A. Background: Kernel Mismatch

In real-world settings, the degradation process can often be complex, involving multiple stages of blurring, downsampling and noise addition. For cases in which this process is not known, super-resolution models are required to be robust to unknown degradations, *i.e.* they need to be able to upsample any natural image in the wild. To this end, Efrat *et al.*[9] first shed light on the kernel mismatch phenomenon: how the super-resolved image would be impacted if the estimated kernel differs from the ground-truth kernel, which is assumed to be Gaussian, regardless of the given prior. Specifically, their study demonstrated that a smoother kernel relative to its corresponding ground-truth kernel leads to sharper images while a sharper kernel leads to blurry images. Hence, accurately estimating the ground-truth kernel is crucial in order for the downstream non-blind SR model to produce visually pleasing super-resolved images.

B. MetaKernelGAN Details

Models. For our generator, we employ a deep linear architecture with six convolutional layers and without non-linear activations. The layers have kernel sizes of $[7, 3, 3, 1, 1, 1]$, a stride of 2 for the last layer and 1 for the earlier layers, representing the linear operation of applying a blur kernel of size 11×11 followed by the subsampling operation. For our discriminator, we adopt KernelGAN's architecture: a 7-layer discriminator with kernel sizes of $[7, 1, 1, 1, 1, 1, 1]$, each followed by a Spectral normalization [27], batch normalization [15], and ReLU activation, except for the last layer which consists of a sigmoid activation at the end.

Weighting the Interval Loss Optimization Steps. Following MZSR [32], we weight each interval loss optimization step equally at the beginning, then slowly decaying the preceding adaptation steps and converging the weight to the last adaptation step:

$$\mathbf{w} \leftarrow \text{GetIntervalLossWeights}(j) \qquad \text{where}$$
$$\mathbf{w}[0], \cdots, \mathbf{w}[N_{\text{val}} - 2] = \min\left(\frac{1}{N_{\text{val}}} - j \cdot \frac{3}{N_{\text{val}} \cdot 10000}, \frac{0.03}{N_{\text{val}}}\right)$$
$$\mathbf{w}[N_{\text{val}} - 1] = 1 - \sum_{m=0}^{N_{\text{val}}-2} w_m$$

where $j \in [1, N_{\text{steps}}]$ is the meta-objective step and N_{val} is the total number of interval loss evaluation for each task, as used in Alg. 1.

Inference Algorithm. To adapt the meta-learned GAN on a new LR image (Alg. 2), we initialize G and D with the meta-learned base parameters, $\hat{\theta}_G$ and $\hat{\theta}_D$ (lines 1-2). We then adapt these parameters using our task adaptation loss as per Eq. (5) for N_{adapt} steps (line 3) and the estimated kernel is derived from MetaKernelGAN's generator as per Eq. (3) (line 5).

Algorithm 2: Inference of MetaKernelGAN

Input: Input image I^{LR}
Number of steps N_{adapt}
Meta-learned $\hat{\theta}_G$ and $\hat{\theta}_D$
Output: Kernel estimate k^* for the given image

1 $\theta_G \leftarrow \hat{\theta}_G, \theta_D \leftarrow \hat{\theta}_D$ ▷ Initialize G and D with meta-learned parameters
2 **for** l in $[1, N_{\text{adapt}}]$ **do** ▷ Adaptation steps over the given image I^{LR}
3 Compute adapted parameters (Eq. (4)):
$\theta_G \leftarrow \theta_G - \alpha_G \nabla_{\theta_G} \mathcal{L}_G^{\text{task}}(\theta_D, \theta_G)$
$\theta_D \leftarrow \theta_D - \alpha_D \nabla_{\theta_D} \mathcal{L}_G^{\text{task}}(\theta_D, \theta_G)$
4 **end**
5 $\hat{k} \leftarrow \text{DK}(\theta_G)$ ▷ Derive kernel estimate

B.1. MetaKernelGAN Flow

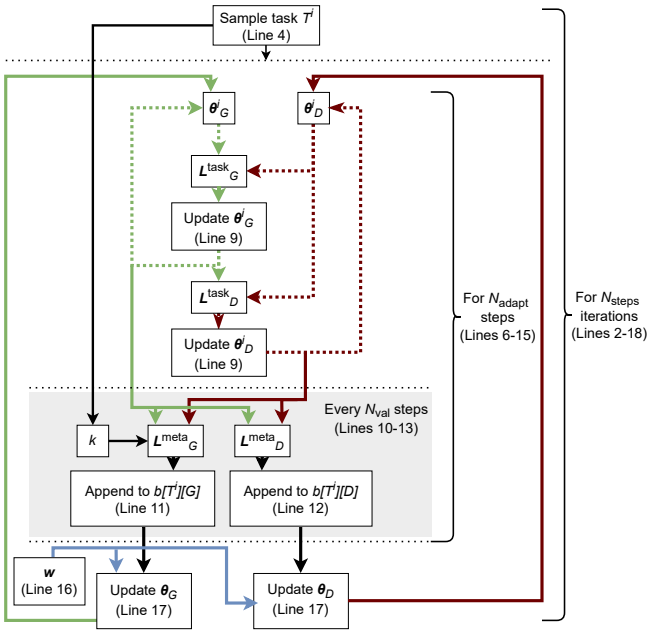

Figure 7. Flow chart of Algo. 1.

Fig. 7 shows the flow diagram of MetaKernelGAN meta-training stage shown in Algo. 1, which is described in Section. 3.1. Green & red lines represent updates to the generator the discriminator respectively, with dotted lines representing the task adaptation stage and solid lines representing the meta-optimization stage. The blue lines represent the weighting of the meta-objectives.

B.2. MetaKernelGAN Task Design

The task-related component of our framework comprise: *i)* the tasks $\mathcal{T}$, and *ii)* the task probability distribution $p(\mathcal{T})$ that determines the strategy of sampling a task from the candidate tasks, *i.e.* $\mathcal{T}^i \sim p(\mathcal{T})$.

Task $\mathcal{T}^i$. Each task $\mathcal{T}^i = \langle I^{\text{LR},i}, k \rangle$ consists of an LR image and a blur kernel k. The LR image is formed in three steps: *i)* an HR image, $I^{\text{HR},i}$, is first randomly sampled from the given dataset; *ii)* random augmentation op-

erations, such as flipping or rotation, are applied on the selected HR image; and, finally, *iii)* an LR image is obtained by applying kernel k on the HR image. Key to our task setup is that each task encapsulates both a *support* and a *query* set, by containing *multiple* patches with the *same* blur kernel. We denote the support/query set as $\mathcal{D}_s^i \cup \mathcal{D}_q^i = \{(p^{\text{LR}}, k)\} \quad \exists p^{\text{LR}} \in patches(I^{\text{LR},i})$, *i.e.* the support ($\mathcal{D}_s^i$) and query ($\mathcal{D}_q^i$) sets consist of different patches from the *same* LR image. With this formulation, the meta-learning method can learn from multiple patches that have been degraded with the same kernel, exploiting in this manner the internal patch recurrence of the given image. The complete set of tasks $\mathcal{T}$ is defined using a dataset of HR images and a distribution of blur kernels.

Task Probability Distribution $p(\mathcal{T})$. The distribution $p(\mathcal{T})$ controls our task *sampling strategy* and encompasses both the blur kernel and the HR image selection. First, the kernel k is chosen by *uniformly* sampling from a predefined distribution, *e.g.* an anisotropic Gaussion distribution. Similarly, the HR image is *uniformly* sampled from the given training dataset. After the sampling process, the final task is constructed by applying k to an augmented version of the HR image to produce its LR counterpart, $I^{\text{LR},i}$.

Patch Sampling Strategy. To evaluate the $\mathcal{L}^{\text{LSGAN}}$ term used by both the task adaptation loss and the meta-objective of our method, patches are sampled from a task's LR image. To sample an image patch, $p^{\text{LR},i}$, we follow a similar strategy as KernelGAN: we assign a selection probability to each patch in $I^{\text{LR},i}$ based on its gradient magnitude. In this manner, patches with higher gradient magnitude have a higher probability to be selected. The rationale behind this strategy is that using flat patches, *i.e.* with low gradient magnitude, would aggravate the ill-posedness of the kernel estimation problem, leading to a typical isotropic Gaussian kernel [21]. Specifically, KernelGAN utilizes the gradient magnitude of $I^{\text{LR},i}$ and its bicubic upsampled super-resolved image to determine each patch's selection probability for the discriminator and generator respectively. Unlike KernelGAN, we utilize the gradient magnitude of $I^{\text{LR},i}$ for the generator instead and take the top-left sub-patch for the discriminator. This empirically results in a slight boost in performance possibly because the discriminator can learn to discriminate between two patches from the same region in the image, as opposed to two randomly sampled regions as implemented in KernelGAN.

C. Evaluation & Reproducibility Details

Hyperparameter Details. We use a task batch size of 1. We divide α_G by 10 after 50 & 200 adaptation steps during inference. For α_G and α_D, we tried values [0.1, 0.2, 0.5, 0.01, 0.02, 0.05] and picked the ones with the highest performance: $\alpha_G = 0.01, \alpha_D = 0.2$.

Implementation Details. MetaKernelGAN was built on top of PyTorch v1.7 and *learn2learn* [3], an open-source meta-learning framework which we extended to support MetaKernelGAN's components and algorithms. We further integrated part of the code of FKP [22], MZSR [32], KernelGAN [5], and USRNet [39].

Baseline Details. Following KernelGAN-FKP, we replace one autoencoder in Double-DIP with a fully-connected network to model the kernel. All prior work results are reported using the associated codebases[4].

Subpixel Alignments & Kernel Shifts. We shift the evaluation set of LR images by shifting its blur kernel following [39] when evaluating all explicit kernel estimation approaches, including MetaKernelGAN. However, the assumed center of mass of the kernel is slightly different in previous implicit degradation estimation works. Hence, to avoid subpixel misalignments in these cases, we regenerate the evaluation set of LR images by shifting its kernel following [5] when evaluating IKC, DAN, and DASR.

Difference in Performance for DIP-FKP. The original FKP work uses a *different* kernel distribution to evaluate DIP-FKP and KernelGAN-FKP. For fair comparison, we use the same distribution when evaluating all previous explicit kernel estimation works. Specifically, we adopt the kernel distribution originally proposed for KernelGAN-FKP.

Covariance of Estimated Kernel. We derive the discretized kernel $\hat{k}$ from Eq. (2) as an $m \times m$ matrix and calculate the covariance matrix as $\hat{\Sigma} = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$ where $a = \text{Var}(col_{\hat{k}})$, $b = \text{Var}(rows_{\hat{k}})$ and $c = \text{Covar}(col_{\hat{k}}, rows_{\hat{k}})$.

D. Ablation

Meta-learning the Generator. Fig. 8 shows the case where we only meta-learn the generator and not the discriminator. As the discriminator is trained from scratch for each image during evaluation, the meta-trained generator dominates the training, leading to inadequate generator feedback from the discriminator and in turn to inferior kernel accuracy.

Number of Iteration Steps. We meta-trained MetaKernelGAN with N_{adapt}=10, 25, 50 and showed the Kernel PSNR and $\mathcal{L}^{\text{K-COV}}$ for $\times 2$ upsampling in Table. R1. We observe that the kernel has not converged when N_{adapt}=10, resulting in worse kernel performance. N_{adapt}=50 also leads to worse performance, possibly attributed to the first-order approximation of FOMAML which is not suited for long inner-loop trajectories.

[4] https://github.com/JingyunLiang/FKP/
https://github.com/greatlog/DAN
https://github.com/yuanjunchai/IKC
https://github.com/ShuhangGu/DASR
https://github.com/zsyOAOA/BSRDM

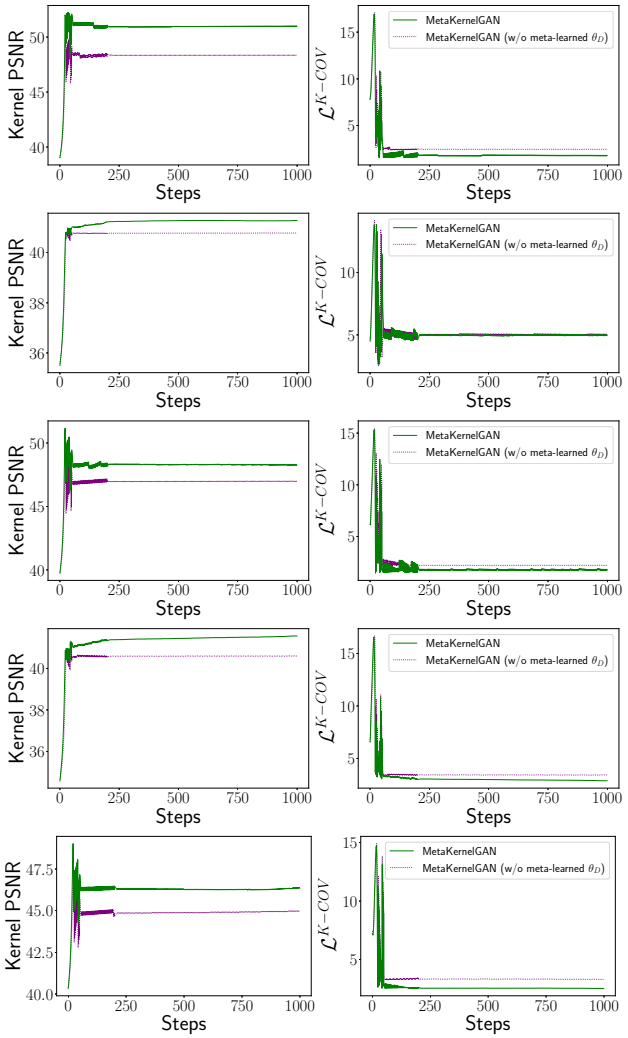

Figure 8. Intermediate kernel results for every adaptation iteration ($\times 2$ upsampling for *0802.png*, *0833.png*, *0838.png*, *0857.png*, and *0853.png* in DIV2K).

Method	N_{adapt}	Set14	B100	Urban100	DIV2K
MetaKernelGAN	10	44.72/2.56	43.42/2.99	45.99/2.39	46.88/2.31
MetaKernelGAN	25	**46.23/2.44**	**45.94/2.38**	**47.37/2.16**	**48.00/2.08**
MetaKernelGAN	50	45.49/2.49	45.41/2.65	46.47/2.42	46.81/2.37

Table R1. Average Kernel PSNR/$\mathcal{L}^{\text{K-COV}}$ on SR benchmarks across five runs for different N_{adapt} for $\times 2$ upsampling.

E. MetaKernelGAN Adaptability to Images with Limited Internal Information.

LR images are severely limited in internal information when their resolution is small, resulting in the fallback on the learned kernel. This is often the case in *i)* $\times 4$ upsampling, where the LR images are tiny, and *ii)* especially in smaller resolution datasets. To quantify this, we compare the estimated discretized covariance matrix of the adapted kernel after 200 steps, $\hat{\Sigma}^{\text{Est200}}$, to that of the initial learned kernel, $\hat{\Sigma}^{\text{Est0}}$, and that of the ground-truth kernel, $\hat{\Sigma}^{\text{GT}}$. Specifically, we compute $\mathcal{L}^T = \max(\mathcal{L}^{\text{K-COV}}(\hat{\Sigma}^{\text{Est200}}, \hat{\Sigma}^{\text{GT}}) - \mathcal{L}^{\text{K-COV}}(\hat{\Sigma}^{\text{Est200}}, \hat{\Sigma}^{\text{Est0}}), 0)$, where $\mathcal{L}^{\text{K-COV}}(a, b) = \sum_{x,y}^{N} \left|\hat{\Sigma}^{\text{a}}_{x,y} - \hat{\Sigma}^{\text{b}}_{x,y}\right|$, in three equal-sized datasets B100, Urban100, and DIV2K, representing small, medium, and large image resolutions, respectively. By definition, the higher $\mathcal{L}^T$ is, the closer the adapted kernel is to the initial kernel relative to its distance from the GT kernel and the higher the fallback rate. For $\times 2$ upsampling, the mean $\mathcal{L}^T$ is 0.0 for all three datasets, indicating no fallback. For $\times 4$, the mean $\mathcal{L}^T$ for B100, Urban100, and DIV2K is 4.98, 3.78, and 3.25, respectively, indicating that lower-resolution images lead to more frequent fallback.

F. Additional Qualitative Results

Fig. 9 & Fig. 10 show the comparisons of both the kernel and image among the explicit kernel estimation methods on our benchmark evaluation datasets for $\times 2$ and $\times 4$ upsampling respectively. Fig. 12 show more examples highighting that DSKernelGAN is more susceptible to adapt to faulty kernels than MetaKernelGAN as the former doesn't learn from the adaptation process. Lastly, we show more real-world results among both implicit and explicit degradation methods from images downloaded from the Internet in Fig. 13.

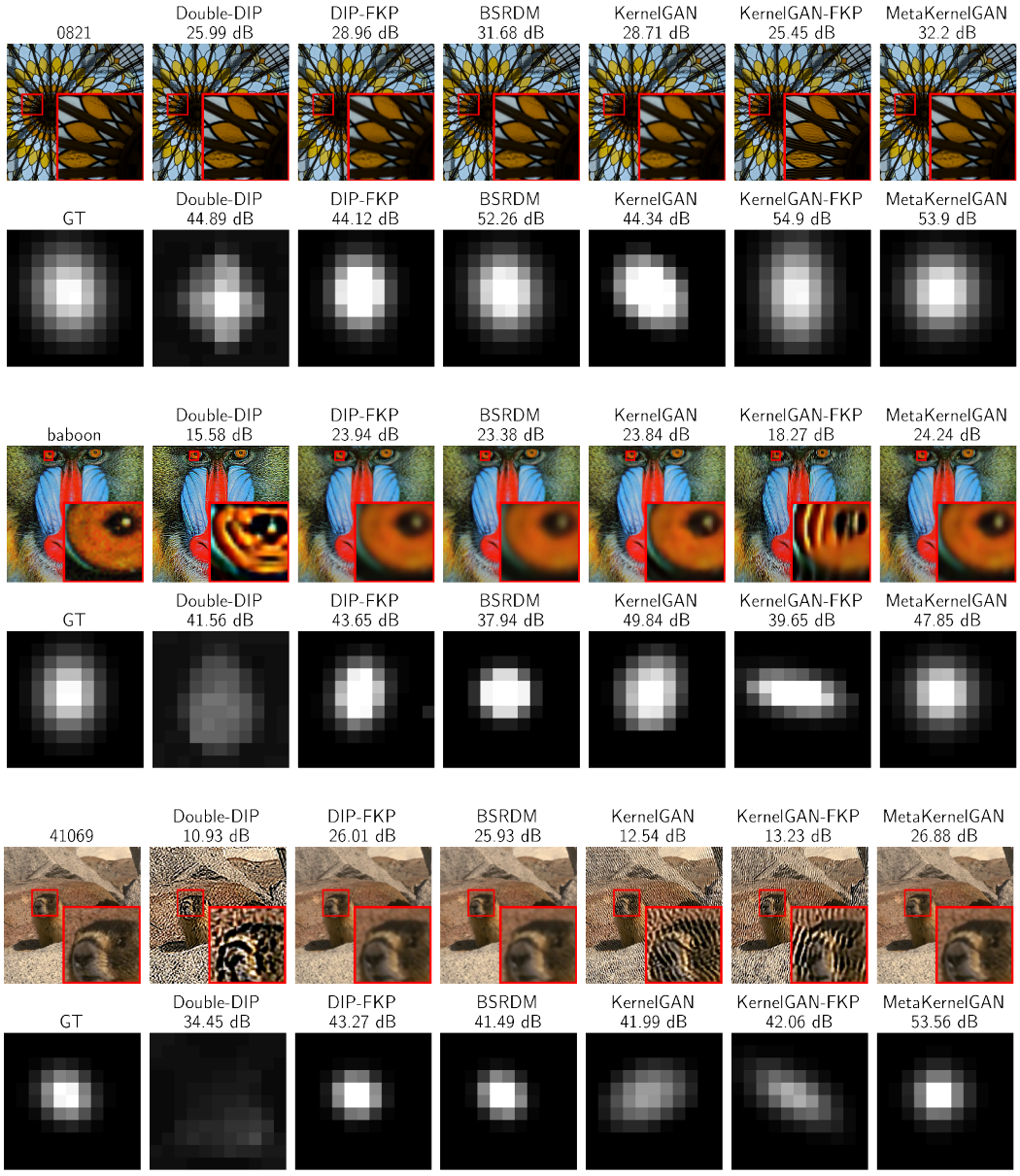

Figure 9. Comparison of estimated kernel, along with its image $\times 2$ upsampled using USRNet [39], among explicit kernel estimation methods across different benchmark datasets. Zoom in for best results.

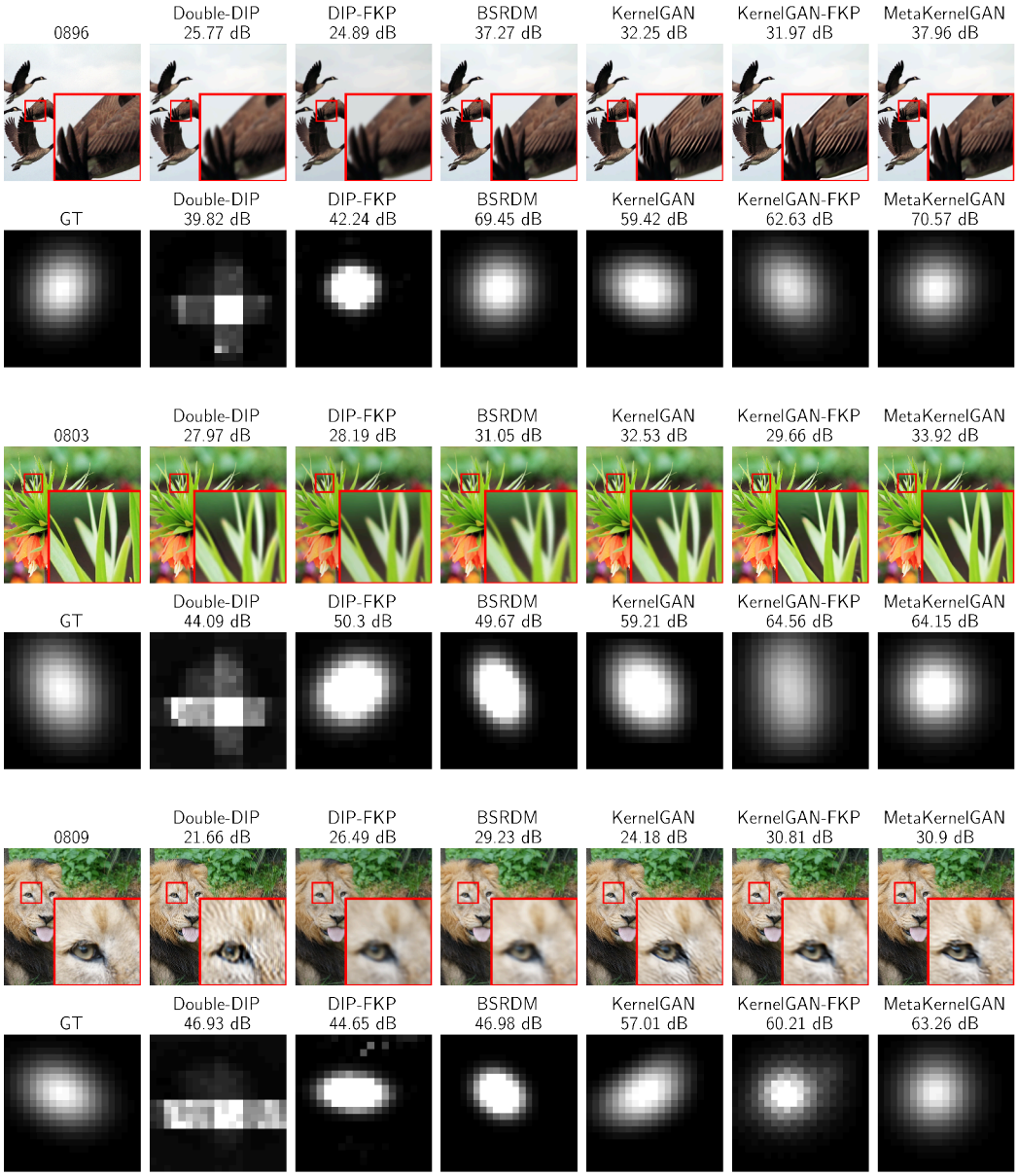

Figure 10. Comparison of estimated kernel, along with its image $\times 4$ upsampled using USRNet [39], among explicit kernel estimation methods across different benchmark datasets. Zoom in for best results. Part 1 of 2.

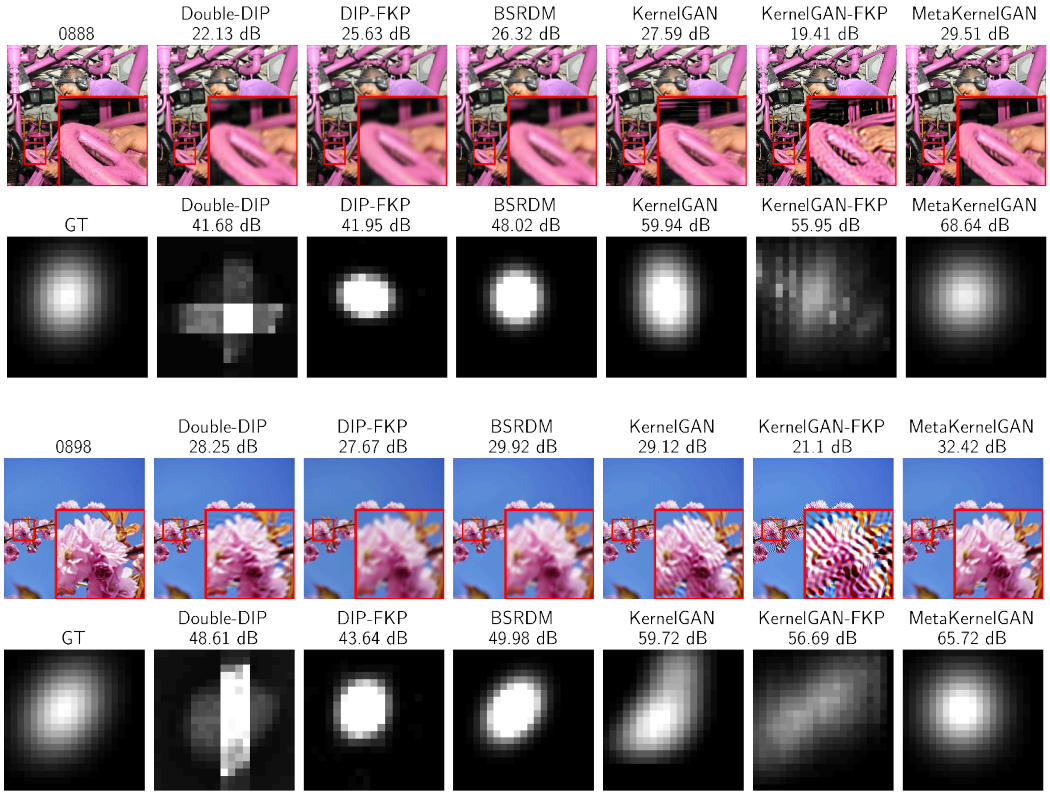

Figure 10. Comparison of estimated kernel, along with its image $\times 4$ upsampled using USRNet [39], among explicit kernel estimation methods across different benchmark datasets. Zoom in for best results. Part 2 of 2.

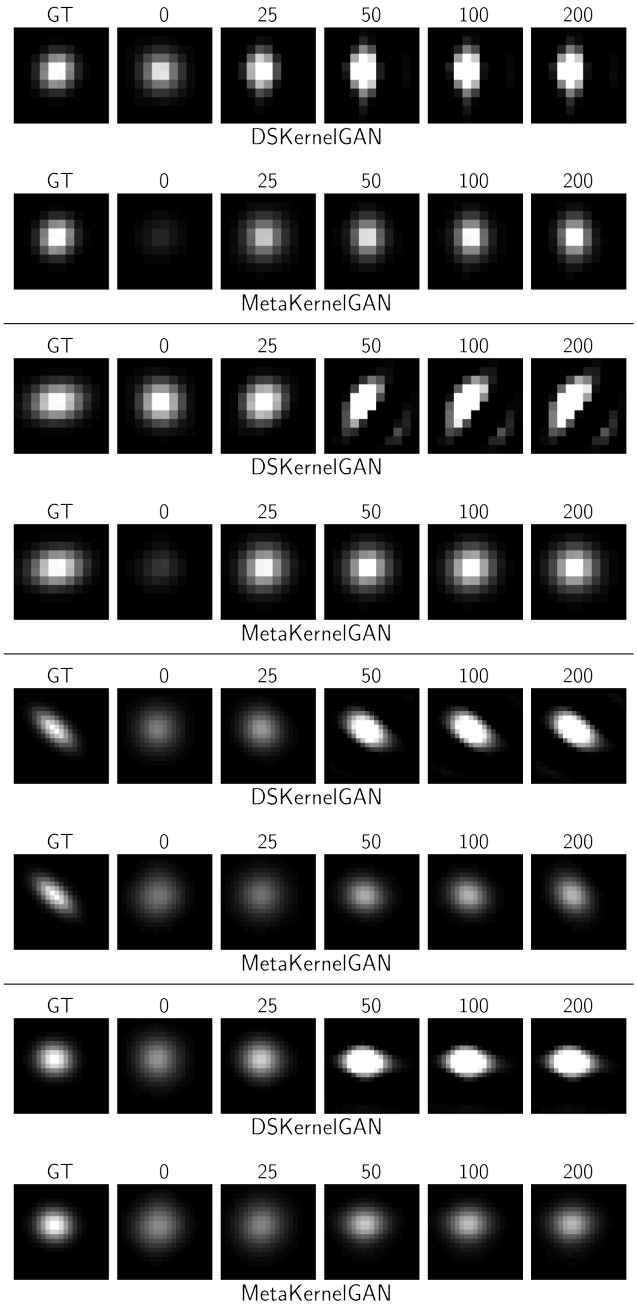

Figure 11. Kernel after 25, 50, 100, 200 adaptation steps for $\times 2$ upsampling on *86000* of B100, *img036* of Urban100, *0821* and *0824* of DIV2K (top to bottom).

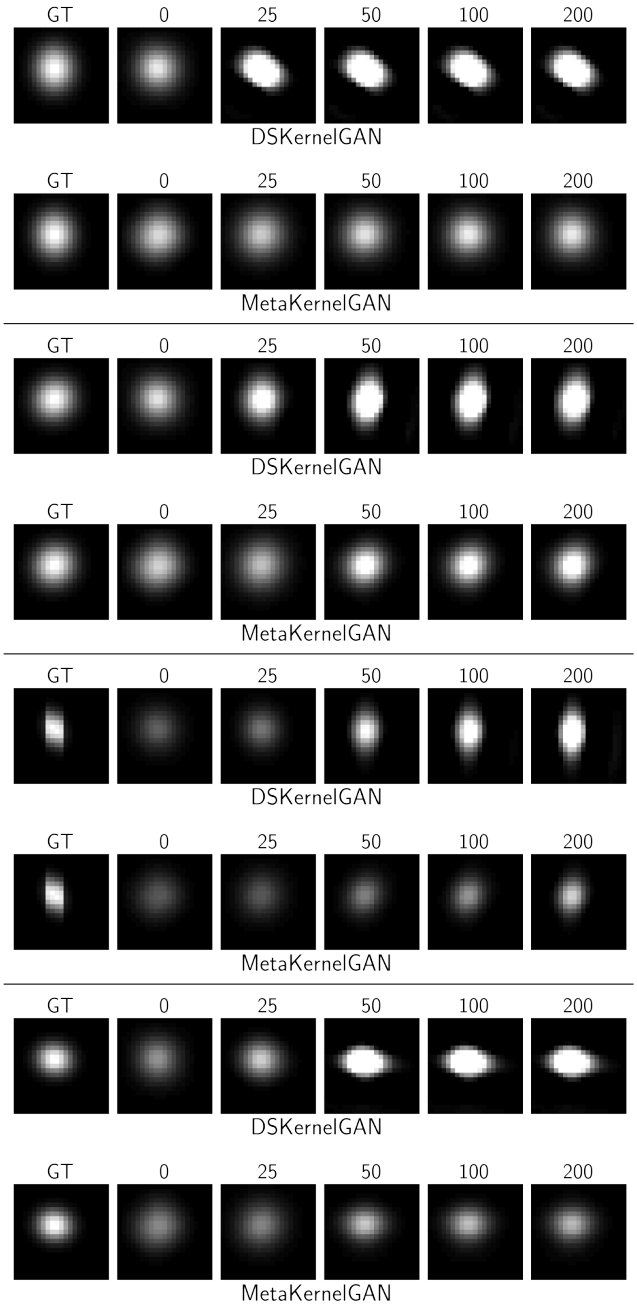

Figure 12. Kernel after 25, 50, 100, 200 adaptation steps for $\times 4$ upsampling on *monarch* and *ppt3* of Set14, and *0879* and *0824* of DIV2K (top to bottom).

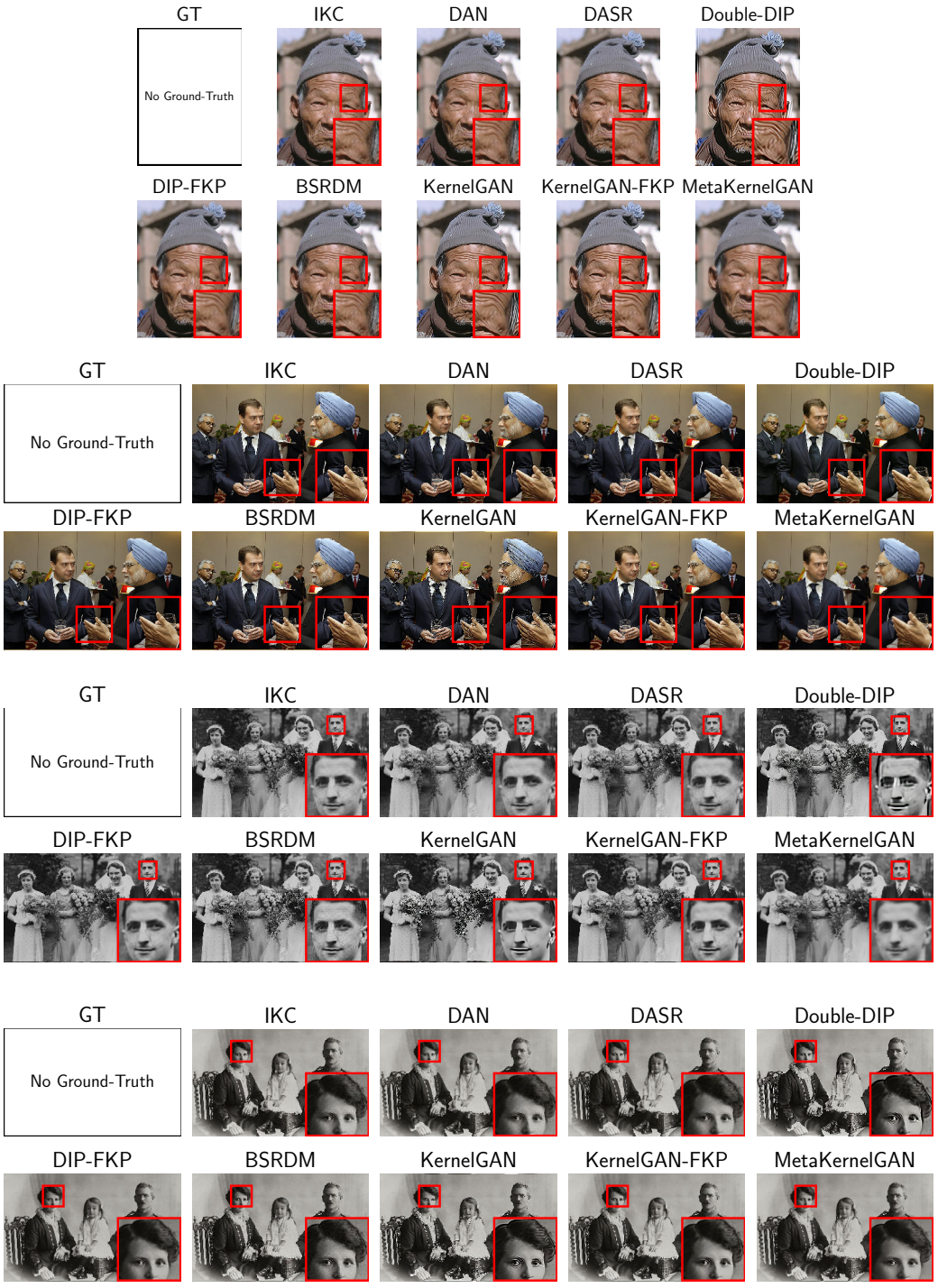

Figure 13. Real-world visual quality comparison on $\times 4$ upsampling among models. Zoom in for best results. No ground truth is available.