

Meta-Learned Kernel For Blind Super-Resolution Kernel Estimation

Royson Lee^{1,2*}, Rui Li², Stylianos Venieris²
 Timothy Hospedales^{2,3}, Ferenc Huszár¹, Nicholas D. Lane^{1,4}

¹ University of Cambridge, UK ² Samsung AI Center, Cambridge, UK
³ University of Edinburgh, UK ⁴ Flower Labs

Abstract

Recent image degradation estimation methods have enabled single-image super-resolution (SR) approaches to better upsample real-world images. Among these methods, explicit kernel estimation approaches have demonstrated unprecedented performance at handling unknown degradations. Nonetheless, a number of limitations constrain their efficacy when used by downstream SR models. Specifically, this family of methods yields i) excessive inference time due to long per-image adaptation times and ii) inferior image fidelity due to kernel mismatch. In this work, we introduce a learning-to-learn approach that meta-learns from the information contained in a distribution of images, thereby enabling significantly faster adaptation to new images with substantially improved performance in both kernel estimation and image fidelity. Specifically, we meta-train a kernel-generating GAN, named *MetaKernelGAN*, on a range of tasks, such that when a new image is presented, the generator starts from an informed kernel estimate and the discriminator starts with a strong capability to distinguish between patch distributions. Compared with state-of-the-art methods, our experiments show that *MetaKernelGAN* better estimates the magnitude and covariance of the kernel, leading to state-of-the-art blind SR results within a similar computational regime when combined with a non-blind SR model. Through supervised learning of an unsupervised learner, our method maintains the generalizability of the unsupervised learner, improves the optimization stability of kernel estimation, and hence image adaptation, and leads to a faster inference with a speedup between 14.24 to 102.1 \times over existing methods.⁰

1. Introduction

Single-image super-resolution (SR) is a low-level vision task that entails the upsampling of a low-resolution (LR) image to high resolution (HR). Despite the significant

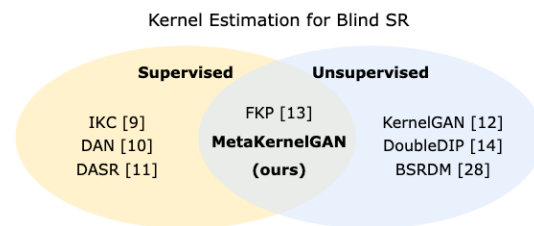


Figure 1. Landscape of kernel estimation methods for blind SR. Supervised methods are computationally efficient, but underperform when faced with out-of-distribution kernels on real-world images. Unsupervised methods can better adapt to diverse kernels, but are undeployable due to their excessive computational overhead. Our *MetaKernelGAN* method blends together both approaches for the first time and introduces the essential techniques to obtain the respective benefits.

progress of deep learning-based SR in recent years, the majority of existing approaches [1, 7, 8, 18, 19, 20, 30, 33] assume a *fixed* degradation process with a known blur kernel, such as bicubic interpolation. This fact often leads to poor performance in real-world cases, due to the variability of the degradation process across images, and puts a strong *retraining* requirement for every new degradation operation.

In this context, an alternative line of SR methods has emerged that aims to improve the performance under unknown degradations. These methods can be taxonomized into non-blind and blind. Non-blind SR leverages a *provided* degradation process in order to maximize the upsampling quality for the given image. Blind SR, on the other hand, *estimates* the degradation process.

Estimating the degradation process can be done either implicitly or explicitly. Implicit methods typically adopt a *supervised learning* approach where the degradation representation is learnt jointly with an SR model on a given dataset [12, 24, 36] (Fig. 1 - left). In such schemes, the features of the degradation process of the given training set are implicitly built into the SR model, yielding high performance on test images with the same degradation. From a computational aspect, as no adaptation is performed at in-

*Corresponding Author: dsr12@cam.ac.uk

⁰Code is available at <https://github.com/royson/metakernelgan>

ference time, these methods only require a forward pass and hence are fast. Nonetheless, the main limitation of these methods is that their performance deteriorates significantly when the test degradation differs from the train distribution. This property makes them unsuitable for the variably degraded images in the wild and renders their applicability narrow to settings with well-known degradation processes.

In contrast, explicit degradation estimation methods [5, 22] learn a degradation process¹ for each given image at inference time. These methods either jointly optimize the super-resolved image together with the kernel or feed both the kernel and the given image to a downstream non-blind SR model in a two-step process. The degradation estimation is performed through *unsupervised learning* and requires a number of training iterations to effectively learn the image-specific degradation process (Fig. 1 - right). By adapting on each image, these methods not only lead to better performance on unseen degradation distributions, but also provide a better interpretation of the kernel in the pixel space through their explicit degradation modeling. Nonetheless, the additional computations per image and their unsupervised nature make these methods slow and unstable, often requiring thousands of training steps for each image to achieve adequate performance.

To counteract the limitations of these unsupervised methods, existing approaches constrain the set of possible kernels solutions by either pretraining the model to start with an initial Gaussian-like kernel [22], jointly optimizing both the image and kernel by minimizing the LR image reconstruction loss [11], explicitly optimizing the precision matrix of the Gaussian kernel [38], or heavily enforcing the use of regularization, such as encouraging the kernel to be a bicubic kernel at the start [5]. Although effective, the aforementioned limitations are not sufficiently mitigated, especially in cases where the given image does not contain sufficient information, *e.g.* noisy and/or tiny images.

In this work, we aim to tackle the same set of challenges of unsupervised kernel estimation through supervised learning while retaining the benefits of unsupervised learning. In other words, our method should only utilize the internal information of the given image during inference to better handle unseen degradations but fall back on a learned kernel that works well on average across a dataset of images in cases where this internal information is limited (see Fig. 1 - center). To this end, we propose MetaKernelGAN, a gradient-based meta-learning approach that learns from the unsupervised kernel estimation learner. Concretely, we build upon KernelGAN [5] as our unsupervised learner and aim to meta-learn the initialization of both the generator and discriminator so that it can rapidly and accurately estimate the kernel for each image. Through extensive experiments, we demonstrate the benefits, limitations, and suitability of

our approach, compared with existing kernel estimation and blind SR approaches on both standard datasets and real-world images. We also provide a detailed analysis of the kernel estimation process, highlighting the advantages and limitations of meta-learning in explicit kernel estimation. Most importantly, we show that this blending of supervised and unsupervised learning holds considerable potential in utilizing external images for unsupervised degradation estimation works, directly counteracting the fundamental limitations of these per-image methods.

2. Related Work

Explicit Kernel Estimation. Recent kernel estimation approaches have been dominated by the use of deep learning. KernelGAN [5] exploits the internal patch recurrence of the image and derives a kernel that preserves the distribution of the image across scales. Double-DIP [11, 29] jointly optimizes the HR image and the blur kernel by minimizing the LR image reconstruction loss. These explicit kernel estimation works learn the kernel in the pixel space using neural networks, starting from a random initialization for each image. Hence, they often require multiple regularization terms to restrict the possible set of solutions, resulting in unstable and slow convergence. To counteract this, FKP [22] constrains the kernel space to an anisotropic Gaussian distribution by mapping a simpler latent distribution to a Gaussian distribution through normalizing flows, which is pretrained in a supervised manner. Optimizing directly on this latent space traverses a learned kernel manifold that improves accuracy when applied to existing kernel estimation methods, such as KernelGAN and Double-DIP. Lastly, BSRDM [38] explicitly learns the precision matrix of the Gaussian kernel and models the noise variance. Instead of learning a restricted space of Gaussian kernel solutions, our approach meta-learns “how to adapt” across multiple unsupervised adaptation steps using a diverse distribution of images.

Meta-Learning. Meta-learning methods aim to extract transferable knowledge from a distribution of tasks so as to solve new tasks more data efficiently. This area is now too wide to review exhaustively here, and we refer the reader to the survey [13]. Gradient-based meta-learners such as MAML [10] are a popular approach that aim to learn a neural network initialization that can be well adapted with a small amount of data and few gradient steps. In the context of zero-shot super-resolution (ZSSR [31]), a few studies such as MZSR [32] and MLSR [28] have applied such meta-learning to learn initializations of SR networks that can be rapidly adapted, accelerating inference compared to prior ZSSR. However, these works are non-blind; they assume the degradation process is known and hence complementary to our work. With respect to GANs, a few studies [6, 23] have tried with limited success to meta-learn GAN initializations for data-efficient adaptation of genera-

¹Generally the blur kernel is assumed to be anisotropic Gaussian.

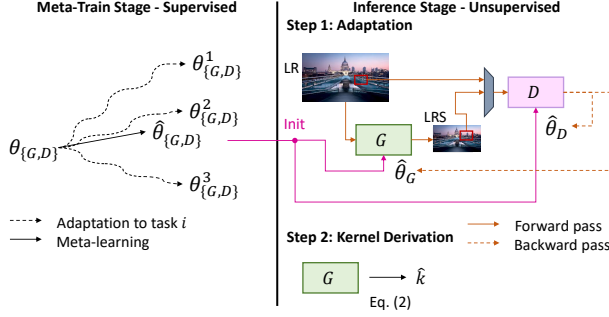


Figure 2. Overview of MetaKernelGAN.

tive models to new categories or domains. In this study, we focus on kernel estimation for blind SR, and for the first time use meta-learning to both accelerate and improve the accuracy of kernel estimation, and hence downstream non-blind super-resolution. Uniquely, we meta-learn the initialization of a kernel-generating GAN from an external distribution of images, so that it can be easily adapted to each testing image with the limited information available from its internal patch recurrence.

3. Meta-Learning for Kernel Estimation

Preliminaries. The degradation process of image super-resolution is commonly expressed as:

$$I^{\text{LR}} = (I^{\text{HR}} * k) \downarrow_s + n \quad (1)$$

where I^{LR} and I^{HR} are the LR and HR image respectively, k is the blur kernel, \downarrow_s is the subsampling operation with scaling factor s , and n is the additive noise. Blind SR aims to estimate both the blur kernel and HR image and the objective is formulated as follows:

$$\hat{I}^{\text{HR}}, \hat{k} = \arg \min_{I^{\text{HR}}, k} \|(I^{\text{HR}} * k) \downarrow_s - I^{\text{LR}}\|_p + \phi(I^{\text{HR}}) + \Phi(k) \quad (2)$$

where $\|(I^{\text{HR}} * k) \downarrow_s - I^{\text{LR}}\|_p$ is the fidelity term, $\phi(I^{\text{HR}})$ is the image prior, and $\Phi(k)$ is the kernel prior. Image priors have been well-investigated and are often implicitly represented through convolutional neural networks (CNNs) as they 1) constitute structurally strong natural image priors [35] and 2) are highly effective at learning priors from a large set of LR-HR pairs [7]. In contrast, kernel priors have only been recently studied through the use of fully-connected layers [29], a deep linear convolutional network [5], or normalizing flows [22], all of which solely rely on the internal information within a given image. In this work, we focus on estimating the kernel and use an existing non-blind SR work for upsampling.

KernelGAN for Internal Learning. A prominent approach to estimate the kernel from an LR image is by exploiting its internal patch recurrence. Specifically, KernelGAN [5] achieves this by employing a deep linear generator as a prior

Algorithm 1: Meta-Training of MetaKernelGAN

Input: Distribution over tasks $p(\mathcal{T})$
Number of steps $N_{\text{steps}}, N_{\text{adapt}}, N_{\text{val}}$
Step size hyperparameters $\alpha_G, \alpha_D, \beta_G, \beta_D$
Loss weighting hyperparameters ω, η, ζ

Output: Meta-learned $\hat{\theta}_G$ and $\hat{\theta}_D$

```

1 Initialize  $\theta_G$  and  $\theta_D$ 
2 for  $j$  in  $[1, N_{\text{steps}}]$  do ▷ Meta-optimization steps
3    $\theta_G^i \leftarrow \theta_G, \theta_D^i \leftarrow \theta_D$ 
4   Sample a batch of tasks  $\mathcal{T}^i = \langle I^{\text{LR},i}, k \rangle \sim p(\mathcal{T})$ 
5   Interval loss dictionary  $b \leftarrow \{\}$ 
6   for all  $\mathcal{T}^i$  do
7      $b[\mathcal{T}^i][G] \leftarrow [\cdot], b[\mathcal{T}^i][D] \leftarrow [\cdot]$ 
8     for  $l$  in  $[1, N_{\text{adapt}}]$  do ▷ Adaptation steps over task  $i$ 
9       Compute adapted parameters (Eq. (5)):
10       $\theta_G^i \leftarrow \theta_G^i - \alpha_G \nabla_{\theta_G} \mathcal{L}_G^{\text{task}}(\theta_D^i, \theta_G^i)$ 
11       $\theta_D^i \leftarrow \theta_D^i - \alpha_D \nabla_{\theta_D} \mathcal{L}_D^{\text{task}}(\theta_D^i, \theta_G^i)$ 
12      if  $(l \bmod N_{\text{val}}) = 0$  then ▷ Interval loss evaluation
13         $b[\mathcal{T}^i][G].\text{append}(\mathcal{L}_G^{\text{meta}}(\theta_D^i, \theta_G^i, k))$ 
14         $b[\mathcal{T}^i][D].\text{append}(\mathcal{L}_D^{\text{meta}}(\theta_D^i, \theta_G^i))$ 
15      end
16    end
17  end
18   $\mathbf{w} \leftarrow \text{GetIntervalLossWeights}(j)$ 
19  Update  $\theta_G$  and  $\theta_D$ :
20    $\theta_G \leftarrow \theta_G - \beta_G \nabla_{\theta_G} \sum_{\mathcal{T}^i \sim p(\mathcal{T})} (b[\mathcal{T}^i][G] \odot \mathbf{w})$ 
21    $\theta_D \leftarrow \theta_D - \beta_D \nabla_{\theta_D} \sum_{\mathcal{T}^i \sim p(\mathcal{T})} (b[\mathcal{T}^i][D] \odot \mathbf{w})$ 
22 end
23  $\hat{\theta}_G \leftarrow \theta_G, \hat{\theta}_D \leftarrow \theta_D$ 

```

to learn the linear degradation process (Eq. (1)). First, the generator, G , takes in a patch from the given LR image and downsamples it to obtain the resulting LR son (LRS) patch. Random patches are sampled from the LR image, *i.e.* $p^{\text{LR}} \sim \text{patches}(I^{\text{LR}})$, and the corresponding LRS patches are generated by the generator G , *i.e.* $G(p^{\text{LR}}; \theta_G)$. These are then fed into the discriminator, D , which distinguishes between the two patch distributions. After training, the estimated kernel is derived from the generator as follows:

$$\hat{k} = \text{DK}(\theta_G) = \hat{G}(\mathbf{J}_{\lceil \frac{m}{2} \rceil \lfloor \frac{m}{2} \rfloor}; \theta_G) \quad (3)$$

where $\hat{k} \in [0, 1]^{m \times m}$ is the estimated kernel, \hat{G} is G with its convolutional strides set to 1, and $\mathbf{J}_{\lceil \frac{m}{2} \rceil \lfloor \frac{m}{2} \rfloor} \in \{0, 1\}^{(2m-1) \times (2m-1)}$ is a single-entry matrix with all zeros except for a value of 1 at its central element.

As KernelGAN directly estimates the kernel in the pixel space, it employs multiple regularization terms to restrict the number of possible solutions to a Gaussian-like kernel such as being similar to the bicubic kernel at the start, incentivizing the optimal estimated kernel, k^* , to sum to one, discouraging boundary pixels, and encouraging sparsity and centrality. The need to retrain both the generator and discriminator from scratch and tune the hyperparameters of these regularization terms per image constitutes two of the main limitations of KernelGAN, resulting in unstable results and excessively long inference times.

3.1. MetaKernelGAN

Our proposed approach, MetaKernelGAN, aims to counteract the drawbacks of KernelGAN to effectively improve

| Method | Steps | Set4 | B100 | Urban100 | DIV2K |
|---|-------|-------------------------|-------------------------|-------------------------|-------------------------|
| ×2 | | | | | |
| Bicubic | - | 24.820/6910/- | 25.170/6603/- | 22.310/6471/- | 26.940/7642/- |
| KernelGAN [5] + USRNet | 3000 | 25.110/7404/3.37/2.2 | 24.810/7081/3.95/0.0 | 23.160/6759/4.11/3.10 | 27.560/8157/4.27/2.82 |
| KernelGAN-FKP [22] + USRNet | 1000 | 22.110/6224/5.25/9.70 | 23.800/6628/4.41/6.61 | 22.670/7099/4.93/4.77 | 28.830/8583/4.94/3.09 |
| DSKernelGAN (our baseline) + USRNet | 200 | 28.340/8210/42.17/4.15 | 28.060/7928/41.26/4.64 | 28.990/8046/42.84/3.74 | 31.070/8732/43.54/3.53 |
| MetaKernelGAN (ours) + USRNet | 200 | 28.710/8331/46.23/2.44 | 28.950/8222/45.94/2.38 | 28.790/8355/47.37/2.16 | 31.670/8944/48.00/2.08 |
| GT + USRNet (upper bound) | - | 32.560/8844/- | 31.530/8717/- | 29.460/8547/- | 34.560/9266/- |
| ×2 with Non-Gaussian Kernel | | | | | |
| Bicubic | - | 24.800/6907/- | 25.170/6609/- | 22.310/6476/- | 26.930/7640/- |
| KernelGAN [5] + USRNet | 3000 | 24.280/7084/1.75/3.16 | 23.840/6780/4.25/0.94 | 22.120/6404/2.86/2.96 | 26.520/7937/4.13/2.83 |
| KernelGAN-FKP [22] + USRNet | 1000 | 20.910/6034/2.35/13.08 | 22.750/6447/2.54/6.31 | 21.540/6742/42.85/4.77 | 27.400/8359/44.75/3.08 |
| DSKernelGAN (our baseline) + USRNet | 200 | 28.000/8067/40.87/4.03 | 27.740/7805/39.94/4.83 | 25.510/7906/41.48/3.68 | 30.740/8672/42.03/3.56 |
| MetaKernelGAN (ours) + USRNet | 200 | 28.650/8153/43.68/2.36 | 28.200/8031/43.69/2.32 | 25.840/8122/44.72/2.12 | 30.960/8830/44.96/2.08 |
| GT + USRNet (upper bound) | - | 32.570/8967/- | 31.540/8803/- | 30.060/8699/- | 34.560/9272/- |
| ×2 with Image Noise of Level 10 (3.92%) | | | | | |
| Bicubic | - | 24.690/6639/- | 24.980/6321/- | 22.200/6162/- | 26.650/7273/- |
| KernelGAN [5] + USRNet | 3000 | 27.280/7671/43.15/3.29 | 27.100/7333/43.69/3.22 | 25.080/6790/44.34/3.10 | 28.500/8286/44.66/2.91 |
| KernelGAN-FKP [22] + USRNet | 1000 | 23.230/7090/4.46/13.40 | 24.870/7188/43.72/7.89 | 23.800/7437/44.26/5.04 | 28.570/8127/44.10/3.68 |
| DSKernelGAN (our baseline) + USRNet | 200 | 27.570/7619/41.27/4.57 | 27.100/7143/39.44/5.55 | 25.150/7807/41.03/4.55 | 29.740/8154/44.11/4.28 |
| MetaKernelGAN (ours) + USRNet | 200 | 28.230/7783/43.99/2.76 | 27.750/7584/43.07/2.98 | 25.860/7780/44.82/2.6 | 30.250/8311/44.91/2.64 |
| GT + USRNet (upper bound) | - | 29.570/7949/- | 28.560/7607/- | 27.170/8114/- | 31.270/8497/- |
| ×4 | | | | | |
| Bicubic | - | 21.150/5280/- | 22.060/5119/- | 19.300/4761/- | 23.200/6329/- |
| KernelGAN [5] + USRNet | 3000 | Not supported | Not supported | Not supported | Not supported |
| KernelGAN-FKP [22] + USRNet | 4000 | Not supported | Not supported | Not supported | Not supported |
| DSKernelGAN (our baseline) + USRNet | 200 | 24.610/6725/58.77/12.97 | 24.300/6127/56.91/18.14 | 21.990/6134/57.34/14.89 | 26.930/7448/58.74/14.10 |
| MetaKernelGAN (ours) + USRNet | 200 | 25.460/6966/63.13/6.54 | 24.360/6200/59.84/10.84 | 22.210/6375/61.74/9.91 | 26.990/7608/61.78/8.34 |
| GT + USRNet (upper bound) | - | 27.800/7498/- | 26.920/6868/- | 24.850/7537/- | 29.460/8668/- |

Table 1. Average Image PSNR/Image SSIM/Kernel PSNR/ $\mathcal{L}^{\text{K-COV}}$ of patch recurrence methods on SR benchmarks across five runs.

its utility and enable fast and stable kernel estimation. Specifically, we adopt a variant of KernelGAN and meta-train the parameters of *both* G and D using a set of diverse images and kernels (Fig. 2). During inference, our approach is fully unsupervised and is similar to KernelGAN. Unlike KernelGAN, G is initialized with a meta-learned parameter that can adapt rapidly to an accurate kernel and D is initialized such that it can keep up to prevent the generator from dominating. Hence, MetaKernelGAN does not rely on the heavy use of regularization and require fewer hyperparameters, resulting in a more stable adaptation process and enabling significantly fewer adaptation steps. A flow diagram of Algo. 1 is provided in the Appendix.

Meta-Training. As presented in Alg. 1, the meta-training process consists of two main loops: the outer loop over meta-optimization steps (line 2) and the inner loop over tasks (line 6) that constitutes the task adaptation stage.

At each outer loop iteration, a batch of new tasks is sampled from the task distribution (line 4). Each task corresponds to our modified KernelGAN process which consists of an LR image and a kernel k , $\mathcal{T}^i = \langle I^{\text{LR},i}, k \rangle$. More details on the sampling process for \mathcal{T}^i can be found in the Appendix. For each task in the batch, we update the task-specific parameters (line 9), θ_G^i and θ_D^i , for N_{adapt} steps, using the following task adaptation loss, which comprises two terms: the L1-norm variant of LSGAN loss [25] (Eq. (4)) and a sum-to-one term $\mathcal{L}^{\text{STO}}(\hat{k}^i) = |1 - \sum_{x,y} \hat{k}_{x,y}^i|$ where \hat{k}^i is the estimated kernel for the i -th task.

$$\begin{aligned} \mathcal{L}_G^{\text{LSGAN}}(\theta_D, \theta_G) &= \mathbb{E}_{p^{\text{LR}} \sim \text{patches}(I^{\text{LR}})} |D(G(p^{\text{LR}}; \theta_G); \theta_D) - 1| \\ \mathcal{L}_D^{\text{LSGAN}}(\theta_D, \theta_G) &= \mathbb{E}_{p^{\text{LR}} \sim \text{patches}(I^{\text{LR}})} \left[\frac{1}{2} |D(p^{\text{LR}}; \theta_D) - 1| \right. \\ &\quad \left. + \frac{1}{2} |D(G(p^{\text{LR}}; \theta_G); \theta_D)| \right] \end{aligned} \quad (4)$$

The \mathcal{L}^{STO} term encourages the estimated kernel to sum to one, resulting in a Gaussian-like kernel at every adaptation step². Finally, we compute the adapted parameters using the

²As the kernel is not the output but rather derived from θ_G (Eq. (3)), the use of a softmax layer would not suffice.

| Method | Steps | Set4 | B100 | Urban100 | DIV2K |
|---|-------|-------------------------|-------------------------|----------------------------|-------------------------|
| ×2 | | | | | |
| Bicubic | - | 24.820/6910/- | 25.170/6603/- | 22.310/6471/- | 26.940/7642/- |
| Double-DIP [11] + USRNet | 1000 | 21.840/5984/1.34/5.96 | 18.520/4475/9.99/8.06 | 19.860/5827/9.95/9.53 | 24.330/7069/7.62/5.06 |
| DIP-FKP [22] + USRNet | 1000 | 28.900/8324/45.32/2.95 | 28.640/8224/46.72/2.95 | 26.000/8064/42.59/3.53 | 30.310/8722/43.19/3.20 |
| BSRDM [38] + USRNet | 440 | 29.380/8240/43.77/2.84 | 28.650/7936/40.76/3.21 | 26.200/8080/44.24/2.70 | 30.680/8694/46.00/2.57 |
| MetaKernelGAN (ours) + USRNet | 200 | 28.710/8331/46.23/2.44 | 28.950/8222/45.94/2.38 | 28.790/8355/47.37/2.16 | 31.670/8944/48.00/2.08 |
| GT + USRNet (upper bound) | - | 32.560/8844/- | 31.530/8717/- | 29.460/8547/- | 34.560/9266/- |
| ×2 with Non-Gaussian Kernel | | | | | |
| Bicubic | - | 24.800/6907/- | 25.170/6609/- | 22.310/6476/- | 26.930/7640/- |
| Double-DIP [11] + USRNet | 1000 | 21.480/5871/40.69/5.99 | 18.280/4367/9.34/8.16 | 19.660/5748/7.44/5.96 | 24.120/6995/7.14/5.10 |
| DIP-FKP [22] + USRNet | 1000 | 28.180/8153/43.30/2.96 | 27.880/8002/43.26/2.87 | 25.510/7888/41.23/3.50 | 29.990/8646/41.63/3.30 |
| BSRDM [38] + USRNet | 440 | 29.010/8155/41.24/2.85 | 28.410/7879/39.77/3.13 | 25.230/7875/42.47/2.70 | 29.900/8613/43.16/2.58 |
| MetaKernelGAN (ours) + USRNet | 200 | 28.050/8153/43.68/2.36 | 28.200/8031/43.69/2.32 | 25.840/8122/44.72/2.12 | 30.960/8830/44.96/2.08 |
| GT + USRNet (upper bound) | - | 32.470/8968/- | 31.340/8803/- | 30.080/8699/- | 34.560/9272/- |
| ×2 with Image Noise of Level 10 (3.92%) | | | | | |
| Bicubic | - | 24.690/6639/- | 24.980/6321/- | 22.200/6162/- | 26.650/7273/- |
| Double-DIP [11] + USRNet | 1000 | 24.320/7023/41.29/5.97 | 21.960/6010/9.59/18.25 | 20.530/6174/7.57/6.65/3.73 | 25.460/7547/57.43/5.03 |
| DIP-FKP [22] + USRNet | 1000 | 28.360/7724/44.70/2.95 | 27.640/7401/42.22/3.11 | 25.470/7644/42.49/3.58 | 29.490/8259/43.10/3.31 |
| BSRDM [38] + USRNet | 440 | 28.370/7716/43.23/3.06 | 27.620/7386/41.15/3.18 | 25.850/7724/45.13/2.59 | 30.280/8325/46.47/2.25 |
| MetaKernelGAN (ours) + USRNet | 200 | 28.230/7773/43.99/2.76 | 27.750/7584/43.07/2.98 | 25.860/7780/44.80/2.60 | 30.250/8311/44.91/2.64 |
| GT + USRNet (upper bound) | - | 29.570/7949/- | 28.560/7607/- | 27.170/8114/- | 31.270/8497/- |
| ×4 | | | | | |
| Bicubic | - | 21.150/5280/- | 22.060/5119/- | 19.300/4761/- | 23.200/6329/- |
| Double-DIP [11] + USRNet | 1000 | 20.480/5099/53.13/16.65 | 18.770/4028/52.26/22.92 | 17.860/4375/50.20/18.05 | 21.480/5704/46.33/15.30 |
| DIP-FKP [22] + USRNet | 1000 | 24.980/6707/54.17/16.28 | 24.570/6167/53.46/19.10 | 21.650/5947/48.57/23.39 | 25.180/7177/46.03/26.76 |
| BSRDM [38] + USRNet | 440 | 24.640/6415/48.09/17.10 | 24.360/6200/59.84/10.84 | 21.940/5888/49.47/18.66 | 26.720/7243/51.43/16.35 |
| MetaKernelGAN (ours) + USRNet | 200 | 25.460/6966/63.13/6.54 | 24.360/6200/59.84/10.84 | 22.210/6375/61.74/9.91 | 26.990/7608/61.78/8.34 |
| GT + USRNet (upper bound) | - | 27.800/7498/- | 26.920/6868/- | 24.850/7537/- | 29.460/8668/- |

Table 2. Average Image PSNR/Image SSIM/Kernel PSNR/ $\mathcal{L}^{\text{K-COV}}$ of MetaKernelGAN and existing joint learning methods on standard benchmarks across five runs.

following objective functions on task \mathcal{T}^i :

$$\begin{aligned} \mathcal{L}_G^{\text{task}}(\theta_D^i, \theta_G^i) &= \mathcal{L}_G^{\text{LSGAN}}(\theta_D^i, \theta_G^i) + \zeta \mathcal{L}^{\text{STO}}(\text{DK}(\theta_G^i)) \\ \mathcal{L}_D^{\text{task}}(\theta_D^i, \theta_G^i) &= \mathcal{L}_D^{\text{LSGAN}}(\theta_D^i, \theta_G^i) \end{aligned} \quad (5)$$

where ζ is a hyperparameter and $\text{DK}(\cdot)$ is the kernel decoder from Eq. (3).

In regular intervals during the task adaptation stage, we compute the meta-objective by introducing an additional loss term: $\mathcal{L}^{\text{K-PIX}}(k, \hat{k}) = \sum_{x,y} |\hat{k}_{x,y} - k_{x,y}|$ that computes the pixel-wise loss given the estimated task kernel \hat{k} . Hence, the covariance constraint is implicitly captured in this supervised learning process. Overall, our meta-objectives are defined as:

$$\begin{aligned} \mathcal{L}_G^{\text{meta}}(\theta_D^i, \theta_G^i, k) &= [\omega \mathcal{L}^{\text{K-PIX}}(k, \text{DK}(\theta_G^i)) \\ &\quad + \eta \mathcal{L}_G^{\text{LSGAN}}(\theta_D^i, \theta_G^i) \\ &\quad + \zeta \mathcal{L}^{\text{STO}}(\text{DK}(\theta_G^i))] \\ \mathcal{L}_D^{\text{meta}}(\theta_D^i, \theta_G^i) &= \mathcal{L}_D^{\text{LSGAN}}(\theta_D^i, \theta_G^i) \end{aligned} \quad (6)$$

where ω , η and ζ are hyperparameters. Following best practices from MAML++ [2], we compute the meta-objective losses for both G and D every N_{val} inner-loop steps and store in dictionary b (lines 10-12). Backpropagating at not only the last but also the preceding steps has been shown to alleviate gradient instability during the meta-training phase.

After the completion of each task adaptation process, we perform an update of the base parameters using a weighted sum of the recorded meta-objectives³ (lines 16-17). This process is repeated for N_{steps} , until the final base parameters, $\hat{\theta}_G$ and $\hat{\theta}_D$, are available to be used for inference.

Inference. To adapt to a new image, we initialize G and D with $\hat{\theta}_G$ and $\hat{\theta}_D$ and adapt these parameters using our task adaptation loss as per Eq. (5). The estimated kernel is derived from MetaKernelGAN’s generator as per Eq. (3). Full algorithm is shown in Appendix Alg. 2.

³For the weight values, we follow the schedule presented by [32]. We defer the details to the Appendix.

| Method | $\times 2$ Peak Memory Usage (GB) | $\times 4$ Peak Memory Usage (GB) | $\times 2$ Latency (Seconds) | $\times 4$ Latency (Seconds) |
|---------------------------|-----------------------------------|-----------------------------------|------------------------------|------------------------------|
| KernelGAN | 0.15 | 0.15 | 133.5 | 109.5 |
| KernelGAN-FKP | 0.23 | 0.23 | 113.9 | 441.6 |
| Double-DIP | 28.42 | 28.42 | 725.1 | 731.0 |
| DIP-FKP | 28.42 | 28.42 | 719.9 | 725.8 |
| BSRDM | 9.66 | 19.41 | 101.1 | 200.2 |
| DSKernelGAN/MetaKernelGAN | 0.15 | 0.15 | 7.1 | 7.4 |

Table 3. Inference cost to estimate the kernel from a LR image corresponding to a 1356×2040 HR image.

4. Experiments

Models. For MetaKernelGAN, we employ a 6-layer deep linear generator and adopt KernelGAN’s discriminator. More details are provided in the Appendix.

Training Setup. We use the DIV2K [34] training set which consists of images that are captured with diverse cameras. We crop the sampled HR image into patches of 192×192 for each task. The cropped image is then randomly rotated by 90° and/or flipped vertically and/or horizontally before being downsampled for each task. We meta-learn both θ_G and θ_D , leaving meta-learning θ_G only as an ablation study in the Appendix.

Optimization Parameters. Our MetaKernelGAN is trained for 100,000 meta-train steps, $N_{\text{steps}}=10^5$. We use SGD [16] and Adam [17] for the task- and meta-optimizer respectively and adopt the first-order MAML [10] algorithm due to the high memory cost of our GAN-based training. We set $N_{\text{adapt}}=25$, and leave experimenting with different N_{adapt} values in the Appendix. We set $N_{\text{val}}=5$, evaluating the meta-objectives at every 5 steps, *i.e.* at steps 5, 10, 15, 20, 25. We set the discriminator’s input patch size to 32 and use the following hyperparameters: $\alpha_G=0.01$, $\alpha_D=0.2$, $\beta_G=1e^{-4}$, $\beta_D=1e^{-4}$, $\omega=1.0$, $\eta=1.0$, and $\zeta=0.5$.

Kernel Distribution. Following KernelGAN-FKP, we consider an anisotropic Gaussian distribution; each kernel is determined based on a covariance matrix, Σ , which is parameterized by a random angle, $\Theta \sim U[0, \pi]$, and two random eigenvalues, $\lambda_1, \lambda_2 \sim U[0.35, 5.0]$. Similarly to both KernelGAN and KernelGAN-FKP, we use the computed $\times 2$ kernel to analytically derive the $\times 4$ kernel of size 21×21 .

Evaluation Setup. We use the benchmark datasets provided by [22] for KernelGAN-FKP in all our experiments. Specifically, we use Set14 [37], B100 [26], Urban100 [14], and DIV2K validation set, in which each I^{HR} image is downsampled using a different randomly generated kernel, k , to form I^{LR} . The benchmark also consists of unseen degradations, namely non-Gaussian kernels and noisy images derived by adding noise to k or I^{LR} , respectively. Concretely, uniform multiplicative noise of up to 40% of the maximum kernel pixel value is added to each randomly generated Gaussian kernel, followed by normalization so that each kernel sums to one. For noisy images, up to 3.92% of the maximum pixel value is added to I^{LR} at the end of the degradation process. Lastly, we utilize USRNet-tiny [39] as our downstream non-blind SR model and report the upper bound of attainable image performance by using the

ground-truth (GT) kernel.

Evaluation Metrics. We follow previous works and quantify the accuracy of the estimated kernel with respect to its ground truth using PSNR, which captures pixel magnitude. Unlike previous works, we also examine its covariance by using the sum of absolute distances between the discretized covariance matrices, $\hat{\Sigma}$, of the kernel estimate and its ground truth: $\mathcal{L}^{\text{K-COV}} = \sum_{x,y}^N |\hat{\Sigma}_{x,y}^{\text{GT}} - \hat{\Sigma}_{x,y}^{\text{Est}}|$, where $\hat{\Sigma}^{\text{GT}}$ and $\hat{\Sigma}^{\text{Est}}$ are the covariance matrix of the GT kernel and the estimated kernel respectively. As $\hat{\Sigma}$ is a discrete representation of Σ , the amount of discretization error depends on the kernel size. Details of how we derive $\hat{\Sigma}$ can be found in the Appendix. Discussions on the choice of this covariance metric can be found in Section 5. For images, we compute PSNR and SSIM on the Y-channel after shaving them by the scale factor. We report the average score across five runs for all quantitative results.

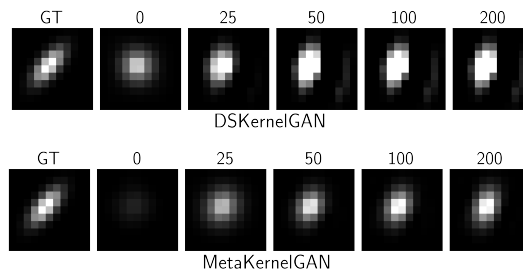


Figure 3. Kernel after 25, 50, 100, 200 adaptation steps for $\times 2$ upsampling on *108005.png* of B100. More examples in Appendix.

4.1. Comparison with Patch Recurrence Methods

In Table 1, we compare our work with other KernelGAN-based approaches, which solely utilize the patch recurrence of the given image during inference. We ran each method per-image using the recommended number of steps in each prior work and report both the average image and kernel performance.

Performance. For most natural images, the larger the image, the stronger the patch recurrence [40], which can be effectively utilized by KernelGAN and its variants. Hence, as KernelGAN-FKP requires a larger patch size than KernelGAN, KernelGAN-FKP’s performance deteriorates more for smaller spatial resolution benchmarks. As a baseline, we pretrained KernelGAN in a supervised manner, without meta-learning, optimizing both the generator and discriminator using the meta-objective (Eq 6). We named this approach DataSetKernelGAN (DSKernelGAN) and trained it with the same hyperparameters as MetaKernelGAN’s. Surprisingly, DSKernelGAN outperforms previous KernelGAN-based approaches by a large margin, even on unseen distributions. This can be attributed to starting with a reasonable kernel, *i.e.* the expected value of the training set, as opposed to randomly initializing the kernel in the parameter space (KernelGAN) or the flow-based latent space (KernelGAN-FKP). Nevertheless, unlike MetaK-

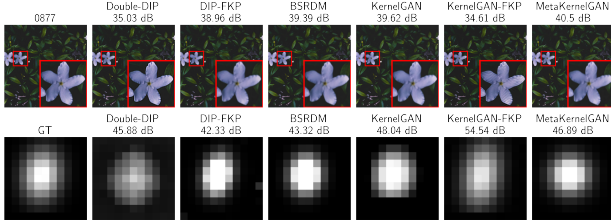


Figure 4. Comparison of estimated kernel, along with its image $\times 2$ upsampling using USRNet, on *0877.png* DIV2K. Zoom in for best results. See Appendix for more enlarged examples.

ernelGAN which directly learns from the adaptation process, DSKernelGAN is more susceptible to faulty kernels *e.g.* having non-zero values that are discontinuous as shown in Fig. 3. For $\times 4$ upsampling, the LR images are tiny and severely limited in internal information. Hence, KernelGAN and KernelGAN-FKP fail when the given image is below the required minimum spatial resolution due to the need for further downsampling. Conversely, DSKernelGAN & MetaKernelGAN supports any spatial resolution by padding the images to the minimum size. In this limited setting, the performance gap between DSKernelGAN & MetaKernelGAN is closer together as both methods start from an informed model initialization and adapt to an isotropic Gaussian-like kernel that performs well on average across multiple tasks. An analysis of the fallback on the initial kernel in tiny images can be found in the Appendix. Benchmark and real-world qualitative results can be found in Fig. 4 and 6 with more examples in the Appendix.

Inference Cost. Table 3 shows the peak memory usage and latency cost for explicit kernel estimation methods during inference. The cost of KernelGAN-based approaches is invariant to the image resolution as it utilizes a fixed-size patch for learning. Due to MetaKernelGAN’s kernel adaptation stability, the recommended number of steps is fixed at 200, resulting in a speedup of up to $18.8\times$ for $\times 2$ upsampling and $59.6\times$ for $\times 4$ upsampling over previous KernelGAN-based methods.

4.2. Comparison with Joint Learning Methods

We further compare with DIP-based approaches, which jointly optimize both the image and kernel, and BSRDM [38], which optimizes for the image noise in addition to the image and kernel (Table 2). In comparison with patch recurrence methods, these joint learning methods use full-blown models and are significantly more costly in both compute and memory. For a fair comparison with our work, we extract the kernel after optimization and feed it into USRNet for upsampling. Due to GPU memory constraints, for DIP-based approaches and BSRDM, we crop 600×600 center image patches to extract \hat{k} for DIV2K.

Performance. As joint optimization is a more challenging task than patch recurrence approaches, the naive approach, Double-DIP, although effective for image deblurring [29],

| Method | Set14 | B100 | Urban100 | DIV2K |
|---|---------------------|---------------------|---------------------|---------------------|
| $\times 2$ | | | | |
| IKC [12] | 27.34/0.8050 | 27.14/0.7712 | 24.64/0.7761 | 29.33/0.8562 |
| DAN [24] | 27.06/0.7787 | 26.98/0.7481 | 24.39/0.7554 | 29.54/0.8490 |
| DASR [36] | 27.34/0.7977 | 27.31/0.7694 | 24.73/0.7811 | 29.74/0.8611 |
| MetaKernelGAN (ours) + USRNet | 28.71/0.8331 | 28.95/0.8222 | 26.78/0.8355 | 31.67/0.8944 |
| $\times 2$ with Non-Gaussian Kernel | | | | |
| IKC [12] | 27.02/0.7765 | 26.82/0.7383 | 24.0/0.7325 | 28.99/0.8357 |
| DAN [24] | 26.66/0.7585 | 26.66/0.7277 | 23.84/0.7208 | 28.90/0.8210 |
| DASR [36] | 26.8/0.7660 | 26.77/0.7344 | 23.97/0.7297 | 29.07/0.8286 |
| MetaKernelGAN (ours) + USRNet | 28.05/0.8153 | 28.2/0.8031 | 25.84/0.8122 | 30.96/0.8830 |
| $\times 2$ with Image Noise of Level 10 (3.92%) | | | | |
| IKC [12] | 26.02/0.6895 | 25.96/0.6524 | 23.23/0.6364 | 27.65/0.7231 |
| DAN [24] | 26.02/0.6828 | 25.95/0.6500 | 23.28/0.6363 | 27.71/0.7222 |
| DASR [36] | 26.07/0.6843 | 25.98/0.6506 | 23.32/0.6375 | 27.75/0.7223 |
| MetaKernelGAN (ours) + USRNet | 28.23/0.7773 | 27.75/0.7354 | 25.86/0.7780 | 30.25/0.8311 |
| $\times 4$ | | | | |
| IKC [12] | 24.51/0.6640 | 24.22/0.6054 | 21.67/0.5952 | 25.83/0.7235 |
| DAN [24] | 24.6/0.6534 | 24.39/0.5917 | 21.69/0.5877 | 26.36/0.7247 |
| DASR [36] | 25.24/0.6719 | 24.86/0.6147 | 22.13/0.6118 | 26.8/0.7404 |
| MetaKernelGAN (ours) + USRNet | 25.46/0.6960 | 24.36/0.6200 | 22.21/0.6375 | 26.99/0.7600 |

Table 4. Comparing Image PSNR/SSIM with implicit degradation blind SR methods on standard benchmarks.

often fails to find adequate solutions. By restricting the possible set of kernel solutions, DIP-FKP outperforms previous KernelGAN-based approaches by a huge margin. BSRDM took a step further by learning the precision matrix of a Gaussian kernel, hence ensuring that the learned kernel is strictly Gaussian. Additionally, they also explicitly modeled the noise, achieving state-of-the-art for kernel estimation on most benchmarks among joint learning methods. These joint learning methods are known to perform significantly better than patch recurrence methods at a cost of heavy compute and memory. Despite being orders-of-magnitude more efficient, our approach helps to bridge this performance gap between joint learning methods and patch recurrence methods, having similar performance or even outperforming them on several benchmarks. Notably, our method is able to achieve similar performance with BSRDM without explicitly modeling the noise. Nonetheless, our approach is still limited in cases where images have weak patch recurrence, namely Set14 and B100. This result highlights the potential of better utilizing external images to improve existing unsupervised blind SR methods.

Inference Cost. We use a 1356×2040 HR image to estimate the kernel from its LR image, cropped to a maximum size of 600×600 , given the recommended number of steps. MetaKernelGAN achieves a speedup of up to $102.1\times$ and $98.7\times$ for $\times 2$ and $\times 4$ upsampling over DIP-based methods, respectively (Table 3). Additionally, DIP-based approaches have a peak memory usage of 28.4GB, two orders of magnitude higher than KernelGAN-based approaches. BSRDM uses a smaller variant of the model used in DIP-based approaches and requires fewer optimization steps. Nonetheless, it is still $14.24\times$ and $27.05\times$ slower and has a peak memory usage of $64.4\times$ and $129.4\times$ more than MetaKernelGAN for $\times 2$ and $\times 4$ upsampling, respectively.

4.3. Comparison with Implicit Estimation

Although our work focuses on explicit kernel estimation, we compare it with several implicit approaches to

| Method Comparisons | Kernel Metrics | Set14 | B100 | Urban100 | DIV2K |
|-------------------------------|-----------------------|---------------|---------------|---------------|---------------|
| ×2 | | | | | |
| MetaKernelGAN & KernelGAN-FKP | PSNR | 0.7157/0.7450 | 0.5558/0.5798 | 0.5161/0.5854 | 0.5189/0.5805 |
| | \mathcal{L}^{K-COV} | 0.7543/0.8373 | 0.3798/0.6451 | 0.5035/0.6247 | 0.5512/0.4508 |
| MetaKernelGAN & DIP-FKP | PSNR | 0.6047/0.8021 | 0.5497/0.6914 | 0.7554/0.7780 | 0.4750/0.6718 |
| | \mathcal{L}^{K-COV} | 0.5082/0.7362 | 0.4786/0.5734 | 0.4556/0.4709 | 0.4181/0.5365 |
| MetaKernelGAN & BSRDM | PSNR | 0.5137/0.6967 | 0.7092/0.8014 | 0.7349/0.7919 | 0.5292/0.7935 |
| | \mathcal{L}^{K-COV} | 0.5991/0.6132 | 0.5463/0.6486 | 0.4792/0.4375 | 0.6953/0.7483 |
| ×4 | | | | | |
| MetaKernelGAN & KernelGAN-FKP | PSNR | - | - | - | 0.5511/0.6820 |
| | \mathcal{L}^{K-COV} | - | - | - | 0.6459/0.6975 |
| MetaKernelGAN & DIP-FKP | PSNR | 0.5496/0.6219 | 0.4537/0.6082 | 0.6984/0.6914 | 0.6506/0.7227 |
| | \mathcal{L}^{K-COV} | 0.3643/0.5252 | 0.4428/0.5210 | 0.5950/0.5808 | 0.5264/0.6077 |
| MetaKernelGAN & BSRDM | PSNR | 0.4342/0.3363 | 0.5171/0.6208 | 0.6933/0.7420 | 0.6499/0.7895 |
| | \mathcal{L}^{K-COV} | 0.2180/0.1604 | 0.5418/0.5508 | 0.6344/0.5832 | 0.5828/0.7059 |

Table 5. Pearson- r /Spearman- ρ correlation between the kernel metrics and Image PSNR on the outputs of MetaKernelGAN and previous state-of-the-art kernel estimation methods.

underline their benefits and limitations. These blind SR methods avoid explicitly estimating the kernel and, instead, learn a degradation representation optimized directly in the image space in a supervised manner. We compare with three prominent works, namely IKC [12], DAN [24], and DASR [36], by evaluating the recommended pre-trained model provided by the authors.

The image performance of these implicit degradation estimation approaches is stellar, in some cases reaching a similar performance as with the use of the ground-truth kernel. However, as they do not adapt to the given image during inference, their performance drops when the test distribution differs from the training distribution, *e.g.* [24] retrains DAN for different distributions. As a result, the performance of these methods is lagging (Table 4), especially in our unseen test distributions, namely non-Gaussian kernels and noisy images, where they perform similarly to one another. Specifically, IKC and DASR $\times 2$ were trained on isotropic kernels and DAN was trained on a narrower kernel distribution. The closest matching distribution to our evaluation set is DASR $\times 4$, which was trained on anisotropic kernels with similar widths ($\lambda_1, \lambda_2 \sim U[0.2, 4.0]$), resulting in a closer performance gap. Hence, implicit degradation methods are ideal for *known* degradation distributions, as they do not require training during inference. Nevertheless, this assumption often does not hold for real-world images in the wild. In terms of memory, IKC, DAN, and DASR have 5.2M, 4.33M, and 5.8M parameters as opposed to MetaKernelGAN’s 0.116M.

4.4. Kernel Analysis

Correlation between Kernel and Image Fidelity. In previous explicit degradation estimation works, kernels are evaluated quantitatively using the PSNR metric. In our experiments, we observe that a higher kernel PSNR does not necessarily imply a higher image fidelity. For instance, KernelGAN-FKP achieves superior kernel PSNR for $\times 2$ upsampling on DIV2K, but is lacking in image PSNR. To elucidate this phenomenon, we measure the correlation between the gain in *i*) image PSNR and kernel PSNR, and

ii) image PSNR and \mathcal{L}^{K-COV} between MetaKernelGAN and previous methods and show them in Table 5. The corresponding Pearson- r and Spearman- ρ correlation coefficients indicate that both the pixel magnitude and the covariance of the estimated kernel are important to evaluate the kernel mismatch for image SR. As such, the approach of previous kernel estimation works [5, 22, 29] that solely evaluates the kernel PSNR to indicate the downstream SR performance does not suffice.

To illustrate the cases where the kernel PSNR is lower but the image PSNR is higher, Fig. 4 shows a super-resolved image together with the estimated kernel. As MetaKernelGAN is learnt using the GT kernels, it puts greater emphasis on pixels closer to the center, thus underestimating kernels that are slightly sharper than the GT and leading to images that may be slightly blurry. In contrast, although KernelGAN-FKP’s kernels look closer to the ground truth, it tends to significantly overestimate kernels, yielding smoother kernels and hence overly sharp images and unpleasant artifacts. Although both under- and over-estimation of kernels are detrimental to the downstream image quality [9], we argue that undesirable artifacts are less perceptually appealing to the human eye than a slight blur. Joint learning methods, on the other hand, estimate kernels whose covariance is closer to that of the ground truth than KernelGAN-FKP’s. However, its performance on kernel pixel magnitude is still lagging (Table 2). Moreover, we observe that for some images, DIP-FKP and KernelGAN-FKP produce kernels that have ill-conditioned Gaussian covariance matrices, aggravating kernel mismatch and image SR.

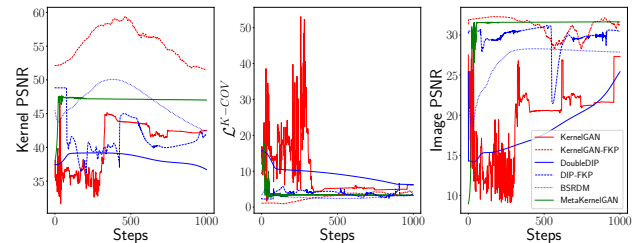


Figure 5. Intermediate kernel results and the corresponding downstream image results generated by USRNet for every adaptation iteration ($\times 2$ upsampling for *0839.png* in DIV2K).

Adaptation Stability. Fig. 5 depicts the convergence behavior of MetaKernelGAN compared to the kernel estimation baselines across adaptation steps. As DIP-FKP and KernelGAN-FKP optimize directly on the learned kernel manifold, they start off with a Gaussian-like kernel, whereas MetaKernelGAN fast adapts to one within the first 25 steps. Furthermore, although KernelGAN-FKP has a more stable convergence than KernelGAN, both methods’ performance can vary drastically depending on the number of update iterations and the given image. Similarly, despite explicitly optimizing for a Gaussian kernel, BSRDM’s performance varies across update iterations. In contrast, MetaKernel-

GAN’s learned adaptation process results in subtle changes in performance and smoother convergence across iterations. This results in a more stable image fidelity result across iterations when the kernels are used by the downstream SR model, underlining the benefits of utilizing external images for the internal learning of a single image.

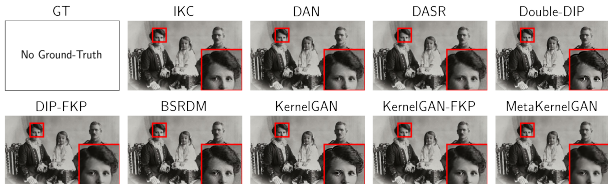


Figure 6. Real-world visual quality comparison on $\times 4$ upsampling. Zoom in for best results. More examples in Appendix.

5. Discussion

MetaKernelGAN’s Use Cases. Although MetaKernelGAN’s task is kernel estimation, it can be deployed in conjunction with an SR model to perform image SR with improved efficiency and quality. To perform SR, MetaKernelGAN first estimates the kernel of the LR image and then passes it downstream to a non-blind SR model (USRNet in our paper) to produce the HR image.

Efficiency Benefits. In real-world use-cases, SR models are deployed in two main scenarios: *i*) on a remote server as-a-service, or *ii*) on a user device, such as a TV or smartphone. In both cases, reducing the *energy consumption* is of critical importance: on the server side, energy bills contribute the most to the monetary cost, while on the device side, excessive energy consumption affects the battery life. Furthermore, minimizing the carbon footprint is a vital necessity.

The amount of inference-time computation directly determines the energy consumption. As such, with MetaKernelGAN providing significantly faster inference than all existing methods (Table 3), we substantially reduce resource usage, latency and energy, while still delivering accurate kernels. As such, MetaKernelGAN constitutes an enabling component for energy-efficient and high-quality SR.

Limitations & Further Directions. Despite our substantial gains, we are still faced with a few remaining challenges.

Kernel Evaluation. We showed that by considering both our proposed \mathcal{L}^{K-COV} and the status-quo metric (PSNR) of evaluating kernels, we can better reason about the performance of the downstream SR task. Nonetheless, these kernel metrics are less effective as the estimated kernel differs more from being Gaussian-like. As covariance matrices lie on a Riemannian manifold and not in Euclidean space, measuring the geodesic distance instead of the Euclidean distance between two covariance matrices would be a better option [4]. However, as the learned kernel estimates do not strictly follow the underlying assumed Gaussian distributions, apart from kernels estimated by

BSRDM, their covariance matrices might not be positive semi-definite, as often seen in the kernel estimates of DIP-FKP and KernelGAN-FKP. Similarly, although there are theoretical works that show how the kernel affects the SR result [9], these cannot be effectively applied to evaluate ill-conditioned covariance matrices, leading to our adoption of \mathcal{L}^{K-COV} as an approximation.

On the other hand, it is arguable whether the estimated kernels need to be strictly Gaussian in real-world scenarios. For example, unlike BSRDM, MetaKernelGAN does not explicitly optimize for image noise or restrict the kernel to be Gaussian (Eq. (5)) while achieving similar performance to BSRDM in the image noise experiments. Moreover, as the degradation process of SR is linear (Eq. (1)), the deep linear generator used in MetaKernelGAN can be learnt to represent both the kernel and the noise.

Weak Patch Recurrence. Another challenge resides in the method we adopted for internal learning; KernelGAN performs poorly on images with weak patch recurrence. Although our approach significantly mitigates this over previous approaches by utilizing an isotropic Gaussian-like kernel, there is still room for improvement in small images.

Optimal Adaptation Steps. Although MetaKernelGAN alleviates the kernel adaptation instability found in previous kernel estimation methods, the optimal number of update iterations is still dependent on the given image at test time as it is unsupervised during inference. Nonetheless, we empirically observe that our method requires significantly fewer adaptation steps compared to previous works.

6. Conclusion

Our work presents the first effective approach for integrating the benefits of learning from a dataset of images in a supervised manner and the unsupervised internal learning of a single image for kernel estimation. By learning the unsupervised adaptation process across diverse images, MetaKernelGAN robustly estimates accurate kernels applicable to high-fidelity downstream SR tasks, while being an order-of-magnitude faster than previous kernel estimation methods at no extra memory cost. Unifying both meta-learning and kernel estimation comes at a cost of creating new challenges not present in either of the original algorithms. In contrast to MAML which targets a purely supervised setting, our problem required blending supervised and unsupervised learning and their disparate objectives, as well as meta-learning both generator and discriminator. We hope that future works would also take advantage of both supervised and unsupervised methodologies such that external images can be used to effectively mitigate existing limitations and better utilize unsupervised methods.

Acknowledgements

This work was supported by Samsung AI and the European Research Council via the REDIAL project.

References

- [1] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, Accurate, and Lightweight Super-Resolution with Cascading Residual Network. In *European Conference on Computer Vision (ECCV)*, 2018. 1
- [2] Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. How to train your maml. In *International Conference on Learning Representations (ICLR)*, 2019. 4
- [3] Sébastien M R Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. learn2learn: A Library for Meta-Learning Research. Aug. 2020. 12
- [4] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):328–347, 2007. 8
- [5] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind Super-Resolution Kernel Estimation using an Internal-GAN. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2019. 2, 3, 4, 7, 12
- [6] Louis Clouatre and Marc Demers. FIGR: few-shot image generation with reptile. *CoRR*, abs/1901.02199, 2019. 2
- [7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016. 1, 3
- [8] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the Super-Resolution Convolutional Neural Network. In *European Conference on Computer Vision (ECCV)*, 2016. 1
- [9] N. Efrat, Daniel Glasner, Alexander Apartsin, B. Nadler, and A. Levin. Accurate Blur Models vs. Image Priors in Single Image Super-resolution. *IEEE International Conference on Computer Vision (ICCV)*, 2013. 7, 8, 11
- [10] Chelsea Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2017. 2, 5
- [11] Yossi Gandelsman, Assaf Shocher, and Michal Irani. “Double-DIP”: Unsupervised Image Decomposition via Coupled Deep-Image-Priors. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4
- [12] Jinjin Gu, Hannan Lu, W. Zuo, and C. Dong. Blind Super-Resolution With Iterative Kernel Correction. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 6, 7
- [13] Timothy M Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. 2
- [14] J. Huang, A. Singh, and N. Ahuja. Single Image Super-Resolution from Transformed Self-Exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 5
- [15] S. Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*, 2015. 11
- [16] J. Kiefer and Jacob Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *Annals of Mathematical Statistics*, 23:462–466, 1952. 5
- [17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5
- [18] Royson Lee, L. Dudziak, M. Abdelfattah, Stylianos I. Venieris, H. Kim, Hongkai Wen, and N. Lane. Journey towards tiny perceptual super-resolution. In *European Conference on Computer Vision (ECCV)*, 2020. 1
- [19] Royson Lee, Stylianos I. Venieris, L. Dudziak, S. Bhat-tacharya, and N. Lane. MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019. 1
- [20] Royson Lee, Stylianos I. Venieris, and Nicholas D. Lane. Deep Neural Network–based Enhancement for Image and Video Streaming Systems: A Survey and Future Directions. *ACM Computing Surveys (CSUR)*, 54:1 – 30, 2021. 1
- [21] Jingyun Liang, Guolei Sun, K. Zhang, Luc Van Gool, and Radu Timofte. Mutual Affine Network for Spatially Variant Kernel Estimation in Blind Image Super-Resolution. *IEEE International Conference on Computer Vision (ICCV)*, 2021. 12
- [22] Jingyun Liang, K. Zhang, Shuhang Gu, Luc Van Gool, and Radu Timofte. Flow-based Kernel Prior with Application to Blind Super-resolution. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 4, 5, 7, 12
- [23] Weixin Liang, Zixuan Liu, and Can Liu. Dawson: A domain adaptive few shot generation framework. *arXiv preprint arXiv:2001.00576*, 2020. 2
- [24] Zhengxiong Luo, Y. Huang, Shang Li, Liang Wang, and Tieniu Tan. Unfolding the Alternating Optimization for Blind Super Resolution. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2020. 1, 6, 7
- [25] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. *IEEE International Conference on Computer Vision (ICCV)*, 2017. 4
- [26] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision (ICCV)*, 2001. 5
- [27] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 11
- [28] Seobin Park, Jin-Su Yoo, Donghyeon Cho, J. Kim, and T. Kim. Fast Adaptation to Super-Resolution Networks via Meta-Learning. In *European Conference on Computer Vision (ECCV)*, 2020. 2
- [29] Dongwei Ren, K. Zhang, Qilong Wang, Qinghua Hu, and Wangmeng Zuo. Neural Blind Deconvolution Using Deep Priors. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3, 6, 7
- [30] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network.

- IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016. [1](#)
- [31] Assaf Shocher, N. Cohen, and M. Irani. “Zero-Shot” Super-Resolution Using Deep Internal Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. [2](#)
- [32] Jae Woong Soh, Sunwoo Cho, and N. I. Cho. Meta-Transfer Learning for Zero-Shot Super-Resolution. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [2](#), [4](#), [11](#), [12](#)
- [33] Dehua Song, Chang Xu, Xu Jia, Yiyi Chen, Chunjing Xu, and Yunhe Wang. Efficient Residual Dense Block Search for Image Super-resolution. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020. [1](#)
- [34] R. Timofte et al. NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017. [5](#)
- [35] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep Image Prior. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. [3](#)
- [36] Longguang Wang, Yingqian Wang, Xiaoyu Dong, Qingyu Xu, Jungang Yang, Wei An, and Yulan Guo. Unsupervised Degradation Representation Learning for Blind Super-Resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. [1](#), [6](#), [7](#)
- [37] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image Super-resolution via Sparse Representation. *Trans. Img. Proc. (TIP)*, 19(11):2861–2873, 2010. [5](#)
- [38] Zongsheng Yue, Qian Zhao, Jianwen Xie, Lei Zhang, Deyu Meng, and Kwan-Yee K Wong. Blind Image Super-resolution with Elaborate Degradation Modeling on Noise and Kernel. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. [2](#), [4](#), [6](#)
- [39] K. Zhang, L. Gool, and R. Timofte. Deep Unfolding Network for Image Super-Resolution. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [5](#), [12](#), [14](#), [15](#), [16](#)
- [40] Maria Zontak and Michal Irani. Internal statistics of a single natural image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. [5](#)

A. Background: Kernel Mismatch

In real-world settings, the degradation process can often be complex, involving multiple stages of blurring, down-sampling 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:

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

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

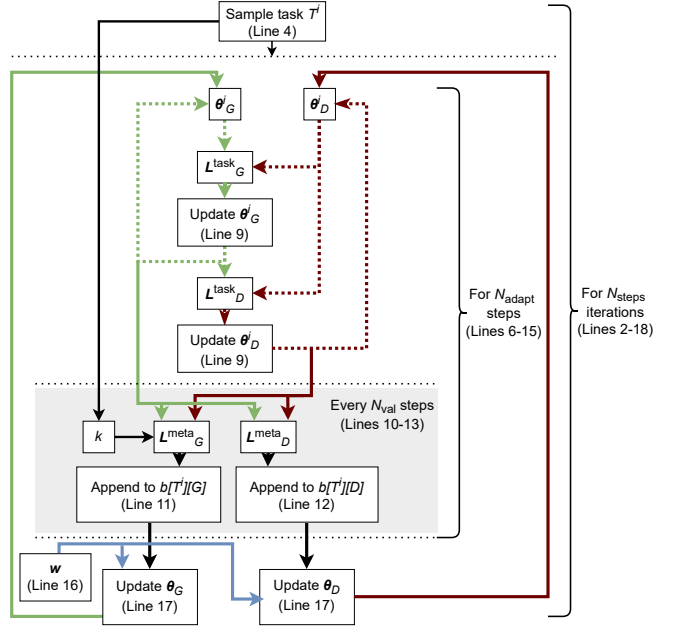


Figure 7. Flow chart of Algo. 1.

Fig. 7 shows the flow diagram of MetaKernelGAN meta-training stage shown in Alg. 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 $\mathcal{D}_s^i \cup \mathcal{D}_q^i = \{(p^{\text{LR}}, k)\} \exists p^{\text{LR}} \in \text{patches}(I^{\text{LR},i})$, *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 Gaussian 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 codebases⁴.

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 $a = \text{Var}(\text{col}_{\hat{k}})$, $b = \text{Var}(\text{rows}_{\hat{k}})$ and $c = \text{Covar}(\text{col}_{\hat{k}}, \text{rows}_{\hat{k}})$.

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 $N_{\text{adapt}}=10, 25, 50$ 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 $N_{\text{adapt}}=10$, resulting in worse kernel performance. $N_{\text{adapt}}=50$ 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>

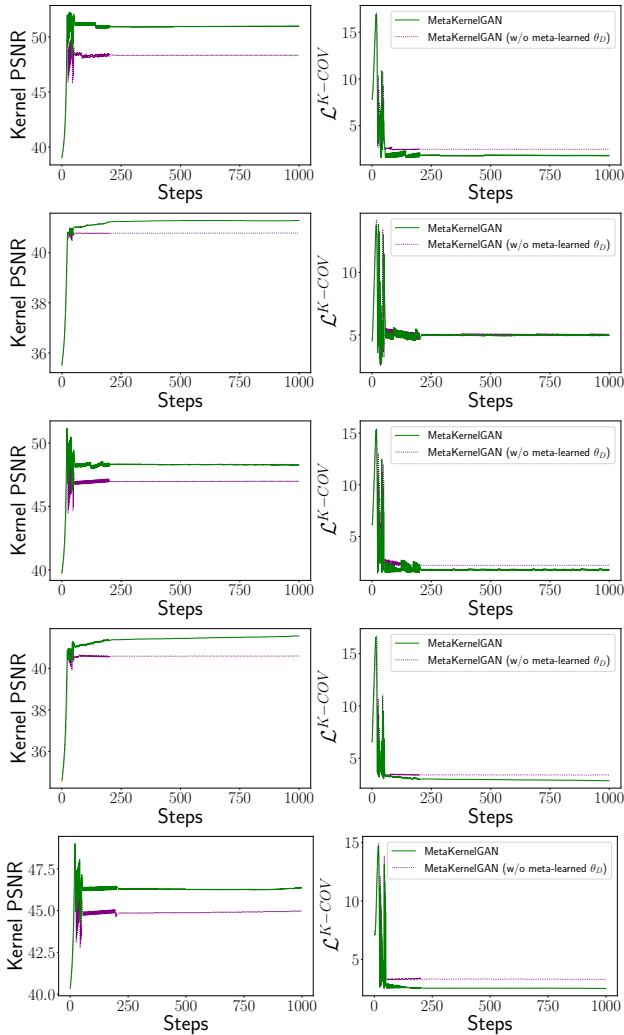


Figure 8. Intermediate kernel results for every adaptation iteration ($\times 2$ upsampling for *0802.png*, *0833.png*, *0838.png*, *0857.png*, and *0853.png* in DIV2K).

| 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 $\mathcal{L}^{\text{K-COV}}(a, b) = \sum_{x,y}^N |\hat{\Sigma}_{x,y}^a - \hat{\Sigma}_{x,y}^b|$, 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 highlighting 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.

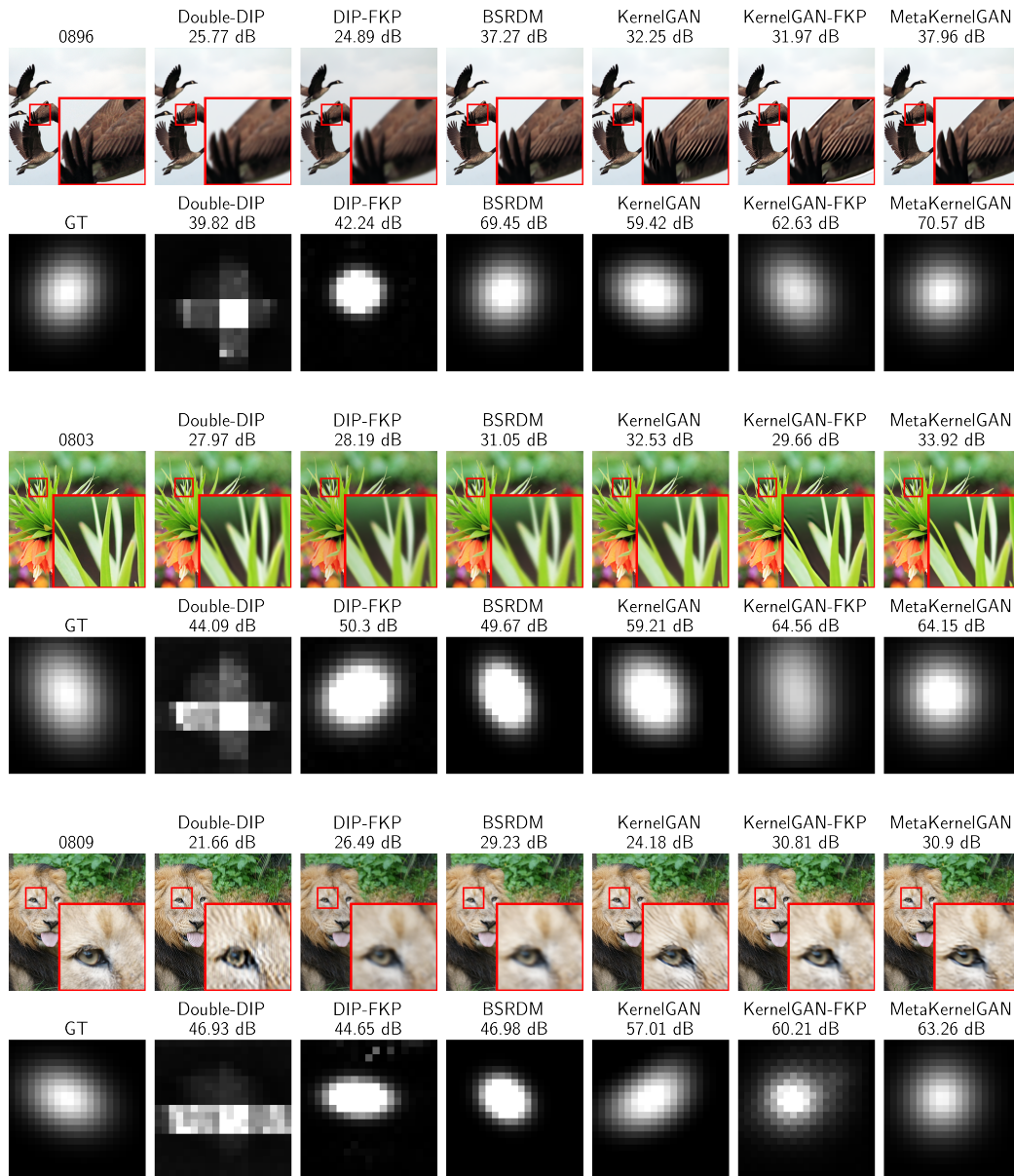


Figure 10. Comparison of estimated kernel, along with its image $\times 4$ upsampled using USRNet [39], among explicit kernel estimation methods across different benchmark datasets. Zoom in for best results. Part 1 of 2.

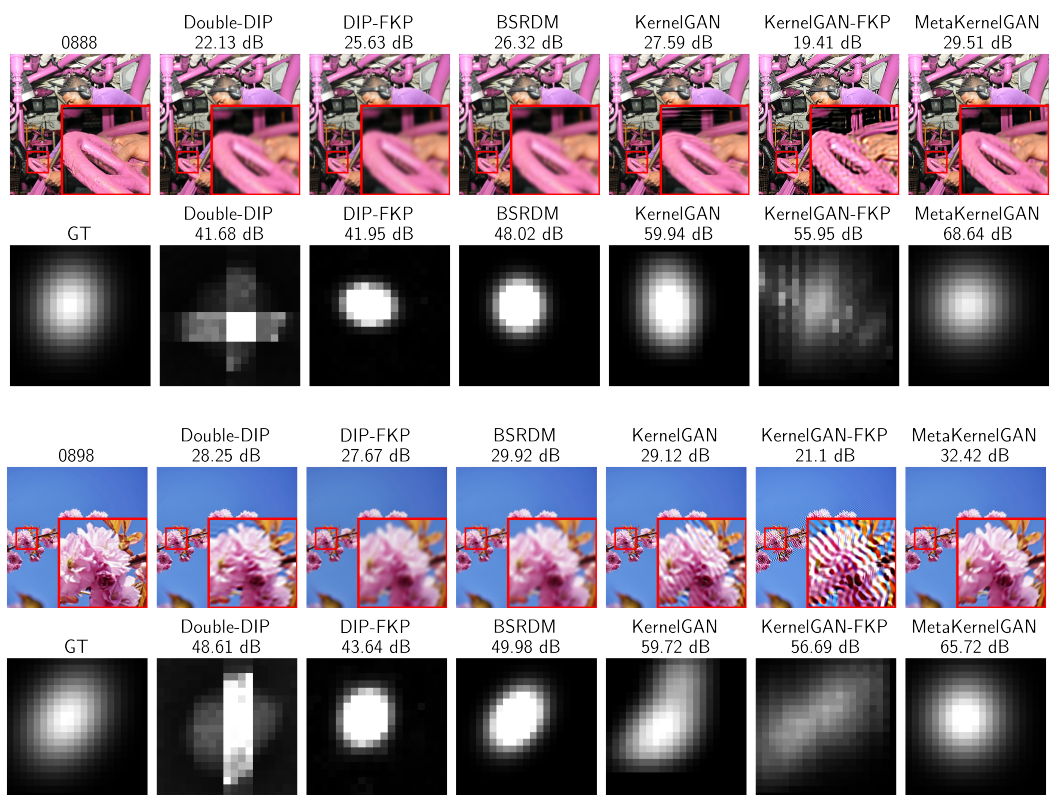


Figure 10. Comparison of estimated kernel, along with its image $\times 4$ upsampled using USRNet [39], among explicit kernel estimation methods across different benchmark datasets. Zoom in for best results. Part 2 of 2.

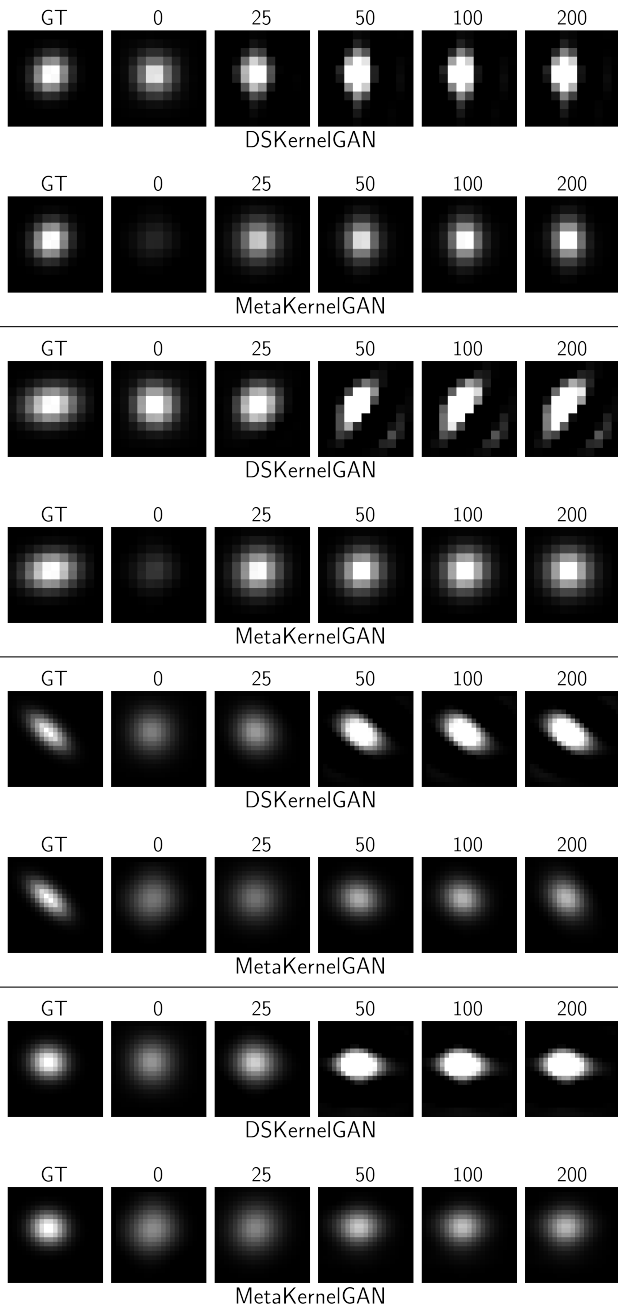


Figure 11. Kernel after 25, 50, 100, 200 adaptation steps for $\times 2$ upsampling on 86000 of B100, *img036* of Urban100, 0821 and 0824 of DIV2K (top to bottom).

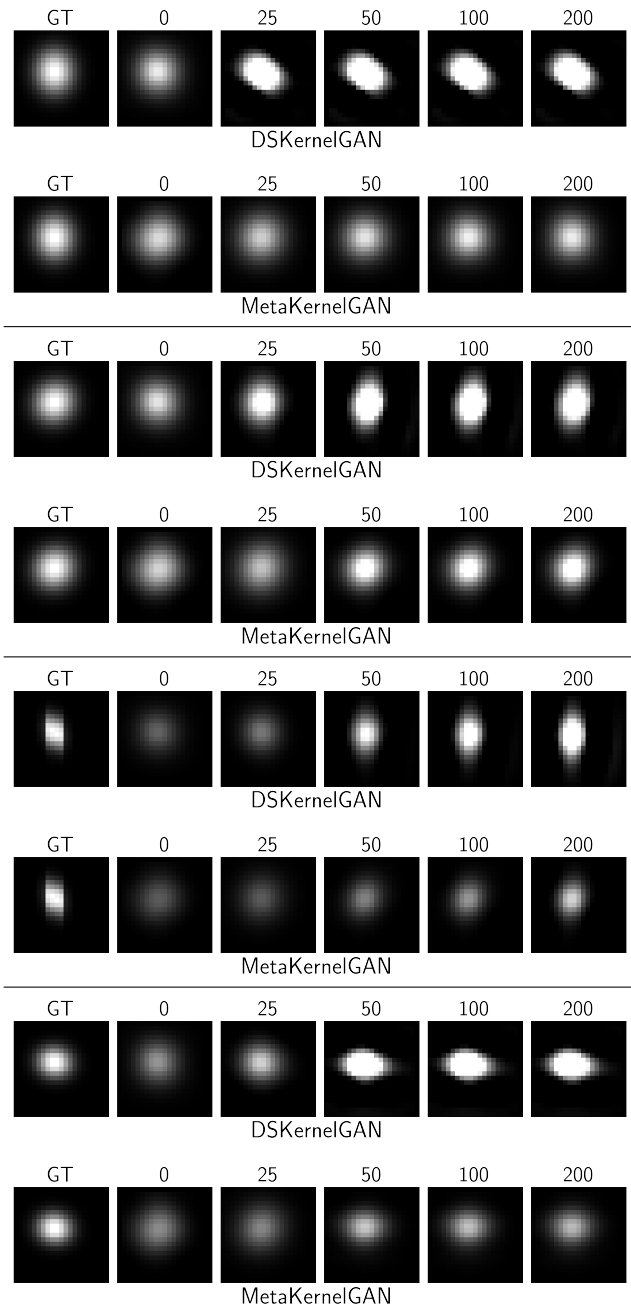


Figure 12. Kernel after 25, 50, 100, 200 adaptation steps for $\times 4$ upsampling on *monarch* and *ppt3* of Set14, and 0879 and 0824 of DIV2K (top to bottom).



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