# Supplementary Material: A Simple Plugin for Transforming Images to Arbitrary Scales

Qinye Zhou*[1]
zhouqinye@sjtu.edu.cn
Ziyi Li*[1]
599lzy@sjtu.edu.cn
Weidi Xie†[1,2]
weidi@sjtu.edu.cn
Xiaoyun Zhang[1]
xiaoyun.zhang@sjtu.edu.cn
Yanfeng Wang†[1,2]
wangyanfeng@sjtu.edu.cn
Ya Zhang[1,2]
ya_zhang@sjtu.edu.cn

[1] Coop. Medianet Innovation Center, Shanghai Jiao Tong University, China
[2] Shanghai AI Laboratory

In this supplementary document, we start by giving more details on the architecture of our ARIS plugin module in Section 1; then describe additional ablation studies that are promised in the main text in Section 2; lastly, we present more visualization results in Section 3.

## 1 Model Architecture

**Baseline SR Encoder** Denote the input image as $\mathcal{X}_{\text{LR}} \in \mathbb{R}^{3\times H\times W}$ (3 means R, G, and B), the encoder of the baseline SR network generates a feature map $\mathcal{F} = \Phi_{\text{ENC}}(\mathcal{X}_{\text{LR}}), \mathcal{F} \in \mathbb{R}^{C\times H\times W}$ with $C$ channels and the same height and width as the input image (here we use $C = 64$ and $H = W = 48$). Note that, in our work, we consider the SR models that encode the input image and generate feature maps of same spatial resolution, for example, IPT [1], SwinIR [4], HAT [2]. We choose this feature map as the input of our ARIS plugin module.

**Transformer Encoder in ARIS** Before inputting the features into our transformer encoder, we first split them into patches and each patch is treated as a "token". Specifically, we reshape the features $\mathcal{F} \in \mathbb{R}^{C\times H\times W}$ into $\mathcal{F}_{\text{flatten}} \in \mathbb{R}^{N\times D}$, where $N = \frac{HW}{p^2} = 256$ is the number of patches, $D = p^2 \times C = 576$ is the dimension of token in transformer encoder and transformer decoder, $p$ is the patch size. Then to emphasize the position information of each patch, we add learnable position embeddings for each feature patch and then feed them into the transformer encoder. The output of transformer encoder ($\mathcal{F}_{\text{LR}} \in \mathbb{R}^{N\times D}$) is of the same dimension as the input ($\mathcal{F}_{\text{flatten}}$). We use the original structure of transformer encoder layer in [5], which consists of a multi-head self-attention module, a feed forward network, and two layer normalization. We also use the same residual structure as [5].


*Both the authors have contributed equally to this project. † denote corresponding authors.

**Transformer Decoder in ARIS** The input of the transformer decoder consists of the scale and the output of the transformer encoder. The scale is used to generate spatial coordinates as described in Section 3.1.2 of the main text. In a nutshell, the spatial coordinates is treated as `query` and be reshaped into dimension $(\gamma^2 \cdot \frac{HW}{p^2\tau^2}) \times D$, where $\gamma$ refers to the scale factor, $\tau$ refers to the upsampling scale of the baseline SR network, the output feature of the transformer encoder is used as `key` and `value` with dimension $N \times D$. Note that we also add position embeddings for each feature patch by firstly constructing a normalised spatial grid with size $\frac{H}{p} \times \frac{W}{p}$ and then project the coordinates into high diemensional vectors with Fourier encoding. We also adopt the original structure of transformer decoder layer in [5] which consists of two multi-head attention modules for self-attention and cross attention respectively, a feed forward network, and three layer normalization. At last, we reshape the output into features with dimension $C \times (\gamma \cdot H/\tau) \times (\gamma \cdot W/\tau)$.

**Baseline SR Decoder** Generally, the decoder of baseline SR network consists of *upsample* modules and *pixelshuffle* modules, which takes feature map $\mathcal{F} \in \mathbb{R}^{C \times H \times W}$ as input and outputs SR image $\mathcal{Y}^{\tau}_{\mathrm{SR}} \in \mathbb{R}^{3 \times (\tau \cdot H) \times (\tau \cdot W)}$. In our case, we all use the baseline SR networks training on $\times 2$ scale, thus $\tau = 2$. After out ARIS plugin module is inserted, the input feature has dimension $C \times (\frac{\gamma \cdot H}{\tau}) \times (\frac{\gamma \cdot W}{\tau})$ and thus the decoder outputs image $\mathcal{Y}^{\gamma}_{\mathrm{SR}} \in \mathbb{R}^{3 \times (\gamma \cdot H) \times (\gamma \cdot W)}$.

# 2 Additional Ablation Study

In this section, we present more ablation results for patch size. We also ablate the training scale factors with different strides.

**Patch Size** In Table 1, we also have an ablation study on the choice of patch size in Transformer, and find that the module works the best when the patch size is 3. Using a smaller patch size will lead to more tokens, thus more computation, while large patch size results in coarse representation, both decrease the performance slightly.

| Patch Size | Set5 | | | Set14 | | |
|---|---|---|---|---|---|---|
| | ×2 | ×3 | ×4 | ×2 | ×3 | ×4 |
| 1 | 38.38 | 33.40 | 31.38 | 34.21 | 29.62 | 28.47 |
| 2 | 38.49 | 34.54 | 31.68 | 34.45 | 30.40 | 28.69 |
| 3 | **38.53** | **34.73** | **32.59** | **34.72** | **30.83** | **29.05** |
| 4 | 38.42 | 34.06 | 30.98 | 33.88 | 30.02 | 28.36 |

**Table 1. Ablation study on patch size.** HAT-ARIS achieves the best performance with patch $3 \times 3$.

**Training Scale** As shown in Table 2, we compare the performance of IPT-ARIS with different training scale factors, which vary from 1 to 4 with different strides, *i.e.*, 0.5, 0.25, 0.1, and the distribution of the scale factors is uniform. The results show that our model can successfully fit the training data and perform well when the stride is equal to 0.5 or 0.25, while fails to fit training data and performs poorly when stride is equal to 0.1.

We conjecture this phenomenon is attributed to the tokenization operation we used, specifically, following ViT [4], we reshape the feature into a sequence of flattened 2D patches before input to the transformer and each patch is regarded as a token, which requires the dimension of query before reshape ($\frac{\gamma H}{\tau}$ and $\frac{\gamma W}{\tau}$) can both be divisible by patch size ($p$). If the dimension of query can not be divisible by patch size, *i.e.*, $p = 3$ in this paper, we use *round* operation to remove the remainder. Therefore, if the stride is 0.1, *e.g.*, for the training scale 1.5, 1.6, we use *round* to change the corresponding dimension of query from 36, 38.4 to 36,

| Dataset | scale | stride | | |
|---|---|---|---|---|
| | | 0.5 | 0.25 | 0.1 |
| Set5 | ×1.5 | 40.91 | 39.99 | 33.17 |
| | ×2 | 38.26 | 37.85 | 33.56 |
| | ×2.5 | 36.06 | 35.59 | 31.00 |
| | ×3 | 34.75 | 34.36 | 31.20 |
| | ×3.5 | 33.56 | 33.10 | 29.21 |
| | ×4 | 32.65 | 32.25 | 31.33 |
| Set14 | ×1.5 | 36.97 | 36.06 | 30.14 |
| | ×2 | 34.11 | 33.64 | 29.96 |
| | ×2.5 | 31.88 | 31.45 | 28.08 |
| | ×3 | 30.68 | 30.38 | 28.13 |
| | ×3.5 | 29.63 | 29.36 | 26.81 |
| | ×4 | 28.94 | 28.67 | 28.08 |
| B100 | ×1.5 | 35.60 | 34.98 | 29.54 |
| | ×2 | 32.38 | 32.16 | 28.95 |
| | ×2.5 | 30.45 | 30.21 | 27.24 |
| | ×3 | 29.33 | 29.14 | 27.23 |
| | ×3.5 | 28.42 | 28.24 | 26.07 |
| | ×4 | 27.80 | 27.64 | 27.29 |
| Urban100 | ×1.5 | 35.97 | 34.49 | 26.58 |
| | ×2 | 33.30 | 32.55 | 26.65 |
| | ×2.5 | 30.60 | 29.91 | 24.88 |
| | ×3 | 29.31 | 28.72 | 25.14 |
| | ×3.5 | 28.00 | 27.46 | 23.72 |
| | ×4 | 27.21 | 26.71 | 25.48 |

**Table 2. Ablation study on training scale.** The training scale factors of the IPT-ARIS vary from 1 to 4 with different strides, *i.e.*,$0.5, 0.25, 0.1$, and the distribution of the scale factors is uniform.

36, respectively. As a result, the model may not be able to distinguish the training scales 1.5 and 1.6 since they have the same query dimension, which may be the reason for the poor performance of model when stride is equal to 0.1.

Such problem can be tackled by simply increasing the size of input LR image, for example, we can increase $H$ and $W$ to 60, then when the stride is equal to 0.1, for the training scale varies from 1 to 4 uniformly, the corresponding dimension of query varies from 30 to 120 uniformly.

## 3 More Qualitative Results

We provide more qualitative results in Figure 1, Figure 2 and Figure 3. Specifically, we compare HAT-ARIS with other arbitrary-scale SR methods on benchmark datasets with different scales. We can observe that other arbitrary-scale SR methods tend to generate blurry regions or artifacts, while our method can consistently achieve better perceptual quality, *i.e.* maintaining the original shape and textual information, and recovering clearer details. In summary, our ARIS plugin module can learn a better continuous image representation.

# References

[1] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proc. CVPR*, 2021.

[2] Xiangyu Chen, Xintao Wang, Jiantao Zhou, and Chao Dong. Activating more pixels in image super-resolution transformer. *arXiv preprint arXiv:2205.04437*, 2022.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[4] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proc. ICCVW*, 2021.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

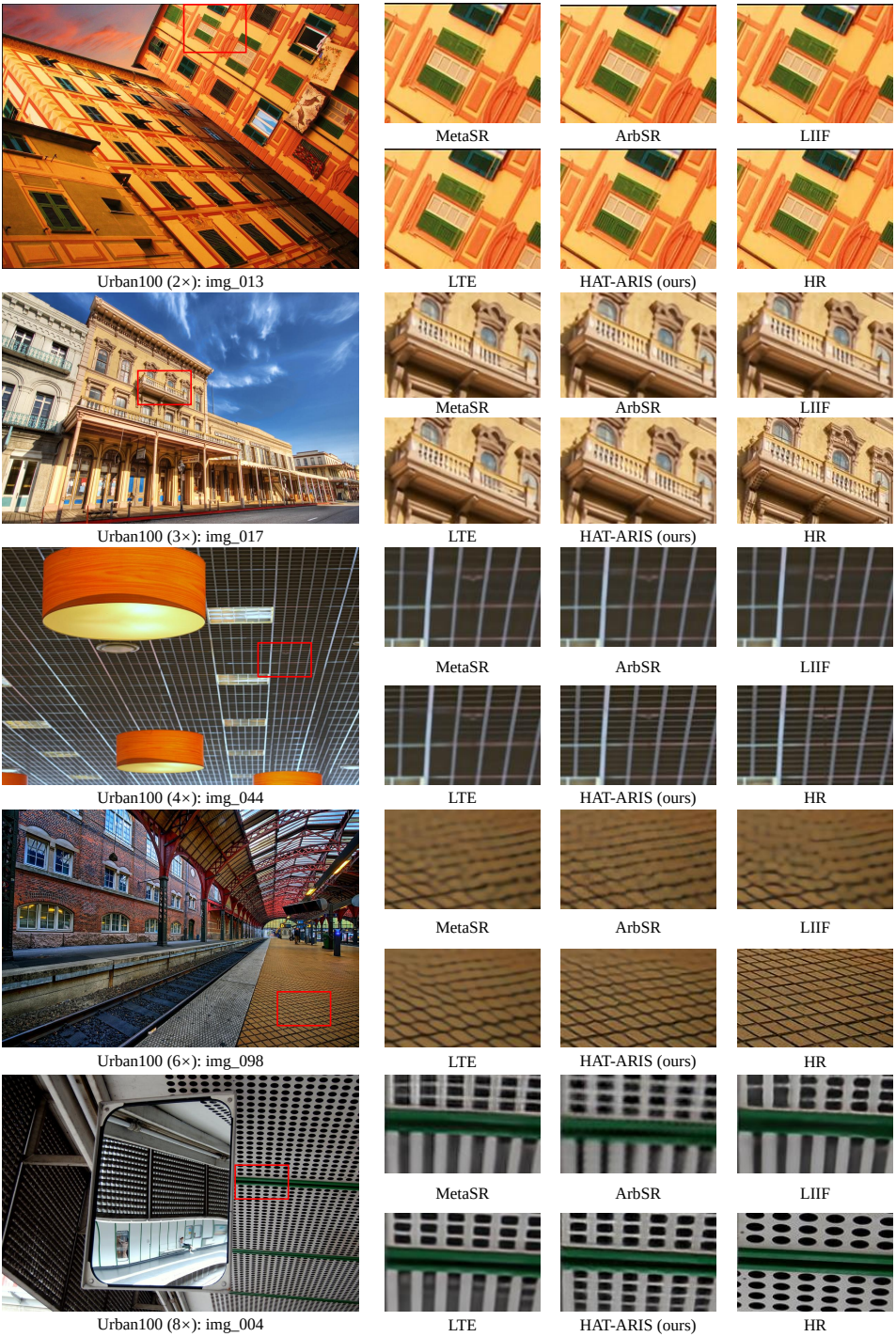


**Figure 1. Qualitative comparisons of different arbitrary-SR methods.** The patches for comparison are marked with red boxes in the original images.

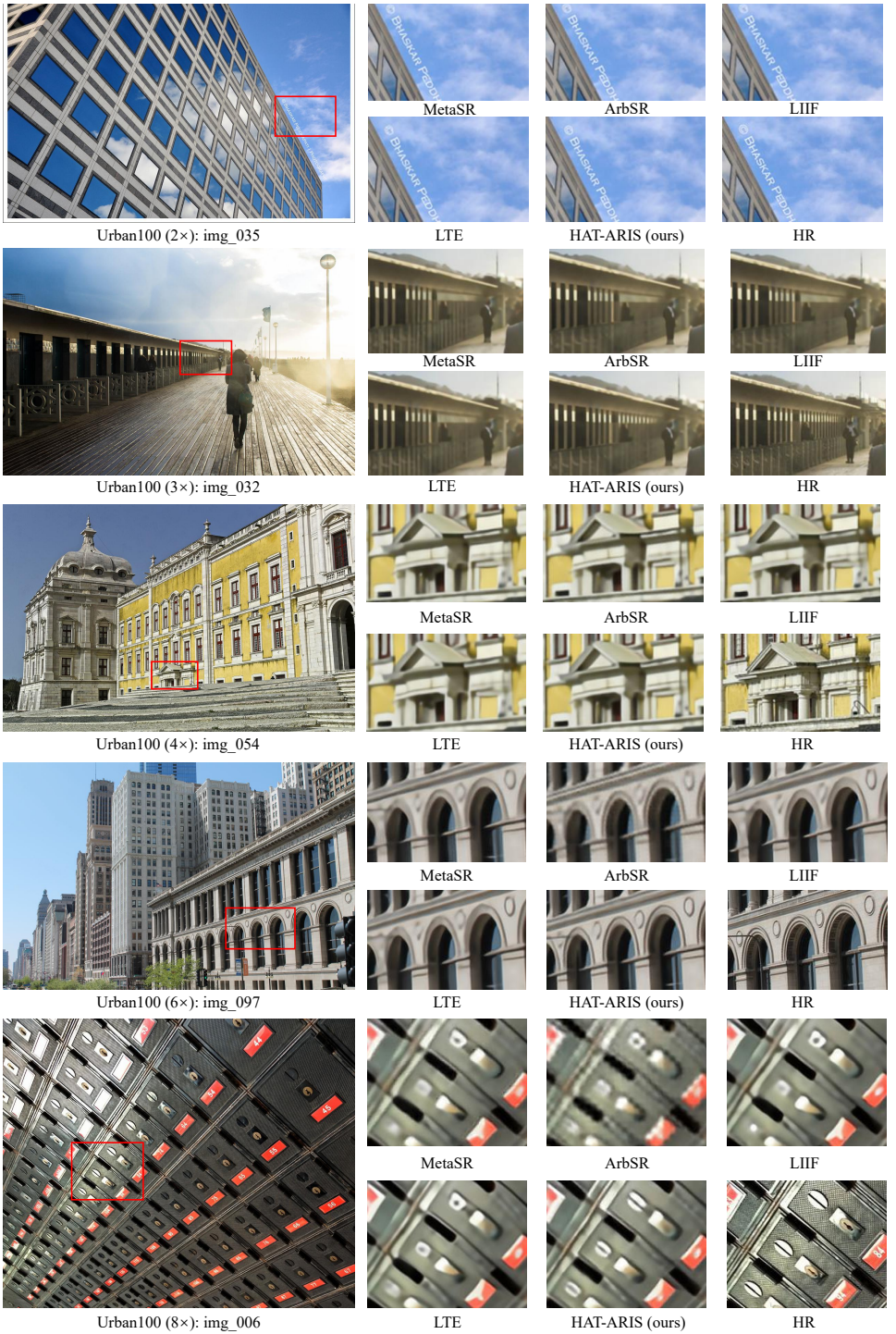


**Figure 2. Qualitative comparisons of different arbitrary-SR methods.** The patches for comparison are marked with red boxes in the original images.

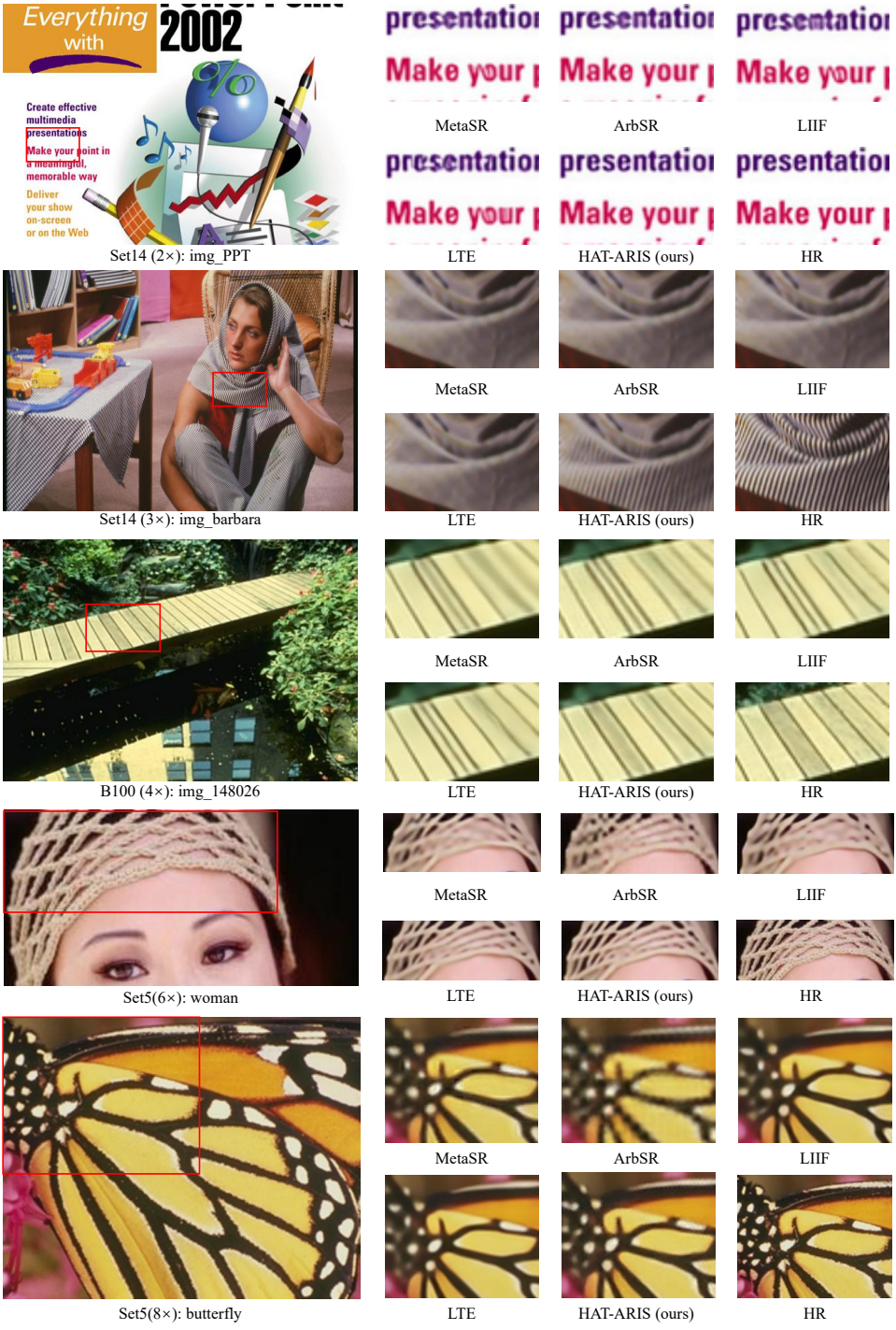


**Figure 3. Qualitative comparisons of different arbitrary-SR methods.** The patches for comparison are marked with red boxes in the original images.