Kernel Aware Resampler - Supplemental Material

Michael Bernasconi[1,2] Abdelaziz Djelouah[2] Farnood Salehi[2] Markus Gross[1,2] Christopher Schroers[2]

[1]ETH Zürich [2]DisneyResearch|Studios

michael.bernasconi@inf.ethz.ch, abdelaziz.djelouah@disney.com

1. Overview

In this supplementary document we show some additional results and provide details of our methods architecture and training procedure in sections 2 and 3. Additional details of the degradation estimation algorithm are presented in section 4.

An important aspect of this work the careful implementation of traditional resampling kernels and their combination with arbitrary warps. Some insights regarding this are provided in section 5. We conclude with more visual results corresponding to our various experiments (ablation, comparisons, etc.).

2. Network Architecture

In this section we describe the network architectures in depth. Our methods architecture can be broken down into four distinct parts: degradation encoder, feature extractor, geometry encoder, and prediction network. The feature extractor is implemented using the ProSR architecture. The degradation encoder, geometry encoder, and prediction network are implemented as MLPs. A detailed breakdown is shown in Table 1. In total KARL c and KARL k consist of $2'909'212$ and $2'914'866$ parameters respectively.

For the error estimation network we choose the ProSR architecture. The network has a total of $241'267$ parameters. A detailed breakdown is shown in Table 2.

Degradation Encoder (MLP)	
input features	225 $(25 \cdot 25)$
intermediate features	256
output features	25
activation	ReLU
number of parameters	$166'681$
Feature Extractor (ProSR)	
input features	28 $(3 + 25)$
intermediate features	160
number of dense compression units	5
number of layers (in dense blocks)	4
growth rate (in dense blocks)	40
number of parameters	$2'157'280$
Geometry Encoder (MLP)	
input features	6 (2 sampling offsets, 4 Jacobian)
intermediate features	256
output features	32
activation	ReLU
number of parameters	$10'016$
Prediction Network (MLP)	
input features	1472 $(3 \cdot 3 \cdot 160 + 32)$
intermediate channels	$256, 256, 256, 256$
output features (KARL c)	3
output features (KARL k)	25 $(5 \cdot 5)$
activation	ReLU
number of parameters (KARL c)	$575'235$
number of parameters (KARL k)	$580'889$

Table 1. Architecture details for KARL c and KARL k.

Error Estimator (ProSR)	
input features	3
intermediate features	64
number of dense compression units	3
number of layers (in dense blocks)	4
growth rate (in dense blocks)	20
number of parameters	$241'267$

Table 2. Architecture details for the error estimator.

3. Training Procedure

In this section we provide further details about the training procedure. All training is performed using PyTorch [2].

3.1. KARL

Our method is trained in a fully supervised way as illustrated in Algorithm 1. The network is trained for $1e6$ steps using image crops of size 64×64 and a batch size of 8. We use the Adam [1] optimizer with a learning rate of $1e^{-4}$.

3.2. Error Estimator

The error estimator is trained in a fully supervised way as illustrated in Algorithm 2. The model is trained for $1e6$ steps using image crops of size 64×64 and a batch size of 16. We use the Adam [1] optimizer with a learning rate of

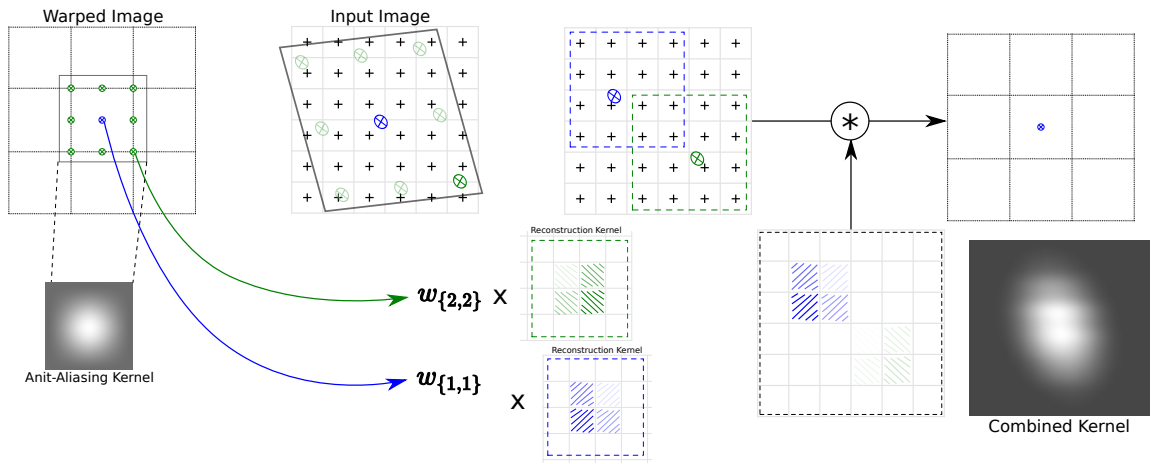

Figure 1. Visual illustration of the kernel combination process to create the low resolution images. In order to compute the combined kernel for a given pixel (blue pixel) in the warped image an anti-aliasing kernel is applied centered around this pixel. The anti-aliasing kernel computes a weighted sum of the pixels in a given neighborhood around the center pixel. The pixels in this neighborhood are shown in green. The color values for each pixel in the neighborhood is computed by first projecting it to the input image and then interpolating the value in the input image using the reconstruction kernel. The reconstruction kernel itself computes a weighted sum of the pixels in the input image. This means that the final pixel in the output image is a weighted sum of weighted sums of pixels in the input image. This sum of sums can be reduced to a single weighted sum (a.k.a a kernel).

Algorithm 1: Pseudocode illustrating how training data is generated and how the loss is computed for our method.

$I \leftarrow$ sample HR image
$W \leftarrow$ sample warp
$\mathcal{K}_R, \mathcal{K}_A \leftarrow$ sample kernels
$\mathcal{K}_W \leftarrow$ combine_kernels$(W, \mathcal{K}_R, \mathcal{K}_A)$
$G_W, G_{W^{-1}} \leftarrow$ compute_warp_grids(W)
$I' \leftarrow$ apply_kernels$(I, G_W, \mathcal{K}_W)$
if *KARL_k* **then**
  $\mathcal{K}_{W^{-1}} \leftarrow$ KARL$(\mathcal{K}_W, I', G_{W^{-1}})$
  $I^* \leftarrow$ apply_kernels$(I', G_{W^{-1}}, \mathcal{K}_{W^{-1}})$
else
  $I^* \leftarrow$ KARL$(\mathcal{K}_W, I', G_{W^{-1}})$
end
$loss \leftarrow L1(I^*, I)$
$loss$.backward()
optimize(KARL)

Algorithm 2: Pseudocode illustrating how the error estimator is trained. We use $N = 5$.

$I \leftarrow$ sample HR image
$W \leftarrow$ sample warp
$\mathcal{K}_{R_{1\ldots N}}, \mathcal{K}_{A_{1\ldots N}} \leftarrow$ sample N kernels
$\mathcal{K}_{W_{1\ldots N}} \leftarrow$ combine_kernels$(W, \mathcal{K}_{R_{1\ldots N}}, \mathcal{K}_{A_{1\ldots N}})$
$G_W, G_{W^{-1}} \leftarrow$ compute_warp_grids(W)
$I'_{1\ldots N} \leftarrow$ apply_kernels$(I, G_W, \mathcal{K}_{W_{1\ldots N}})$
for $i \in 1\ldots N$ **do**
  $I^* \leftarrow$ KARL$(\mathcal{K}_{W_i}, I'_i, G_{W^{-1}})$
  for $j \in i\ldots N$ **do**
    $\hat{I}^* \leftarrow$ KARL$(\mathcal{K}_{W_j}, I'_i, G_{W^{-1}})$
    $E \leftarrow |\hat{I}^* - I^*|$
    $E^* \leftarrow$ ErrorEstimator$(\hat{I}^*)$
    $loss \leftarrow L1(E^*, E)$
    $loss$.backward()
  end
end
optimize(ErrorEstimator)

$1e^{-4}$.

4. Degradation Estimation

We estimate the degradation map K_W in two steps. First, a coarse grid search over a number of different reconstruction kernels and anti-aliasing kernels is performed. Then, the best combination of kernels is further optimized using gradient descent. This procedure is illustrated in Algorithm 3. In order to reduce memory consumption we perform the estimation on randomly sampled 48×48 patches.

5. Combining Kernels

Figure 1 illustrates the combination of kernels to create the low resolution images. In order to compute the combined kernel for a given pixel (blue pixel) in the warped im-

Algorithm 3: Pseudocode illustrating how the degradation map is estimated.

I' : Input Image W : Warp
$G_W, G_{W^{-1}} \leftarrow$ compute_warp_grids(W)
$\mathcal{K}_{R_{1\ldots N}}, \mathcal{K}_{A_{1\ldots N}}$: initial set of kernels
α, β : kernel support weights

$E^*_{min} \leftarrow inf$
for $i \in 1..N$ **do**
  $\mathcal{K}_{W_i} \leftarrow$ combine_kernels$(W, \mathcal{K}_{R_i}, \mathcal{K}_{A_i})$
  $I^* \leftarrow$ KARL$(\mathcal{K}_{W_i}, I', G_{W^{-1}})$
  $E^* \leftarrow$ mean(ErrorEstimator(I^*))
  if $E^* < E^*_{min}$ **then**
    $E^*_{min} \leftarrow E^*$
    $\mathcal{K}_R, \mathcal{K}_A \leftarrow \mathcal{K}_{R_i}, \mathcal{K}_{A_i}$
  end
end

for $j...M$ **do**
  $\mathcal{K}_W \leftarrow$ combine_kernels$(W, \mathcal{K}_R, \mathcal{K}_A)$
  $I^* \leftarrow$ KARL$(\mathcal{K}_W, I', G_{W^{-1}})$
  $e \leftarrow$ mean(ErrorEstimator(I^*))
  $loss \leftarrow e + \alpha \cdot |\mathcal{K}_R| + \beta \cdot |\mathcal{K}_A|$
  $loss$.backward()
  optimize$(\mathcal{K}_R, \mathcal{K}_A)$
end

$\mathcal{K}_W \leftarrow$ combine_kernels$(W, \mathcal{K}_R, \mathcal{K}_A)$
return $\mathcal{K}_W$

age an anti-aliasing kernel is applied centered around this pixel. The anti-aliasing kernel computes a weighted sum of the pixels in a given neighborhood around the center pixel. The pixels in this neighborhood are shown in green. The color values for each pixel in the neighborhood is computed by first projecting it to the input image and then interpolating the value in the input image using the reconstruction kernel. The reconstruction kernel itself computes a weighted sum of the pixels in the input image. This means that the final pixel in the output image is a weighted sum of weighted sums of pixels in the input image. This sum of sums can be reduced to a single weighted sum (a.k.a a kernel).

6. Ablation

6.1. Geometric information

Figure 2 visually illustrates the importance of providing information about the local distortion. Especially the sampling offsets are required in order for the model to be able to adapt to sub pixel offsets.

Algorithm 4: Pseudocode illustrating how a spatially variant kernel is applied to an image.

$I \in \mathbb{R}^{H \times W}$: Image
$G_W \in \mathbb{R}^{H' \times W' \times 2}$: Warp Grid
$K_W \in \mathbb{R}^{H' \times W' \times h \times w}$: Kernel Map

$I' \leftarrow 0^{H' \times W'}$
for $i, j \in 1...H', 1...W'$ **do**
  $x, y \leftarrow G_{W_{i,j}}$
  for $l, m \in 1...h, 1...w$ **do**
    $I'_{i,j} += I_{\lfloor y+l-\frac{h}{2} \rceil, \lfloor x+m-\frac{w}{2} \rceil} \cdot K_{W_{l,m}}$
  end
end

6.2. Degradation

Figures 3 and 4 visually illustrate the importance of training on images that were downscaled using a variety of kernels as well as providing the degradation map to the model. The "fixed deg (m)" column shows the results for a model that was trained using a fixed kernel (corresponding to the medium kernel). This model performs well when evaluated on the medium kernel but struggles to generalize to other kernels. Especially when aliasing is present in the input image (small kernel) the model produces artifacts. These artifacts are reduces when the model is trained on a variety of kernels ("variable deg" column). This model, however, tends to produce blurry results especially when the input image was downscaled using a large kernel. Finally, the "deg aware" column shows our degradation aware model. This model does not produce artifacts when aliasing is present in the input image and also manages to produce sharp results when the input image is overblurred.

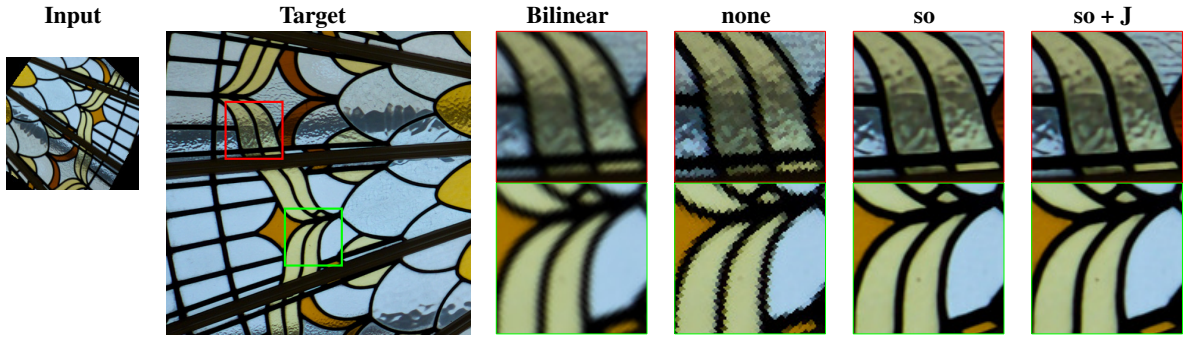

Figure 2. Visual comparison of models with access to different geometric information. When no geometric information is provided ("none" column) the method cannot adapt to sub pixel offsets. This results in strong artifacts. Providing the sampling offsets ("so" column) allows the method adapt to sub pixel offsets and results in a large increase in performance. Finally, providing both sampling offsets and the local Jacobian ("so+J" column) still produces additional improvements but less significant.

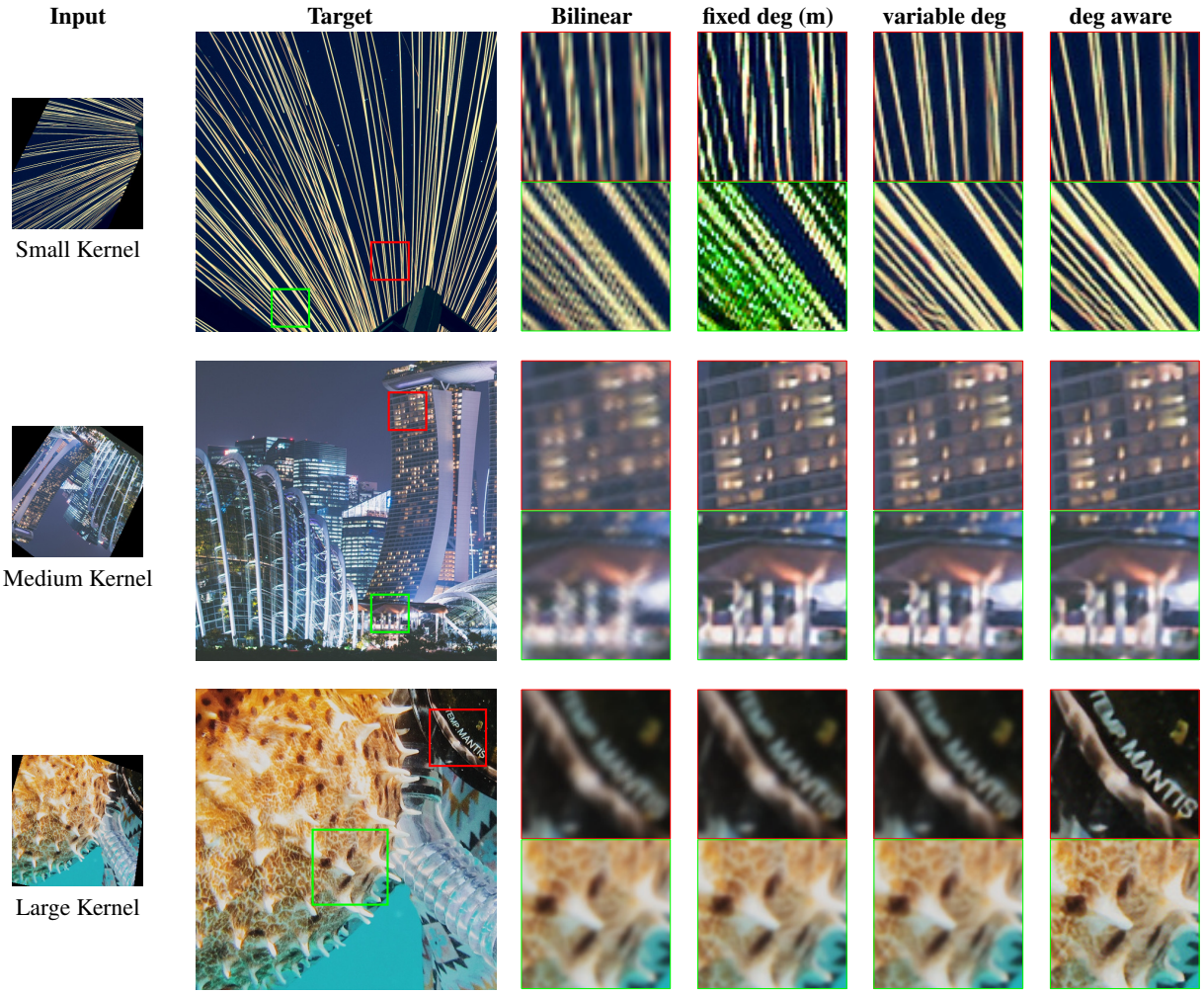

Figure 3. Visual results for the ablation study on the degradation map, using a projective transforms with an average local scaling factor of $\times 2$. Providing the degradation map to the model leads to the best results.

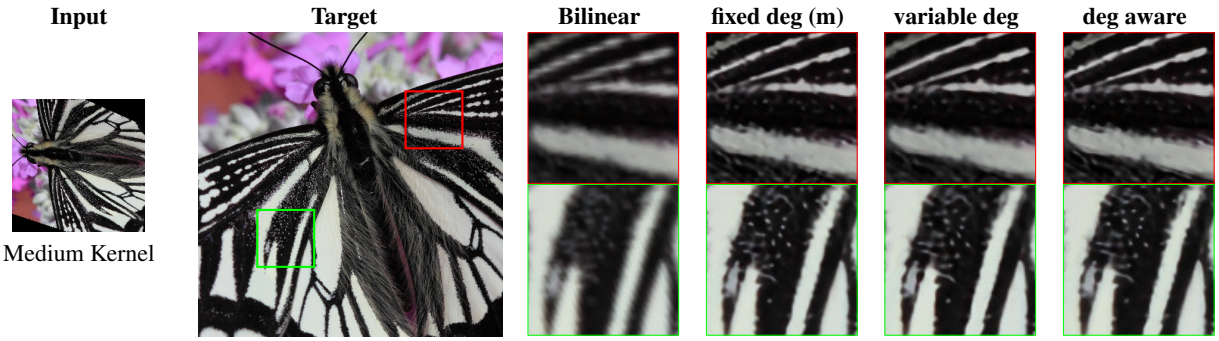

Figure 4. Visual results for the ablation study on the degradation map, using a projective transforms with an average local scaling factor of $\times 3$. Providing the degradation map to the model leads to the best results

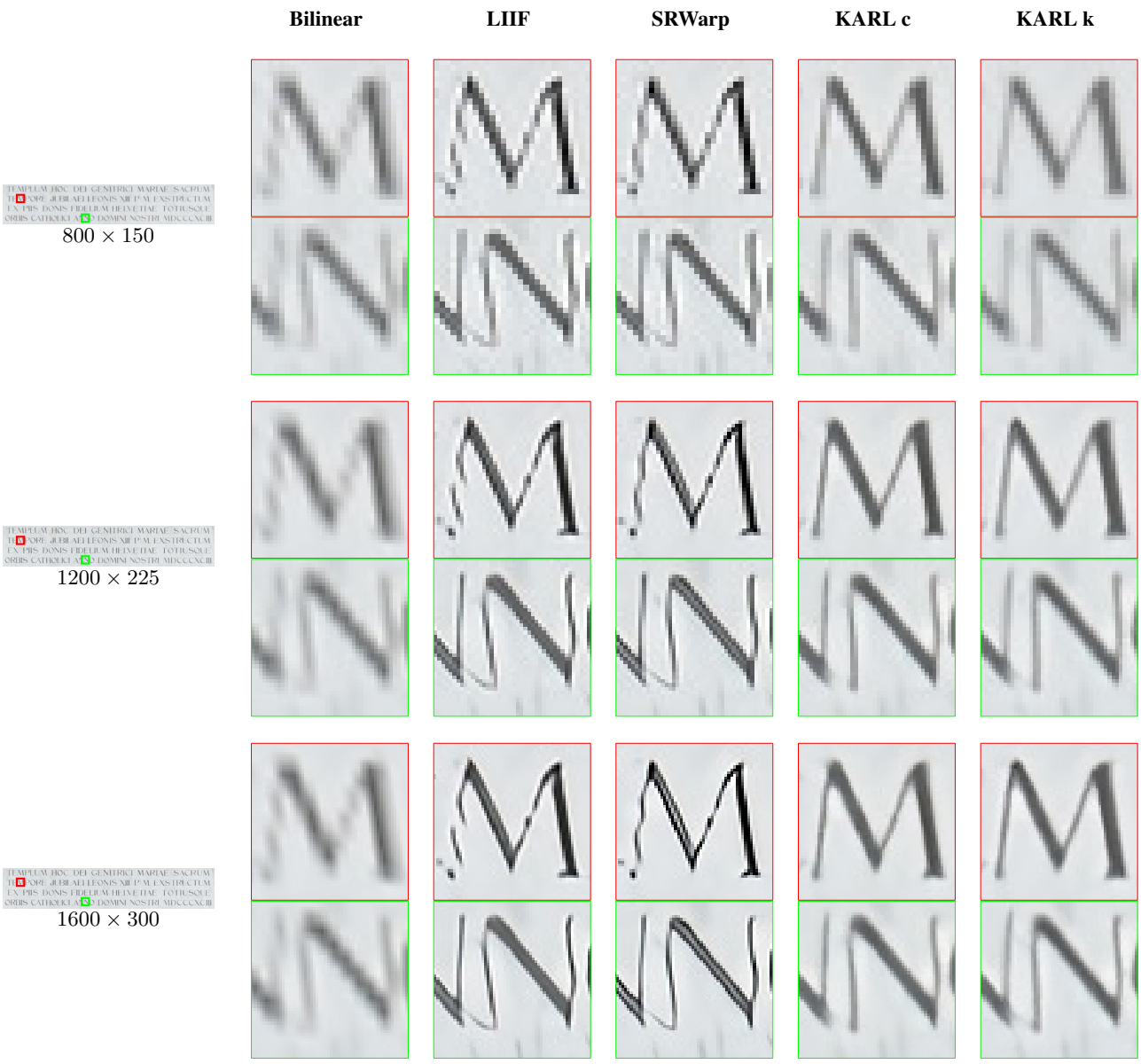

Figure 5. Comparison of our method against LIIF and SRWarp on an image rectification example across different scales. The input is directly taken by a camera and not downscaled. The degradation map is unknown and automatically estimated by our model. We can see that our method manages to consistently produce high quality results for every scale factor. LIIF and SRWarp on the other hand are not able tp properly reconstruct the letters and produce oversharpening artifacts.

	bilinear	LIIF	SRWarp	KARL c	KARL k	KARL c est. deg	KARL k est. deg
$\times 1.5$ bicubic s	25.53	27.32	27.44	30.60	30.55	27.98	27.97
$\times 1.5$ bicubic m	27.03	34.10	34.05	33.54	33.72	28.10	29.24
$\times 1.5$ bicubic l	25.09	26.48	26.45	30.45	31.07	28.99	28.73
$\times 1.5$ gaussian s	26.92	31.08	31.07	32.49	32.63	28.46	28.88
$\times 1.5$ gaussian m	26.41	30.59	30.50	33.67	33.96	27.44	29.15
$\times 1.5$ gaussian l	23.83	24.40	24.38	28.51	29.49	26.99	27.64
$\times 1.5$ lanczos s	23.82	21.37	21.77	27.69	27.68	26.10	26.12
$\times 1.5$ lanczos m	24.82	25.36	25.70	32.55	32.41	31.88	31.57
$\times 1.5$ lanczos l	23.83	24.93	24.97	24.31	26.11	26.20	26.37
$\times 2.0, 1.5$ bicubic s	25.73	27.80	27.84	29.64	29.74	27.11	27.66
$\times 2.0, 1.5$ bicubic m	26.30	31.79	31.73	31.38	31.47	31.15	28.28
$\times 2.0, 1.5$ bicubic l	24.28	25.37	25.32	29.15	29.66	27.83	28.03
$\times 2.0, 1.5$ gaussian s	26.35	29.82	29.79	30.85	30.94	27.28	28.37
$\times 2.0, 1.5$ gaussian m	25.61	29.01	28.88	31.49	31.56	30.72	28.31
$\times 2.0, 1.5$ gaussian l	23.10	23.56	23.54	27.64	27.96	25.99	26.58
$\times 2.0, 1.5$ lanczos s	23.89	20.57	20.70	27.88	27.89	26.71	26.75
$\times 2.0, 1.5$ lanczos m	24.31	24.10	24.66	30.60	30.50	30.07	29.47
$\times 2.0, 1.5$ lanczos l	23.17	24.02	24.06	23.61	25.15	22.45	25.33
$\times 2.0$ bicubic s	25.94	28.44	28.28	29.96	29.88	26.79	28.08
$\times 2.0$ bicubic m	25.75	30.47	30.47	30.15	30.09	29.94	28.17
$\times 2.0$ bicubic l	23.71	24.63	24.63	28.24	28.49	27.15	27.37
$\times 2.0$ gaussian s	25.93	28.83	28.70	29.95	29.88	26.78	27.95
$\times 2.0$ gaussian m	25.04	28.01	28.00	30.13	30.08	30.11	27.31
$\times 2.0$ gaussian l	22.59	22.99	22.99	26.98	27.10	25.36	26.16
$\times 2.0$ lanczos s	23.76	19.65	19.26	28.46	28.32	27.53	28.08
$\times 2.0$ lanczos m	23.93	23.33	23.67	29.53	29.43	29.05	28.81
$\times 2.0$ lanczos l	22.70	23.39	23.40	23.67	24.45	21.46	24.56
$\times 2.5, 2.0$ bicubic s	24.31	24.85	24.91	28.06	28.05	25.56	26.99
$\times 2.5, 2.0$ bicubic m	24.64	29.18	29.16	28.88	28.93	25.15	28.18
$\times 2.5, 2.0$ bicubic l	23.00	23.95	23.92	27.46	27.50	26.48	26.29
$\times 2.5, 2.0$ gaussian s	24.66	26.78	26.77	28.53	28.54	28.35	27.38
$\times 2.5, 2.0$ gaussian m	24.10	27.04	26.93	28.96	29.01	28.93	27.08
$\times 2.5, 2.0$ gaussian l	22.01	22.43	22.42	26.22	26.30	24.95	25.26
$\times 2.5, 2.0$ lanczos s	22.27	18.52	18.52	26.70	26.64	25.59	26.47
$\times 2.5, 2.0$ lanczos m	22.95	22.55	23.11	28.35	28.41	27.90	27.78
$\times 2.5, 2.0$ lanczos l	21.93	22.68	22.72	23.05	23.71	20.46	23.86
$\times 2.5$ bicubic s	23.39	23.12	23.12	27.11	27.13	24.90	26.44
$\times 2.5$ bicubic m	23.90	28.31	28.31	28.04	28.09	27.98	27.65
$\times 2.5$ bicubic l	22.50	23.46	23.44	26.84	26.70	25.88	25.71
$\times 2.5$ gaussian s	23.84	25.44	25.43	27.60	27.62	27.29	26.76
$\times 2.5$ gaussian m	23.47	26.34	26.27	28.12	28.20	28.06	26.58
$\times 2.5$ gaussian l	21.60	22.04	22.03	25.64	25.76	24.56	25.03
$\times 2.5$ lanczos s	21.40	17.73	17.79	25.73	25.76	24.39	25.61
$\times 2.5$ lanczos m	22.30	22.05	22.58	27.61	27.65	26.95	27.05
$\times 2.5$ lanczos l	21.37	22.14	22.18	22.46	22.82	19.61	23.16
$\times 3.0$ bicubic s	22.56	21.91	21.83	25.77	25.77	23.91	25.26
$\times 3.0$ bicubic m	23.02	26.82	26.86	26.62	26.66	23.37	26.57
$\times 3.0$ bicubic l	21.76	22.63	22.61	25.64	25.62	25.02	24.64
$\times 3.0$ gaussian s	22.96	24.05	23.97	26.22	26.24	25.42	25.61
$\times 3.0$ gaussian m	22.64	25.19	25.13	26.70	26.75	26.62	25.51
$\times 3.0$ gaussian l	20.94	21.35	21.34	24.59	24.84	24.22	24.38
$\times 3.0$ lanczos s	20.52	16.68	16.73	24.39	24.47	22.87	24.13
$\times 3.0$ lanczos m	21.57	21.18	21.70	26.27	26.30	25.28	25.90
$\times 3.0$ lanczos l	19.31	19.75	19.84	22.12	21.77	21.38	21.73
$\times 3.5$ bicubic s	21.95	21.10	20.92	24.75	24.77	23.29	24.16
$\times 3.5$ bicubic m	22.39	25.77	25.82	25.56	25.58	22.64	25.59
$\times 3.5$ bicubic l	21.20	21.99	22.01	24.61	24.69	24.61	23.96
$\times 3.5$ gaussian s	22.32	23.14	22.99	25.20	25.21	23.92	25.21
$\times 3.5$ gaussian m	22.02	24.30	24.28	25.59	25.61	24.74	24.16
$\times 3.5$ gaussian l	20.43	20.80	20.81	23.77	24.10	23.69	24.05
$\times 3.5$ lanczos s	19.95	16.02	16.00	23.36	23.47	21.77	23.29
$\times 3.5$ lanczos m	21.05	20.57	21.00	25.26	25.27	23.76	24.93
$\times 3.5$ lanczos l	18.11	18.38	18.43	20.91	20.92	20.96	20.97
$\times 4.0$ bicubic s	21.56	20.59	20.11	23.98	23.97	22.10	23.04
$\times 4.0$ bicubic m	21.95	24.97	25.10	24.64	24.65	22.84	22.54
$\times 4.0$ bicubic l	20.77	21.48	21.57	23.59	23.78	20.98	20.17
$\times 4.0$ gaussian s	21.91	22.49	22.13	24.38	24.39	22.69	24.39
$\times 4.0$ gaussian m	21.58	23.62	23.75	24.50	24.55	23.39	22.17
$\times 4.0$ gaussian l	20.02	20.36	20.42	23.13	23.51	21.75	19.44
$\times 4.0$ lanczos s	19.61	15.61	15.30	22.43	22.58	20.82	22.50
$\times 4.0$ lanczos m	20.70	20.13	20.13	24.33	24.10	22.52	24.10
$\times 4.0$ lanczos l	19.04	19.46	19.38	20.35	19.84	20.92	20.84
$P_{\times 2}$ bicubic s	26.27	27.54	27.70	29.96	30.11	27.00	28.14
$P_{\times 2}$ bicubic m	25.44	29.84	28.46	30.33	30.53	29.63	28.14
$P_{\times 2}$ bicubic l	23.40	24.20	24.03	27.86	28.01	25.79	25.86
$P_{\times 2}$ gaussian s	25.98	29.48	28.61	30.20	30.38	28.83	28.01
$P_{\times 2}$ gaussian m	24.59	26.97	26.32	30.20	30.40	28.85	27.96
$P_{\times 2}$ gaussian l	22.31	22.70	22.62	26.46	26.63	23.96	24.22
$P_{\times 2}$ lanczos s	26.58	21.97	23.71	29.09	29.19	28.43	28.29
$P_{\times 2}$ lanczos m	27.35	25.34	27.10	29.48	29.60	28.96	28.64
$P_{\times 2}$ lanczos l	23.63	23.25	23.34	23.41	24.16	22.07	23.93
$P_{\times 3}$ bicubic s	23.89	22.81	23.49	26.39	26.47	26.28	26.14
$P_{\times 3}$ bicubic m	23.36	26.87	25.83	26.81	26.88	26.77	26.28
$P_{\times 3}$ bicubic l	21.70	22.34	22.24	25.38	25.33	23.20	23.19
$P_{\times 3}$ gaussian s	23.74	24.77	24.80	26.63	26.72	26.60	26.28
$P_{\times 3}$ gaussian m	22.68	24.73	24.24	26.74	26.82	26.33	25.70
$P_{\times 3}$ gaussian l	20.80	21.13	21.09	24.29	24.58	21.66	21.63
$P_{\times 3}$ lanczos s	23.66	17.42	18.96	25.40	25.48	24.40	25.52
$P_{\times 3}$ lanczos m	24.54	22.16	23.64	26.29	26.31	25.45	25.57
$P_{\times 3}$ lanczos l	20.58	19.87	20.02	21.87	21.73	18.72	20.19

Table 3. Detailed numerical evaluation of our method. The metric that is shown is PSNR. All numbers were computed on 100 hard 512×512 crops from images in the DIV2K test set.

	bilinear	LIIF	SRWarp	KARL c	KARL k	KARL c est. deg	KARL k est. deg
$\times 1.5$ bicubic s	0.9093	0.9373	0.9394	0.9595	0.9604	0.9332	0.9299
$\times 1.5$ bicubic m	0.9253	0.9784	0.9783	0.9760	0.9770	0.9352	0.9457
$\times 1.5$ bicubic l	0.8668	0.8960	0.8948	0.9544	0.9584	0.9378	0.9346
$\times 1.5$ gaussian s	0.9280	0.9660	0.9664	0.9706	0.9719	0.9394	0.9417
$\times 1.5$ gaussian m	0.9075	0.9583	0.9573	0.9768	0.9779	0.9284	0.9450
$\times 1.5$ gaussian l	0.8213	0.8384	0.8373	0.9323	0.9442	0.9064	0.9183
$\times 1.5$ lanczos s	0.8773	0.8200	0.8335	0.9303	0.9307	0.9093	0.9067
$\times 1.5$ lanczos m	0.9054	0.9211	0.9269	0.9721	0.9721	0.9697	0.9667
$\times 1.5$ lanczos l	0.8584	0.8815	0.8819	0.8765	0.8912	0.8917	0.8903
$\times 2.0, 1.5$ bicubic s	0.9066	0.9375	0.9387	0.9490	0.9499	0.9166	0.9214
$\times 2.0, 1.5$ bicubic m	0.9092	0.9638	0.9636	0.9608	0.9616	0.9584	0.9307
$\times 2.0, 1.5$ bicubic l	0.8395	0.8679	0.8661	0.9384	0.9439	0.9189	0.9219
$\times 2.0, 1.5$ gaussian s	0.9148	0.9534	0.9537	0.9571	0.9579	0.9194	0.9314
$\times 2.0, 1.5$ gaussian m	0.8868	0.9392	0.9374	0.9619	0.9626	0.9550	0.9312
$\times 2.0, 1.5$ gaussian l	0.7921	0.8083	0.8070	0.9141	0.9241	0.8818	0.8961
$\times 2.0, 1.5$ lanczos s	0.8785	0.7934	0.8034	0.9278	0.9273	0.9126	0.9088
$\times 2.0, 1.5$ lanczos m	0.8914	0.8989	0.9093	0.9563	0.9559	0.9527	0.9434
$\times 2.0, 1.5$ lanczos l	0.8312	0.8529	0.8534	0.8485	0.8617	0.8327	0.8656
$\times 2.0$ bicubic s	0.9045	0.9394	0.9384	0.9467	0.9468	0.9088	0.9242
$\times 2.0$ bicubic m	0.8950	0.9518	0.9516	0.9493	0.9492	0.9456	0.9259
$\times 2.0$ bicubic l	0.8173	0.8448	0.8442	0.9265	0.9296	0.9052	0.9088
$\times 2.0$ gaussian s	0.9036	0.9425	0.9417	0.9470	0.9471	0.9099	0.9229
$\times 2.0$ gaussian m	0.8694	0.9235	0.9228	0.9506	0.9503	0.9480	0.9142
$\times 2.0$ gaussian l	0.7690	0.7844	0.7838	0.9003	0.9089	0.8641	0.8845
$\times 2.0$ lanczos s	0.8753	0.7616	0.7562	0.9285	0.9280	0.9183	0.9234
$\times 2.0$ lanczos m	0.8794	0.8831	0.8903	0.9442	0.9442	0.9414	0.9343
$\times 2.0$ lanczos l	0.8088	0.8293	0.8296	0.8347	0.8343	0.8015	0.8418
$\times 2.5, 2.0$ bicubic s	0.8704	0.8891	0.8921	0.9217	0.9225	0.8828	0.9040
$\times 2.5, 2.0$ bicubic m	0.8665	0.9366	0.9360	0.9320	0.9336	0.8755	0.9212
$\times 2.5, 2.0$ bicubic l	0.7897	0.8213	0.8200	0.9108	0.9110	0.8900	0.8865
$\times 2.5, 2.0$ gaussian s	0.8746	0.9166	0.9173	0.9277	0.9288	0.9286	0.9104
$\times 2.5, 2.0$ gaussian m	0.8409	0.9046	0.9023	0.9332	0.9351	0.9330	0.9067
$\times 2.5, 2.0$ gaussian l	0.7434	0.7611	0.7601	0.8806	0.8890	0.8494	0.8604
$\times 2.5, 2.0$ lanczos s	0.8350	0.7202	0.7261	0.9007	0.9011	0.8840	0.8958
$\times 2.5, 2.0$ lanczos m	0.8512	0.8626	0.8751	0.9273	0.9296	0.9240	0.9181
$\times 2.5, 2.0$ lanczos l	0.7782	0.8037	0.8045	0.8077	0.8116	0.7675	0.8145
$\times 2.5$ bicubic s	0.8452	0.8535	0.8562	0.9047	0.9058	0.8657	0.8913
$\times 2.5$ bicubic m	0.8433	0.9242	0.9233	0.9183	0.9201	0.9205	0.9110
$\times 2.5$ bicubic l	0.7677	0.8020	0.8009	0.8967	0.8931	0.8748	0.8722
$\times 2.5$ gaussian s	0.8513	0.8949	0.8959	0.9120	0.9133	0.9132	0.8971
$\times 2.5$ gaussian m	0.8180	0.8890	0.8867	0.9195	0.9216	0.9201	0.8950
$\times 2.5$ gaussian l	0.7234	0.7427	0.7418	0.8643	0.8734	0.8361	0.8515
$\times 2.5$ lanczos s	0.8059	0.6897	0.6970	0.8817	0.8823	0.8559	0.8773
$\times 2.5$ lanczos m	0.8284	0.8479	0.8598	0.9140	0.9159	0.9097	0.9046
$\times 2.5$ lanczos l	0.7530	0.7822	0.7832	0.7837	0.7812	0.7371	0.7897
$\times 3.0$ bicubic s	0.8135	0.8198	0.8211	0.8753	0.8768	0.8367	0.8635
$\times 3.0$ bicubic m	0.8087	0.8969	0.8962	0.8897	0.8917	0.8211	0.8879
$\times 3.0$ bicubic l	0.7322	0.7668	0.7658	0.8658	0.8653	0.8525	0.8395
$\times 3.0$ gaussian s	0.8180	0.8631	0.8637	0.8829	0.8848	0.8806	0.8695
$\times 3.0$ gaussian m	0.7824	0.8576	0.8553	0.8908	0.8933	0.8930	0.8678
$\times 3.0$ gaussian l	0.6919	0.7111	0.7103	0.8303	0.8442	0.8187	0.8272
$\times 3.0$ lanczos s	0.7718	0.6444	0.6522	0.8494	0.8522	0.8128	0.8416
$\times 3.0$ lanczos m	0.7940	0.8170	0.8306	0.8848	0.8870	0.8759	0.8783
$\times 3.0$ lanczos l	0.6930	0.7226	0.7260	0.7410	0.7262	0.7448	0.7368
$\times 3.5$ bicubic s	0.7861	0.7909	0.7909	0.8475	0.8497	0.8338	0.8333
$\times 3.5$ bicubic m	0.7789	0.8718	0.8713	0.8629	0.8645	0.8006	0.8638
$\times 3.5$ bicubic l	0.7033	0.7378	0.7378	0.8362	0.8379	0.8300	0.8139
$\times 3.5$ gaussian s	0.7895	0.8363	0.8357	0.8556	0.8580	0.8466	0.8557
$\times 3.5$ gaussian m	0.7521	0.8286	0.8272	0.8646	0.8661	0.8609	0.8313
$\times 3.5$ gaussian l	0.6672	0.6864	0.6863	0.7997	0.8153	0.8013	0.8114
$\times 3.5$ lanczos s	0.7469	0.6112	0.6176	0.8191	0.8235	0.7763	0.8163
$\times 3.5$ lanczos m	0.7642	0.7900	0.8036	0.8560	0.8575	0.8387	0.8533
$\times 3.5$ lanczos l	0.6464	0.6757	0.6798	0.7122	0.7075	0.7148	0.7022
$\times 4.0$ bicubic s	0.7652	0.7679	0.7627	0.8232	0.8254	0.7986	0.7994
$\times 4.0$ bicubic m	0.7555	0.8482	0.8501	0.8379	0.8398	0.7922	0.7812
$\times 4.0$ bicubic l	0.6811	0.7138	0.7169	0.8052	0.8105	0.6908	0.6733
$\times 4.0$ gaussian s	0.7676	0.8120	0.8097	0.8316	0.8338	0.8122	0.8332
$\times 4.0$ gaussian m	0.7285	0.8031	0.8060	0.8382	0.8401	0.8264	0.7692
$\times 4.0$ gaussian l	0.6490	0.6671	0.6689	0.7730	0.7892	0.7618	0.6378
$\times 4.0$ lanczos s	0.7298	0.5873	0.5879	0.7880	0.7943	0.7425	0.7945
$\times 4.0$ lanczos m	0.7389	0.7664	0.7724	0.8296	0.8236	0.8018	0.8279
$\times 4.0$ lanczos l	0.6444	0.6735	0.6751	0.6912	0.6865	0.6952	0.6769
$P_{\times 2}$ bicubic s	0.9045	0.9370	0.9343	0.9515	0.9527	0.9196	0.9322
$P_{\times 2}$ bicubic m	0.8839	0.9509	0.9349	0.9548	0.9563	0.9480	0.9320
$P_{\times 2}$ bicubic l	0.8202	0.8471	0.8412	0.9267	0.9276	0.8888	0.8902
$P_{\times 2}$ gaussian s	0.8976	0.9518	0.9415	0.9537	0.9550	0.9420	0.9310
$P_{\times 2}$ gaussian m	0.8603	0.9152	0.9017	0.9538	0.9555	0.9399	0.9290
$P_{\times 2}$ gaussian l	0.7783	0.7942	0.7909	0.8996	0.9068	0.8391	0.8479
$P_{\times 2}$ lanczos s	0.9136	0.8479	0.8721	0.9433	0.9444	0.9345	0.9321
$P_{\times 2}$ lanczos m	0.9246	0.9218	0.9353	0.9491	0.9507	0.9436	0.9397
$P_{\times 2}$ lanczos l	0.8386	0.8398	0.8401	0.8413	0.8436	0.8264	0.8442
$P_{\times 3}$ bicubic s	0.8415	0.8548	0.8584	0.8971	0.8997	0.8985	0.8904
$P_{\times 3}$ bicubic m	0.8197	0.9054	0.8870	0.9035	0.9053	0.9024	0.8929
$P_{\times 3}$ bicubic l	0.7517	0.7782	0.7732	0.8738	0.8700	0.8088	0.8085
$P_{\times 3}$ gaussian s	0.8347	0.8866	0.8797	0.9007	0.9031	0.9020	0.8932
$P_{\times 3}$ gaussian m	0.7935	0.8603	0.8457	0.9031	0.9045	0.8930	0.8808
$P_{\times 3}$ gaussian l	0.7154	0.7295	0.7270	0.8384	0.8510	0.7509	0.7497
$P_{\times 3}$ lanczos s	0.8462	0.7011	0.7379	0.8816	0.8843	0.8623	0.8825
$P_{\times 3}$ lanczos m	0.8605	0.8526	0.8715	0.8961	0.8971	0.8854	0.8826
$P_{\times 3}$ lanczos l	0.7499	0.7433	0.7467	0.7612	0.7536	0.7229	0.7483

Table 4. Detailed numerical evaluation of our method. The metric that is shown is SSIM. All numbers were computed on 100 hard 512×512 crops from images in the DIV2K test set.

References

[1] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NIPS*. 2019. 1