Supplementary Material for: Stochastic Frequency Masking to Improve Super-Resolution and Denoising Networks

Majed El Helou, Ruofan Zhou, and Sabine Süsstrunk

School of Computer and Communication Sciences, EPFL, Switzerland
{majed.elhelou,ruofan.zhou,sabine.sussstrunk}@epfl.ch

Abstract. We present, in this supplementary material, extended theoretical explanations, various ablation studies, and additional experimental evaluation.
The theory section begins with a basic background discussion on anti-aliasing filtering, and specifically Gaussian kernels. We then expand on different points addressed in our main paper. We conduct different ablation studies of our proposed SFM, for instance varying the SFM rate or the masked frequency bands. Lastly, we present additional experimental evaluation. We assess the super-resolution and denoising performances in the frequency domain with and without SFM, present visual results, and evaluate different upscaling factors for super-resolution.

1 Theory

1.1 Anti-aliasing preliminaries

For completeness, we present in this section very basic background on the anti-aliasing necessary before downsampling, focusing on Gaussian kernels.

We discuss in our main paper the effect of downsampling on the frequency domain and the potential aliasing problem that can occur. Avoiding or reducing this aliasing imposes the need for pre-filtering with a low-pass filter before downsampling. We analyze the frequency-domain filtering for a one-dimensional Gaussian function $g(x)$

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}, \tag{1}$$

where σ is the standard deviation. The Fourier transform $G(f)$ of the Gaussian function is then given by

$$G(f) = e^{-\frac{f^2}{2/\sigma^2}}. \tag{2}$$

If we set $p \in [0,1]$ as the maximal accepted threshold, in the filtered frequency region, to limit the effect of aliasing, and this filtered region is $[f_c, +\infty]$ for a low-pass filter with cutoff f_c, then under these conditions we need

$$\sigma > \frac{\sqrt{-2\ln p}}{f_c}, \tag{3}$$

σ being by convention positive, and f_c being imposed by the downsampling rate and Nyquist's theorem. The Gaussian function's spectrum, not being band-limited, cannot be sampled with a sufficiently-small period for faithful reconstruction. For discrete images, different approximate discrete versions of a Gaussian blur kernel can thus be obtained by direct sampling or local integration. However, the choice of σ is still dictated by the downsampling rate if we are to avoid aliasing. The value of σ needs to be large enough to avoid or significantly attenuate the effect of aliasing, and its value is *proportional to the downsampling rate* (both of which being inversely related to the cutoff frequency).

In our paper, we construct the discrete Gaussian filters with different standard deviations through direct sampling of the corresponding continuous Gaussian functions. Other low-pass kernels are also indirectly used in the bicubic downsampling or in obtaining the real low-resolution images. The low-pass filter in the physical domain is modeled as the point spread function (PSF) of the imaging system. The PSF is related to the lens and the aperture, but also to the captured wavelength and the depth of the point source in the scene [5]. It therefore varies from image to image.

The main points are summarized as follows. (1) A more severe blur, translated for instance into a larger standard deviation of the Gaussian kernel, erases a larger part of the high-frequency components but also has some attenuation effect on the low frequencies. (2) A blind super-resolution method should be able to restore different ranges of high-frequency bands, because the different unknown degradation kernels affect different frequency components.

1.2 Implicit conditional learning

We propose in our main paper that a super-resolution network, in order to reconstruct high frequencies, implicitly learns a conditional probability

$$P\left(I^{HR} \circledast F^{HP} \mid I^{HR} \circledast F^{LP}\right), \tag{4}$$

where F^{HP} and F^{LP} are ideal high-pass and low-pass filters, applied to the high-resolution image I^{HR}, and $\circledast$ is the convolution operator. The cutoff point of the ideal filters varies depending on the downsampling rate and Nyquist's theorem, as discussed in Sec. 1.1. Being ideal, the two high-pass and low-pass filters sum to one, with product zero, on all frequency values in the frequency domain.

We analyze the practical case with non-ideal anti-aliasing filters that we mention in the main paper. As explained in Sec. 1.1, the downsampling filters being non-ideal, a trade-off imposes itself. Some high frequencies are not completely removed or some low/mid frequencies are attenuated. In practice, aliasing is removed as it is very visually disturbing, and this is done at the cost of losing some information in the low-frequency range. We define F_o^{LP} to be a practical non-ideal low-pass filter. The underlying conditional probability distribution, needed to recover the missing information, becomes

$$P\left(I^{HR} - I^{HR} \circledast F_o^{LP} \mid I^{HR} \circledast F_o^{LP}\right), \tag{5}$$

where the frequency components of $(I^{HR} \circledast F_o^{LP})$ are those remaining in the low-resolution input image. We note here the similarity with the residual learning introduced in [19]. The difference relative to the ideal-filter case is that $(1 - F_o^{LP})$ does not correspond to an ideal high-pass filter anymore. We can nonetheless separate the frequency components of the residual image $I^{HR} - I^{HR} \circledast F_o^{LP} =$

$$(I^{HR} - I^{HR} \circledast F_o^{LP}) \circledast F^{LP} + (I^{HR} - I^{HR} \circledast F_o^{LP}) \circledast F^{HP}, \tag{6}$$

where again F^{HP} and F^{LP} are complementary *ideal* high-pass and low-pass filters. We note two properties of the filters, first,

$$F_o^{LP} \circledast F^{LP} = F_o^{LP}, \tag{7}$$

which is true for any anti-aliasing filter F_o^{LP} that completely removes aliasing effects and for any ideal low-pass filter F^{LP}, and second,

$$F_o^{LP} \circledast F^{HP} = 0. \tag{8}$$

The proof of Eq. (7) becomes straight-forward when translated into the frequency domain, where the convolution becomes an element-wise product. Indeed, F^{LP} is an ideal filter that does not affect low frequencies and completely removes high frequencies. Also, F_o^{LP} removes aliasing and therefore removes all high frequencies (above π/T, for a downsampling rate T). Effectively, applying F^{LP} on F_o^{LP} only removes the high frequency values which are already zero. The proof of Eq. (8) can be derived using Eq. (7), and the fact that filters are ideal, as follows

$$F_o^{LP} \circledast F^{HP} = F_o^{LP} \circledast (1 - F^{LP}) = F_o^{LP} - F_o^{LP} = 0. \tag{9}$$

By expanding Eq. (6) and using Eq. (7) and Eq. (8) we can derive that $I^{HR} - I^{HR} \circledast F_o^{LP} =$

$$\underbrace{I^{HR} \circledast F^{LP} - I^{HR} \circledast F_0^{LP}}_{low-freq\ residual} + \underbrace{I^{HR} \circledast F^{HP}}_{high-freq}. \tag{10}$$

The interesting result is the separation between low-frequency components and high-frequency ones, which we can assume to be conditionally independent (conditioned on $I^{HR} * F_o^{LP}$). With this assumption, we can factorize Eq. (5) into the two factors

$$\begin{cases} P\left(I^{HR} \circledast F^{LP} - I^{HR} \circledast F_0^{LP} \mid I^{HR} \circledast F_o^{LP}\right) \\ P\left(I^{HR} \circledast F^{HP} \mid I^{HR} \circledast F_o^{LP}\right). \end{cases} \tag{11}$$

We first note that this also leads us to an implicit conditional distribution for predicting the high frequencies

$$P\left(I^{HR} \circledast F^{HP} \mid I^{HR} \circledast F_o^{LP}\right), \tag{12}$$

which is the same as Eq. (4) except for the conditional term. Indeed, instead of learning to predict the high-frequency components given the low-frequency ones, the network is given a degraded version of the low frequencies. But also, the network must learn to predict the residual of the degraded low frequencies

$$P\left(I^{HR} \circledast F^{LP} - I^{HR} \circledast F_0^{LP} \mid I^{HR} \circledast F_o^{LP}\right). \tag{13}$$

While the target components predicted through the distribution in Eq. (12) are the same irrespective of the degradation kernel F_o^{LP}, the target residual predicted through the distribution in Eq. (13) depends on that kernel. The network trained using that degradation kernel could therefore overfit and always produce the same residual, irrespective of the degradation of the test image. This issue is illustrated in Fig. 1. The networks in Fig. 1 that are tested on images degraded with a kernel that removes more frequencies than the training kernel do not predict the missing frequency components (plots above the diagonal). Inversely, the networks tested on images degraded with a kernel that removes fewer frequency components end up adding residual frequency components that are already in the input image (plots below the diagonal). The former can be visualized as gaps of missing frequencies, and the latter as an addition of redundant frequency components (Fig. 1).

Our SFM method masks different frequency components from the input during training to boost the implicit learning and consequently the performance of the networks. Also, as SFM masks different frequency bands in a stochastic way, it pushes the network to adapt to different F_o^{LP} that are thus simulated by the masks in the frequency domain (Sec. 1.5). This improves the learning under blind settings and the generalization to unseen degradation blur kernels.

1.3 Noise PSD

We analyze in our main paper the power spectral density (PSD) functions of white Gaussian noise (WGN) and of typical images to draw a parallel between denoising and our super-resolution formulation. We discuss here in more detail the PSD of additive Gaussian noise and Poisson noise. In the presence of Poisson-Gaussian noise, i.e. both additive Gaussian and Poisson noise components, the measured pixel intensity y at pixel i is given by

$$y[i] = x[i] + n_P(x[i]) + n_G, \tag{14}$$

where $x[i]$ is the noise-free signal at pixel i, n_G is a noise sample taken from a Gaussian distribution with standard deviation tied to the Gaussian noise level, and $n_P(\lambda)$ is a noise sample from a Poisson distribution with mean λ *from which we subtract* λ. This means that $(x[i] + n_P(x[i])) \sim \mathcal{P}(a \cdot x[i])$ [7], for some $a > 0$.

The additive Gaussian noise samples are modeled as independent and identically distributed with zero mean [18]. The corresponding PSD is thus uniform and only depends on the noise level (the samples are taken from a Gaussian stochastic process of length equal to the number of pixels, they are uncorrelated and have zero mean). The PSD can be calculated from the co-variance function and is equal to σ^2 where σ is the standard deviation of the Gaussian distribution. The additive Poisson noise is, however, not necessarily white with a uniform PSD. The different Poisson noise samples are taken from different probability distributions. Together with the clean signal, $(x[i] + n_P(x[i]))$, they are taken from a Poisson distribution with mean $a \cdot x[i]$ that varies with i. The Poisson noise components, signal aside, have zero mean. The autocorrelation function for the noise (referred to as n in what follows) is

$$R_n(\Delta) = \mathbb{E}[n[i]n[i+\Delta]] = \begin{cases} \mathbb{E}[n[i]^2], \Delta = 0 \\ \mathbb{E}[(y[i] - x[i])(y[i+\Delta] - x[i+\Delta])], o.w., \end{cases} \tag{15}$$

and we condition then run expectation on x for both terms (entire x vector). The first terms leads to the variance of $n[i]$ conditioned on x, whose conditional distribution is a Poisson distribution of mean $ax[i]$ but that is zero-shifted. After running the expectation over x we have: $a\mathbb{E}[x[i]] = a\mathbb{E}[x]$. The second term then becomes

$$\begin{aligned}\mathbb{E}[(y[i]-x[i])(y[i+\Delta]-x[i+\Delta])] = \mathbb{E}_x[x[i]x[i+\Delta]] + ...\\ \mathbb{E}_x[-x[i]\mathbb{E}[y[i+\Delta]|x] - x[i+\Delta]\mathbb{E}[y[i]|x] + \mathbb{E}[y[i]y[i+\Delta]|x]],\end{aligned} \tag{16}$$

which, because y instances are independent conditioned on x, and because $\mathbb{E}[y[i]|x] = ax[i]$, leads to

$$R_n(\Delta) = a\mathbb{E}[x]\delta(\Delta) + [R_x(\Delta) - aR_x(\Delta) - aR_x(\Delta) + a^2R_x(\Delta)](1-\delta(\Delta)), \tag{17}$$

where $\delta(\cdot)$ is the Dirac delta function. Taking the Fourier transform on both sides, we obtain that the SNR is given by

$$\frac{S_x(f)}{a\mathbb{E}[x] + (1-a)^2S_x(f) - (1-a)^2\mathbb{E}[x^2]}, \tag{18}$$

where f is the frequency and $S_x(f)$ is the PSD of x. This shows that, although to a lesser degree than with Gaussian noise, the SNR goes to zero at higher frequencies as the PSD of x itself goes to zero.

1.4 Details and choice of DCT

This section presents the mathematical details of the frequency transform that we use. In the main paper, we use the discrete cosine transform DCT type 2, also called DCT-II. The DCT-II of a one-dimensional discrete signal x of length N is defined by

$$z[k] = \sqrt{\frac{2}{N}}\sum_{j=1}^{N} x[j]\frac{1}{\sqrt{1+\delta_{k1}}}\cos\left(\frac{\pi}{2N}(2j-1)(k-1)\right), \tag{19}$$

where δ_{k1} is the Kronecker delta [1,12]. The inverse is obtained by swapping j and k as the DCT is orthogonal. The two-dimensional DCT is obtained by applying the DCT along the first dimension then along the second, and forms the basis of the JPEG compression standard [16]. The DCT-II of a length-N discrete sequence is equivalent to the DFT of a sequence of length $2N$, created by mirroring the original length-N sequence to avoid DFT artifacts [10]. Such artifacts are due to the fact that the signal is assumed to be circulantly continuous by the DFT. We mediate this issue by using the DCT-II, which we adopt in our proposed method.

1.5 Blur kernel frequency-domain basis

We present in this section a simple explanation linking our frequency-masking method to simulating blur kernels. As mentioned in the main paper, blur kernels are defined spatially with convolution. They are defined inside $\mathbb{R}^{K\times K}$ for kernels with support

K. Synthesizing all possible kernels in this space is computationally impractical. It is also not sensible because realistic kernels only form a sub-space of $\mathbb{R}^{K\times K}$, which is however not well-defined analytically.

Translating a blur kernel to the frequency domain, for instance with the DCT, provides a dissected view of the kernel's effect. A kernel acts in a multiplicative manner over every frequency band. Therefore, the effect of applying a certain blur kernel can be distributed into a basis of independent multiplications on every frequency component. If we segment frequency components into a set of M bands, from the limit of the Riemann sum, we know that $\forall\epsilon \in \mathbb{R}^+$, $\exists M$ large enough such that the absolute error of approximating the kernel function with a set of constant steps of equal width is $< \epsilon$. We can thus simulate the effect of different kernels, through a finite set of M steps in the frequency domain, with a controllable trade-off between accuracy and computation. We approximate this filtering effect in a binary way (our multiplicative step values are 0 or 1), with our SFM, by stochastically masking different frequency bands. SFM hence leverages a spanning set for the space of degradation kernels that improves generalization to unknown kernels.

1.6 Frequency visualization (extended)

We present in the main paper an experimental setup to analyze the learning and degradation-kernel overfitting through a frequency-domain visualization. That experimental setup is illustrated in Fig 2 of the main text. The RRDB [17] network is trained, as explained in our main paper, with a fixed degradation kernel F_1^{LP}. We also train the same network with 50% SFM (masking applied to half of the training set). The two networks are then tested with degradation kernel F_2^{LP}.

Here, we further vary the training degradation kernel F_1^{LP} and the testing degradation kernel F_2^{LP}, and repeat the same experiment. We present our frequency visualization, with networks trained and tested using different degradation kernels, with and without 50% SFM, in Fig. 1. For the SR networks trained without our proposed SFM, the restored SR output images from the networks (red fill) have gaps of missing frequency components when the testing degradation kernel has a lower cutoff frequency (larger Gaussian standard deviation) than the training degradation kernel. Also as explained in Sec. 1.2, when the testing degradation kernel actually has a larger cutoff than the training degradation kernel, the SR networks trained without SFM reconstruct redundant frequency components in the restored SR output (instead of a gap, we see a very clear surplus over the ground-truth PSD). This is shown in the plots below the diagonal of Fig. 1. The missing or the redundant frequency components are largely resolved by the same network architecture trained with SFM (light blue fill).

2 Ablation studies

2.1 Super-resolution

Varying masked bands This ablation study investigates the effect of the frequency-domain masking on SR when applied on different *targeted* narrow frequency bands

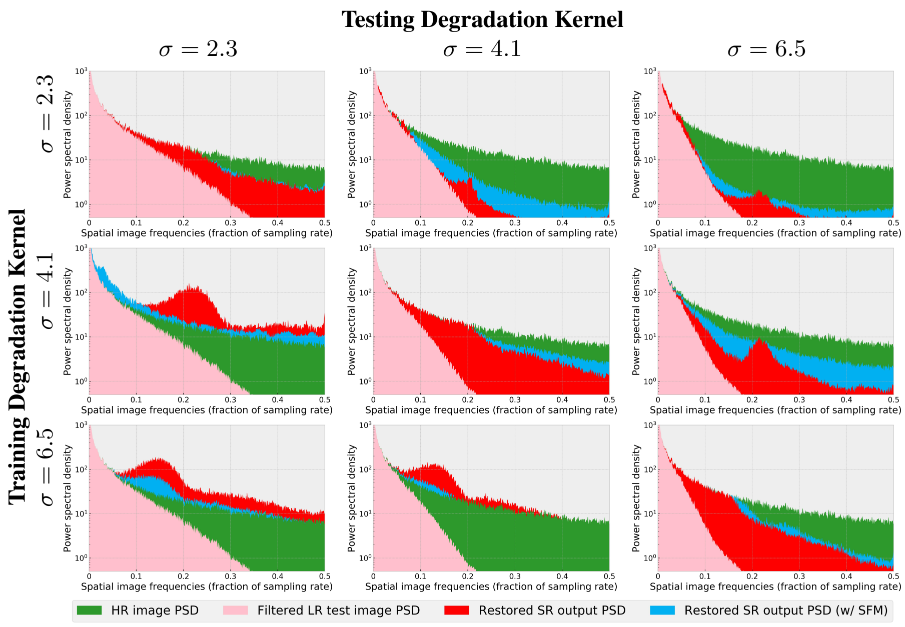


Fig. 1: Frequency visualization of SR reconstructions with different training and testing degradation kernels. We use the same experimental settings as in Sec. 3.2.1 of our main paper. We train a 20-block RRDB [17] x4 SR network with images filtered by different Gaussian blur kernels, each at a time, and evaluate it with three different Gaussian blur kernels. Results are averaged over 100 random samples. We can see that SFM improves the SR reconstructions, as the PSD of the restored images is closer to the ground-truth, and follows the average PSD power law [4,6,14,15]. SFM largely resolves the gap of missing frequencies when the test kernel has a lower frequency cutoff (larger Gaussian standard deviation) than the training kernel (plots above the diagonal). SFM also resolves the issue of redundant frequency components restored when the test kernel has a larger frequency cutoff value than the training kernel (plots below the diagonal). We lastly note that SFM slightly improves even the methods *trained and tested on the same degradation kernel* (the three plots on the diagonal).

	Test blur kernel (g_σ is a Gaussian kernel, standard deviation σ)										
	bicubic	$g_{1.7}$	$g_{2.3}$	$g_{2.9}$	$g_{3.5}$	$g_{4.1}$	$g_{4.7}$	$g_{5.3}$	$g_{5.9}$	$g_{6.5}$
RCAN [21]	29.18	23.80	24.08	23.76	23.35	22.98	22.38	22.16	21.86	21.72
RCAN with 50% LFM	29.23	23.91	24.50	24.12	23.70	**23.29**	22.53	**22.25**	**21.91**	21.77
RCAN with 50% SFM	**29.32**	**24.21**	**24.64**	**24.19**	**23.72**	23.27	**22.54**	22.23	**21.91**	**21.79**
RCAN with 50% HFM	29.20	23.78	24.07	23.79	23.39	22.98	22.40	22.17	21.88	21.75

Table 1: PSNR (dB) results of blind image SR on the DIV2K validation set for RCAN, trained with different frequency-domain maskings, on different degradation kernels. Kernels seen in the training are shaded in gray. The proposed SFM outperforms not only the baseline but also the low-frequency masking (LFM) and high-frequency masking (HFM) on almost all the degradation kernels.

rather than the wide-band frequency masking carried out by the *central mode* of SFM. We test on two different frequency bands, namely Low-Frequency Masking (LFM), in which the target band is set to $r_C = 0.25\, r_M$, and High-Frequency Masking (HFM), in which the target band is set to $r_C = 0.75\, r_M$. We control the average width of the band by setting $\sigma_\Delta = 0.15\, r_M$, as in the *targeted mode* used for denoising.

We train RCAN [21] without any masking, with 50% SFM, with 50% LFM and with 50% HFM on x4 SR on the DIV2K dataset using the same experimental settings as in the main paper. Table 1 shows the PSNR results. We see that with SFM (and LFM), RCAN gains improvements on all the test degradation kernels. This further supports that frequency-domain masking improves the learning of SR networks. SFM outperforms LFM and HFM on most of the degradation kernels, which shows the effectiveness of the proposed *central mode* frequency masking of SFM.

	Test blur kernel (g_σ is a Gaussian kernel, standard deviation σ)										
	bicubic	$g_{1.7}$	$g_{2.3}$	$g_{2.9}$	$g_{3.5}$	$g_{4.1}$	$g_{4.7}$	$g_{5.3}$	$g_{5.9}$	$g_{6.5}$
RCAN [21]	29.18	23.80	24.08	23.76	23.35	22.98	22.38	22.16	21.86	21.72
RCAN with 25% SFM	**29.35**	24.18	24.59	**24.21**	23.67	23.25	22.48	**22.31**	21.90	21.78
RCAN with 50% SFM	29.32	**24.21**	**24.64**	24.19	**23.72**	**23.27**	**22.54**	22.23	21.91	21.79
RCAN with 75% SFM	29.21	24.02	24.32	24.04	23.62	23.17	22.46	22.24	**21.95**	21.82
RCAN with 100% SFM	29.28	23.78	24.11	23.69	23.44	23.05	22.41	22.25	21.93	**21.85**

Table 2: PSNR (dB) results of blind image SR on the DIV2K validation set for RCAN on different degradation kernels. We present an ablation study over different rates of SFM. Note that 100% SFM means we mask every training input using SFM. Kernels seen in the training are shaded in gray. The results show that with any rates between 25% and 75%, SFM improves the performance of the SR network on all the degradation kernels.

Varying SFM rates This ablation study investigates the effect of varying the rate of SFM that is applied during training. We train RCAN [17] on x4 SR with a varying percentage of patches being masked with SFM. The results are shown in Table 2. With 25% and 50% SFM we achieve the best performance on most of the degradation kernels. The network trained with rates [25, 50, 75]% SFM outperforms the baseline method under all test degradation kernels, and even the one with 100% masking outperforms the baseline on all test kernels except for two. This shows that our proposed SFM is effective with different rates, and is not very sensitive to the chosen percentage of masked training patches.

2.2 Denoising

Low-frequency masking This ablation study investigates the effect of the frequency-domain masking when applied on low-frequency components rather than the high-frequency masking carried out by SFM. We use the same name as in the SR experiments and call this masking LFM, for Low-Frequency Masking, although the target frequency is smaller, as described next. The same denoiser training pipeline is used, with the same approach for applying the masking, except that the central band is set to $r_C = 0.15 * r_M$ rather than $r_C = 0.85 * r_M$ as in our SFM. All other settings are preserved, as described in our main paper.

We present the results on additive white Gaussian noise removal, without any masking, with 50% SFM, and with 50% LFM, in Table 3. The results are given for the various denoising methods analyzed in the main paper. The masking of low-frequency components is always worse than the high-frequency masking of SFM (the only exception being with RIDNet at noise level 100). LFM is almost always worse than the baseline, with some exceptions when the noise level is significantly high and therefore even relatively low frequency components are actually overtaken by the additive noise. The results show the importance of not simply masking any frequency components but specifically high-frequency ones.

Varying SFM rates This ablation study investigates the effect of varying the rate of SFM that is applied during training. In other words, we vary the percentage of total training patches that are masked using SFM and analyze the resulting performance. We carry out this ablation study on the state-of-the-art method, namely N2N [9], on the real fluorescence microscopy image benchmark test sets [20]. All training settings follow exactly the description in the paper, with the only variable being the percentage of SFM-masked training patches. We thus repeat the training with 0, 10, 25, 50, 75, 90 and 100% of images masked using SFM.

We present the results of the real-image denoising task on the mixed microscopy test set and the two-photon test set [20] in Table 4. The top results are distributed between 25 and 90% SFM, with the best ones usually at 25 or 50% SFM rates. The gray background highlights the cases where N2N with a certain SFM rate does not improve the baseline. These few exceptions are either at extreme SFM rates (10 or 100%), or at very low noise levels (the lowest one in fact). This further validates the theoretical proposition we make in our main paper that SFM becomes more applicable when the noise level is higher.

	Test noise level (standard deviation of the stationary AWGN)									
	10	20	30	40	50	60	70	80	90	100
DnCNN-B [19]	33.33	29.71	27.66	26.13	24.88	23.69	22.06	19.86	17.88	16.35
DnCNN-B with 50% LFM	33.01	29.36	27.31	25.87	24.71	23.65	22.25	20.39	18.61	17.08
DnCNN-B with 50% SFM	**33.35**	**29.78**	**27.73**	**26.27**	**25.09**	**24.02**	**22.80**	**21.24**	**19.46**	**17.87**
Noise2Noise [9]	**32.67**	28.84	26.61	25.02	23.76	22.69	21.74	20.88	20.11	19.41
Noise2Noise with 50% LFM	27.32	26.15	25.16	24.21	23.34	22.57	21.83	21.10	20.40	19.73
Noise2Noise with 50% SFM	32.55	**28.94**	**26.84**	**25.31**	**24.11**	**23.05**	**22.14**	**21.32**	**20.61**	**19.95**
Blind* N3Net [11]	**33.53**	**30.01**	**27.84**	26.30	25.04	23.93	22.87	21.84	20.87	19.98
N3Net with 50% LFM	29.24	27.62	26.42	25.44	24.56	23.72	22.90	22.10	21.35	20.65
N3Net with 50% SFM	33.41	29.86	**27.84**	**26.38**	**25.19**	**24.15**	**23.20**	**22.32**	**21.51**	**20.78**
Blind* MemNet [13]	**33.51**	29.75	27.61	26.06	24.87	23.83	22.67	21.00	18.92	17.16
MemNet with 50% LFM	32.90	29.27	27.21	25.76	24.61	23.55	22.40	21.01	19.64	18.45
MemNet with 50% SFM	33.36	**29.80**	**27.76**	**26.31**	**25.14**	**24.09**	**23.09**	**22.00**	**20.77**	**19.46**
RIDNet [2]	**33.65**	**29.87**	27.65	26.04	24.79	23.65	22.25	20.05	18.15	17.09
RIDNet with 50% LFM	31.48	28.06	25.99	24.55	23.45	22.52	21.70	20.96	20.27	**19.65**
RIDNet with 50% SFM	33.43	29.81	**27.76**	**26.30**	**25.12**	**24.08**	**23.11**	**22.08**	**20.74**	19.17

Table 3: PSNR (dB) results of blind AWGN image denoising on the standard BSD68 test set for different methods and noise levels. SFM improves the various methods, and the improvement increases with increasing noise levels, validating our hypothesis. We clamp noisy test images to [0,255] as in camera pipelines, to follow practical settings. LFM stands for low-frequency masking, which is similar to applying SFM but on low-frequency components rather than high-frequency ones, i.e. opposite to our proposed SFM approach. *We re-train under blind noise settings. The gray background indicates noise levels seen during training.

The results also show that they are not very sensitive with respect to the SFM rate, since all SFM models with rates in [25, 90]% outperform the previous state-of-the-art method on practically all noise levels in both test sets.

3 Extended experimental evaluation

3.1 Super-resolution

Different upscaling factors We present the PSNR results of RCAN [21], RRDB [17], ESRGAN [17] and IKC [8], trained without and with SFM as described in our main paper. The results with x2 and x8 upscaling factors are given, respectively, in Tables 5 and 6. We carry out the evaluation with all the different degradation kernels presented in our main paper. We note that SFM improves the results of the various methods, on both SR upscaling factors, and on practically all degradation kernels except the smallest Gaussian standard deviation ones for only IKC [8], which explicitly models and estimates all the *test* blur kernels *during training*.

DCT evaluation In this section, we analyze the reconstruction performance of x4 SR networks, trained with and without SFM, in the DCT frequency domain. We present

Method	# raw images for averaging				
	16	**8**	**4**	**2**	**1**
	Mixed test set [20]				
N2N [9]	41.45	39.43	37.59	36.40	35.40
N2N 10% SFM	41.35	39.35	37.73	36.32	35.45
N2N 25% SFM	**41.50**	<u>39.45</u>	**37.79**	36.41	**35.52**
N2N 50% SFM	<u>41.48</u>	**39.46**	<u>37.78</u>	36.43	<u>35.50</u>
N2N 75% SFM	41.40	39.44	37.75	<u>36.46</u>	<u>35.50</u>
N2N 90% SFM	41.38	**39.46**	37.76	**36.47**	<u>35.50</u>
N2N 100% SFM	41.25	39.40	37.70	36.43	35.45
	Two-photon test set [20]				
N2N [9]	38.37	35.82	34.56	33.58	32.70
N2N 10% SFM	38.68	35.98	34.79	<u>33.90</u>	33.03
N2N 25% SFM	**38.81**	36.06	<u>34.84</u>	**33.95**	**33.10**
N2N 50% SFM	<u>38.78</u>	**36.10**	**34.85**	<u>33.90</u>	<u>33.05</u>
N2N 75% SFM	38.71	36.02	34.76	33.81	33.01
N2N 90% SFM	38.69	<u>36.07</u>	34.80	33.87	<u>33.05</u>
N2N 100% SFM	38.39	35.78	34.52	33.61	32.84

Table 4: PSNR (dB) denoising results on real fluorescence microscopy images with Poisson-Gaussian noise. We present an ablation study over different rates of SFM. Note that 100% SFM means we mask every training input image using SFM. We highlight with gray background the results that *do not* outperform the previous state of the art on both benchmark datasets. Results confirm that even with very small (10%), or with very extreme SFM rates (100%), using SFM improves the results on the high noise level scenarios where our theory is most applicable.

the results of RCAN [21], ESRGAN [17] and IKC [8] in Fig. 2, with one method per row. The first column shows the image PSNR improvement of models trained with SFM compared to the models trained without it, for different Gaussian blur kernels. The second and the third columns show the MSE improvement on low-frequency and high-frequency components in the DCT domain. Low and high frequencies are split in the DCT domain using an ideal frequency filtering with cutoff at $\pi/4$. We choose this separation cutoff rather than, for instance, $\pi/2$, because the SR network is performing x4 upscaling, therefore, the anti-aliasing filter required before the downsampling must filter frequencies above $\pi/4$ to avoid high-frequency aliasing. We thus adopt this definition for high- vs low-frequency content.

The results show that SFM improves the reconstruction of SR networks for both low-frequency and high-frequency components (note that the scales are different as the typical image PSD is not uniform with respect to frequency, as discussed earlier). This improvement is consistent across the different test degradation kernels and SR methods. The results also show that the improvement on reconstructing high-frequency components does not come at the expense of the low-frequency reconstruction.

	Test blur kernel (g_σ is a Gaussian kernel, standard deviation σ)									
	bicubic	$g_{1.7}$	$g_{2.3}$	$g_{2.9}$	$g_{3.5}$	$g_{4.1}$	$g_{4.7}$	$g_{5.3}$	$g_{5.9}$	$g_{6.5}$
RCAN [21]	32.07	27.01	25.92	24.97	24.27	23.73	23.00	22.72	22.36	22.24
RCAN with 50% SFM	**32.20**	**27.19**	**26.21**	**25.35**	**24.63**	**24.10**	**23.38**	**22.91**	**22.44**	**22.29**
RRDB [17]	31.93	27.00	25.95	24.83	24.16	23.69	22.89	22.65	22.19	22.13
RRDB with 50% SFM	**31.99**	**27.08**	**26.14**	**25.21**	**24.49**	**24.02**	**23.25**	**22.88**	**22.29**	**22.18**
ESRGAN [17]	30.87	26.72	24.07	22.53	22.74	22.26	21.52	21.29	20.89	19.99
ESRGAN with 50% SFM	**30.90**	**26.81**	**24.25**	**22.66**	**22.94**	**22.49**	**21.78**	**21.40**	**21.95**	**20.03**
IKC [8]	**31.68**	**28.65**	27.43	26.33	25.78	25.29	24.44	24.20	23.89	23.61
IKC with 50% SFM	31.60	28.64	**27.51**	**26.46**	**25.99**	**25.42**	**24.67**	**24.51**	**24.08**	**23.79**

Table 5: Image SR PSNR (dB) results, with **x2 upscaling** factor, on the DIV2K validation set. Kernels seen in the trainng are shaded in gray. SFM improves the results of the various methods on different test blur kernels.

	Test blur kernel (g_σ is a Gaussian kernel, standard deviation σ)									
	bicubic	$g_{1.7}$	$g_{2.3}$	$g_{2.9}$	$g_{3.5}$	$g_{4.1}$	$g_{4.7}$	$g_{5.3}$	$g_{5.9}$	$g_{6.5}$
RCAN [21]	22.67	21.49	21.61	21.80	21.86	21.86	21.51	21.51	21.45	21.20
RCAN with 50% SFM	**22.89**	**21.70**	**21.85**	**22.09**	**22.17**	**22.19**	**21.80**	**21.78**	**21.86**	**21.35**
RRDB [17]	22.52	21.38	21.47	21.58	21.63	21.65	21.30	21.26	21.14	21.08
RRDB with 50% SFM	**22.59**	**21.45**	**21.56**	**21.80**	**21.82**	**21.87**	**21.53**	**21.48**	**21.37**	**21.18**
ESRGAN [17]	21.64	18.94	19.16	19.35	19.63	19.72	19.12	19.08	19.01	18.97
ESRGAN with 50% SFM	**21.92**	**19.04**	**19.37**	**19.62**	**19.87**	**19.99**	**19.36**	**19.31**	**19.29**	**19.15**
IKC [8]	**22.33**	**22.64**	**22.87**	22.93	23.02	22.87	22.65	22.61	22.58	22.33
IKC with 50% SFM	22.28	22.58	22.84	**22.97**	**23.09**	**23.01**	**22.78**	**22.73**	**22.69**	**22.50**

Table 6: Image SR PSNR (dB) results, with **x8 upscaling** factor, on the DIV2K validation set. Kernels seen in the training are shaded in gray. SFM improves the results of the various methods on different test blur kernels.

Visual results: synthetic kernels We present more visual results of synthetic x4 SR similar to those in Sec. 5.1 of the main paper. We show the results of RCAN [21], ESRGAN [17] and IKC [8], trained with and without 50% SFM, with different degradation kernels, in Fig. 3, 4, 5, and 6. For each of these methods, we show in the bottom row the results of the same method trained with the same settings and starting from the same network initialization but using our proposed SFM with a 50% rate, as explained in our main paper. With SFM, the SR networks are able to produce sharper results.

Visual results: real datasets We present more visual results from real SR datasets, as in Sec. 5.2 of our main paper, in Fig 7, 8 and 9. We show SR results of RCAN [21], KMSR [22] and IKC [8]. For each of these three methods, we also show the results of the same version trained with 50% SFM as described in our main paper. We clearly see that SFM improves the visual quality of the SR networks' results.

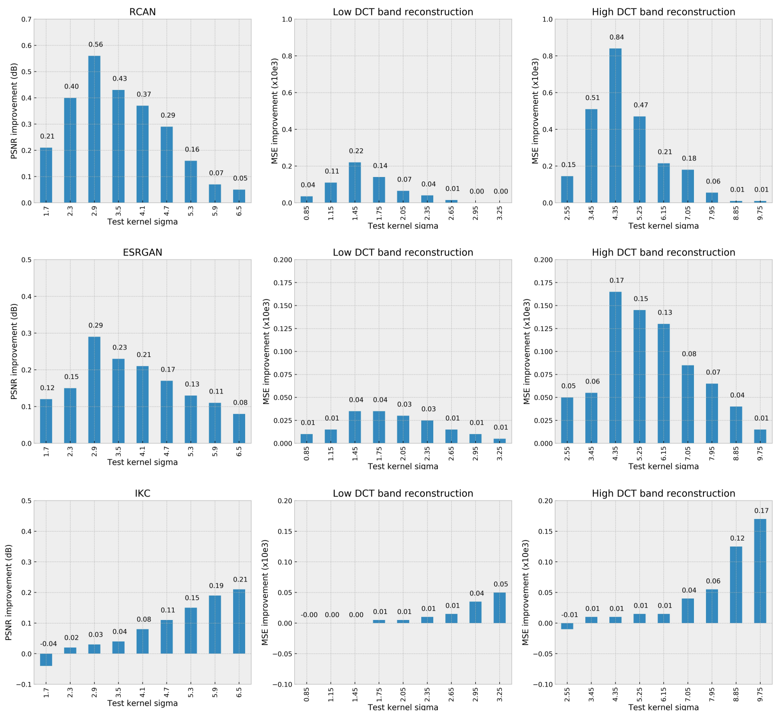
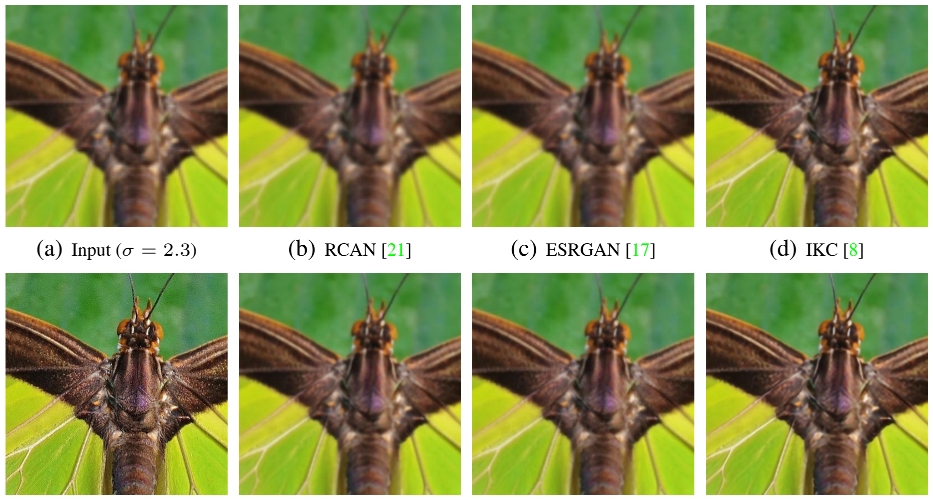


Fig. 2: The first column shows the PSNR improvements due to SFM for blind SR on the DIV2K for varying degradations, as reported in our main paper. The second and third columns show the improvements in MSE computed, respectively, on low and high frequencies. The results show that the improvements obtained on reconstructing the high-frequency content does not come at the cost of low-frequency content reconstruction, on the contrary, both are improved.

3.2 Denoising

DCT evaluation We evaluate, in our main paper, the performance of the trained denoisers, with and without SFM, using the standard PSNR metric. In this section, we are interested in analyzing the reconstruction performance in the DCT frequency domain. Fig. 10 shows results with different methods, one method per row. The first column shows the image PSNR improvement of methods trained with SFM relative to without it, for every noise level in the range 10 to 100 with steps of 10. The second and third columns also show the improvement per noise level but evaluated in the DCT domain. In the second column, we show the improvement with SFM in terms of MSE computed on the low frequencies of the image. Similarly, in the third column we show the MSE improvement of using SFM but on the high-frequency components. Low and high frequencies are split in the DCT domain into two equal ranges, thus simulating an ideal frequency filtering with cutoff at $\pi/2$.



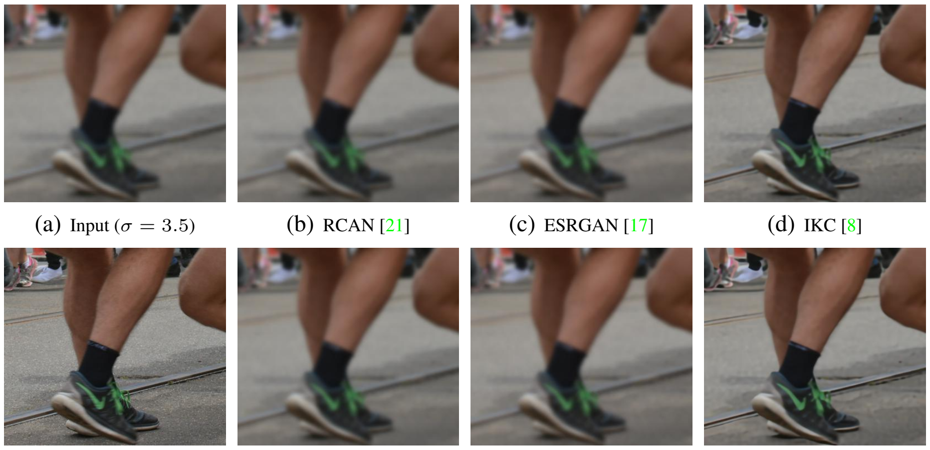
(a) Input ($\sigma = 2.3$) (b) RCAN [21] (c) ESRGAN [17] (d) IKC [8]

(e) Ground-truth (f) RCAN + SFM (g) ESRGAN + SFM (h) IKC + SFM

Fig. 3: Cropped SR results (x4 upscaling) with different methods (top row), and with the same methods trained with our SFM (bottom row), for image 0829 of the DIV2K benchmark.

The results illustrate the increase in improvement as the noise level increases, supporting our main paper's hypothesis. Furthermore, we see that the improvement in reconstruction is notable in both low and high frequencies, for the different methods. SFM does indeed improve the reconstruction of high frequencies by pushing the network during training to predict them from their low-frequency counterpart (corresponding to the bottom conditional distribution in Eq. (11)). Also importantly, this procedure is not damaging the denoising of low frequencies (corresponding to the top conditional distribution in Eq. (11)), as shown by the results in the third column. Our SFM even improves the reconstruction of those low frequencies, possibly because the network has a direct view of them during training when SFM masks the high-frequency counterpart in the input.

Visual results: AWGN We present visual denoising results for additive white Gaussian noise (AWGN) removal. We show the results of DnCNN [19], N2N [9], N3Net [11], MemNet [13], and RIDNet [2] on different images from the BSD68 benchmark, for different noise levels. For each of these methods, we also show in the bottom row the results of the same method trained with the same settings and starting from the same network initialization but using our proposed SFM with a 50% rate, as explained in our main paper. The results are shown in Fig. 11, 12, 13, 14, 15, 16, 17, 18, and 19. The first column shows the noisy input image (and the corresponding standard deviation of the AWGN) in the top row, and the ground-truth image in the bottom row.



(a) Input ($\sigma = 3.5$) (b) RCAN [21] (c) ESRGAN [17] (d) IKC [8]

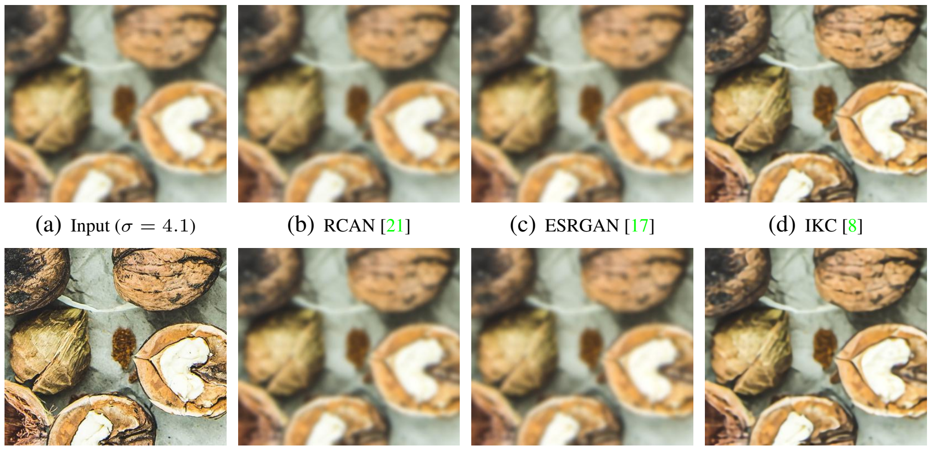
(e) Ground-truth (f) RCAN + SFM (g) ESRGAN + SFM (h) IKC + SFM

Fig. 4: Cropped SR results (x4 upscaling) with different methods (top row), and with the same methods trained with our SFM (bottom row), for image 0832 of the DIV2K benchmark.

Visual results: real Poisson-Gaussian images We present visual denoising results from the real-image fluorescence microscopy dataset in Fig. 20, 21, 22. The first column shows the noisy images, obtained by averaging a different number of raw images to indirectly control the noise level, and the ground-truth images in the last column are estimated by averaging 50 raw images for every scan. We present the results of the two methods that can be trained without ground-truth data for such real-image datasets, namely, N2S [3] and N2N [9]. For each of these two methods, we also show the results of the same version trained with 50% SFM as described in our main paper.

References

1. Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE Transactions on Computers **100**(1), 90–93 (1974) 5
2. Anwar, S., Barnes, N.: Real image denoising with feature attention. ICCV (2019) 10, 14, 21, 22, 23, 24, 25, 26, 27, 28, 29
3. Batson, J., Royer, L.: Noise2Self: Blind denoising by self-supervision. In: ICML (2019) 15, 30, 31, 32
4. Burton, G.J., Moorhead, I.R.: Color and spatial structure in natural scenes. Applied Optics **26**(1), 157–170 (1987) 7
5. El Helou, M., Dümbgen, F., Süsstrunk, S.: AAM: An assessment metric of axial chromatic aberration. In: ICIP (2018) 2
6. Field, D.J.: Relations between the statistics of natural images and the response properties of cortical cells. JOSA **4**(12), 2379–2394 (1987) 7
7. Foi, A., Trimeche, M., Katkovnik, V., Egiazarian, K.: Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. IEEE Transactions on Image Processing **17**(10), 1737–1754 (2008) 4

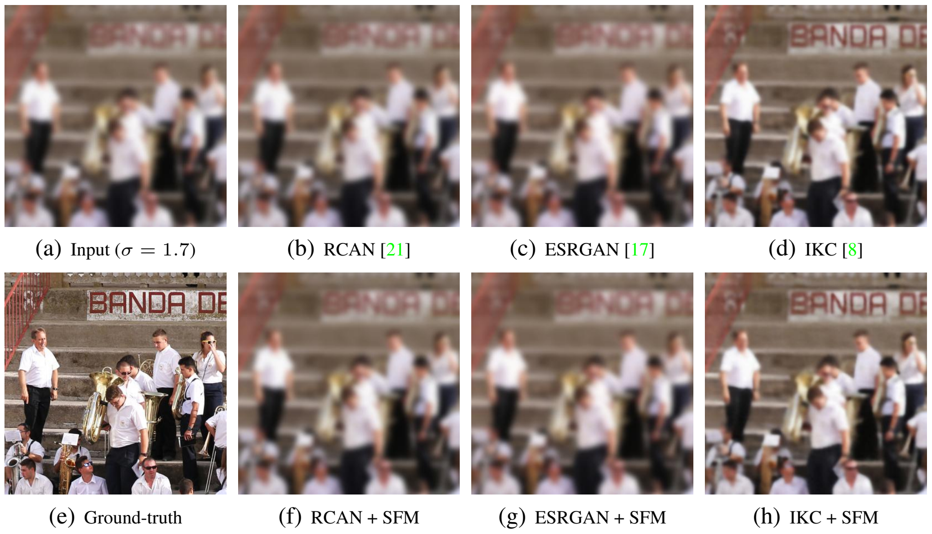


(a) Input ($\sigma = 4.1$) (b) RCAN [21] (c) ESRGAN [17] (d) IKC [8]

(e) Ground-truth (f) RCAN + SFM (g) ESRGAN + SFM (h) IKC + SFM

Fig. 5: Cropped SR results (x4 upscaling) with different methods (top row), and with the same methods trained with our SFM (bottom row), for image 0872 of the DIV2K benchmark.

8. Gu, J., Lu, H., Zuo, W.Z., Dong, C.: Blind super-resolution with iterative kernel correction. In: CVPR (2019) 10, 11, 12, 14, 15, 16, 17, 18, 19
9. Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., Aila, T.: Noise2Noise: Learning image restoration without clean data. In: ICML (2018) 9, 10, 11, 14, 15, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
10. Mahmood, F., Toots, M., Öfverstedt, L.G., Skoglund, U.: 2D discrete Fourier transform with simultaneous edge artifact removal for real-time applications. In: IEEE International Conference on Field Programmable Technology. pp. 236–239 (2015) 5
11. Plötz, T., Roth, S.: Neural nearest neighbors networks. In: NeurIPS (2018) 10, 14, 21, 22, 23, 24, 25, 26, 27, 28, 29
12. Strang, G.: The discrete cosine transform. SIAM **41**(1), 135–147 (1999) 5
13. Tai, Y., Yang, J., Liu, X., Xu, C.: MemNet: A persistent memory network for image restoration. In: ICCV (2017) 10, 14, 21, 22, 23, 24, 25, 26, 27, 28, 29
14. Tolhurst, D., Tadmor, Y., Chao, T.: Amplitude spectra of natural images. Ophthalmic and Physiological Optics **12**(2), 229–232 (1992) 7
15. Torralba, A., Oliva, A.: Statistics of natural image categories. Network: Computation in Neural Systems **14**(3), 391–412 (2003) 7
16. Wallace, G.K.: The JPEG still picture compression standard. IEEE Transactions on Consumer Electronics **38**(1) (1992) 5
17. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., Loy, C.C.: ESRGAN: Enhanced super-resolution generative adversarial networks. In: ECCV Workshops (2018) 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19
18. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: NeurIPS (2012) 4
19. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Transactions on Image Processing **26**(7), 3142–3155 (2017) 3, 10, 14, 21, 22, 23, 24, 25, 26, 27, 28, 29

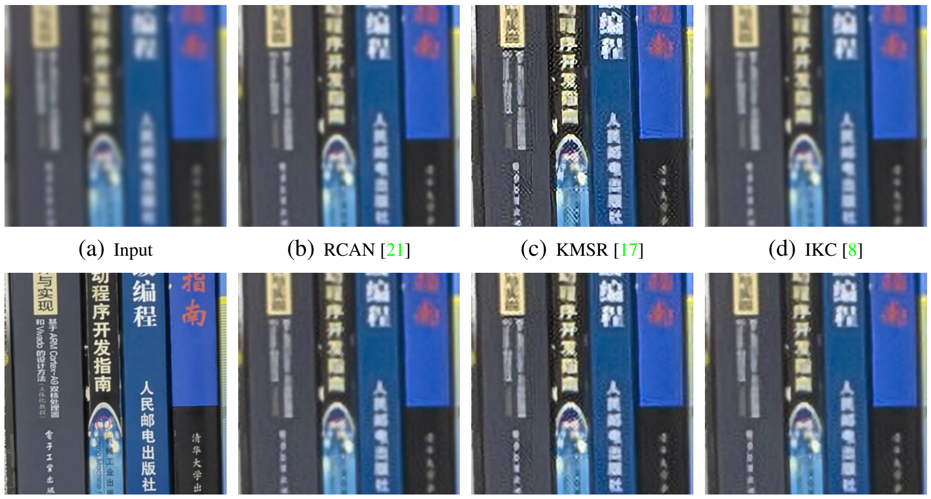


(a) Input ($\sigma = 1.7$) (b) RCAN [21] (c) ESRGAN [17] (d) IKC [8]

(e) Ground-truth (f) RCAN + SFM (g) ESRGAN + SFM (h) IKC + SFM

Fig. 6: Cropped SR results (x4 upscaling) with different methods (top row), and with the same methods trained with our SFM (bottom row), for image 0825 of the DIV2K benchmark.

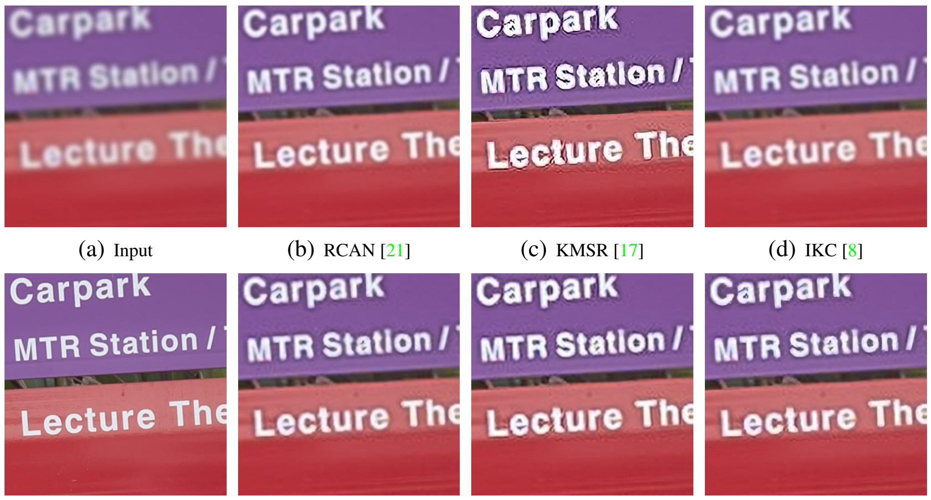
20. Zhang, Y., Zhu, Y., Nichols, E., Wang, Q., Zhang, S., Smith, C., Howard, S.: A Poisson-Gaussian denoising dataset with real fluorescence microscopy images. In: CVPR (2019) 9, 11, 30, 31, 32
21. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: ECCV (2018) 8, 10, 11, 12, 14, 15, 16, 17, 18, 19
22. Zhou, R., Süsstrunk, S.: Kernel modeling super-resolution on real low-resolution images. In: ICCV (2019) 12



(a) Input (b) RCAN [21] (c) KMSR [17] (d) IKC [8]

(e) Ground-truth (f) RCAN + SFM (g) KMSR + SFM (h) IKC + SFM

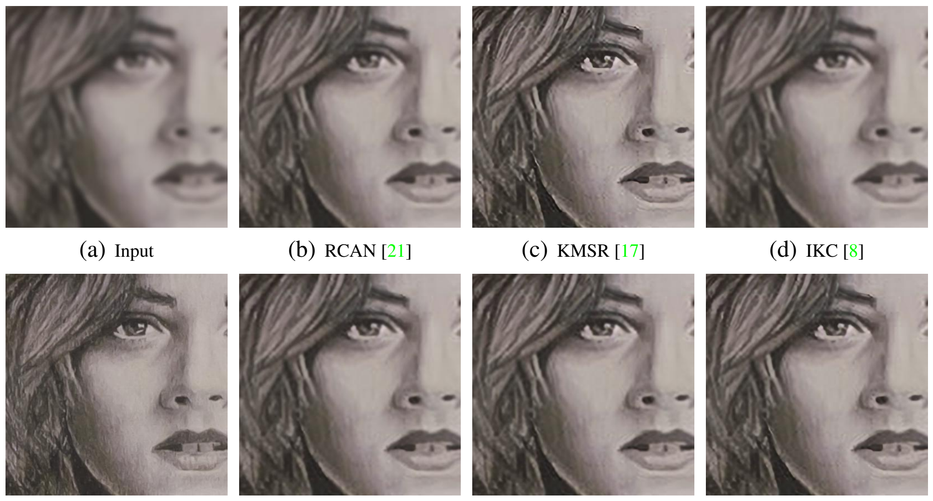
Fig. 7: Cropped SR results (x4 upscaling) with different methods (top row), and with the same methods trained with our SFM (bottom row), for image Canon_013 of the Real SR benchmark.



(a) Input (b) RCAN [21] (c) KMSR [17] (d) IKC [8]

(e) Ground-truth (f) RCAN + SFM (g) KMSR + SFM (h) IKC + SFM

Fig. 8: Cropped SR results (x4 upscaling) with different methods (top row), and with the same methods trained with our SFM (bottom row), for image Nikon_004 of the Real SR benchmark.



(a) Input (b) RCAN [21] (c) KMSR [17] (d) IKC [8]

(e) Ground-truth (f) RCAN + SFM (g) KMSR + SFM (h) IKC + SFM

Fig. 9: Cropped SR results (x4 upscaling) with different methods (top row), and with the same methods trained with our SFM (bottom row), for image Nikon_011 of the Real SR benchmark.

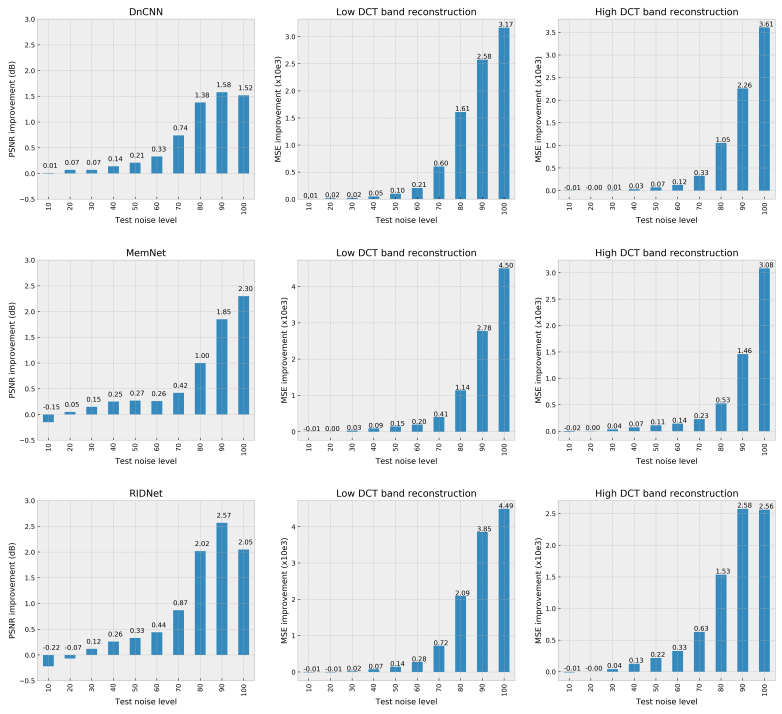
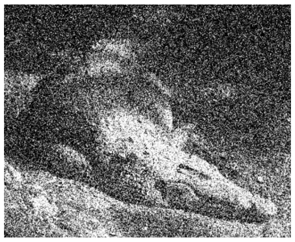
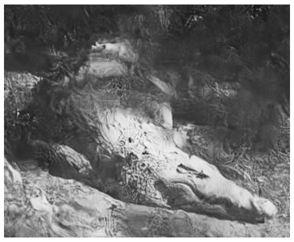
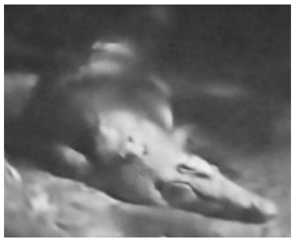
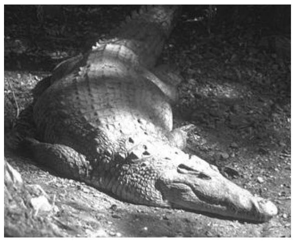


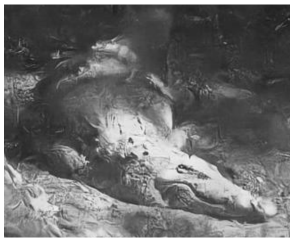
Fig. 10: The first column shows the PSNR improvements due to SFM for blind AWGN denoising on the BSD68 benchmark for varying noise levels from 10 to 100, with steps of 10, as reported in our main paper. The second and third columns show the improvements in MSE computed, respectively, on low and high frequencies. The results show that the improvements obtained on reconstructing the high-frequency content does not come at the cost of low-frequency content reconstruction, on the contrary, both are improved. We also note that the improvement increases with increasing noise levels, supporting our original hypothesis.

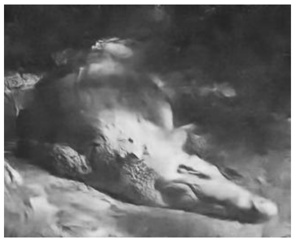

(a) Noisy ($\sigma = 70$)

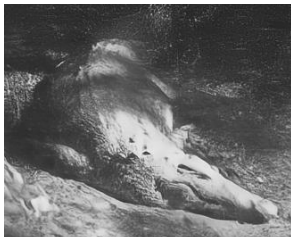

(b) DnCNN [19] 21.25

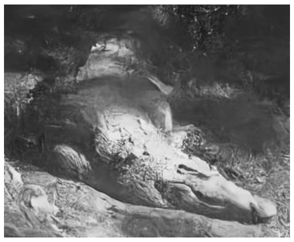

(c) N2N [9] 20.51

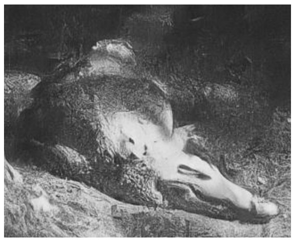

(d) Ground-truth

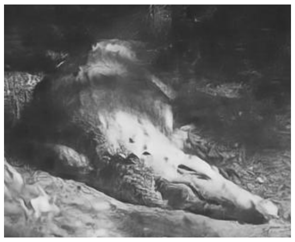

(e) DnCNN+SFM 21.65

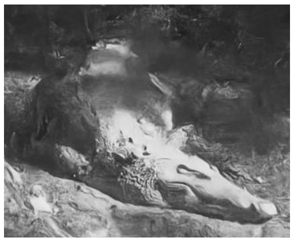

(f) N2N+SFM 21.18


(g) N3Net [11] 21.64


(h) MemNet [13] 21.59


(i) RIDNet [2] 20.96


(j) N3Net+SFM 21.88


(k) MemNet+SFM 21.80

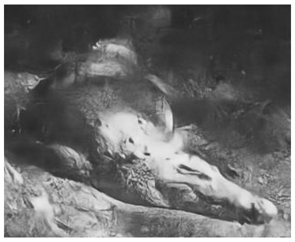
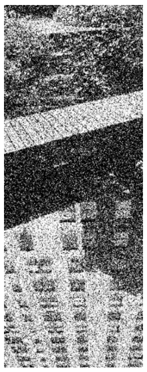
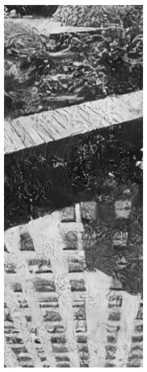
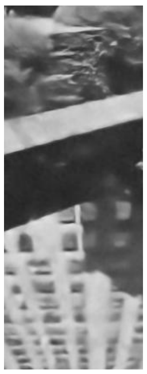
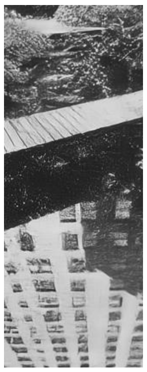

(l) RIDNet+SFM 21.88

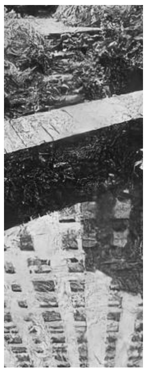
Fig. 11: Denoising results with different methods (1st and 3rd row), and with the same method trained with our SFM (2nd and 4th row), for image 14 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

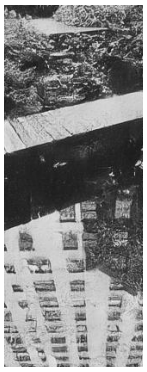

(a) Noisy ($\sigma = 80$)

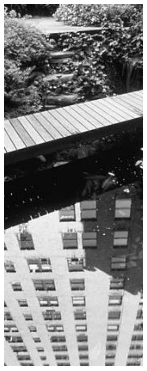

(b) DnCNN [19] 18.27

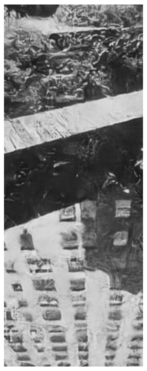

(c) N2N [9] 17.78

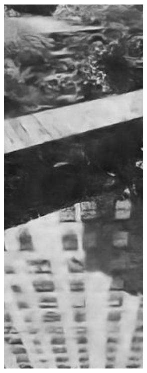

(d) N3Net [11] 18.93

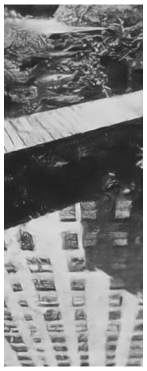

(e) MemNet [13] 18.35

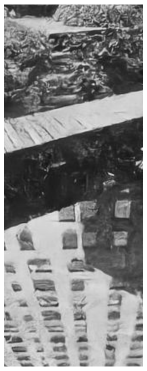

(f) RIDNet [2] 18.41


(g) Ground-truth


(h) DnCNN+SFM 18.81


(i) N2N+SFM 18.48


(j) N3Net+SFM 19.22


(k) MemNet+SFM 19.06

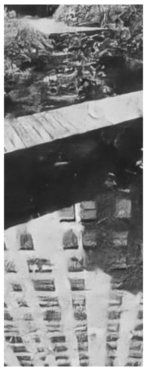
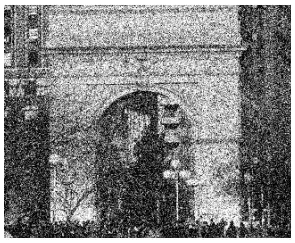
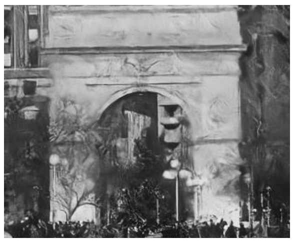
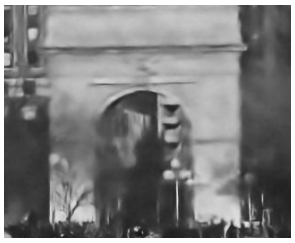
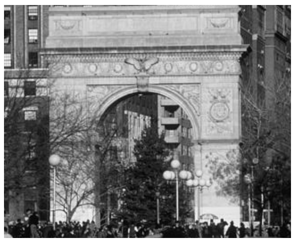

(l) RIDNet+SFM 19.01

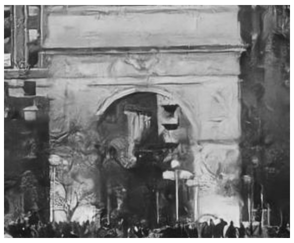
Fig. 12: Denoising results with different methods (top row), and with the same method trained with our SFM (bottom row), for image 20 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

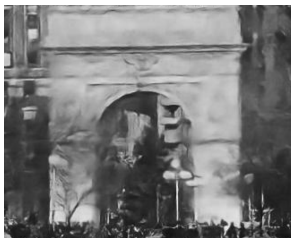

(a) Noisy ($\sigma = 60$)

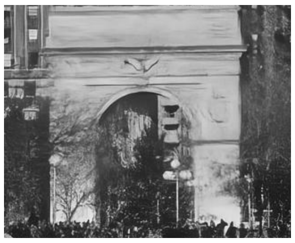

(b) DnCNN [19] 21.92

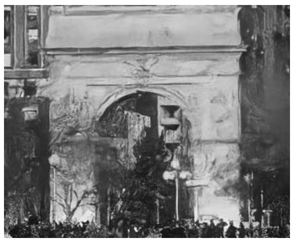

(c) N2N [9] 20.98

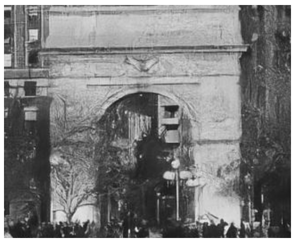

(d) Ground-truth

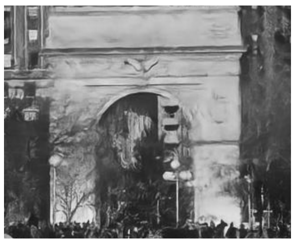

(e) DnCNN+SFM 22.12

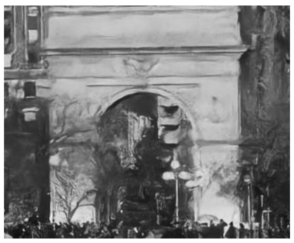

(f) N2N+SFM 21.42


(g) N3Net [11] 21.95


(h) MemNet [13] 21.74


(i) RIDNet [2] 21.98


(j) N3Net+SFM 22.26


(k) MemNet+SFM 22.13

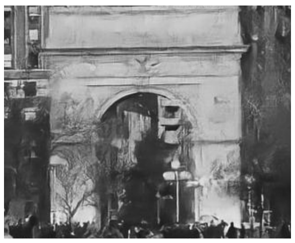
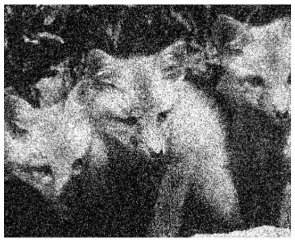
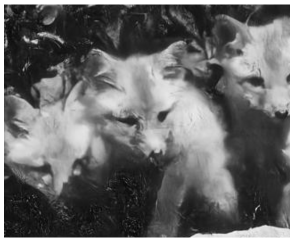
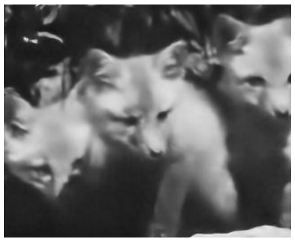
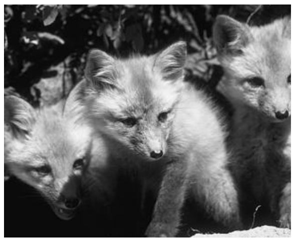

(l) RIDNet+SFM 22.12

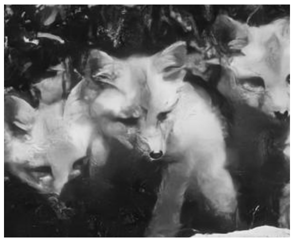
Fig. 13: Denoising results with different methods (1st and 3rd row), and with the same method trained with our SFM (2nd and 4th row), for image 21 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

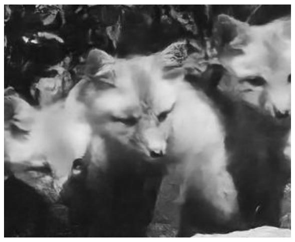

(a) Noisy ($\sigma = 50$)

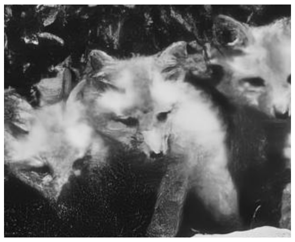

(b) DnCNN [19] 23.51

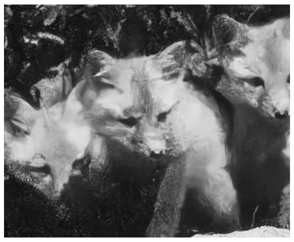

(c) N2N [9] 23.04

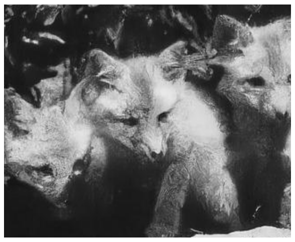

(d) Ground-truth

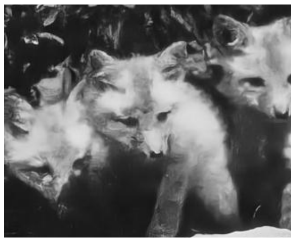

(e) DnCNN+SFM 23.87

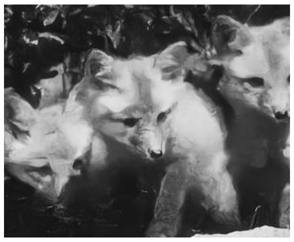

(f) N2N+SFM 23.36


(g) N3Net [11] 23.40


(h) MemNet [13] 23.05


(i) RIDNet [2] 23.15


(j) N3Net+SFM 23.95


(k) MemNet+SFM 24.01

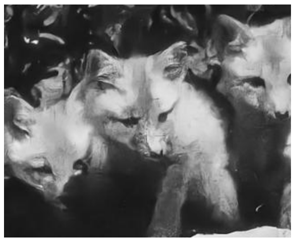
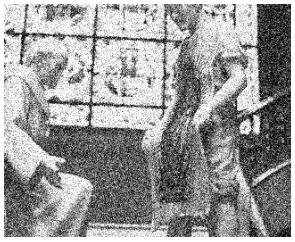
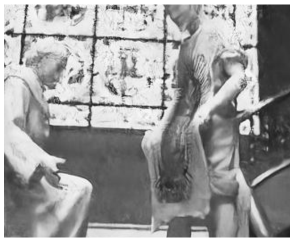
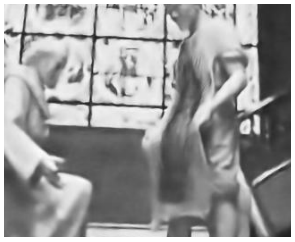
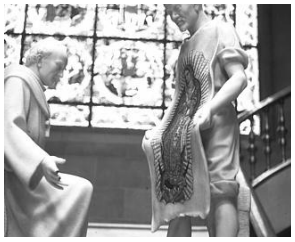

(l) RIDNet+SFM 23.84

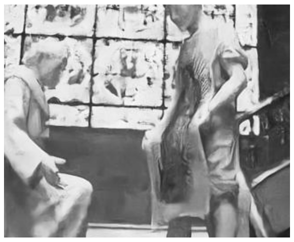
Fig. 14: Denoising results with different methods (1st and 3rd row), and with the same method trained with our SFM (2nd and 4th row), for image 23 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

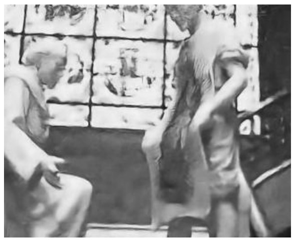

(a) Noisy ($\sigma = 50$)

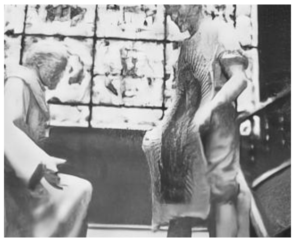

(b) DnCNN [19] 23.77

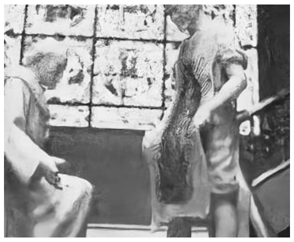

(c) N2N [9] 23.21

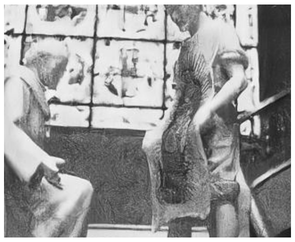

(d) Ground-truth

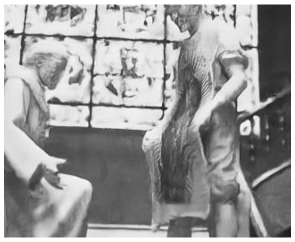

(e) DnCNN+SFM 24.26

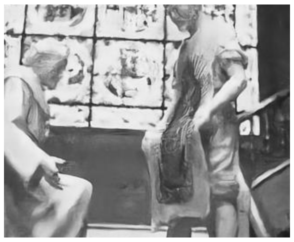

(f) N2N+SFM 23.83


(g) N3Net [11] 24.10


(h) MemNet [13] 23.58


(i) RIDNet [2] 23.45


(j) N3Net+SFM 24.30


(k) MemNet+SFM 24.31

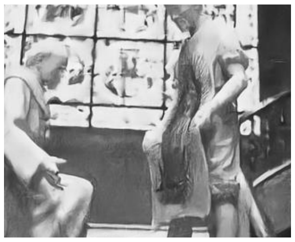
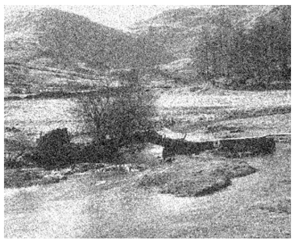
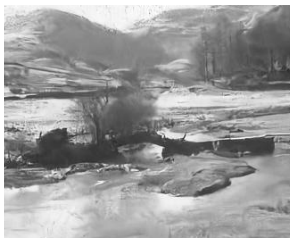
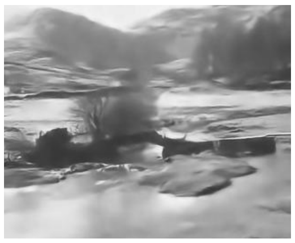
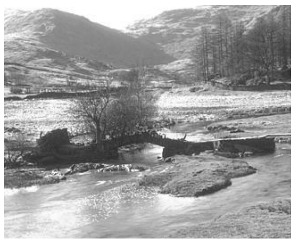

(l) RIDNet+SFM 24.30

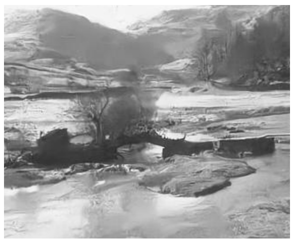
Fig. 15: Denoising results of different methods (1st and 3rd row), and of the same methods trained with our SFM (2nd and 4th row), for image 47 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

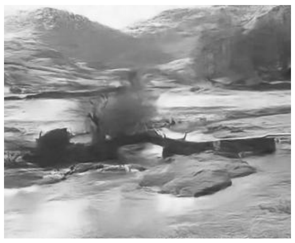

(a) Noisy ($\sigma = 30$)

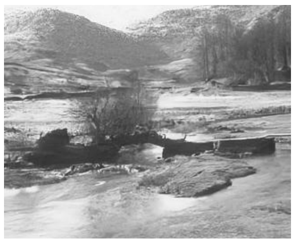

(b) DnCNN [19] 25.35

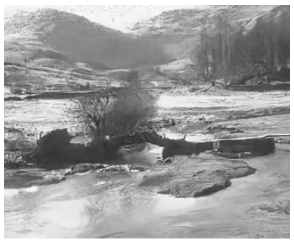

(c) N2N [9] 24.30

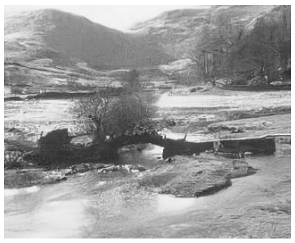

(d) Ground-truth

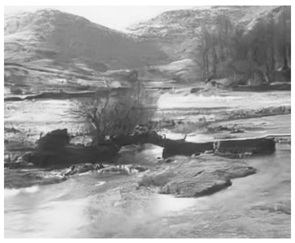

(e) DnCNN+SFM 25.60

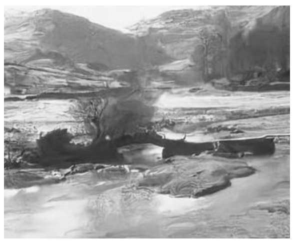

(f) N2N+SFM 24.99


(g) N3Net [11] 25.49


(h) MemNet [13] 25.48


(i) RIDNet [2] 25.47


(j) N3Net+SFM 25.58


(k) MemNet+SFM 25.63

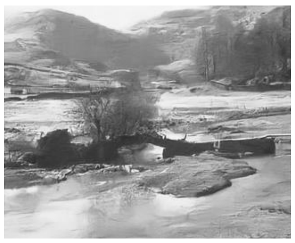
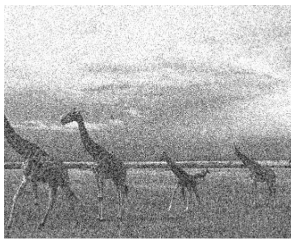
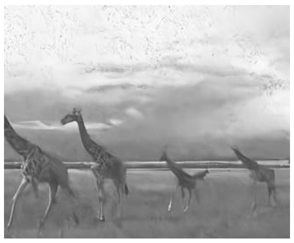
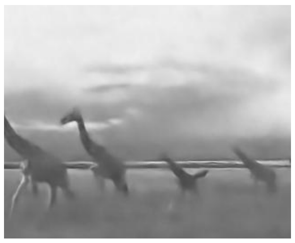
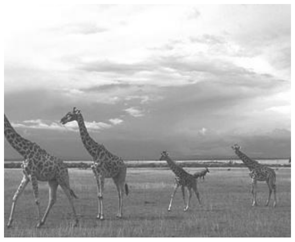

(l) RIDNet+SFM 25.66

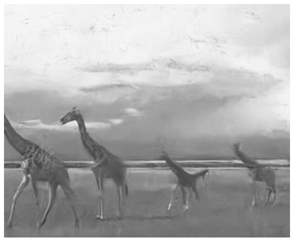
Fig. 16: Denoising results with different methods (1st and 3rd row), and with the same method trained with our SFM (2nd and 4th row), for image 49 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

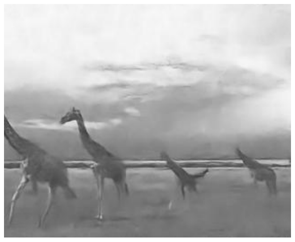

(a) Noisy ($\sigma = 30$)

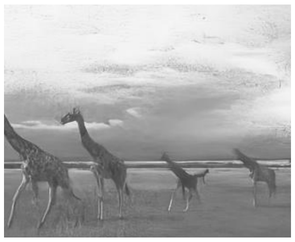

(b) DnCNN [19] 28.82

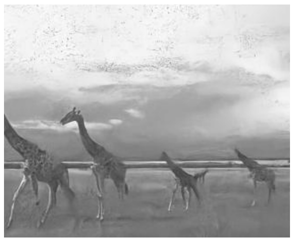

(c) N2N [9] 26.92

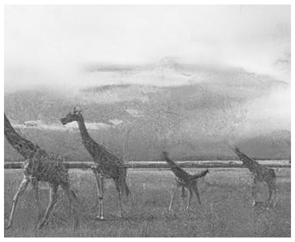

(d) Ground-truth

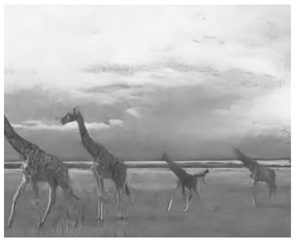

(e) DnCNN+SFM 29.25

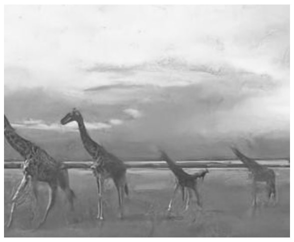

(f) N2N+SFM 28.69


(g) N3Net [11] 29.08


(h) MemNet [13] 28.81


(i) RIDNet [2] 27.22


(j) N3Net+SFM 29.37


(k) MemNet+SFM 29.38

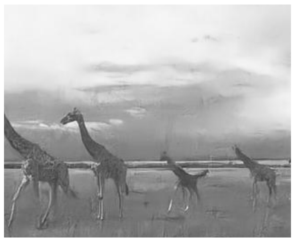
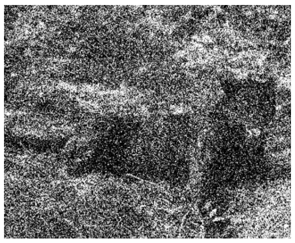
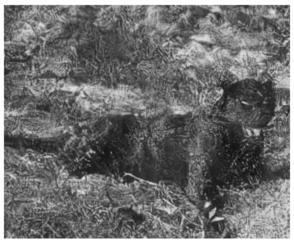
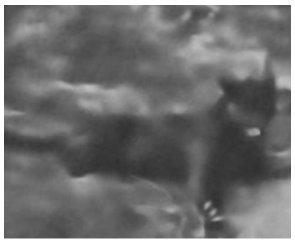
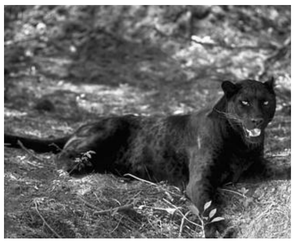

(l) RIDNet+SFM 29.47

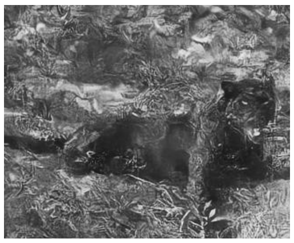
Fig. 17: Denoising results with different methods (1st and 3rd row), and with the same method trained with our SFM (2nd and 4th row), for image 51 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

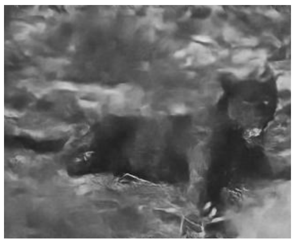

(a) Noisy ($\sigma = 90$)

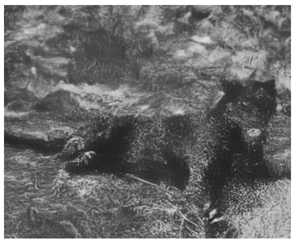

(b) DnCNN [19] 17.36

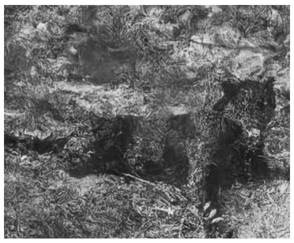

(c) N2N [9] 18.46

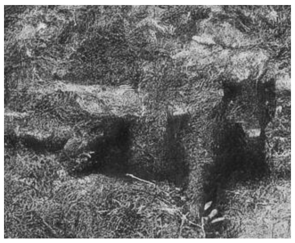

(d) Ground-truth

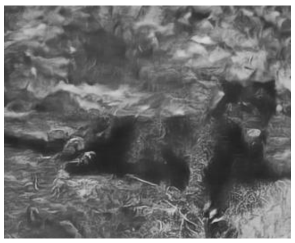

(e) DnCNN+SFM 18.65

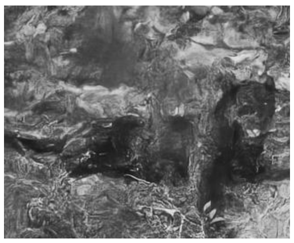

(f) N2N+SFM 19.43


(g) N3Net [11] 19.29


(h) MemNet [13] 18.03


(i) RIDNet [2] 17.62


(j) N3Net+SFM 19.93


(k) MemNet+SFM 19.39

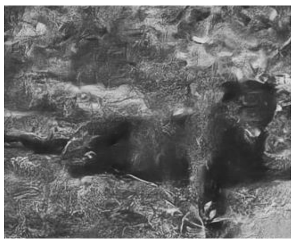
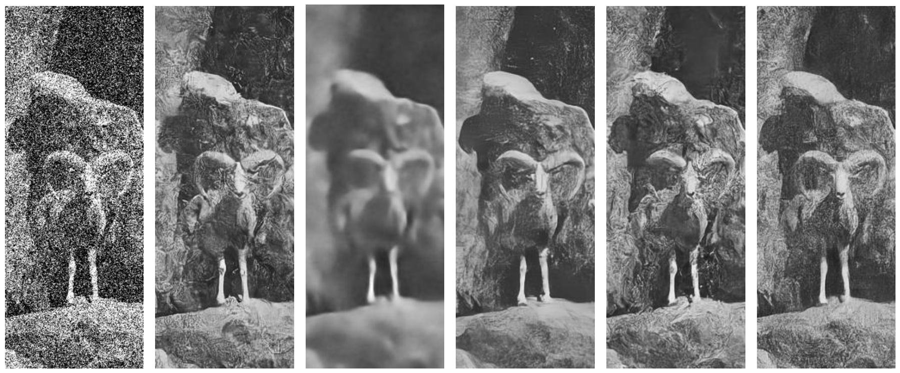

(l) RIDNet+SFM 19.44

Fig. 18: Denoising results with different methods (1st and 3rd row), and with the same method trained with our SFM (2nd and 4th row), for image 62 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).


(a) Noisy (σ = 80) (b) DnCNN [19] 19.95 (c) N2N [9] 20.67 (d) N3Net [11] 21.45 (e) MemNet [13] 20.72 (f) RIDNet [2] 20.35

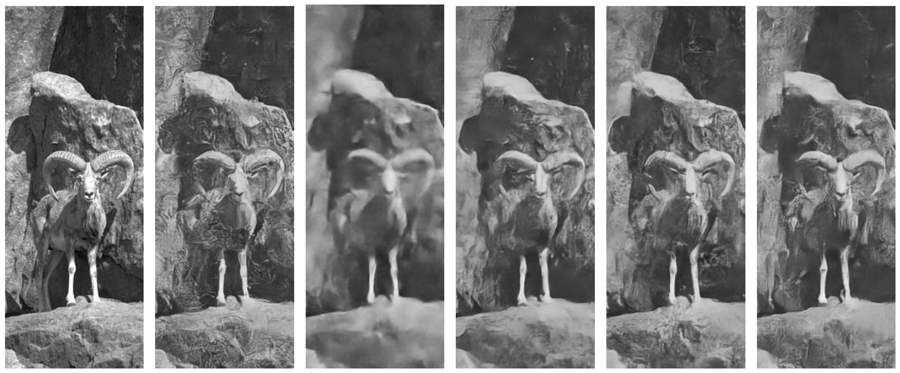
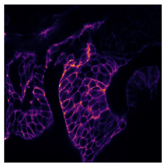
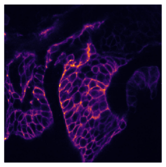
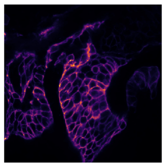
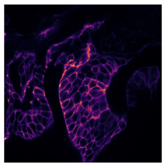

(g) Ground-truth (h) DnCNN+SFM 20.99 (i) N2N+SFM 21.14 (j) N3Net+SFM 21.89 (k) MemNet+SFM 21.72 (l) RIDNet+SFM 21.71

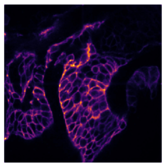
Fig. 19: Denoising results with different methods (top row), and with the same method trained with our SFM (bottom row), for image 63 of the BSD68 benchmark. We also show the PSNR values of every denoised result (in dB).

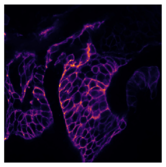

(a) Noisy (16 raw avg)

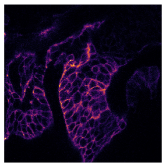

(b) N2S [3]

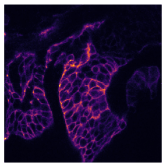

(c) N2N [9]

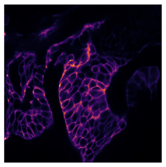

(d) GT

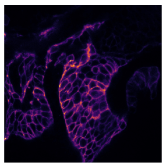

(e) N2S + SFM

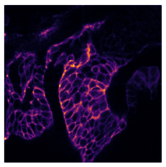

(f) N2N + SFM

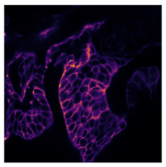

(g) Noisy (4 raw avg)

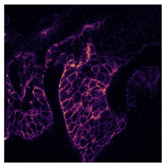

(h) N2S [3]

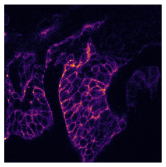

(i) N2N [9]

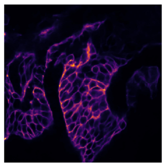

(j) GT

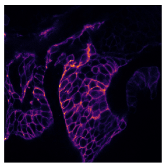

(k) N2S + SFM

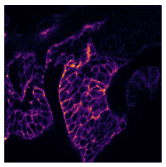

(l) N2N + SFM


(m) Noisy (one raw)


(n) N2S [3]


(o) N2N [9]


(p) GT


(q) N2S + SFM

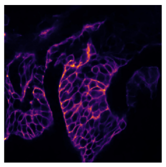
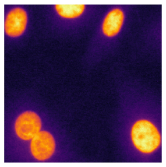
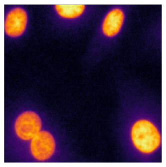
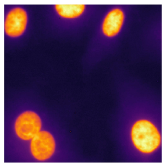
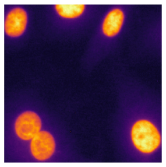

(r) N2N + SFM

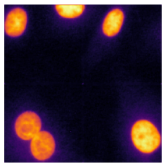
Fig. 20: Confocal microscopy sample results for denoising the noisy input image from the real fluorescence microscopy denoising dataset [20]. The first image (a) averages 16 raw images to obtain the noisy input, the second one (g) averages 4 raw images, and the last one (m) is directly a raw image. The 'ground-truth' images are estimated by averaging 50 raw images [20].

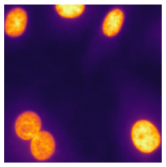

(a) Noisy (16 raw avg)

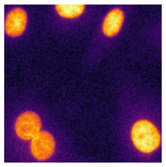

(b) N2S [3]

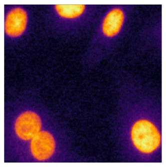

(c) N2N [9]

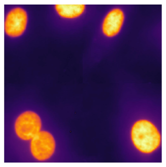

(d) GT

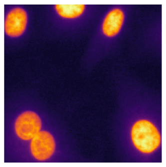

(e) N2S + SFM

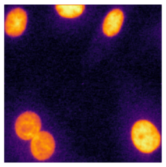

(f) N2N + SFM

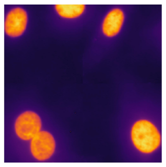

(g) Noisy (4 raw avg)

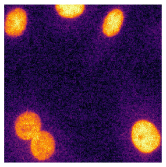

(h) N2S [3]

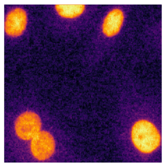

(i) N2N [9]

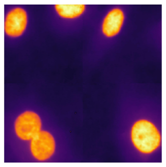

(j) GT

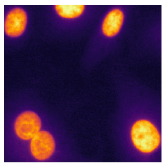

(k) N2S + SFM

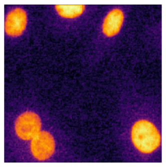

(l) N2N + SFM


(m) Noisy (one raw)


(n) N2S [3]


(o) N2N [9]


(p) GT


(q) N2S + SFM

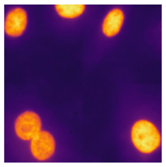
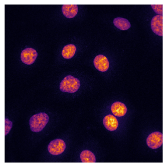
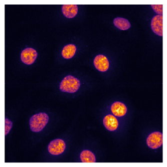
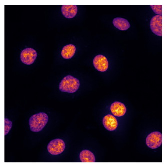
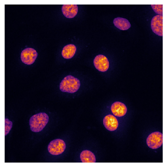

(r) N2N + SFM

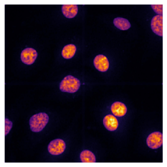
Fig. 21: Widefield microscopy sample results for denoising the noisy input image from the real fluorescence microscopy denoising dataset [20]. The first image (a) averages 16 raw images to obtain the noisy input, the second one (g) averages 4 raw images, and the last one (m) is directly a raw image. The ‘ground-truth’ images are estimated by averaging 50 raw images [20].

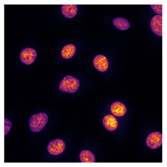

(a) Noisy (16 raw avg)

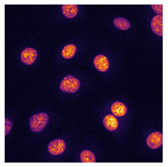

(b) N2S [3]

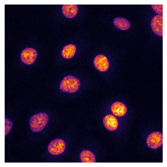

(c) N2N [9]

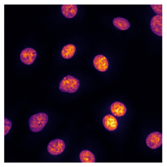

(d) GT

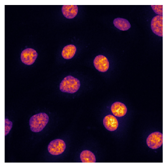

(e) N2S + SFM

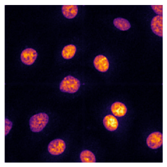

(f) N2N + SFM

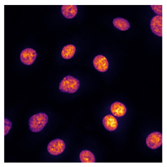

(g) Noisy (4 raw avg)

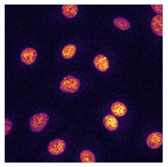

(h) N2S [3]

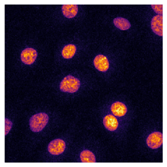

(i) N2N [9]

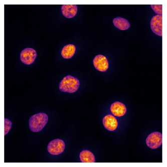

(j) GT

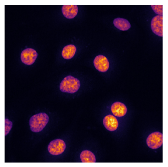

(k) N2S + SFM

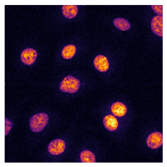

(l) N2N + SFM


(m) Noisy (one raw)


(n) N2S [3]


(o) N2N [9]


(p) GT


(q) N2S + SFM

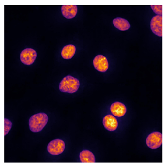

(r) N2N + SFM

Fig. 22: Two-photon microscopy sample results for denoising the noisy input image from the real fluorescence microscopy denoising dataset [20]. The first image (a) averages 16 raw images to obtain the noisy input, the second one (g) averages 4 raw images, and the last one (m) is directly a raw image. The 'ground-truth' images are estimated by averaging 50 raw images [20].