KernelNet: A Blind Super-Resolution Kernel Estimation Network Supplemental Document

Mehmet Yamac Baran Ataman Aakif Nawaz
Huawei Technologies Oy (Finland) Co. Ltd
{mehmet.yamac, baran.ataman, aakif.nawaz}@huawei.com

1. Introduction

This document includes supplementary information, which will be published as a PDF linked to the primary article. This supplementary document presents the following information that would be beneficial for the readers:

1. The extended visual comparisons,
2. implementation details,
3. ablation studies:
   (a) the proposed SR kernel estimation algorithm's stress test, which is the robustness analysis against additive noise,
   (b) the stability analysis of the proposed modular neural network structure against different gradient estimation modules,
   (c) the experimental analysis that shows the importance of the extended kernel pool.

2. Kernel Estimation in Synthetic Data

Realistic SR kernel pool: The preparation of the realistic SR kernel pool is summarized in Section 5.1 of the primary article. Two sub-set from the Huawei P20 dataset [16] were arbitrarily partitioned. The first subset was used to estimate realistic SR kernels, and the second subset was used as the test set for blind SR. In order to create an SR kernel pool, 512 × 512 patches were extracted from the center of the images in the first sub-set. Then, from these patches, SR kernels were estimated by using both KernelGAN and model-based approach. In addition to these estimated kernels, anisotropic and isotropic Gaussian kernels were also generated and added to the SR kernel pool. The variances of Gaussian kernels were chosen in accordance with the estimated SR kernels.

For Scale 2, we had a total of 1368 training kernels from the model-based solution (600), KernelGAN estimation (280), different anisotropic (340), and isotropic Gaussian kernels (148) with standard deviations randomly drawn from the interval $\left[\sqrt{2} * 1.2, \sqrt{2} * 2\right]$. The test set of SR kernel pools is composed of 368 kernels with model-based estimations (148), KernelGAN estimations (64), different anisotropic (80), and isotropic Gaussian kernels (56) with standard deviations randomly drawn from the interval $\left[\sqrt{2} * 1.2, \sqrt{2} * 2\right]$.

For scale 3, instead of 512 × 512 patches, 510 × 510 patches were used as 510 is divisible to 3. Moreover, as KernelGAN is not optimized for odd scale factors like 3, the estimated kernels using KernelGAN for scale factor 3 were self-convolved to obtain approximate SR kernels for scale factor 3. In numbers, we had 749 SR kernels in the training set. From them, 215 kernels are estimated via model-based 347 are self-convolved $sf = 2$ KernelGAN estimations, 201 are anisotropic, and 76 are isotropic Gaussian with the standard deviation randomly drawn from the interval $\left[\sqrt{2} * 1.2, \sqrt{2} * 2\right]$. For the test set, we had 134 SR kernels composed of model-based estimations (30), KernelGAN estimations for Scale 2 were self-convolved (60), anisotropic (31), and isotropic Gaussian Kernels (13).

For scale 4, the training set of SR kernel pool consists of 638 SR kernels, including model-based estimations (47), KernelGAN estimations and KernelGAN estimations (for $sf = 2$) 2 times self-convolved (347), different anisotropic Gaussian kernels (149), and isotropic Gaussian kernels (100) with the standard deviation randomly drawn from the interval $[2.5, 3.5]$. The test kernel set has 129 SR kernels which are model-based estimations (12), KernelGAN estimations, and KernelGAN estimations (for $sf = 2$) 2 times self-convolved (63), different anisotropic Gaussian kernels (33), and isotropic Gaussian kernels (21).

Synthetic Data: Synthetic Dataset (training and test image sets) were produced using DIV2K training images and test images. These HR images were degraded using the SR degradation model with SR kernels from our SR kernel pools. The training procedure is explained in Section 4 of the primary article in detail.

For the synthetic test set, 100 1024 × 1024 patches are randomly extracted from the images in DIV2K dataset. Then, these patches were degraded by the SR degradation

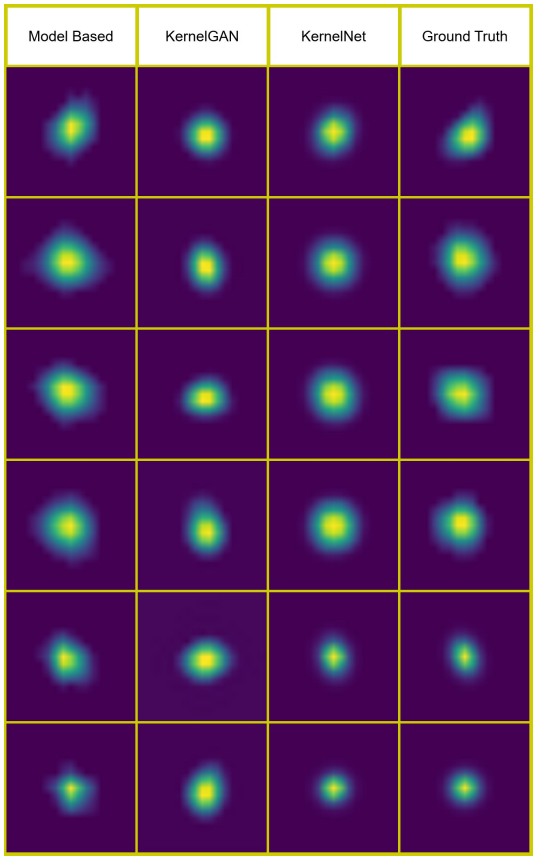

Figure 1. Visual comparison of estimated kernels by different SR kernel estimators for 6 different examples.

model in Equation 1 of the primary article. The SR kernels are randomly chosen from the SR kernel pool (test pool).

Comparison of SR Kernel Estimations: We added a visual comparison of estimated kernels by different SR kernel estimators, which was not presented in the primary article. The visual comparison is given in Figure 1.

More visual comparison on the synthetic dataset: We also give more visual examples from the outputs of competing blind SR methods for both scale factor 2 (Figure 2) and scale factor 3 (Figure 3).

3. Visual Comparison on Real Images

Compared to the primary article, we present more visual examples in Figure 4 and Figure 5 for Scale 2 and Scale 3. Moreover, for scale 4, state-of-the-art GAN-based blind SR algorithms are also included, and more visual examples from the real dataset are given in Figure 6.

4. Ablation Studies

4.1. Robustness to additive noise

In order to make a stress test of competing SR kernel estimators, we corrupted the observation (LR image), $\mathbf{y}$, with Additive White Gaussian Noise (AWGN) with standard deviations, 25, 50, and 75. The performance comparison is given in Table 4.1. From the results, we can claim that the proposed SR kernel estimation network, KernelNet, is performing similary with the other competing algorithms while its computational complexity is less. It remains state-of-the SR kernel estimators for realistic noise levels, e.g., standard variance in the interval 1-50.

4.2. Different Structures for Sharp Gradient Estimation Module

Inspired by [38], our Sharp Gradient Estimation Module consists of six convolutional layers. The five hidden layers have 128 neurons, and we use ReLu activation functions for these layers. For the output layer, Tanh is used. The kernel sizes are, 9×9, 1×1, 3×3, 5×5, 1×1 and 3×3, respectively. We chose this structure because it gives slightly better performance on SR kernel estimation problem com-

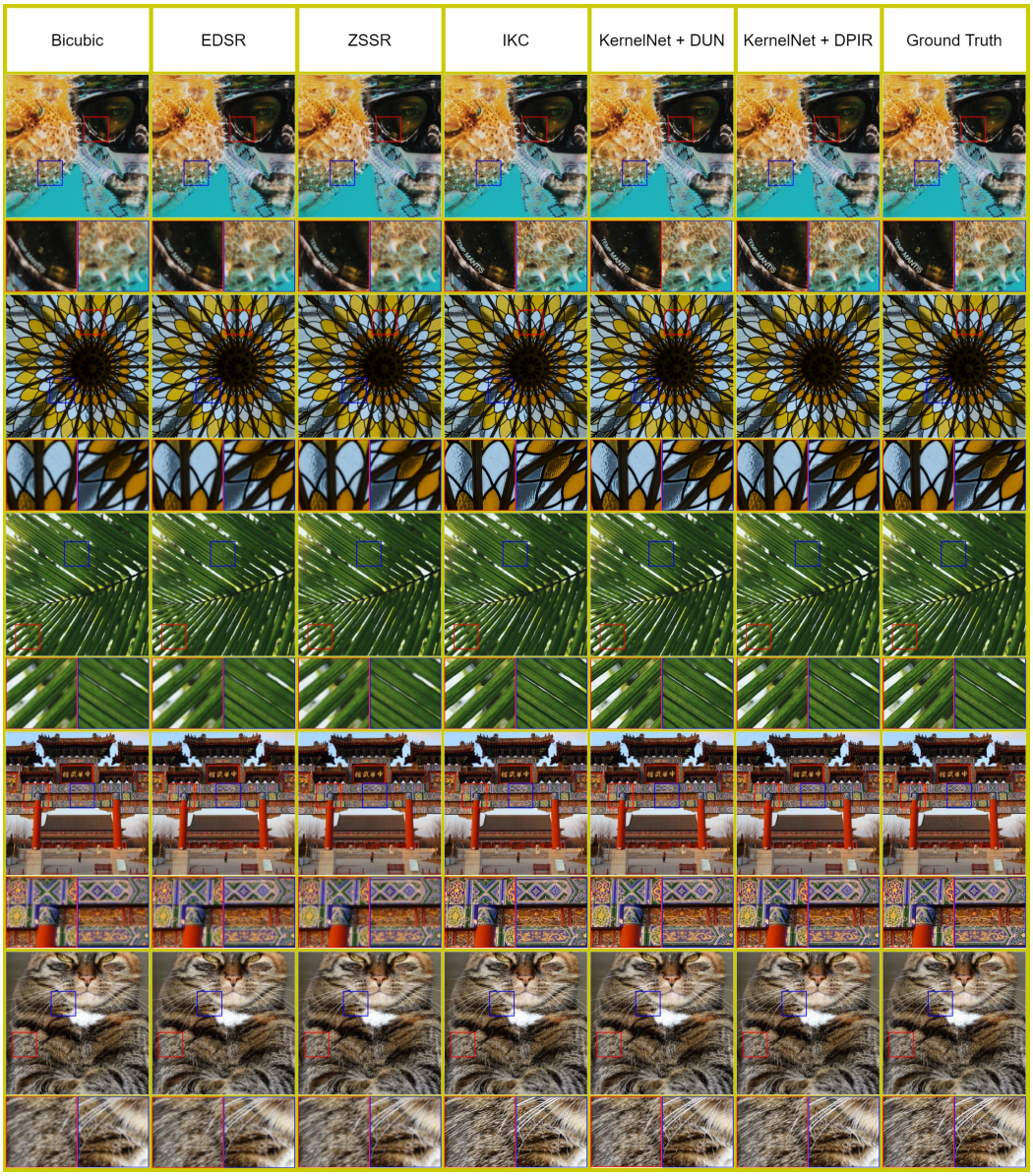

Figure 2. (Scale 2) Visual comparison of SR algorithms. Examples from Synthetic Test Data.

Table 1. The performance comparision in noisy enviroment

Noise Level	Methods	Kernel Estimation Error	PSNR	SSIM
25.0/255.0	Model Based	0.429	25.21	0.837
	KernelGAN	0.437	25.73	0.832
	KernelNet	0.461	26.37	0.838
50.0/255.0	Model Based	0.429	24.96	0.84
	KernelGAN	0.497	25.59	0.837
	KernelNet	0.458	26.75	0.849
75.0/255.0	Model Based	0.495	25.35	0.833
	KernelGAN	0.437	25.82	0.834
	KernelNet	0.457	26.37	

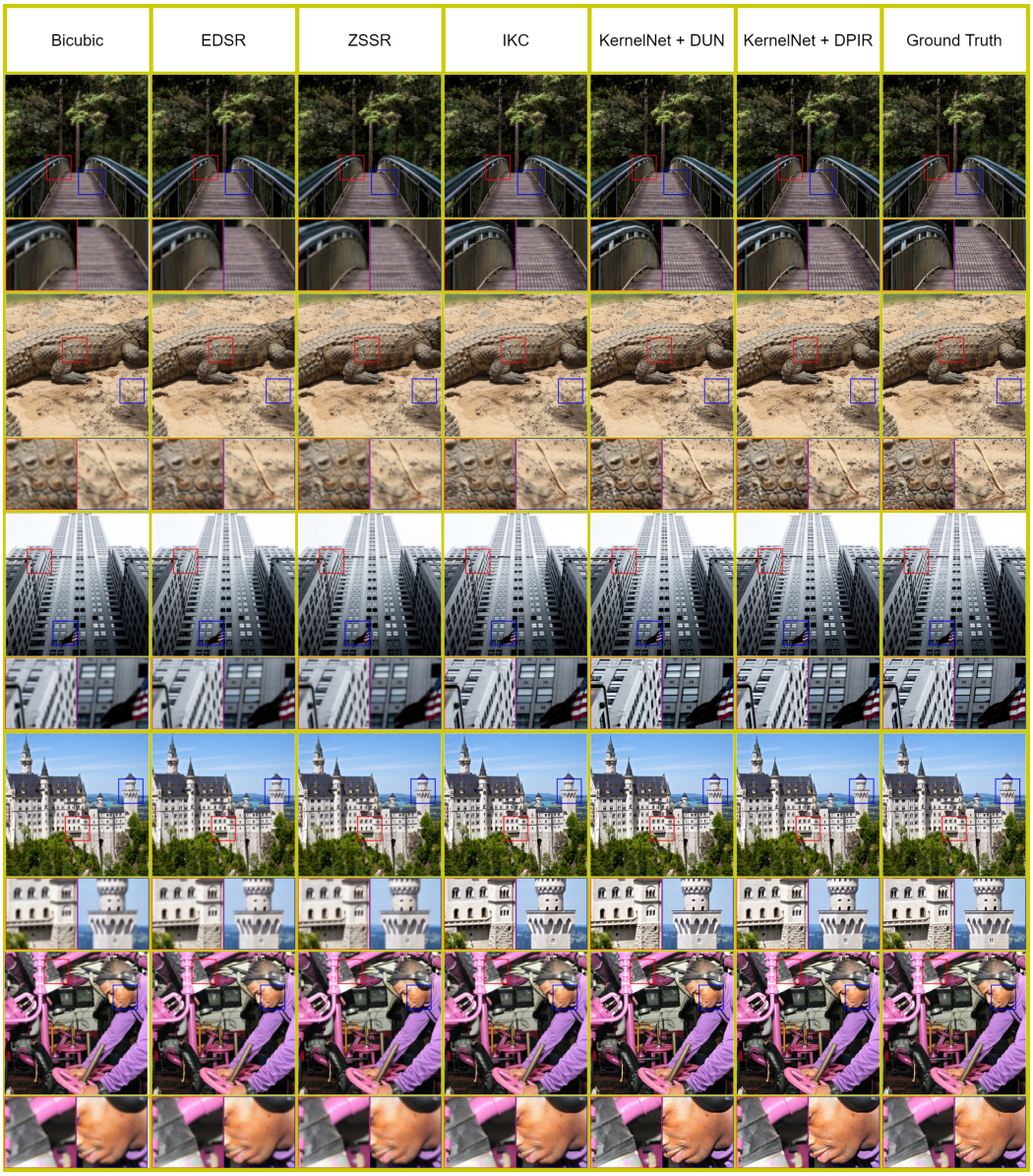

Figure 3. (Scale 3) Visual comparison of SR algorithms. Examples from Synthetic Test Data.

pared to alternative approaches.

Fortunately, our KernelNet does not heavily depend on the structure of the Sharp Gradient Estimation Module. Namely, a proper alternative can also be replaced with it. In order to show it, we designed the following experiments: We replace the current module with three alternative sharp gradient estimators. In [1], the authors proposed convolutional support estimator networks (CSEN1- CSEN2) that can estimate sharp image gradients from the corrupted images. Although their corruption is compressive sensing, which can be defined as a related type of inverse problem, the networks can be alternative chooses in our experimental setup. In addition, we also design a simple CNN structure, as we call it ordinary, a simple CNN mapping (OSC). OSC consists of 3 hidden layers. The layers have 48, 24, 24 neurons, and the ReLu activation function is used for hidden layers. For the output layer, Tanh is used. The kernel sizes are, 3×3, 2×2, 3×3, 3×3, respectively. After the first layer, a Max-pool layer with stride two was used.

The performance comparison of different KernelNet structures with different Sharp Gradient Estimation Modules is given in Table 2. The current module gives slightly better performance but the network still stably works with alternative modules.

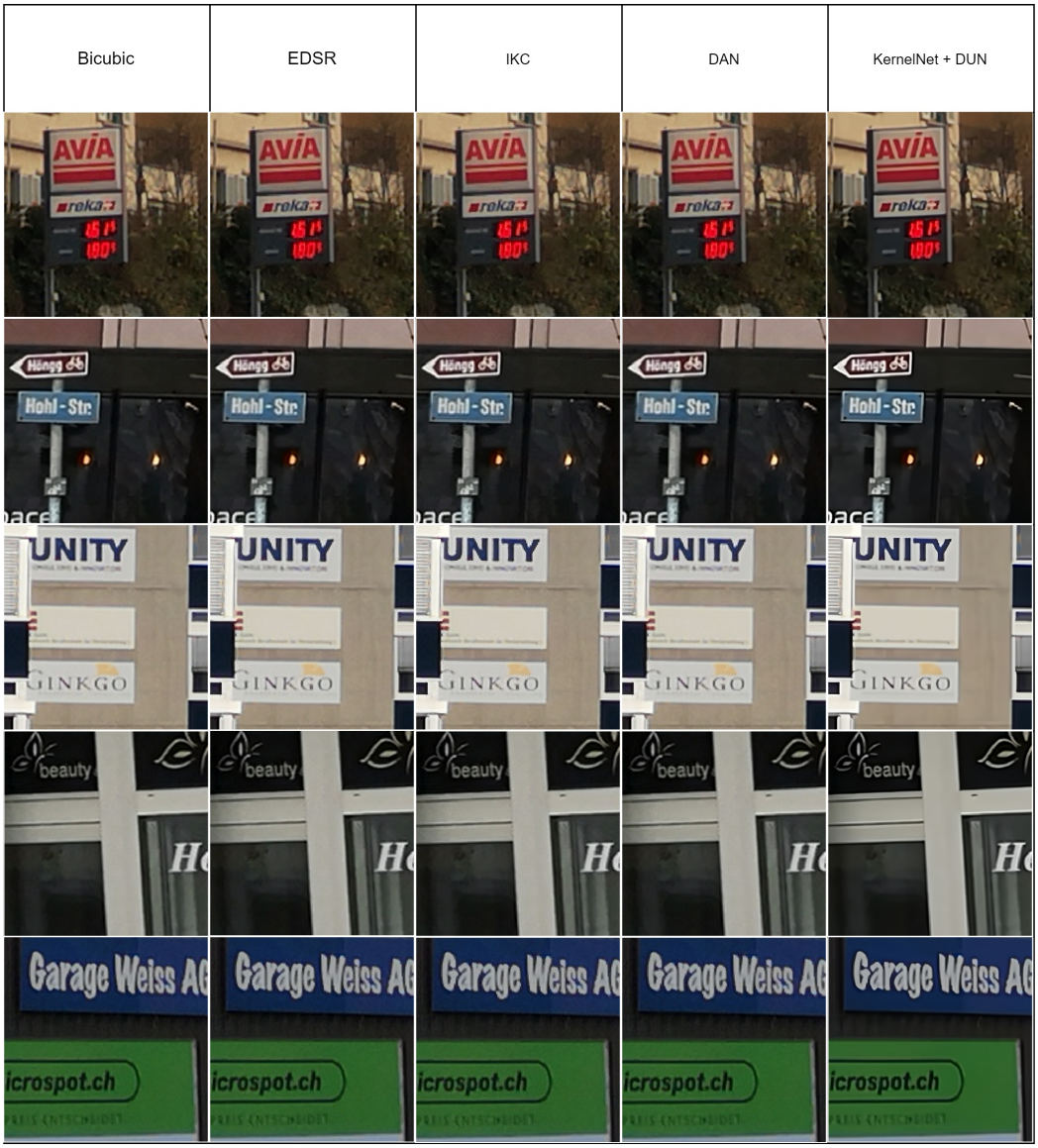

Figure 4. (Scale 2) Visual comparison of SR algorithms on real images. Examples from Huawei P20 Test Data.

Table 2. Different gradient estimation networks.

Networks	Kernel Error	PSNR	SSIM
Current Network	0.224	28.82	0.905
CSEN II	0.2476	28.68	0.901
OSC	0.2328	27.46	0.894
CSEN I	0.2357	27.82	0.896

4.3. Importance of Extended Kernel Pool

The recent works [17, 44] have proposed to estimate the realistic SR kernels from a specific camera dataset to create an SR kernel pool. Then, they used these kernel pools to degrade HR images in order to obtain a training set consisting of realistic LR-HR image pairs. In that way, they increased their CNN based SR solutions' generalization capabilities. From them, the study [44] used a model-based solution explained in Section 2a and [17] used KernelGAN.

As explained in Section 2c, the existing SR kernel estimation solutions have limited performances, which causes the kernel mismatch problem. In this study, we proposed to use an extended SR kernel pool. To illustrate the importance of the proposed solution for SR kernel pool generation, we designed the following experiments: To train KernelNet, we used only estimated kernels using KernelGAN as it is done in [17]. However, for the test SR kernel pool, we take the

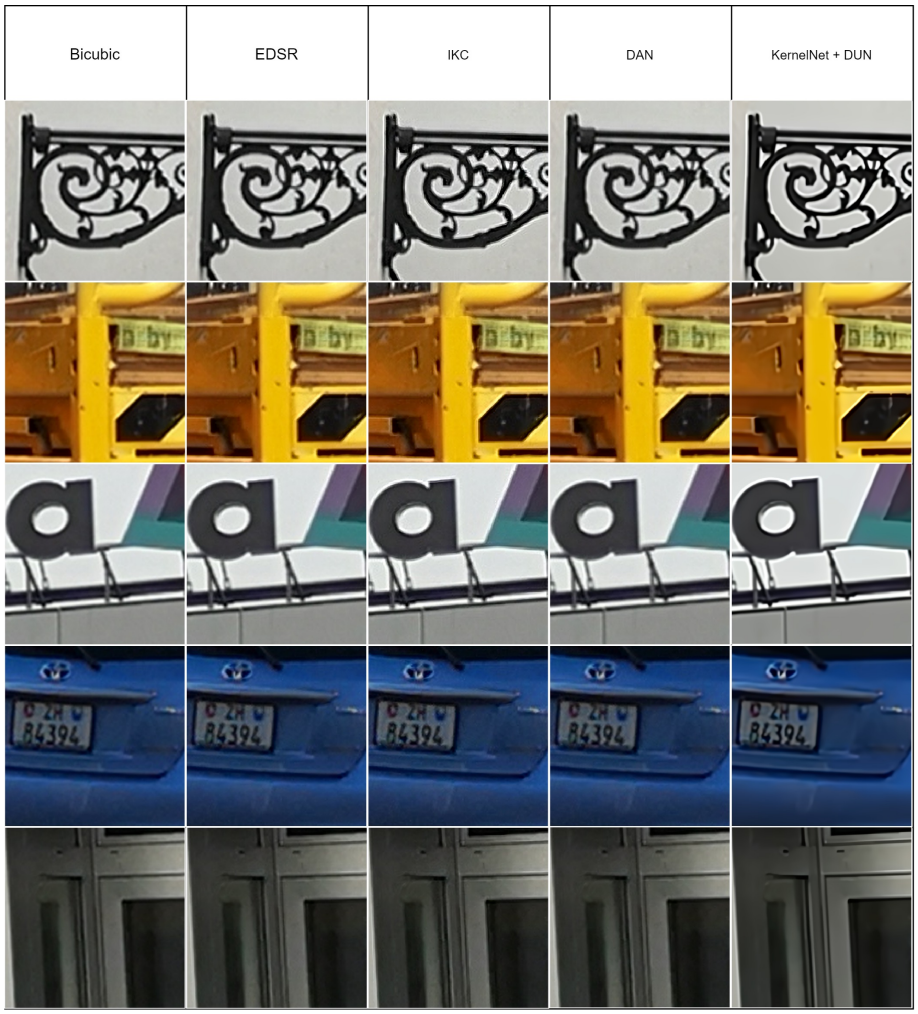

Figure 5. (Scale 3) visual comparison of SR algorithms on real images. Examples from Huawei P20 Test Data.

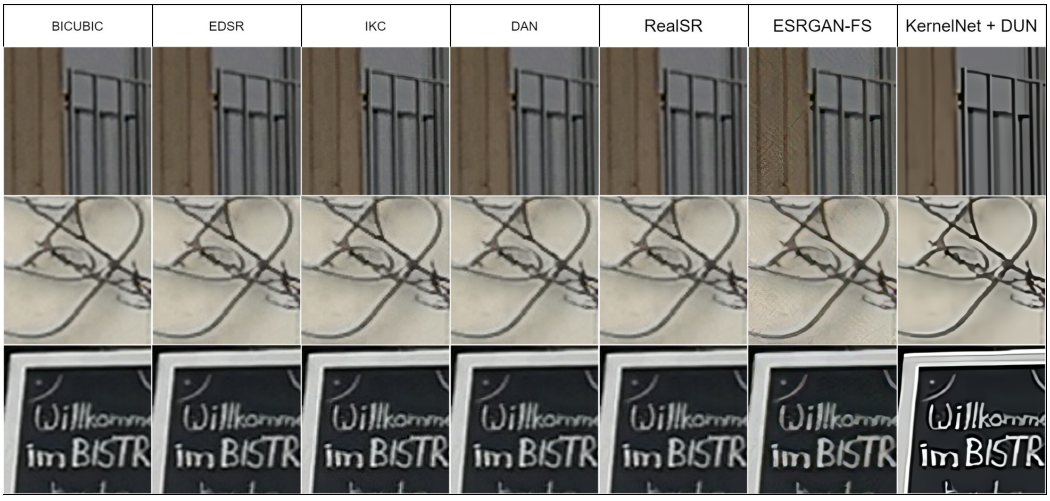

Figure 6. (Scale 4) visual comparison of SR algorithms on real images. Examples from Huawei P20 Test Data.

estimated kernels using the model-based solution. The performance drops from error 0.224 to 0.40. Although KernelNet still performs better than KernelGAN, the performance drops drastically. The reason is that the current SR kernel estimators are not accurate enough; they produce different kernel estimations that can even yield different types of kernel pools.

References

[1] Mehmet Yamac, Mete Ahishali, Serkan Kiranyaz, and Moncef Gabbouj. Convolutional sparse support estimator network (csen) from energy efficient support estimation to learning-aided compressive sensing. *arXiv preprint arXiv:2003.00768*, 2020.