Crime Scene Evidence (2017XCWZK01) and the Interdisciplinary Program of Shanghai Jiao Tong University (YG2019QNA09).
References

[1] Vivek Boominathan, Kaushik Mitra, and Ashok Veeraraghavan. Improving resolution and depth-of-field of light field cameras using a hybrid imaging system. In ICCP, pages 1–10, 2014.
[2] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In CVPR, pages 11065–11074, 2019.
[3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaou Tang. Image super-resolution using deep convolutional networks. TPAMI, 38(2):295–307, 2015.
[4] Chao Dong, Chen Change Loy, and Xiaou Tang. Accelerating the super-resolution convolutional neural network. In ECCV, pages 391–407, 2016.
[5] Gilad Freedman and Raanan Fattal. Image and video upscaling from local self-examples. ACM Transactions on Graphics (TOG), 30(2):1–11, 2011.
[6] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. IEEE Computer graphics and Applications, 22(2):56–65, 2002.
[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NeurIPS, pages 2672–2680, 2014.
[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In NeurIPS, pages 5767–5777, 2017.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, pages 770–778, 2016.
[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In CVPR, pages 4700–4708, 2017.
[11] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In CVPR, pages 5197–5206, 2015.
[12] Michal Irani and Shmuel Peleg. Improving resolution by image registration. CVGIP, 53(3):231–239, 1991.
[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In ECCV, pages 694–711, 2016.
[14] Jiwon Kim, Jang Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In CVPR, pages 1646–1654, 2016.
[15] Jiwon Kim, Jang Kwon Lee, and Kyoung Mu Lee. Deeply-recurive convolutional network for image super-resolution. In CVPR, pages 1637–1646, 2016.
[16] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In CVPR, pages 4681–4690, 2017.
[17] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In CVPR Workshops, pages 136–144, 2017.
[18] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. In NeurIPS, pages 1673–1682, 2018.
[19] Shuang Ma, Jianlong Fu, Chang Wen Chen, and Tao Mei. Da-gan: Instance-level image translation by deep attention generative adversarial networks. In CVPR, pages 5657–5666, 2018.
[20] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. Multimedia Tools and Applications, 76(20):21811–21838, 2017.
[21] Ozan Oktay, Wenjia Bai, Matthew Lee, Ricardo Guerrero, Konstantinos Kamnitsas, Jose Caballero, Antonio de Marvao, Stuart Cook, Declan ORegan, and Daniel Rueckert. Multi-input cardiac image super-resolution using convolutional neural networks. In MICCAI, pages 246–254, 2016.
[22] Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In ICCV, pages 4491–4500, 2017.
[23] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In CVPR, pages 1874–1883, 2016.
[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
[25] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. arXiv preprint arXiv:1904.04545, 2019.
[26] Libin Sun and James Hays. Super-resolution from internet-scale scene matching. In ICCP, pages 1–12, 2012.
[27] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep residual recursive network. In CVPR, pages 3147–3155, 2017.
[28] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In ICCV, pages 4539–4547, 2017.
[29] Radu Timofte, Vincent De Smet, and Luc Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In ICCV, pages 1920–1927, 2013.
[30] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In ICCV, pages 4799–4807, 2017.
[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, pages 5998–6008, 2018.
[32] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In ECCV Workshops, 2018.
[33] Yuwang Wang, Yebin Liu, Wolfgang Heidrich, and Qionghai Dai. The light field attachment: Turning a dslr into a light field camera using a low budget camera ring. IEEE TVCG, 23(10):2357–2364, 2016.
[34] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. In CVPR, pages 1316–1324, 2018.

[35] Deniz Yıldırım and Oğuz Güngör. A novel image fusion method using ikonos satellite images. Journal of Geodesy and Geoinformation, 1(1):75–83, 2012.

[36] Huanjing Yue, Xiaoyan Sun, Jingyu Yang, and Feng Wu. Landmark image super-resolution by retrieving web images. IEEE TIP, 22(12):4865–4878, 2013.

[37] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. Learning pyramid-context encoder network for high-quality image inpainting. In CVPR, pages 1486–1494, 2019.

[38] Wenlong Zhang, Yihao Liu, Chao Dong, and Yu Qiao. Ranksrgan: Generative adversarial networks with ranker for image super-resolution. In ICCV, pages 3096–3105, 2019.

[39] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In ECCV, pages 286–301, 2018.

[40] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In CVPR, pages 2472–2481, 2018.

[41] Zhifei Zhang, Zhaowen Wang, Zhe Lin, and Hairong Qi. Image super-resolution by neural texture transfer. In CVPR, pages 7982–7991, 2019.

[42] Haitian Zheng, Mengqi Ji, Lei Han, Ziwei Xu, Haoqian Wang, Yebin Liu, and Lu Fang. Learning cross-scale correspondence and patch-based synthesis for reference-based super-resolution. In BMVC, 2017.

[43] Haitian Zheng, Mengqi Ji, Haoqian Wang, Yebin Liu, and Lu Fang. Crossnet: An end-to-end reference-based super-resolution network using cross-scale warping. In ECCV, pages 88–104, 2018.
Supplementary

In this supplementary material, Section A illustrates the details of TTSR’s network structure. Section B provides additional analyses about the texture transformers on different scales. Section C describes the comparison of the running time and the parameter number. Finally, more visual comparison results will be shown in Section D.

A. Details of Network Structure

In this section, we will illustrate the detailed network structure of our approach TTSR, including the learnable texture extractor in the texture transformer, the generator with three stacked texture transformers and the discriminator. The structure of the learnable texture extractor is shown in Table A.1, in which the layers $0$, $3$ and $6$ are used to search and transfer texture features in the texture transformer. Table A.2 shows the details of the generator, and Table A.3 illustrates the discriminator.

Table A.1. Network structure of the learnable texture extractor. Conv($N_{in}$, $N_{out}$) indicates the convolutional layer with $N_{in}$ input channels and $N_{out}$ output channels. The kernel size is $3 \times 3$ for all convolutional layers. Pool($2 \times 2$) is the $2 \times 2$ pooling layer with stride 2.

| Id | Layer Name                     |
|----|--------------------------------|
| 0  | Conv(3,64), ReLU               |
| 1  | Conv(64,64), ReLU              |
| 2  | Pool($2 \times 2$)             |
| 3  | Conv(64,128), ReLU             |
| 4  | Conv(128,128), ReLU            |
| 5  | Pool($2 \times 2$)             |
| 6  | Conv(128, 256), ReLU           |

B. Texture Transformers on Different Scales

Our proposed TTSR contains three stacked texture transformers. The texture transformer at each scale fuses HR texture features of different levels from the Ref image. Here we conduct experiments of using the texture transformers on different scales. The model here is without CSFI since CSFI is designed for multi-scale stacked texture transformers. Table B.1 shows the results. The larger scale the texture transformer is applied at, the more performance it brings, which demonstrates that the texture features at a larger scale have a less loss of details. When we gradually add the texture transformers at other scales, the performance can be further improved.

C. Running Time and Model Size

In this section, the running time and the model size of TTSR will be discussed. We compare the proposed TTSR with state-of-the-art SISR and RefSR approaches, RCAN [39], RSRGAN [38], CrossNet [43] and SRNTT [41]. For running time, all approaches are run on a Tesla V100 PCIe GPU and tested on an $83 \times 125 \times 3$ LR input image with the up-sampling factor of $4 \times$. Table C.1 shows the comparison results. Specifically, the stacked texture transformers cost 0.037s and the other parts cost 0.059s, and TTSR takes a total time of 0.096s. The results show that TTSR achieves the best performance with a relatively small parameter number and running time.

D. More Visual Comparison

In this section, we show more comparison results among the proposed TTSR and other SR methods, including RDN [40], RCAN [39], RSRGAN [38], CrossNet [43] and SRNTT [41]. RCAN has achieved state-of-the-art performance on both PSNR and SSIM in recent years and RSRGAN is considered to achieve the state-of-the-art visual quality. CrossNet and SRNTT are two state-of-the-art RefSR approaches which significantly outperform previous RefSR methods. The visual comparison results on CUFED5 [41], Sun80 [26], Urban100 [11] and Manga109 [20] are shown in Figure D.1-D.4, Figure D.5-D.6, Figure D.7-D.8 and Figure D.9-D.10, respectively.
Table A.2. Network structure of the generator. Conv($N_{in}$, $N_{out}$) indicates the convolutional layer with $N_{in}$ input channels and $N_{out}$ output channels. The kernel size is $3 \times 3$ for all convolutional layers except that the last convolution uses $1 \times 1$ kernel. RB denotes the residual block without batch normalization layers and the ReLU layer after the skip connection. TT represents the texture transformer and PS is the $2 \times 2$ pixel shuffle layer. ↑ indicates bicubic up-sampling followed by a $1 \times 1$ convolution, while ↓ denotes the strided convolution.

| Id  | Layer Name (scale1×) | Id  | Layer Name (scale2×) | Id  | Layer Name (scale4×) |
|-----|----------------------|-----|----------------------|-----|----------------------|
|     |                      |     |                      |     |                      |
| Stage0 |                      | 1-0 | Conv(3,64), ReLU    |     |                      |
|       |                      | 1-1 | RB ×16               |     |                      |
|       |                      | 1-2 | Conv(64,64)          |     |                      |
|       |                      | 1-3 | $1-0 + 1-2$          |     |                      |
|       |                      | 1-4 | TT                   |     |                      |
|       |                      | 1-5 | RB ×16               |     |                      |
|       |                      | 1-6 | Conv(64,64)          |     |                      |
|       |                      | 1-7 | $1-4 + 1-6$          |     |                      |
| Stage1 |                      | 2-0 | Conv(64,256), PS, ReLU($1-7$) | 2-1 | TT                   |
|       |                      | 2-2 | Concat($1-7 \uparrow || 2-1\downarrow$) | 2-3 | Conv(128,64), ReLU |
|       |                      | 2-4 | RB ×8                | 2-5 | Conv(64,64)          |
|       |                      | 2-6 | $1-7 + 1-11$         | 2-7 | $1-4 + 1-6$          |
| Stage2 |                      | 3-0 | Conv(64,256), PS, ReLU($2-6$) | 3-1 | TT                   |
|       |                      | 3-2 | Concat($1-7 \uparrow || 2-1\downarrow$) | 3-3 | Conv(128,64), ReLU |
|       |                      | 3-4 | RB ×8                | 3-5 | Conv(64,64)          |
|       |                      | 3-6 | $1-7 + 1-11$         | 3-7 | $1-4 + 1-6$          |
| Stage3 |                      | 4-0 | Conv(64,256), PS, ReLU($2-6$) | 4-1 | TT                   |
|       |                      | 4-2 | Concat($1-7 \uparrow || 2-1\downarrow$) | 4-3 | Conv(128,64), ReLU |
|       |                      | 4-4 | RB ×4                | 4-5 | Conv(64,64)          |
|       |                      | 4-6 | $1-7 + 1-11$         | 4-7 | $1-4 + 1-6$          |
| Stage4 |                      | 5-0 | Conv(64,256), PS, ReLU($2-6$) | 5-1 | TT                   |
|       |                      | 5-2 | Concat($1-7 \uparrow || 2-1\downarrow$) | 5-3 | Conv(128,64), ReLU |
|       |                      | 5-4 | RB ×4                | 5-5 | Conv(64,64)          |
|       |                      | 5-6 | $1-7 + 1-11$         | 5-7 | $1-4 + 1-6$          |

Table A.3. Network structure of the discriminator. Conv($N_{in}$, $N_{out}$, S) indicates the convolutional layer with $N_{in}$ input channels, $N_{out}$ output channels and stride S. The kernel size is $3 \times 3$ for all convolutional layers. The parameter is 0.2 for all the leaky ReLU layers. The size of HR and SR input is $160 \times 160 \times 3$.

| Id | Layer Name              |
|----|-------------------------|
| 0  | Conv(3,32,1), LReLU     |
| 1  | Conv(32,32,2), LReLU    |
| 2  | Conv(32,64,1), LReLU    |
| 3  | Conv(64,64,2), LReLU    |
| 4  | Conv(64,128,1), LReLU   |
| 5  | Conv(128,128,2), LReLU  |
| 6  | Conv(128,256,1), LReLU  |
| 7  | Conv(256,256,2), LReLU  |
| 8  | Conv(256,512,1), LReLU  |
| 9  | Conv(512,512,2), LReLU  |
| 10 | FC(12800,1024), LReLU   |
| 11 | FC(1024,1)              |

Table B.1. Performance on CUFED5 testing set using texture transformers on different scales.

| scale1× | scale2× | scale4× | PSNR/SSIM |
|---------|---------|---------|-----------|
| ✓       | ✓       | ✓       | 26.56 / .785 |
| ✓       | ✓       | ✓       | 26.77 / .793 |
| ✓       | ✓       | ✓       | 26.80 / .797 |
| ✓       | ✓       | ✓       | 26.82 / .795 |

Table C.1. Running time and parameter number of different approaches. The last column shows the PSNR/SSIM performance on CUFED5 testing set.

| Approach | Time | Param. | PSNR/SSIM |
|----------|------|--------|-----------|
| RCAN [39]| 0.108s | 16M | 26.06 / .769 |
| RSRGAN [38] | 0.007s | 1.5M | 22.31 / .655 |
| CrossNet [43] | 0.229s | 3.6M | 25.48 / .764 |
| SRNTT [41] | 4.977s | 4.2M | 26.24 / .784 |
| TTSR | 0.096s | 6.4M | 27.09 / .804 |
Figure D.1. Visual comparison of different SR methods on CUFED5 [41] dataset.
Figure D.2. Visual comparison of different SR methods on CUFED5 [41] dataset.
Figure D.3. Visual comparison of different SR methods on CUFED5 [41] dataset.
Figure D.4. Visual comparison of different SR methods on CUFED5 [41] dataset.
Figure D.5. Visual comparison of different SR methods on Sun80 [26] dataset.
Figure D.6. Visual comparison of different SR methods on Sun80 [26] dataset.
Figure D.7. Visual comparison of different SR methods on Urban100 [11] dataset.
Figure D.8. Visual comparison of different SR methods on Urban100 [11] dataset.
Figure D.9. Visual comparison of different SR methods on Manga109 [20] dataset.
Figure D.10. Visual comparison of different SR methods on Manga109 [20] dataset.