CUF: Continuous Upsampling Filters

Supplementary Material

A. Experiment details

Training Hyper-parameters. Table 8 contains the hyper-parameters used for training the ablated models, taken from their single-scale (Sub-Pixel Conv.) training settings. All models are trained for $1K$ epochs with $\mathcal{L}_1$ loss and ADAM optimizer by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. We use a step-wise learning rate schedule that is halved at epochs $[500, 800, 900, 950]$. Unless referred as single-scale, models were trained with random scales by sampling the scale factor uniformly within the continuous interval $[1, 4]$. In order to ensure that the dimensions within the training mini-batch match despite heterogeneous scale factors, we first scale each image as required and then apply the same crop size to both LR and HR images, such that the HR random crop contains $1/s^2$ of the content of the LR crop, and fix the relative grid coordinates to point to the random sub-region.

CUF's hyper parameters. Table 9 describes the number of neurons used in the presented ablations across different encoders. C_e was chosen to replicate the original number of output features of each encoder. CUF's hyper-parameters were obtained by grid search on the EDSR-baseline and Set5 dataset and replicated on the remaining encoders. CUF's positional encoding hyper-parameters enforce a small number of basis per input parameter. For each input (represented in $2D$ space), the number of basis N^2 was searched within the set $\{0, 1^2, 2^2, ..., 5^2\}$ while f_{max} within the set $\{0, 0.5, ..., 4.0\}$ for $\Pi(\delta^s(\mathbf{x}))$ and $\Pi(s)$, and within the set $\{0, 0.5, .., 3.0\}$ for $\Pi(k)$. The final positional encoding hyper-parameters adopted are $\Pi(\delta^s(\mathbf{x});)$: $\{N^2 = 25; f_{\text{max}} = 2.0\}$, $\Pi(s)$: $\{N^2 = 25; f_{\text{max}} = 2.0\}$ and $\Pi(k)$: $\{N^2 = 9; f_{\text{max}} = 1.0\}$.

Encoder	Batch	Crop	Initial LR
EDSR-baseline [21]	16	48	$1e^{-4}$
RDN [40]	16	48	$1e^{-4}$
SWINIR [20]	32	48	$2e^{-4}$
SWINIR-lightweight [20]	64	64	$2e^{-4}$
ABPN [10]	16	64	$1e^{-3}$

Table 8. **Training hyper-parameters**: replicate each encoder's original setting.

Encoder	C_e	C_h	Params(K)
EDSR-baseline RDN SWINIR	64	32	10
SWINIR-lightweight	60	32	10
ABPN (k as input)	28	28	5
ABPN (k as output)			11

Table 9. **CUF's hyper-parameters**: C_e chosen as each the encoder output features.

B. On the use of ensemble

Multi-scale up-sampling methods - DIV2k								
Encoder	Upsampler	Ens.	seen scales			unseen scales		
			×2	×3	×4	×6	×12	
EDSR-baseline [21]	Sub-pixel conv.		34.69	30.94	28.97	–	–	
	LIIF		34.63	30.95	28.97	26.72	23.66	
	LTE		34.63	30.99	29.01	26.77	23.74	
	CUF (ours)		34.70	30.99	29.01	26.76	23.73	
	Sub-pixel conv.	$+_{\text{geo}}$	34.78	31.03	29.06	–	–	
	LIIF	$+_{\text{geo}}$	34.74	31.05	29.07	26.80	23.76	
	LTE	$+_{\text{geo}}$	34.72	31.07	29.08	26.83	23.79	
	CUF (ours)	$+_{\text{geo}}$	34.79	31.07	29.09	26.82	23.78	

Table 10. **Disentangling the effect of ensembling on optimization**: Models trained under same supervision (no ensemble), and tested with (marked with $+_{geo}$) and without geometric self-ensemble. Results on DIV2K's validation subset [25].

The baseline settings from LIIF [6] and LTE [19] include locally ensembling pixels around the target sub-pixel, by shifting by its position by half pixel in the low resolution grid, and averaging their results. This procedure introduces a training and inference overhead, as the sampled points is increased by a factor of four. As a direct consequence, during training models adopting local self-ensemble evaluate four times more gradients per optimization step. In order to disentangle possible optimization side effects, in this section we ablate the models LIIF and LTE under same optimization conditions as other models, that is, no ensemble is adopted during training, but on inference time only. LTE presents a strong result on scales 3 and larger, but a reduction in performance on scale 2, in which LIIF matches or surpass its performance. Overall, this ablation confirms the benefits from CUF as the lighter arbitrary scale upsampler with strong performance under single pass and self-ensemble settings, across both smaller and larger scales.

C. On the use of positional encoding

Figure 11 contains a comparison of CUF models trained using the neural-fields parameters as raw values versus the projection using positional encoding. The stronger impact of using positional encoding is observed on ABPN encoder (Figure 12) and Urban-100 dataset [17]. We note that the content of this dataset is characterized by sharp straight lines and geometric structures, thus the quantitative gain is aligned with the expected behaviour.

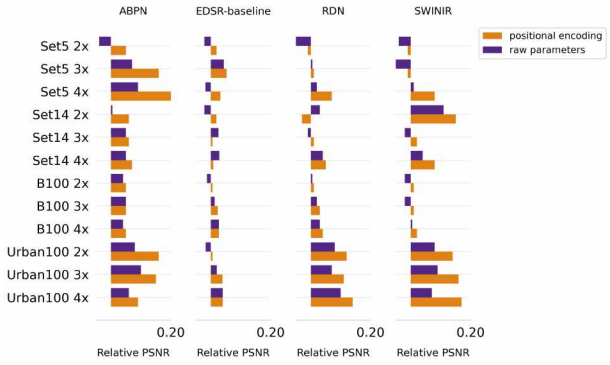

Figure 11. Impact of CUF's **positional encoding** on different datasets and encoders. Bar plots represent PSNR differences relative to baseline models adopting sub-pixel convolutions and corresponding encoder.

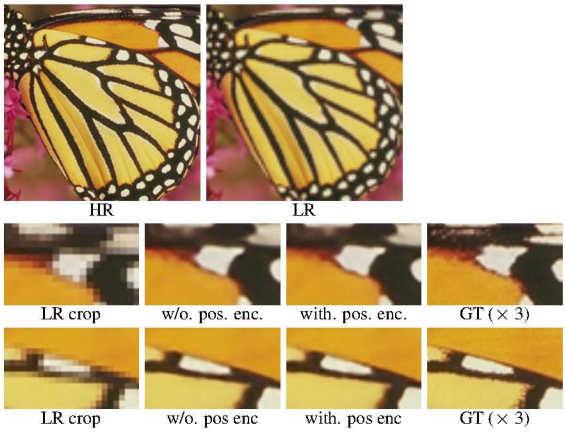

Figure 12. **Qualitative evaluation on mobile-compatible encoder** – ABPN-CUF with and without positional encoding. Butterfly image from Set5 dataset.

D. Conditioning on the kernel indexes

Figure 13 contains a comparison between representing the kernel indexes k_i, k_j as input parameters to the neural-fields versus representing their discrete set as individual neurons at the output of the hyper-network. On the efficiency side, setting them as the hyper-network output neurons reduces memory and computation used, as hidden layers are shared. On the other hand, the layers of the hyper-network and its nonlinearities provide additional expressiveness compared to the linear transformation used in the multi-headed version. This additional expressiveness results in in performance improvement in stronger encoders (RDN, SWINIR), but not on smaller ones (ABPN, EDSR). Thus, we recommend conditioning on kernel indexes only for those encoders that take advantage of it.

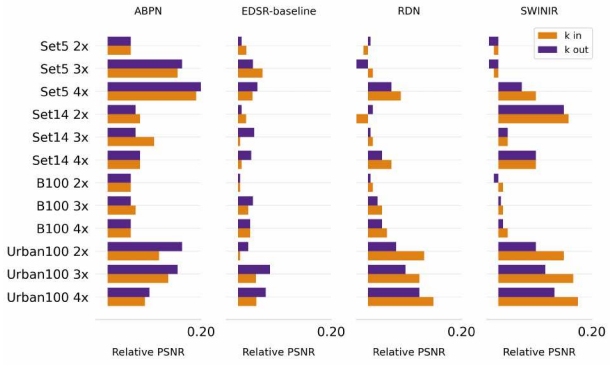

Figure 13. **Comparison between conditioning the neural-fields on the kernel indexes (k_i, k_j) versus its discretization at the hyper-network output layer.** Stronger encoders take advantage of the hyper-network depth and non linearities. Bar plots represent PSNR differences relative to baseline models adopting sub-pixel convolutions and corresponding encoder.

E. Sub-Pixel Convolution vs. CUF-instantiated

In this section we compare the costs associated with Sub-Pixel Convolution and CUF-instantiated upsampling heads. The presented comparison contrasts their designs choices based on full convolution (Sub-Pixel Convolution) versus depthwise-pointwise decomposition (CUF). As unitary element of comparison we evaluate the number of multiplications performed to produce a single output pixel. We assume an input feature map with C_{in} channels, the resulting image with C_{out} channels and that both Sub-Pixel Convolution and CUF-instantiated adopt filters of same size K.

CUF-instantiated architecture is composed with a depthwise convolution and pointwise projections. Its three layers perform respectively $C_{in} * k^2$, $C_{in} * C_{in}$ and $C_{in} * C_{out}$ multiplications per output pixel.

Next, we cover two common compositions with Sub-Pixel Convolution. The most common design for upsampler heads targeting high quality results is to combine a Sub-Pixel Convolution layer with a pointwise layer projecting from C_{in} into RGB channels (C_{out}) ([20, 21, 40]). In this setting, both Sub-Pixel Conv. and CUF-instantiated have identical output layers, that is removed from our analysis. The number of multiplications performed by the Sub-Pixel Convolution layer alone per target subpixel is: $C_{in} * k * k * C_{in}$. Thus, the fraction of multiplications performed by CUF-instantiated in relation to Sub-Pixel Convolution can be expressed as: $\frac{k^2+C_{in}}{k^2*C_{in}} = \frac{1}{C_{in}} + \frac{1}{k^2}$. That is, the decomposition has the desired effect of saving computation whenever $C_{in}, k > 1$.

An alternative use of a Sub-Pixel Convolution layer is its direct use as output layer. In this case, the three layers that compose CUF's upsampling head are compared to the full

expansion convolution alone. Thus, the total operations performed by CUF-instantiated is smaller that those performed by Sub-Pixel Convolution whenever $k^2 + C_{in} + C_{out} < k^2 * C_{out}$.

In practice, the depthwise-pointwise decomposition adopted by CUF-instantiated faces a memory drawback of storing an extra feature map created in-between the decomposition layers C_{in} (Figure 1). The reduction of such drawback with fused-convolutions is left as future work [1].

F. Qualitative comparisons

The difference between existent arbitrary-scale upsamplers can only be observed at textured regions of the image. In this section we disentangle the rule of the encoder and upsampler in the perceived quality of the results. Figure 16 contain additional results, with non-integer scale factors.

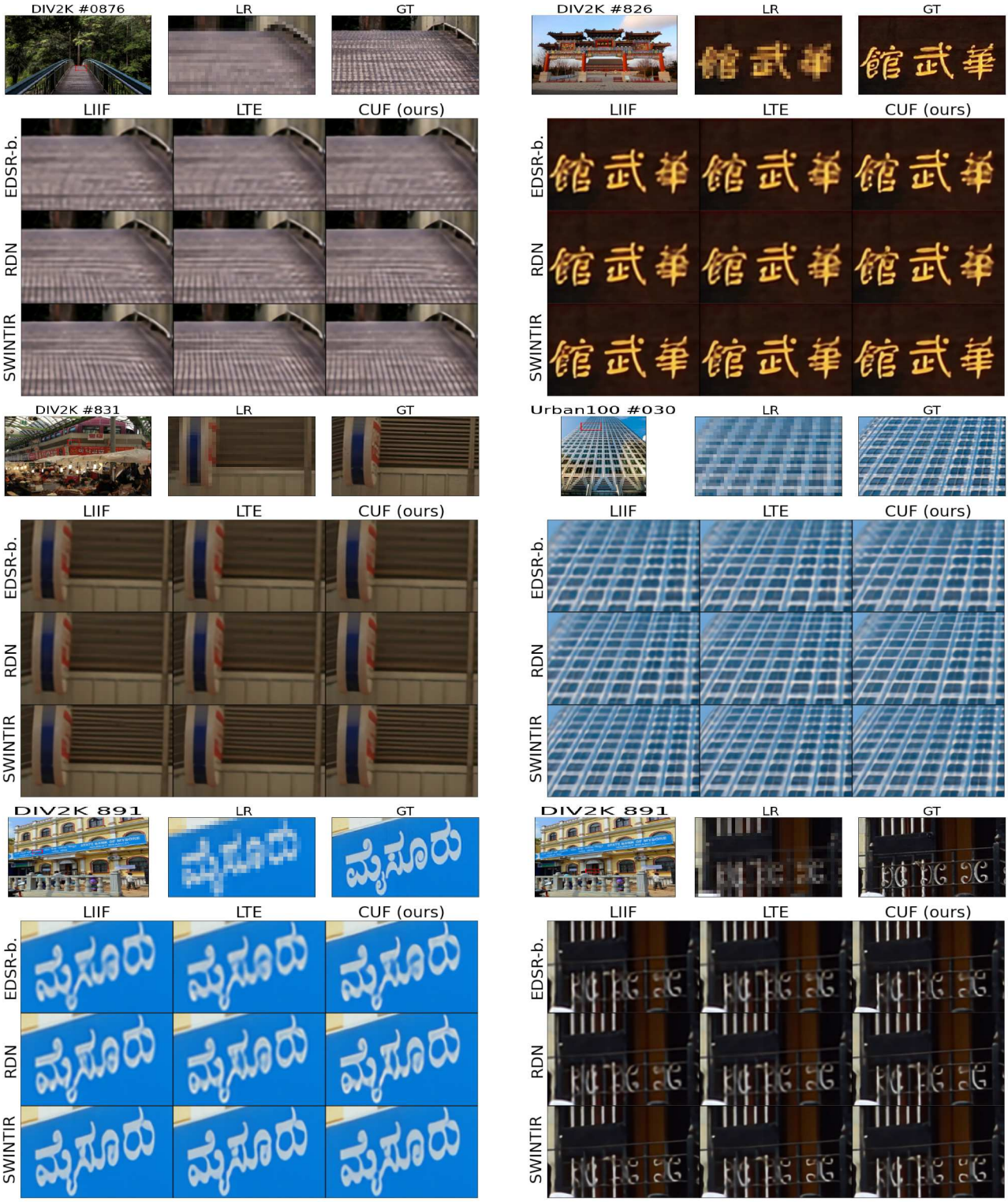

Figure 14. Qualitative comparisons of arbitrary-scale super resolution methods using different encoders. Scale factor $4\times$

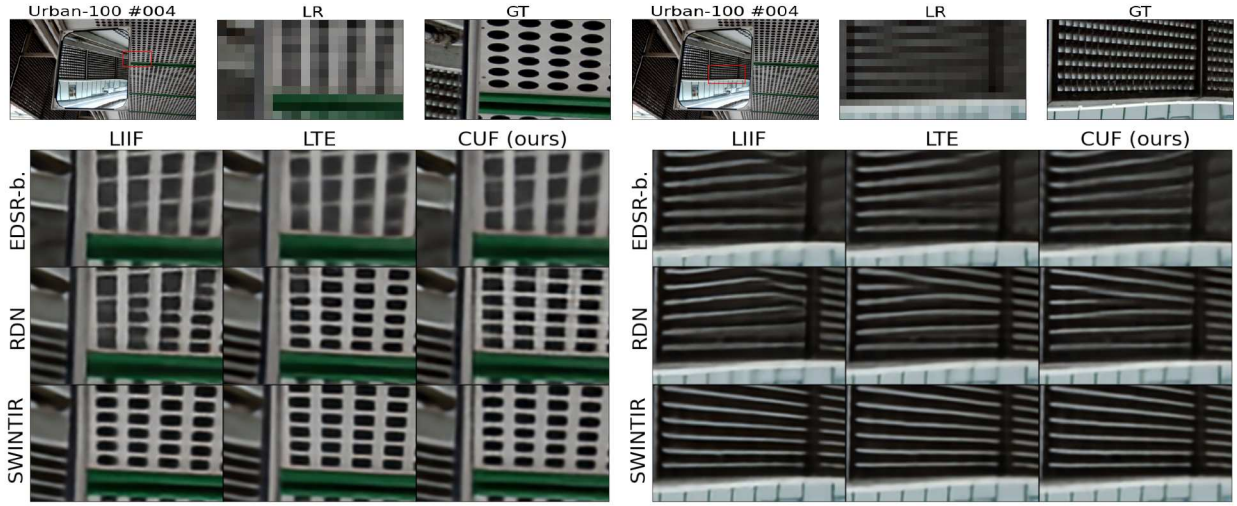

Figure 15. Hard cases: Aliasing artifacts observed on hard cases. Our upsampler produces the sharper results. Scale factor 8×

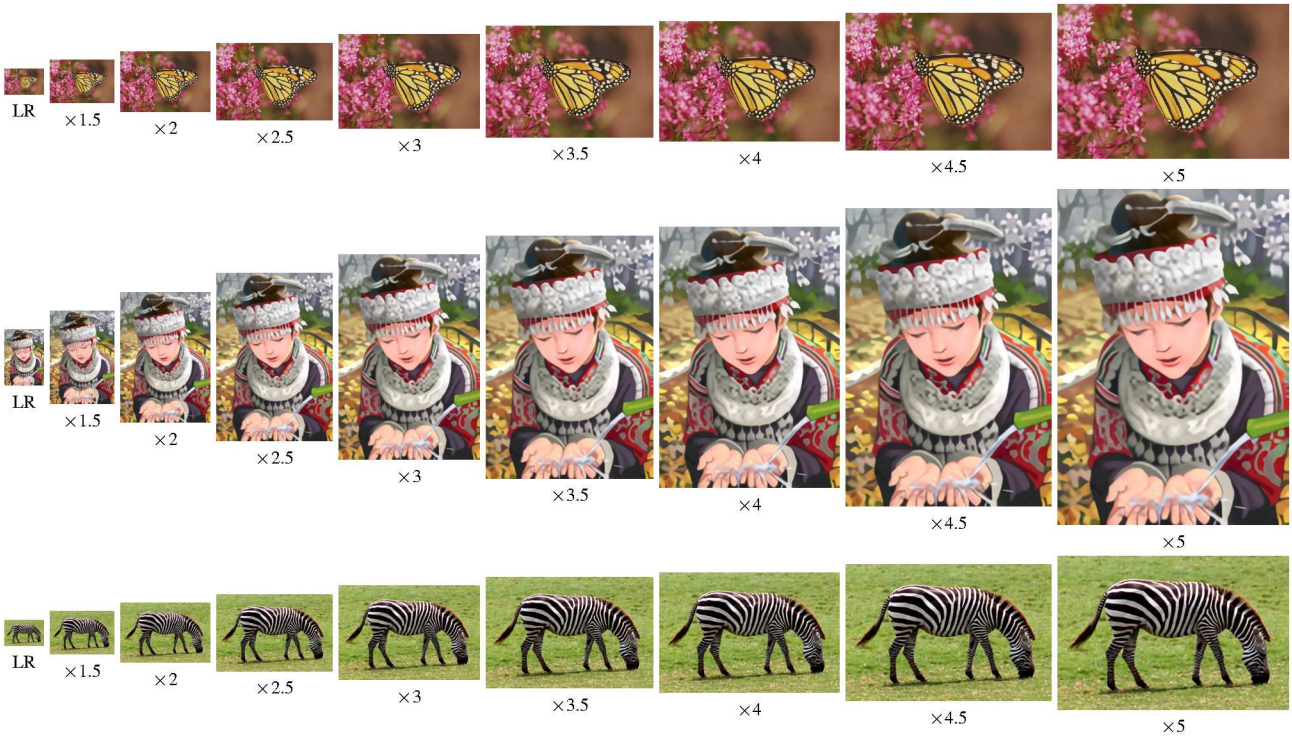

Figure 16. Qualitative results using non-integer scales. Images from Set14 dataset.