Imitating the Functionality of Image-to-Image Models Using a Single Example

Supplementary Material

A. More results for stealing biological image translation models

In Figs. S1-S2 we show examples for stealing virtual staining models for translating H&E images into Masson Trichrome staining and PAS staining. The average PSNR we obtain over the testing samples is 35.58dB and 34.69dB for the Masson Trichrome staining and PAS staining, respectively. In Figs. S3-S4 we show additional results for stealing biological models, with different architectures for the imitating model g_θ. We can see that overall the results are similar and the effect of the architecture is quite negligible. As part of a controlled experiment, we show an example of stealing a model for translating actin staining to nuclear staining (Bio-imageing). Using the ZeroCostDL4Mic toolbox [38], we trained two U-Net models for this task, one with an MSE loss and the other with a GAN loss and a perceptual loss (following the pix2pix method [46]). We then experimented with stealing those models with a 512×512 single example. The results are summarized in Tab. S1 We can see that the MSE-based model is easier to steal. However this model has lower perceptual quality than the Pix2Pix model, which was trained with perceptual and GAN losses in addition to the MSE loss. We provide further discussion regarding the loss with which the original model was trained in Sec. I. In Fig. S5, we present visual examples. As can be seen, the PSNR (w.r.t. the GT) achieved by our models is quite close to that of the original models.

Method	PSNR (f, g_θ)	PSNR (f,GT)	PSNR (g_θ,GT)
MSE	34.56	26.15	24.58
pix2pix	28.56	25.13	23.86

Table S1. **Stealing Biological image to image translation models.** The "Method" column specifies the loss with which the original model had been trained, where pix2pix means MSE + perceptual + adversarial losses. The MSE-based model is easier to steal.

B. Performance evaluation procedure

In Table 1, we report the imitation evaluation for the *non-blind restoration* and *blind restoration* scenarios. In each row we specified the loss that was used for training the original model (column "Loss"), the architectures of the original model f and of our model g_θ (column "Architecture"), the dataset (synthetically degraded) from which the single example and the test samples were extracted (column "Test set"). In the last two columns, we report the PSNR between the outputs of the original model and of our model for different sizes of the single examples. The evaluation procedure is as follows. We randomly choose a single example from the test set, crop it to either 128×128 or 320×480 or 512×512, train our model, and compute the PSNR on the rest of the test set. We repeat this 10 times and report the average PSNR. For example, in the first row in Tab. 1, we extracted a single example from the degraded dataset of BSD100, trained our model (randomly initialized SRCNN), and compared the outputs of the original model and our model on the remaining 99 degraded images from the degraded dataset of BSD100. In Sec. C.1-D.2 we elaborate on the exact datasets and degradations that were used in our experiments.

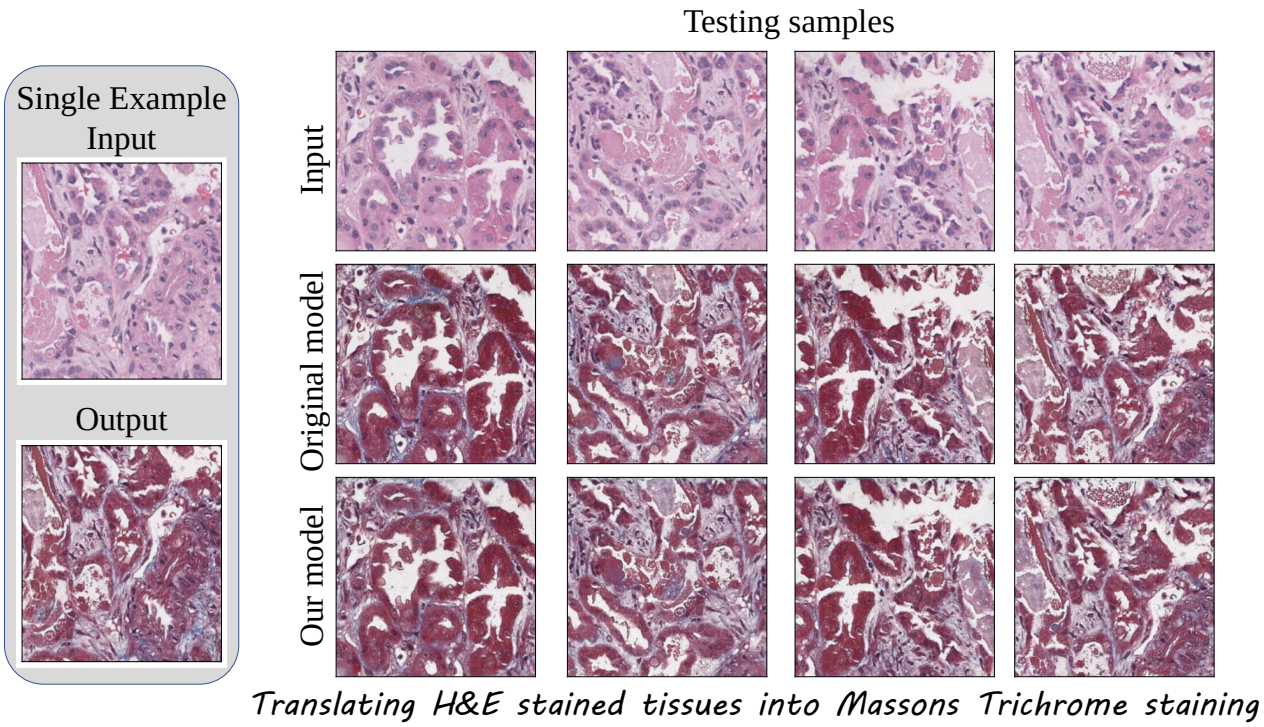

Figure S1. **More results for stealing biological image translation models.** Here we show stealing of a model for translating H&E stained images into Massons Trichrome staining. At the left (gray background) we show the single example used to steal the model and at the right, we compare the outputs of our model and the original model on test samples.

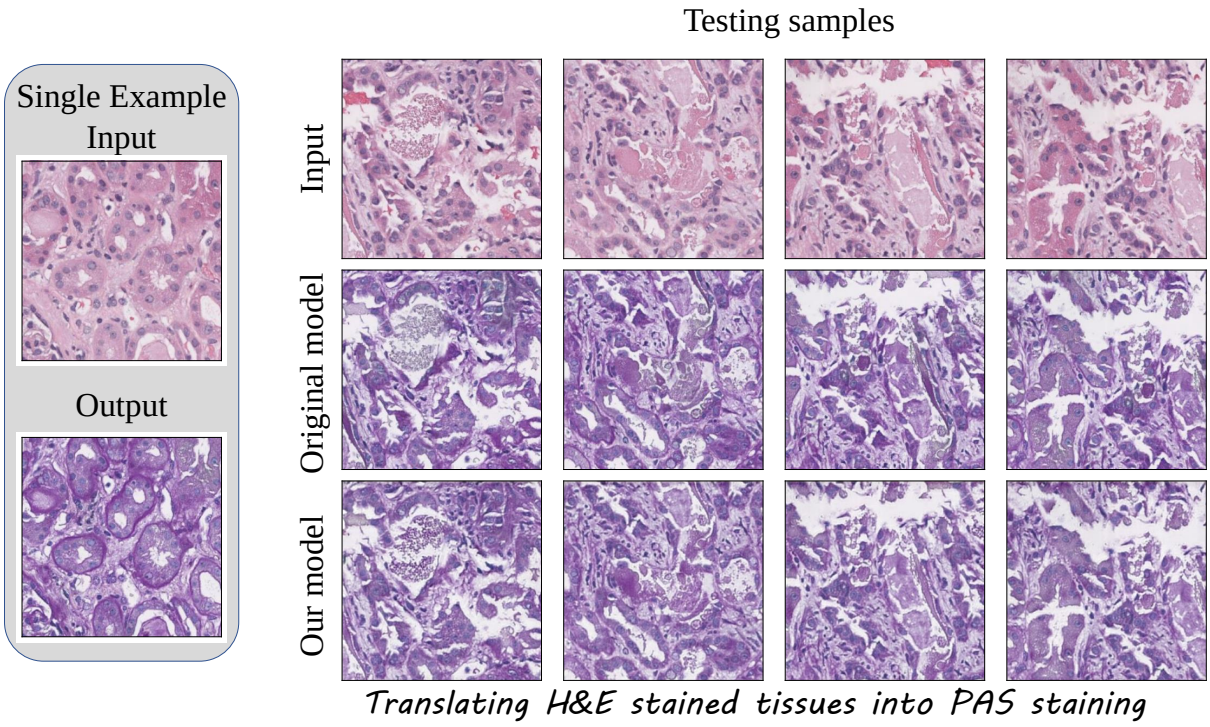

Figure S2. **More results for stealing biological image translation models.** Here we show stealing of a model for translating H&E stained images into PAS staining. At the left (gray background) we show the single example used to steal the model and at the right, we compare the outputs of our model and the original model on test samples.

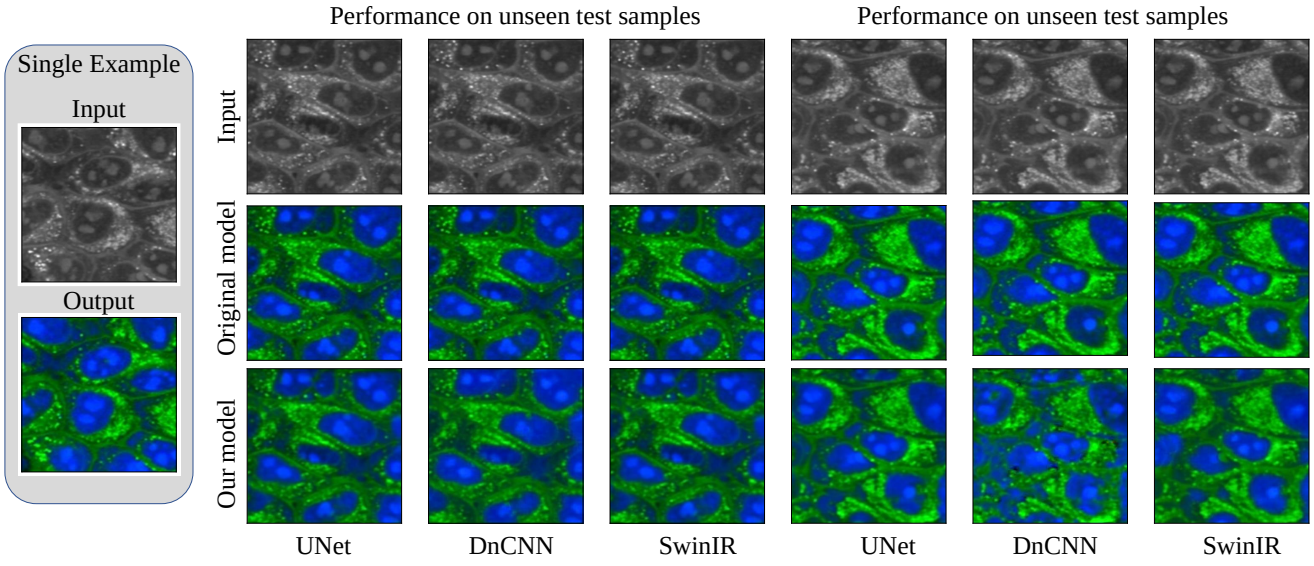

Figure S3. **More results for stealing a model for spectrally resolving femto-stimulated Raman scattering images.** Here we use several different model architectures for the imitating model g_θ. As can be seen, the architecture has no significant effect on the performance. At the left (gray background) we show the single example used to steal the model and at the right, we compare the outputs of our model and the original model on test samples.

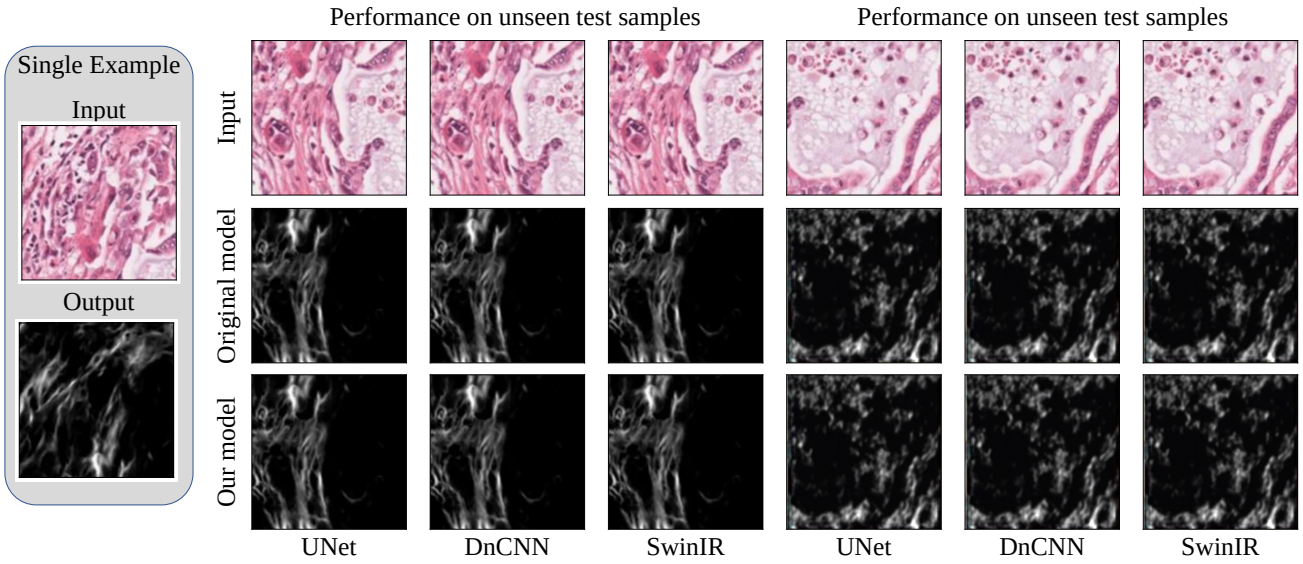

Figure S4. **More results for stealing a model for predicting collagen fibers from H&E stained images.** Here again we see that g_θ has no significant effect on the performance. At the left (gray background) we show the single example used to steal the model and at the right, we compare the outputs of our model and the original model on test samples.

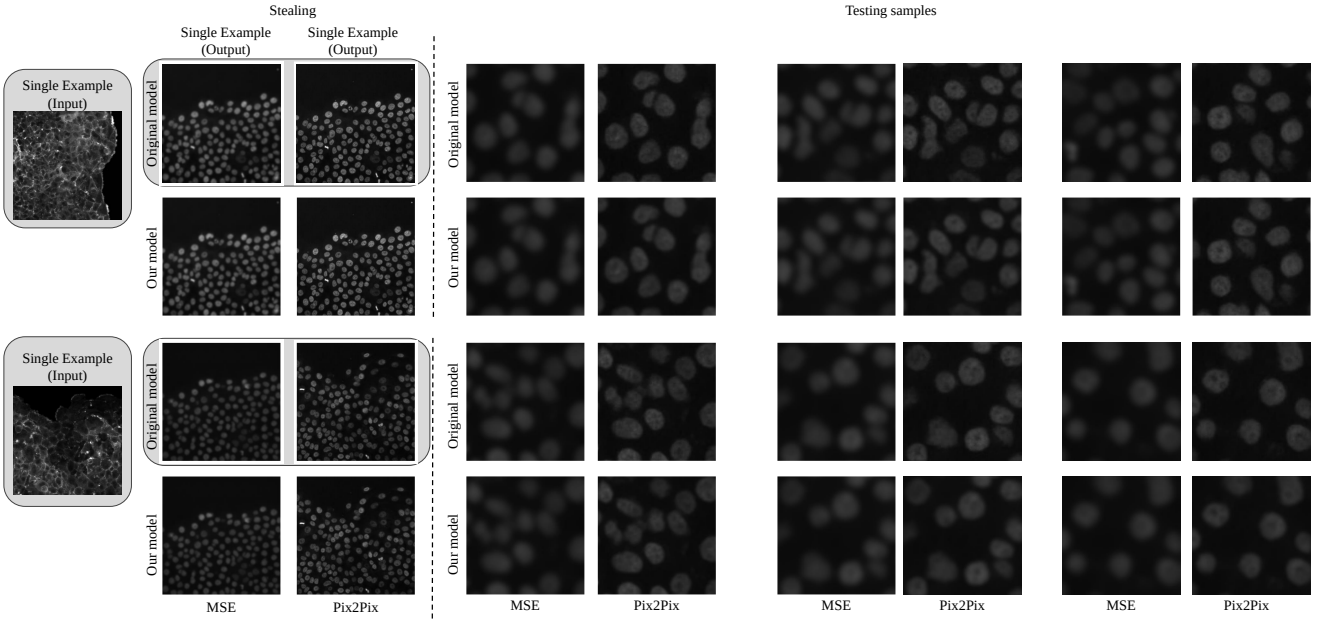

Figure S5. **Biological image-to-image translation for virtual staining.** Here we show two examples for stealing U-Net models trained with MSE loss (MSE) and with additional perceptual and GAN losses (Pix2Pix). On the left we show the input related to the single example with its output obtained by the original model (gray background). We also show the output of our imitating model on this input. Since the model was trained on that example, the outputs are similar. On the right, we present zoomed-in test images. We can see that for both examples, our model generalizes well to unseen test images.

C. Datasets and degradations

We now elaborate on the exact datasets and degradations that were used in each task for evaluating the stealing performance for the *non-blind restoration* and *blind restoration* scenarios.

C.1. Super-resolution

In the *non-blind restoration* scenario, we steal models that were trained for $4\times$ super-resolution with bicubic downsampling, and test on such images as well. The single examples and the testing images we used here are the bicubic down-sampled images of the DIV2K [2], Urban100 [11] and BSD100 [18] datasets. In Figs. S6-S8, we show several visual examples.

In the *blind restoration* scenario, we tested the models on the "unknown" downsampled dataset within DIV2K and as for the Urban100 and BSD100 datasets, the images were synthetically degraded with the downsampling pipeline provided by [44] which involves convolution with isotropic Gaussian kernels, random down-sampling by either nearest, bilinear or bicubic followed by adding a Gaussian noise with noise level chosen uniformly at random from the range $\sigma \in [1, 10]$. As can be seen under the "blind PSNR" category of Tab. 1, models can still be stolen successfully in this case, at least when the size of the single example is 512×512.

C.2. Denoising

In the *non-blind restoration* scenario, we used models trained on a noise level of STD $= 25$ and tested our models on noisy images with the same noise level. The single example and the testing images we used here are taken from the CBSD68 [31] and Kodak [17] datasets. As can be seen under the "denoising" category in Tab. 1, all models can be successfully stolen in this setting, except for SCUNetG on CBSD68. This seems to be related to its high perceptual quality. In general, models trained with an adversarial loss are often harder to steal (see discussion in Sec. I). In Fig. S9, we show visual examples for denoising on the Kodak24 dataset. Note how despite the dissimilarity between the single example and the test image (a face), we hardly observe any difference in the results produced by the original and by our models. In the *blind restoration* scenario, the images in the dataset are randomly noised with noise levels in the range of $\sigma \in [5, 55]$.

C.3. Defocus Deblurring

In the *non-blind restoration* scenario, the single example and the test images are blurred by a Gaussian kernel width, $\sigma_k = 5$ and contaminated by additive Gaussian noise with $\sigma_n = 0.1$. Here we re-train the Uformer and Restormer architectures on DIV2K dataset with that degradation. Then, we steal these models. We randomly chose the single examples from the synthetically blurred BSD dataset and tested on the remaining images in the dataset (*e.g.* 99). As mentioned, we repeat this for 10 times and report the average PSNR. In Fig. S10 we show visual examples for defocus deblurring within the setting of *non-blind restoration*. In the *blind restoration* scenario, we steal and test on images blurred by a Gaussian kernel with width chosen uniformly at random from the range $\sigma_k \in [3, 13]$. The additive Gaussian noise level remains constant as in the *non-blind restoration* scenario (*i.e.* $\sigma_n = 0.1$). Also in this case, we re-train the Uformer and Restormer architectures on DIV2K dataset with this degradation and test our model on the synthesized blurred BSD dataset. In Tables 1 we see that in this setting, stealing is typically successful.

D. Additional tasks of blind restoration

Table S2 reports experiments on additional blind restoration tasks.

D.1. Motion Deblurring

Motion deblurring is considered a blind restoration problem, since in practice, the motion blur in different images is almost always different (*e.g.* different direction and speed of the camera and of moving objects). Therefore, here we have only the blind setup. Following the DeblurGAN [14] evaluation method, we use the YOLO object detection dataset [27] in the *blind restoration* scenario.

D.2. Deraining

Deraining is also considered a blind problem. Since even in synthetic scenarios, the "rain" streak direction and its strength can vary significantly between images. Nevertheless, as can be seen in Tab. S2, we successfully steal the Restormer model when the single example and the test images are both taken from the Rain100L/H [41] or Test2800 [8] datasets. In Fig. S13 we show visual examples of stealing CycleGAN and Restormer for deraining on Rain100L dataset.

Task	Loss	Architecture	Test set	PSNR (f, g_θ) Size of single example image		
				128^2	320×480	512^2
Motion deblurring	L_2	DeblurGAN	YOLO	29.58	31.11	**36.49**
		DeblurGAN-V2	YOLO	30.05	32.70	**36.88**
	Charbonnier	Uformer	YOLO	31.12	33.46	**38.23**
	L_1	Restormer	YOLO	32.22	**34.28**	**37.01**
Blind deraining	L_1	Restormer	Rain100L	29.16	32.08	**35.91**
		Restormer	Rain100H	26.54	27.23	28.99
		Restormer	Test2800	28.05	31.28	**34.51**
	L_2 + Perceptual + Adversarial	CycleGAN	Rain100L	27.10	28.85	32.21
		CycleGAN	Rain100H	24.53	29.18	30.69
		CycleGAN	Test2800	27.79	28.98	32.47

Table S2. **Stealing models for *blind restoration*.** This table is complementary to the blind PSNR performance in Tab. 1. The evaluation protocol is as in Tab. 1. Here the single example suffers from a different degradation than the rest of the images in the test set (*e.g.* different rain streak direction). Nevertheless, stealing still succeeds with a large enough single example image.

E. blind restoration for severe degradations

We now introduce an additional blind restoration scenario, which we call *blind restoration for severe degradations*. In this case, the degradation is "real" (*i.e.* not synthetic) and varies drastically between images. Here, the performance of stealing with a single image is significantly reduced. However, we observe that when using more then a single example to train our model, the performance is improved. In Tab. S3 we report the PSNR as a function of the number of 128×128 single example images. Specifically, we use 1, 10, 20 or the entire dataset (*i.e.* "all") except for a small set that is excluded for testing. We now elaborate on this setting for super-reslution, denoising, defocus deblurring and motion deblurring.

E.1. Super-resolution

We steal and test on the RealSRset dataset [44] ("real SR" in Tab. S3). This dataset includes images from different domains (*e.g.* cartoon and natural images) and is therefore very challenging for stealing with a single example. However, as can be seen in the table, increasing the number of single examples helps. It is certainly possible that augmentation techniques could assist, but we keep this for future work.

E.2. Denoising

We steal and test on real noisy images taken from the RN15 [15], Real3 [43] and Real9 [43] datasets. Because of the small number of images in each of these datasets, we take our test set to be the union of all three datasets, resulting in 27 test images. As can be seen in Tab. S3 under "real denoising", despite the challenging setting, we manage to successfully steal DnCNN-B

Task	Loss	Architecture	Test set	PSNR(f, g_θ)			
				1	10	20	all
Real SR	L_1	SRCNN	RealSR	21.15	25.36	27.26	**35.87**
		EDSR	RealSR	24.68	26.89	29.56	**34.50**
	L_1 + Perc. + Adv.	SwinIR-G	RealSR	23.48	26.78	28.11	**34.89**
	L_2 + Perc. + Adv.	SRGAN	RealSR	22.38	26.57	32.54	**38.54**
Real denoising	L_2	DnCNN-B	RN15	25.69	27.56	**34.82**	**36.55**
		DnCNN-B	Real3+Real9	26.97	28.24	30.02	34.12
	L_1	SwinIR	RN15	27.11	29.67	32.58	**36.14**
		SwinIR	Real3+Real9	26.66	28.49	29.87	32.59
		Restormer	RN15	26.54	27.13	29.07	33.54
		Restormer	Real3+Real9	26.03	28.18	30.10	**34.56**
	L_1 + Perc. + Adv.	SCUNetG	RN15	25.69	26.08	29.87	32.54
		SCUNetG	Real3+Real9	29.54	33.54	**36.07**	**37.58**
Defocus Deblurring	L_1	Restormer	DPDD	27.65	29.07	31.55	**34.54**
	Charbonnier	Uformer	DPDD	26.54	28.56	29.09	**35.15**
Motion Deblurring	L_1	Restormer	GoPro	22.35	24.56	28.31	31.59
	Charbonnier	Uformer	GoPro)	26.11	27.35	29.79	32.54

Table S3. **Blind restoration for severe degradations.** In all experiments here, we train the model with training examples in size of 128×128 training examples. Using more training examples significantly improve the imitation performance.

f Arch.	g_θ Arch.	PSNR (f_θ, g_θ)
SRGAN	EDSR	32.86
SRCNN	EDSR	**41.46**
EDSR	SwinIR	**35.05**
EDSR	SRCNN	31.84
SwinIR	EDSR	**37.54**
SwinIR	SRGAN	29.84
SwinIR	SRCNN	**34.65**

Table S4. **Imitating super-resolution models with an unknown architecture.** We employ a single 320×480 input-output example, randomly chosen from BSD100, to imitate models using a mismatched student architecture. Note how even small models, like SwinIR which is based on attention modules can be used to imitate large models, like EDSR which is a fully convolutional model.

f Arch.	g_θ Arch.	PSNR (f_θ, g_θ)
DnCNN	SwinIR	**35.17**
SwinIR	DnCNN	31.87
SwinIR	SCUNet	**34.91**
Restormer	SwinIR	**35.88**
SCUNet	Restormer	**34.86**

Table S5. **Imitating denoising models with an unknown architecture.** We employ a single 320×480 input-output example, randomly chosen from CBSD68, to imitate models using a mismatched student architecture.

and SwinIR when training with a single example sampled from RN15, and SCUNetG with a single example sampled from the Real3+Real9 dataset.

E.3. Defocus deblurring

We steal and test on the defocus deblurring dual pixel dataset (DPDD) [1]. Here, the task is to remove blur from a pair of images captured by a dual-pixel (DP) sensor. The two images are referred to as DP sub-aperture views (InputR and InputL in Fig. S11), and generate an all-in-focus image, which should be similar to a corresponding image that was captured with a small aperture. Please refer to [1] for more details. As can be seen in Tab. S3, here we successfully steal Restormer and Uformer.

E.4. Motion Deblurring

We steal and test on the GoPro [20] dataset, which is characterized by faster motion and very different degradation for each image, which is thus more challenging for stealing (see Tab. S3). In Fig. S12 we show an example of stealing the model with 1, 10, 20, and 1050 randomly chosen example. We can see that enlarging the number of training examples significantly improve the results.

F. Effect of g_θ's architecture

In Tab. S4-S5 we examine the imitation performance in case of mismatched architectures and observe that imitation is still successful. In Figs S3-S4, we show visual examples of stealing biological models with different architectures and, again, observe that the effect of the architecture is negligible, as already mentioned.

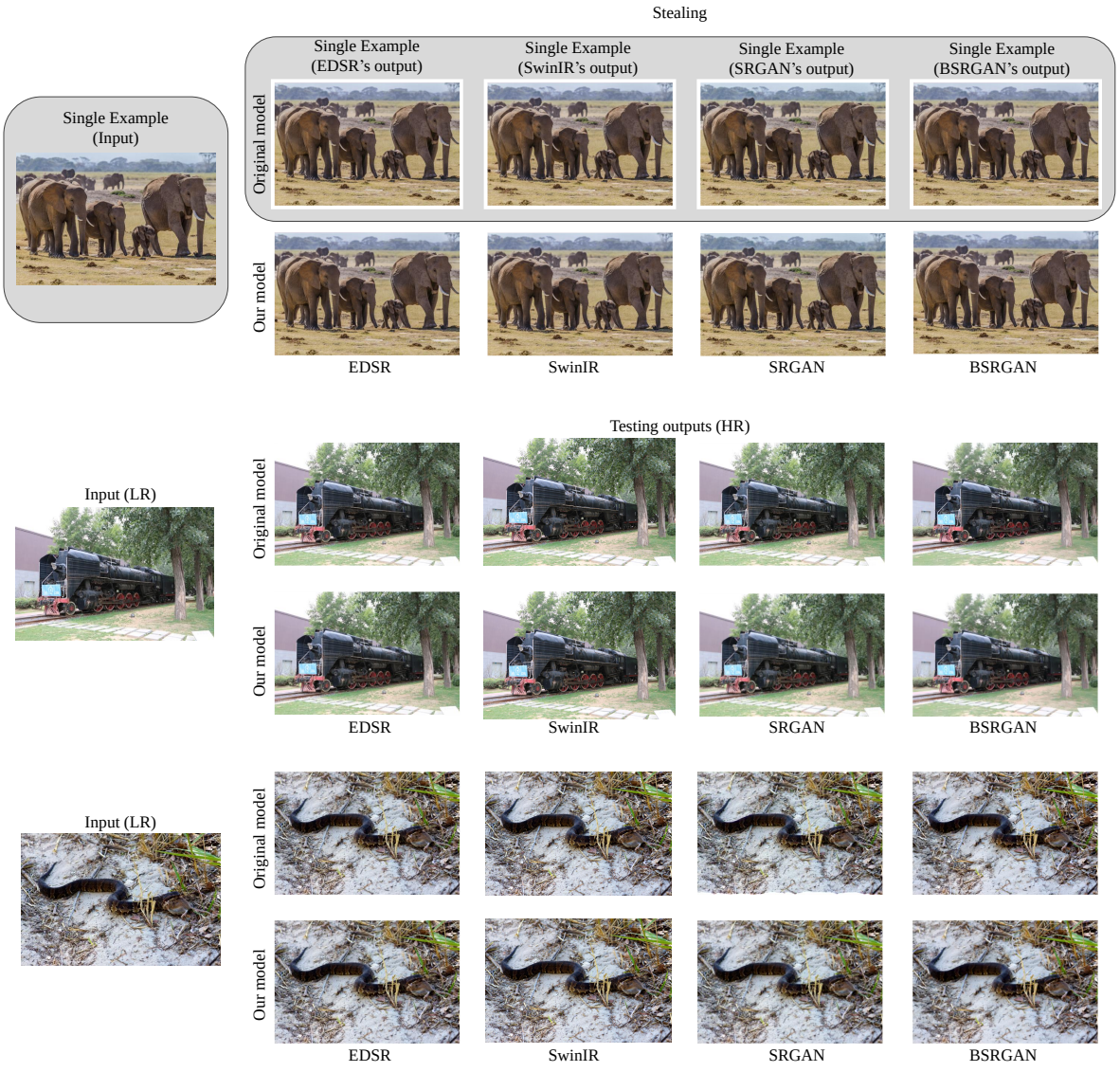

Figure S6. **Super resolution on the DIV2K dataset.** Visual examples of stealing different super resolution models where the single example (on the left) and the test images are from the DIV2K dataset.

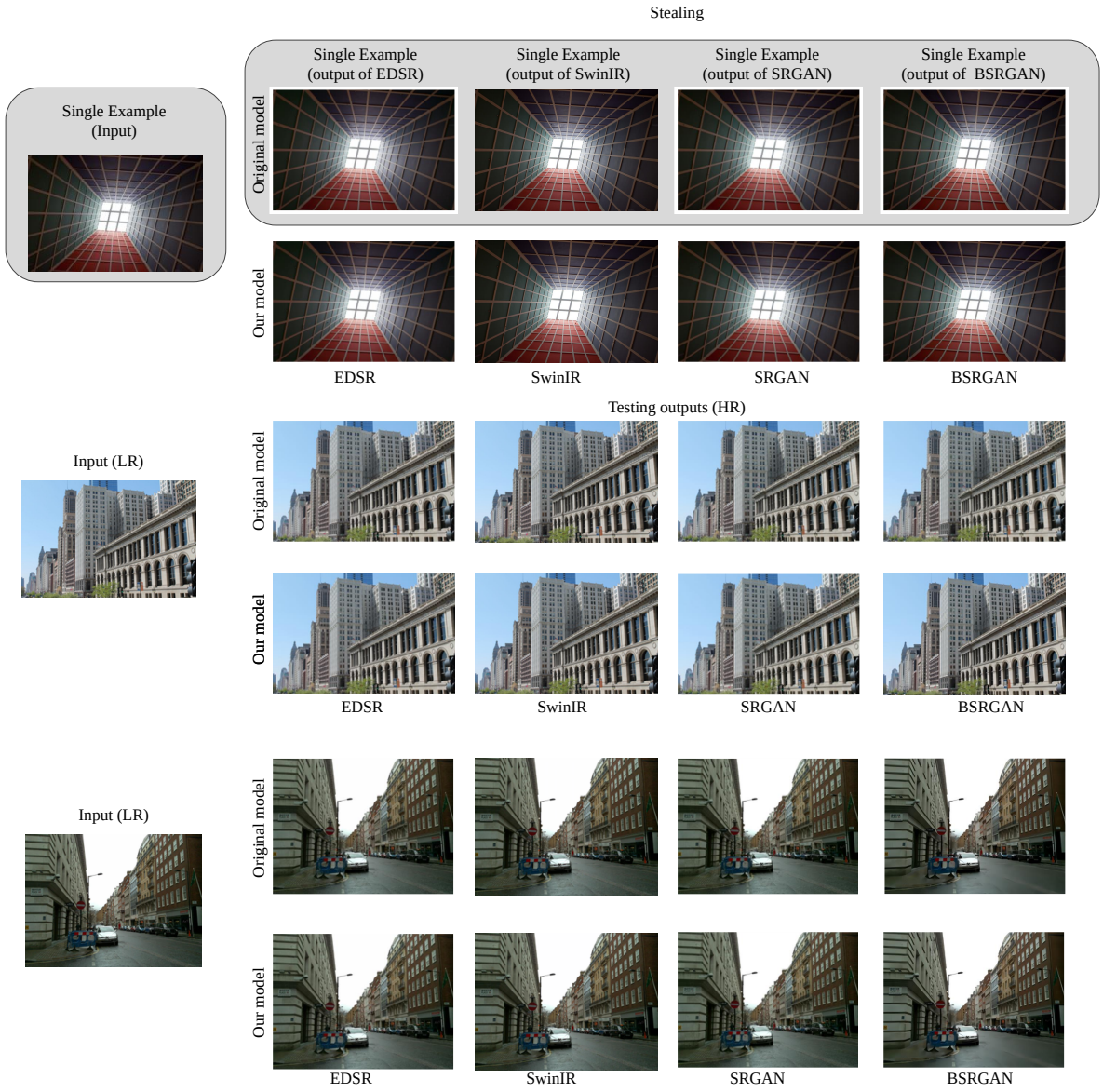

Figure S7. **Super resolution on the URBAN100 dataset.** Visual examples of stealing different super resolution models where the single example (gray background) and the testing images are from the URBAN100 dataset.

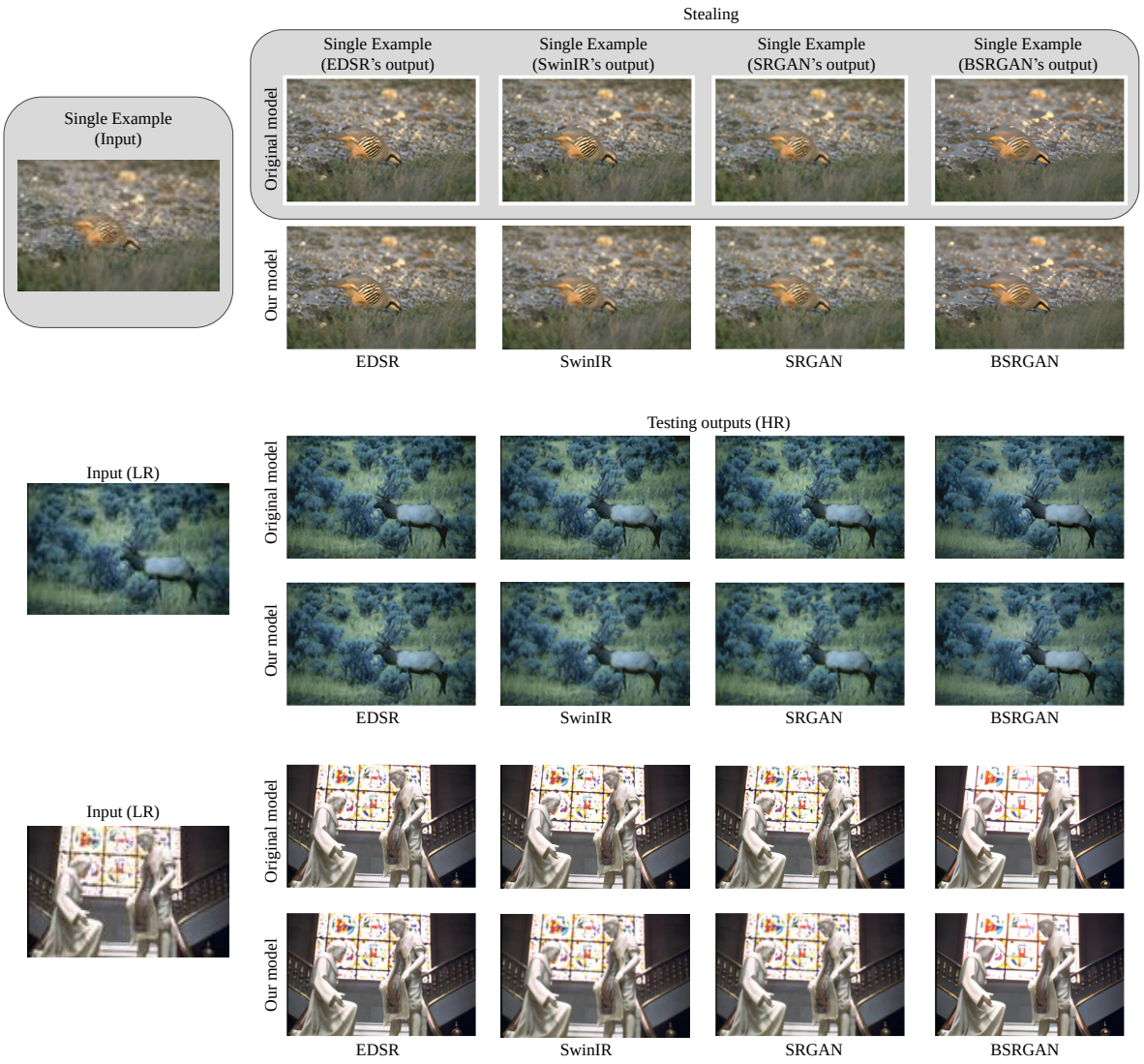

Figure S8. **Super resolution on the BSD100 dataset.** Visual examples of stealing different super resolution models where the single example (gray background) and the testing images are from the BSD100 dataset.

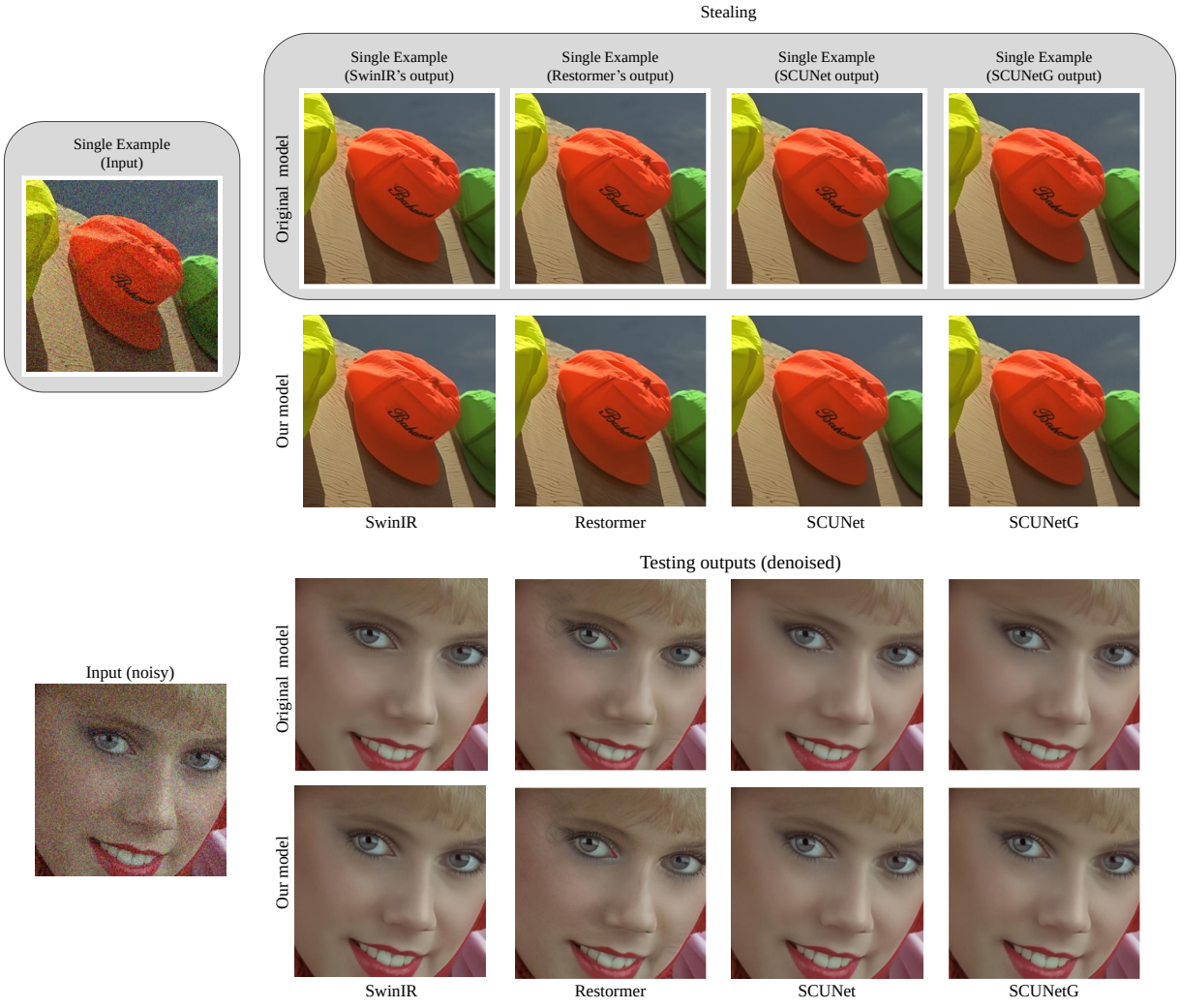

Figure S9. **Denoising on the Kodak24 dataset.** Visual examples of stealing different denoising models where the single example (gray background) and the testing images are from the BSD100 dataset.

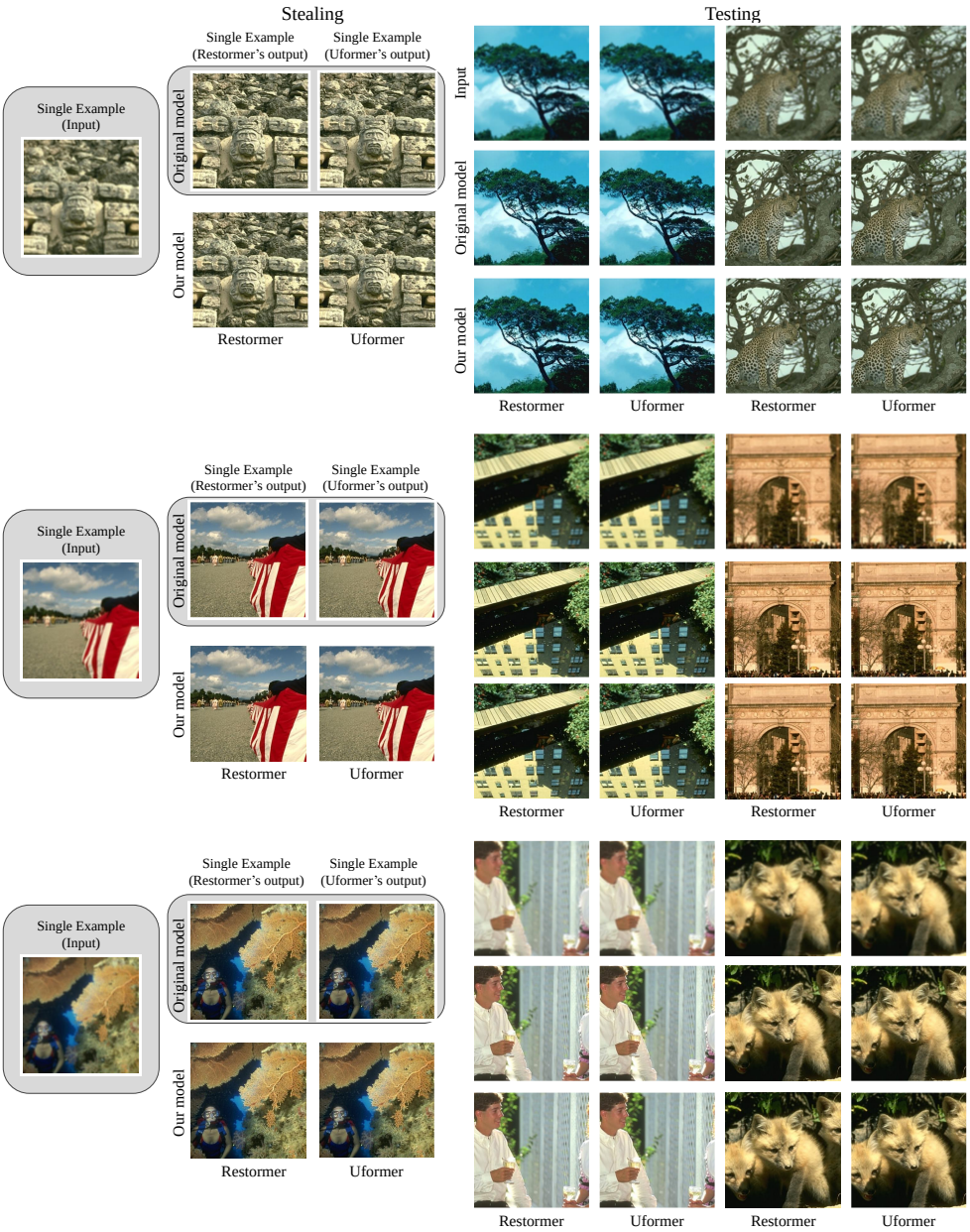

Figure S10. **Defocus deblurring on the synthetically blurred BSD100 dataset** Visual examples of stealing different defocus deblurring models where the single example (gray background) and the testing images are from the BSD100 dataset.

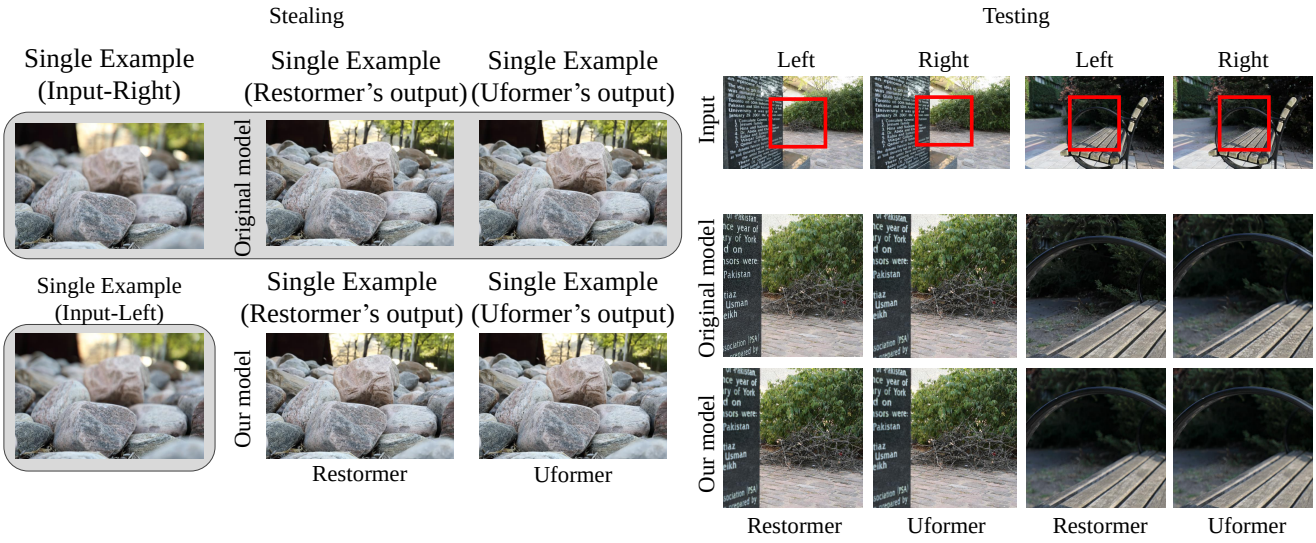

Figure S11. **Defocus deblurring on real blurry images from the DPDD dataset for the *blind restoration* scenario.**

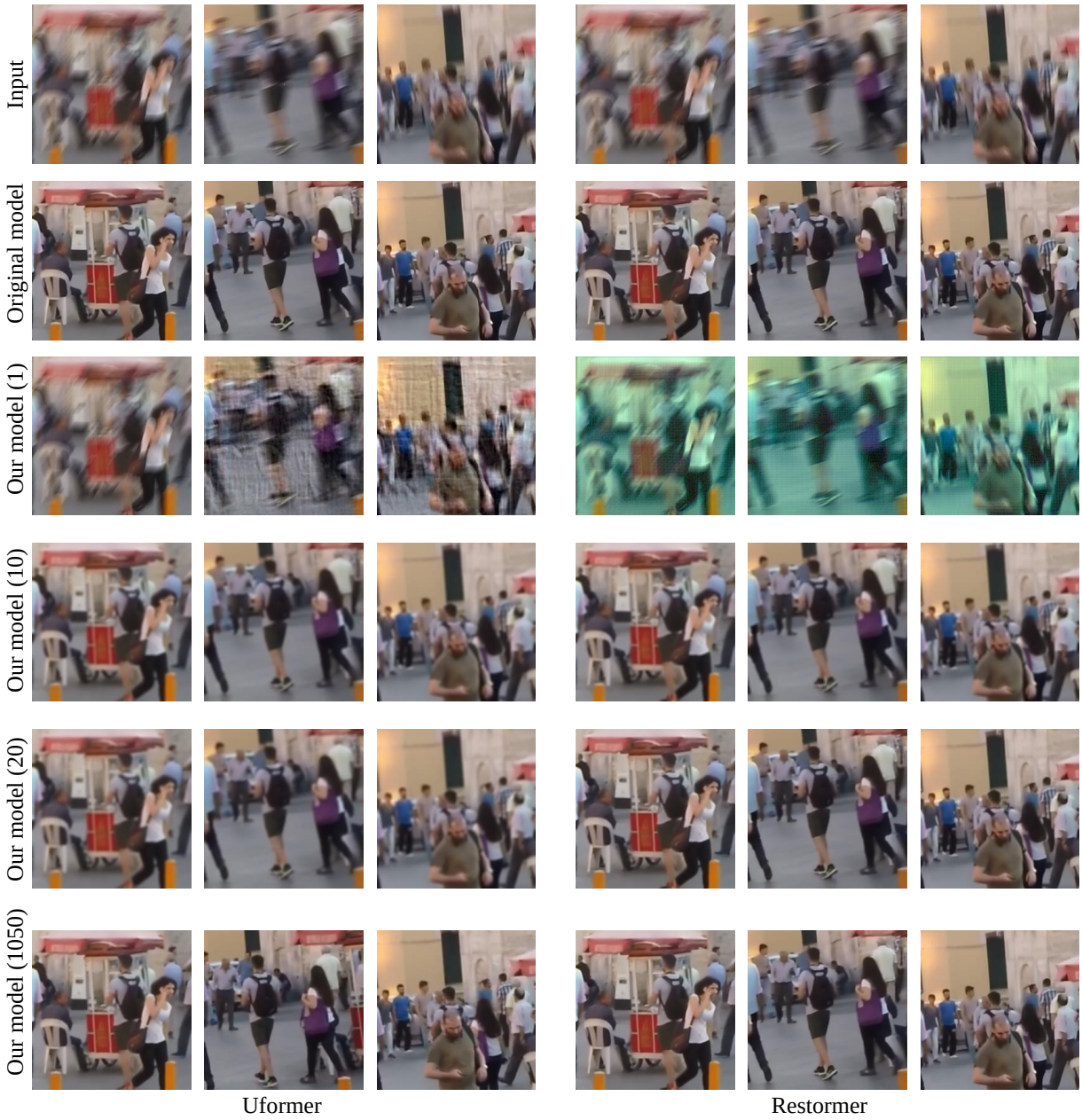

Figure S12. **Blind motion deblurring on GoPro.** The GoPro dataset is the most challenging among our experiments. This is because of the significant variations in the degradation between different images. Here we show visual example of stealing the Uformer and Restormer models using a single example (3^{rd} row), 10 examples (4^{th} row), 20 examples (5^{th} row) and 1050 examples (6^{th} row). We can see that it is impossible to steal the model with a single example. But, when enlarging the number of examples, even with 10 examples, we already see a significant improvement.

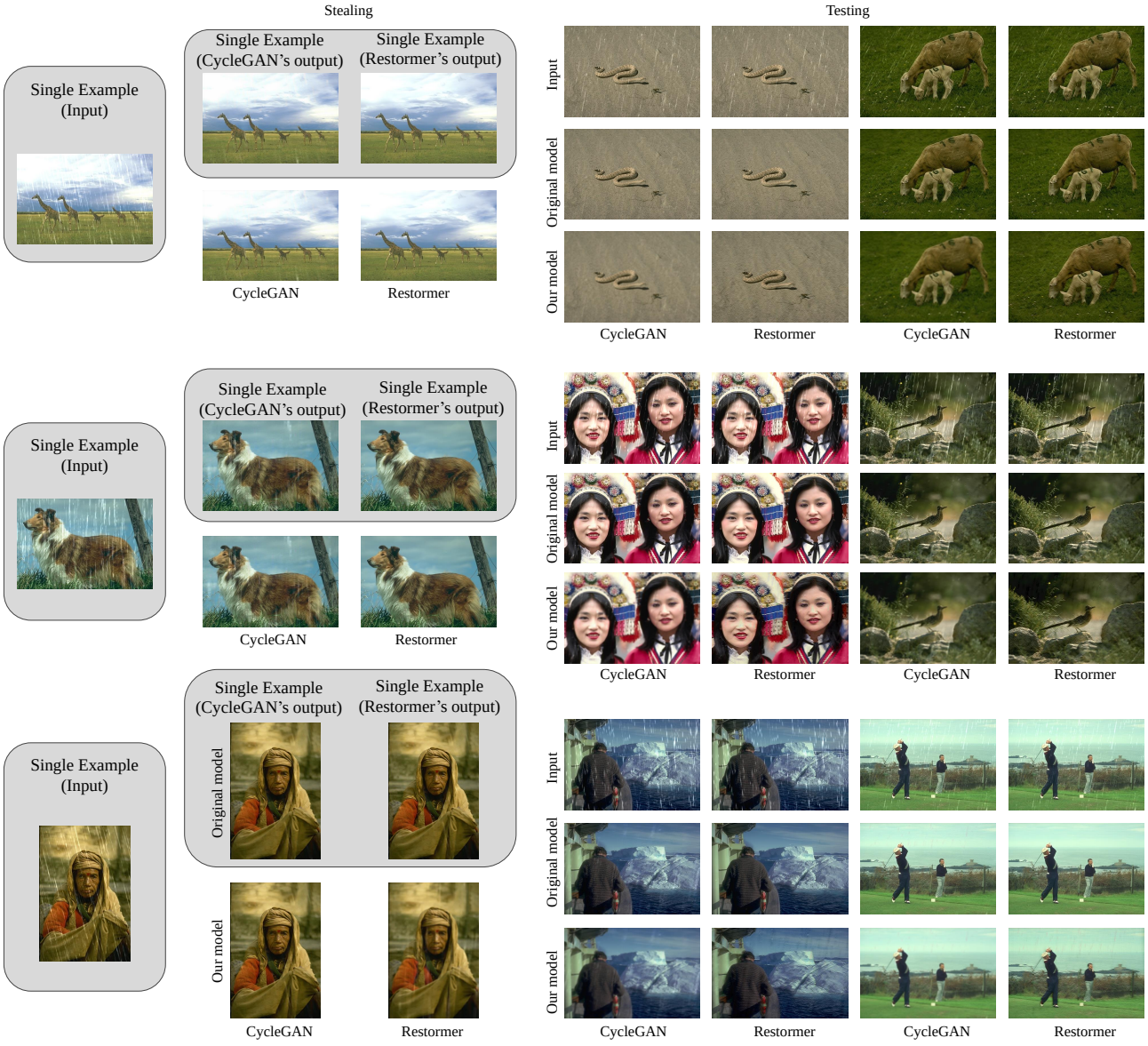

Figure S13. **Deraining on Rain100L dataset.** Here we show stealing performance for CycleGAN and Restormer on the Rain100L dataset. This task is within the *blind restoration* category since the direction of the rain steaks and their intensity is varies between the single example and the test images. However we still manage to steal both models and produce visually plausible results.

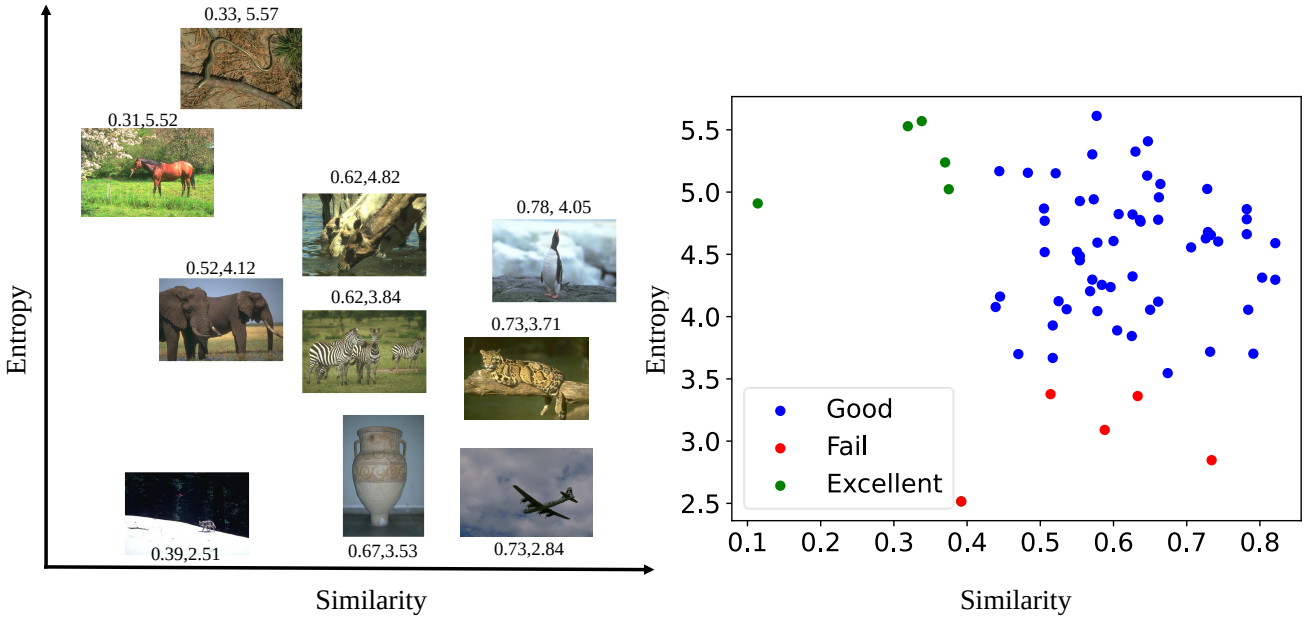

Figure S14. **Similarity vs. Entropy of the single example.** Left: here we show few visual examples of images corresponding to different similarity and entropy measures. Right:, we demonstrate stealing performance of SCUNet denoiser on CBSD68 dataset, as function of different entropy and similarity values. We observe that the best stealing performance is obtained where the single example has high entropy and low similarity However stealing is successful for most of single example choices.

single example	Tested	PSNR (f, g_θ)
BSD100	DIV2K	**36.47**
DIV2K	BSD100	**37.05**
Urban100	DIV2K	**36.54**
DIV2K	Urban100	**38.15**

Table S6. **Effect of single example choice on EDSR for Super-resolution.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

G. Effect of the choice of the single-example

Since it serves as many small training patches, the single example image has some effect on the imitation success. In most cases, where the single example is taken from the same domain as the test images, the results are satisfactory. However, to find whether specific characteristics of the single example are responsible for the success of imitation, we define two relevant measures. The first is related to the entropy of the image. We divide the image to local patches and calculate the minimum number of bits which are required for encoding each patch. Then, we take the average. The second measure is based on the normalized cross correlation (NCC) which is related to the similarity between patches in the images. Specifically, we calculate the NCC between all patches and average the values in the upper triangular part of the resulting matrix. In Fig. S14 on the left, we show a few examples of images with different values of entropy and similarity. On the right, we examine the stealing performance where the single example image is characterized by different values of entropy and similarity. As can be deduced from the plot, in 63 out of 68 cases (CBSD68 dataset) stealing of the SCUNet denoiser succeeded (*e.g.* PSNR>34.15). The best results ("excellent") are when the similarity measure is low and the Entropy is high. In these cases, the PSNR between the outputs of the original model and our model is above $38.58dB$ (corresponding to 3 RMSE<3).

In Tables S6-S13 we demonstrate the effect of taking the single example from one dataset and evaluating the stealing on another. As we also observed when testing with SRCNN for super-resolution in the main paper, the imitation performance is slightly affected by domain shifts, especially when the URBAN100 dataset is involved.

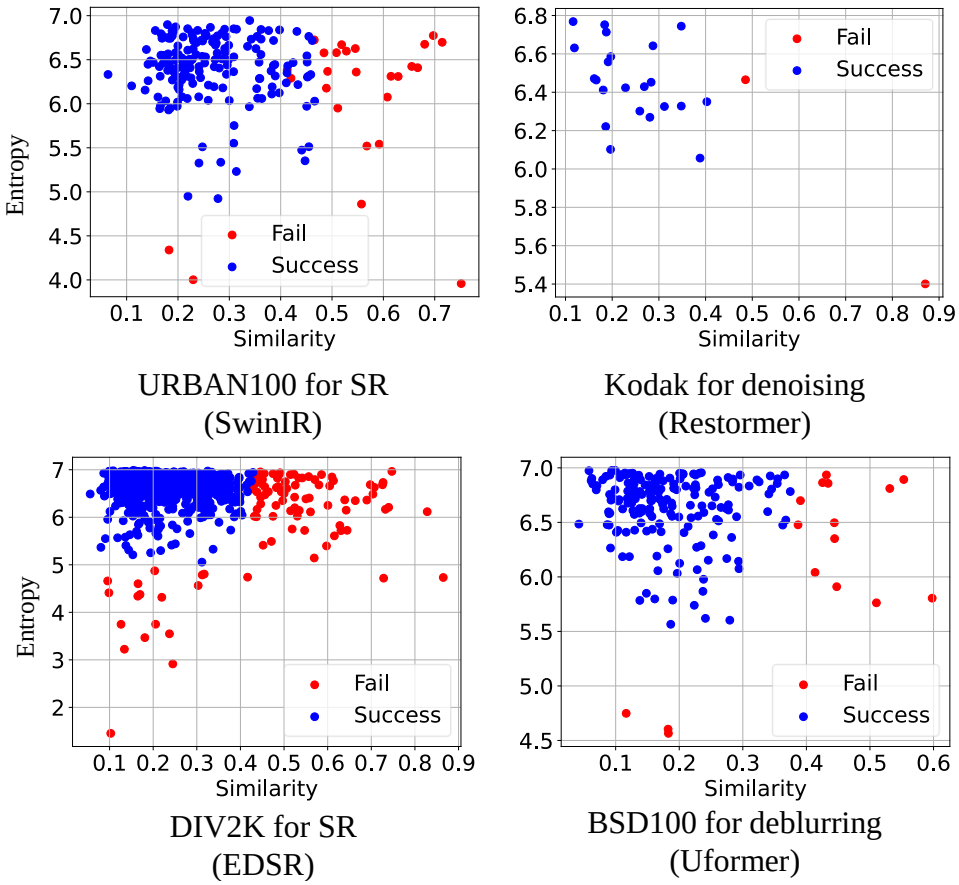

Figure S15. **Similarity vs. Entropy of the single example.** Across all examined models and tasks, whenever the entropy is high and the similarity is low, imitation is successful.

single example	Tested	PSNR (f, g_θ)
BSD100	DIV2K	**37.88**
DIV2K	BSD100	**37.03**
Urban100	DIV2K	**37.72**
DIV2K	Urban100	**38.36**

Table S7. **Effect of single example choice on SwinIR for Super-resolution.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

single example	Tested	PSNR (f, g_θ)
BSD100	DIV2K	**36.15**
DIV2K	BSD100	**36.24**
Urban100	DIV2K	33.87
DIV2K	Urban100	**35.44**

Table S8. **Effect of single example choice on SRGAN for Super-resolution.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

H. Effective receptive field

In order to measure the receptive field, we calculate the difference between the outputs of each model for two input images that are completely identical, apart for a small change in the center pixel. In that way, we actually manage to observe what is

single example	Tested	PSNR (f, g_θ)
BSD100	DIV2K	**36.99**
DIV2K	BSD100	**36.01**
Urban100	DIV2K	34.18
DIV2K	Urban100	**36.65**

Table S9. **Effect of single example choice on BSRGAN for Super-resolution.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

single example	Tested	PSNR (f, g_θ)
CBSD68	Kodak24	**36.45**
Kodak24	CBSD68	**35.95**

Table S10. **Effect of single example choice on DnCNN for denoising.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

single example	Tested	PSNR (f, g_θ)
CBSD68	Kodak24	**36.48**
Kodak24	CBSD68	**37.75**

Table S11. **Effect of single example choice on SwinIR for denoising.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

single example	Tested	PSNR (f, g_θ)
CBSD68	Kodak24	**36.78**
Kodak24	CBSD68	**37.97**

Table S12. **Effect of single example choice on Restormer for denoising.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

single example	Tested	PSNR (f, g_θ)
CBSD68	Kodak24	**36.11**
Kodak24	CBSD68	**34.52**

Table S13. **Effect of single example choice on SCUNet for denoising.** At each row, we choose a 320×480 single example from one datastet and test the model on images from a different dataset.

the impact of the change in the center pixel in the input on the output of the model.

In Figs. S16-S17, we show the receptive field of the original models and of our models for different tasks and scenarios.

I. The loss with which the original model was trained

Some image restoration models are trained only with a distortion loss (*e.g.* L^1 or L^2). These models achieve high PSNR values, but their restored images tend to be somewhat blurry. Other models incorporate GAN and perceptual losses in addition to the distortion loss. These models achieve lower PSNR values, but produce images with better perceptual quality. We observe that generally, stealing models that were trained only with a distortion loss is easier than stealing models that were trained with additional perceptual and GAN losses. This highlights a fundamental difference between those two families of models. In particular, it implies that perceptual models tend to operate quite differently on different images, so that observing their input-output relation for one image is insufficient for training a student model that performs well on new test images. This behavior was also observed for the bio-imageing model which we train once with MSE loss and once with MSE in addition to perceptual and GAN losses.

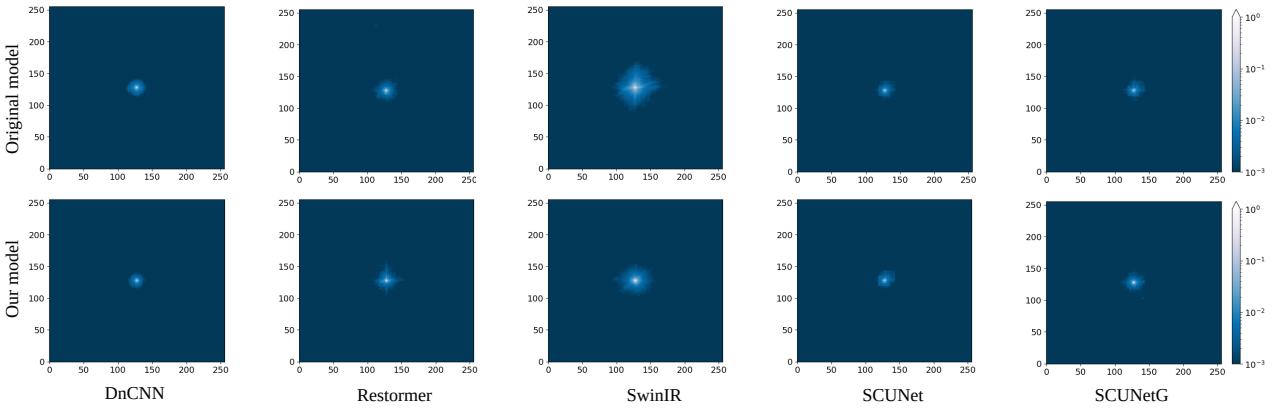

Figure S16. **Effective receptive field for denoisers in the *non blind restoration* scenario.** Here we compare between the effective receptive field of our model (bottom row) and the original model (top row). We observe that sometimes the effective receptive field of our model is smaller than that of the original model. We tend to believe that the reason for that inherits in the fact that it might be too complicated to study all global relations from a single image. Nevertheless, it does not impact the stealing performance.

J. Implementation details

J.1. Training

In all our experiments, we use an NVIDIA Quadro RTX 8000 GPU. Through training, we minimize (1) using the Adam optimizer [13] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for 2000 epochs. We use linear scheduling for the learning rate, starting with a learning rate of 10^{-3} and reducing it by a factor of 10 at epochs 1200 and 1800. In Table S15 we report the time it takes to train our models and the average GPU memory usage during the training. We focus on the non-blind scenario and on cases in which the architectures of f and g_θ are the same.

J.2. Code credits

We used the following shared code for stealing or re-training the original models:

- Restormer – https://github.com/swz30/Restormer.
- DnCNN – https://github.com/cszn/DnCNN
- SwinIR – https://github.com/JingyunLiang/SwinIR
- DeblurGAN – https://github.com/KupynOrest/DeblurGAN
- DeblurGAN-V2 – https://github.com/VITA-Group/DeblurGANv2
- SRGAN – https://github.com/Lornatang/SRGAN-PyTorch
- BSRGAN – https://github.com/cszn/BSRGAN
- SCUNet – https://github.com/cszn/SCUNet
- SRCNN – https://github.com/Lornatang/SRCNN-PyTorch
- EDSR – https://github.com/sanghyun-son/EDSR-PyTorch
- Uformer – https://github.com/ZhendongWang6/Uformer
- CycleGAN (deraining) – https://github.com/OaDsis/DerainCycleGAN
- CycleGAN (Virtual staining) – https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

K. Permission and rights

All used and presented images, which we extracted from published papers, are allowed for use under the terms of the Creative Commons CC-BY license. Here we share the permission and right pages of each paper For paper [19] please refer to: https://s100.copyright.com/AppDispatchServlet?publisherName=ELS&contentID=S016502702100306X&orderBeanReset=true. Papers [5], [7], [16] and [12] appeared in the Nature Communication journal; please see their common rights page here: https://creativecommons.org/licenses/by/4.0/ For the images we downloaded from the VISIOPHARM website, we obtained a written permission from the company.

Figure S18 illustrates this in the context of stealing the SCUNet denoising model. The airplane image in the first row is dominated by bluish-gray colors (the clouds and the airplane), having high similarity of 0.73 and low entropy of 2.84

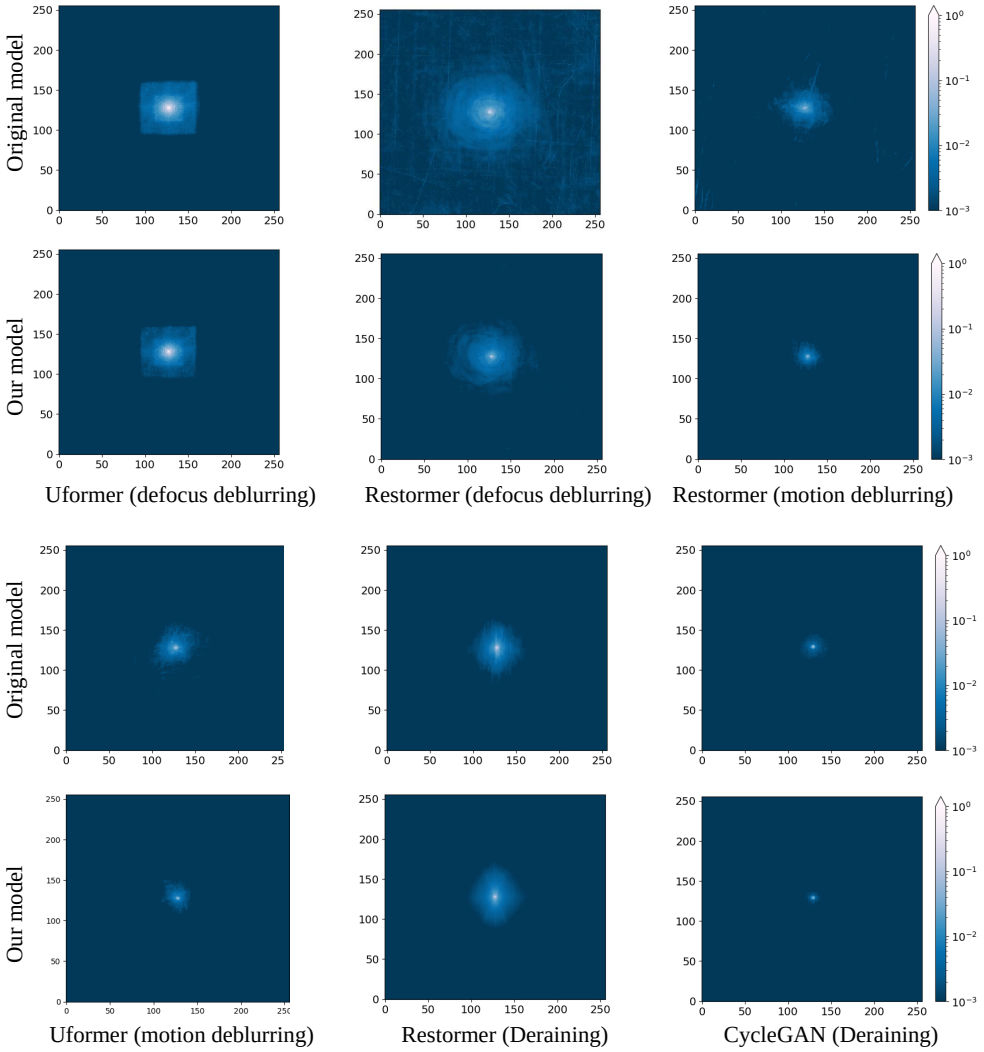

Figure S17. **Effective receptive field of image restoration models in the *blind restoreation* scenario.** Here the observation that the effective receptive field of our model is less spreads (smaller) then this of the original model, is even more dominant.

(See Fig. S14). In this case, our model fails to generalize to test images that contain other colors. The colors in the single example Zebras image in the second row are slightly more similar to the test images. We can see that in this case, the our model succeeds much better. In the third row, we did an interesting experiment and use a noisy constant image (with noise of STD 25, as the model expects). Here, the our model performs well on this image, but generalizes poorly to other images. Worth mentioning that this behavior is not precisely replicated in other models. However, what does in common is that if the training image sharing similar image statistics as the test images stealing always better. Also, better results obtained when the similarity and entropy measures are lower and higher, respectively. Please see Fig. S15.

L. The effectiveness of simple defense methods

We now briefly discuss two simple defenses against stealing attacks: (i) adding a visible watermark and (ii) applying JPEG compression to the images returned by the model. We note that these defenses may not always be tolerable in practical applications, but we leave the design of more involved techniques for future work.

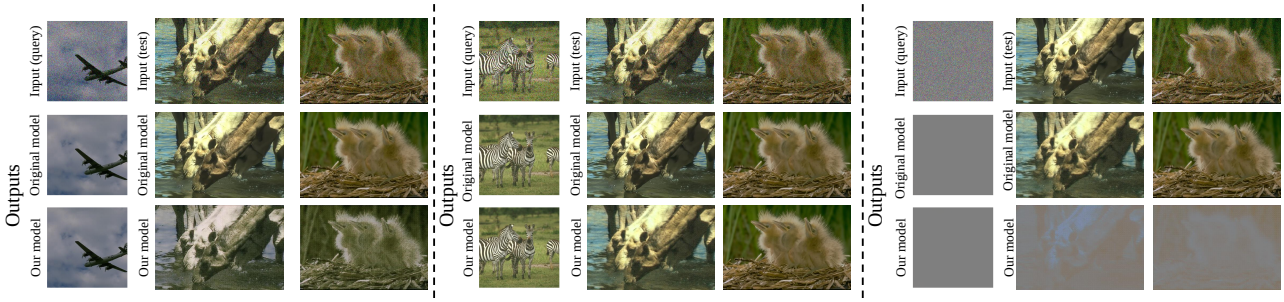

Figure S18. **Effect of the choice of the single example on SCUNet model for denoising.** Stealing is more successful when the single example is from the same domain as the test images. Moreover, many details (edges, smooth areas and colors) can significantly improve the stealing quality.

Watermarking. This defense works by adding to every output of the original model f a visible watermark (Figs. S19-S22), so that the attacker does not have direct access to $f(x)$, but rather only to Watermark$(f(x))$. We use these modified images in place of $f(x)$ for training the student model g_θ. In Table S17, column "WM", we report the stealing quality in this setting for the *non-blind restoration* scenario. As can be seen, the naive stealing attack (1) completely fails in this setting. However, watermarks can be easily detected [21] and masked in the stealing process. Column "Masking WM" reports the stealing

Task	f Arch. = g_θ Arch.	Dataset	single example size	Time [minutes]	Memory [MiB]
SR (bicubic kernel)	SRCNN	BSD100	128 × 128	3.55	85
	SRCNN	DIV2K	320 × 480	6.68	205
	SRCNN	Urban100	512 × 512	8.25	528
	EDSR	BSD100	128 × 128	4.21	1822
	EDSR	DIV2K	320 × 480	6.58	2200
	EDSR	Urban100	512 × 512	10.89	2366
	SwinIR	BSD100	128 × 128	8.17	8450
	SwinIR	DIV2K	320 × 480	16.58	12550
	SwinIR	Urban100	512 × 512	21.56	32548
	SRGAN	BSD100	128 × 128	5.87	515
	SRGAN	DIV2K	320 × 480	6.74	954
	SRGAN	Urban100	512 × 512	7.28	1082
	BSRGAN	BSD100	128 × 128	5.51	1520
	BSRGAN	DIV2K	320 × 480	7.89	2598
	BSRGAN	Urban100	512 × 512	12.77	4589
Denoising (STD = 25)	DnCNN	Kodak24	128 × 128	6.12	356
	DnCNN	Kodak24	320 × 480	8.11	708
	DnCNN	Kodak24	512 × 512	16.56	1450
	SwinIR	DIV2K	128 × 128	8.98	8950
	SwinIR	DIV2K	320 × 480	17.11	12520
	SwinIR	DIV2K	512 × 512	22.48	33518
	Restormer	Kodak24	128 × 128	12.15	16705
	Restormer	Kodak24	320 × 480	17.59	31522
	Restormer	Kodak24	512 × 512	22.22	36541
	SCUNet	Kodak24	128 × 128	4.33	2274
	SCUNet	Kodak24	320 × 480	7.33	3652
	SCUNet	Kodak24	512 × 512	20.15	18928
Defocus Deblurring (σ = 5)	Uformer	BSD	128 × 128	4.11	2585
	Uformer	BSD	320 × 480	10.25	6824
	Restormer	Kodak24	128 × 128	13.81	16915
	Restormer	Kodak24	320 × 480	18.11	32582

Table S14. **Time and memory performance for blind restoration scenario.** The time taken to train our model (the student) on a single single example is minutes and significantly shorter then training the original model from scratch.

quality with such masking. In this case, the PSNR is much higher than with the watermark (column "WM") and only slightly lower than in the baseline non-watermark setting (column "W/O"). We conclude that visible watermarks are not effective in preventing stealing, unless the watermark covers a large portion of the image.

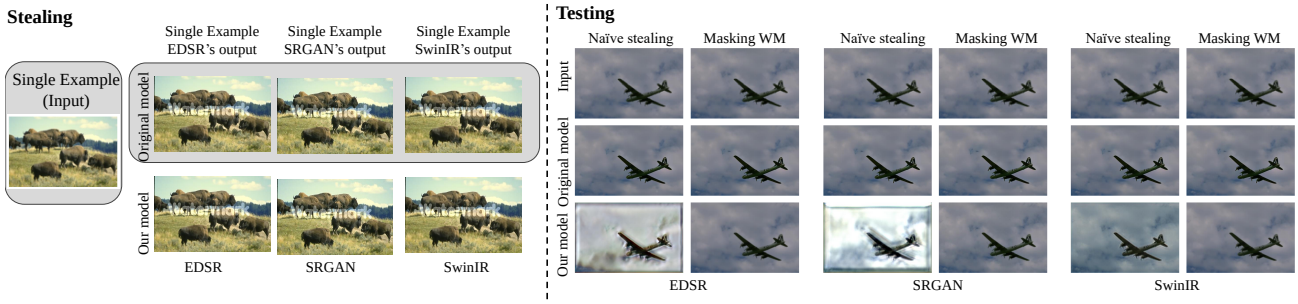

Figure S19. **Overcoming watermark defense.** Adding watermark to the output of the original model deteriorates the stealing performance (See Naive stealing). However, when applying the loss only outside the watermark, we can improve the stealing performance significantly. Here we show example for three models: EDSR, SRGAN and SwinIR. We can observe that SwinIR is relatively less vulnerable to watermark protection then the convolution-based models.

JPEG compression. We now briefly discuss a defense that works by applying JPEG compression [39] to the images returned by the model (Figs. S23-S26) so that the attacker does not have direct access to $f(x)$, but rather only to JPEG$(f(x))$. Table S16 reports the stealing quality for *non-blind restoration* models under two JPEG quality factors (QFs): 5 and 75. For

Task	f Arch. = g_θ Arch.	Dataset	single example size	Time [minutes]	Memory [MiB]
SR (bicubic kernel)	SRCNN	BSD100	128 × 128	3.55	85
	SRCNN	DIV2K	320 × 480	6.68	205
	SRCNN	Urban100	512 × 512	8.25	528
	EDSR	BSD100	128 × 128	4.21	1822
	EDSR	DIV2K	320 × 480	6.58	2200
	EDSR	Urban100	512 × 512	10.89	2366
	SwinIR	BSD100	128 × 128	8.17	8450
	SwinIR	DIV2K	320 × 480	16.58	12550
	SwinIR	Urban100	512 × 512	21.56	32548
	SRGAN	BSD100	128 × 128	5.87	515
	SRGAN	DIV2K	320 × 480	6.74	954
	SRGAN	Urban100	512 × 512	7.28	1082
	BSRGAN	BSD100	128 × 128	5.51	1520
	BSRGAN	DIV2K	320 × 480	7.89	2598
	BSRGAN	Urban100	512 × 512	12.77	4589
Denoising (STD = 25)	DnCNN	Kodak24	128 × 128	6.12	356
	DnCNN	Kodak24	320 × 480	8.11	708
	DnCNN	Kodak24	512 × 512	16.56	1450
	SwinIR	DIV2K	128 × 128	8.98	8950
	SwinIR	DIV2K	320 × 480	17.11	12520
	SwinIR	DIV2K	512 × 512	22.48	33518
	Restormer	Kodak24	128 × 128	12.15	16705
	Restormer	Kodak24	320 × 480	17.59	31522
	Restormer	Kodak24	512 × 512	22.22	36541
	SCUNet	Kodak24	128 × 128	4.33	2274
	SCUNet	Kodak24	320 × 480	7.33	3652
	SCUNet	Kodak24	512 × 512	20.15	18928
Defocus Deblurring (σ = 5)	Uformer	BSD	128 × 128	4.11	2585
	Uformer	BSD	320 × 480	10.25	6824
	Restormer	Kodak24	128 × 128	13.81	16915
	Restormer	Kodak24	320 × 480	18.11	32582

Table S15. **Time and memory performance for blind restoration scenario.** The time taken to train our model (the student) on a single single example is minutes and significantly shorter then training the original model from scratch.

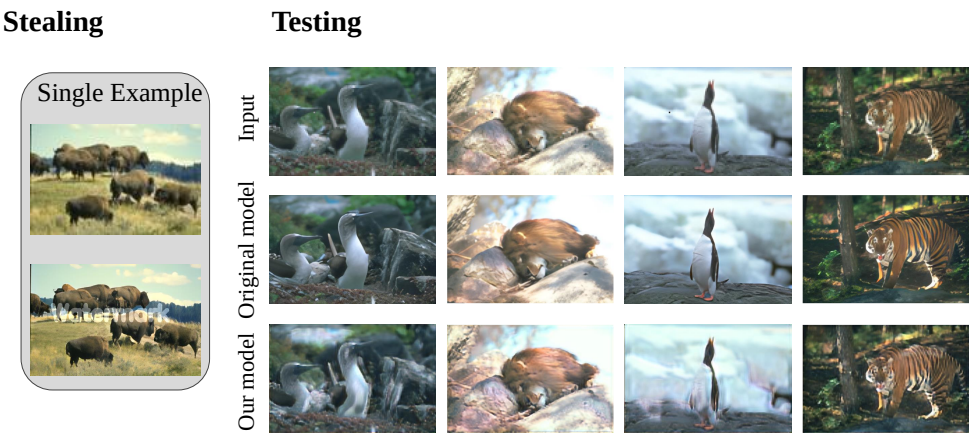

Figure S20. **Protecting the BSRGAN [44] super resolution model from stealing by adding a watermark.** Here we use a random chosen single example from the BSD100 dataset (left), and added watermark to the BSRGAN output. This pair was used for stealing the model. At the right we see that we fail to generalize on test images. The output images of our model suffer from changes in color and looks quite blurry.

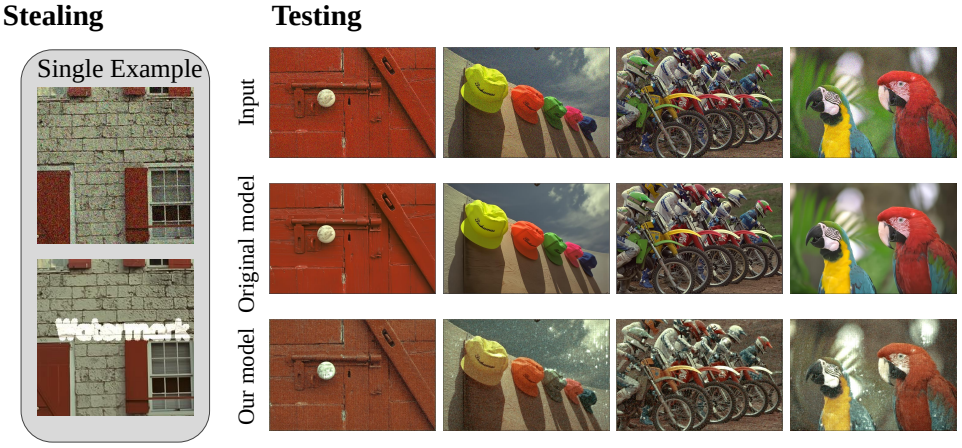

Figure S21. **Protecting the Restormer [42] denoising model from stealing by adding a watermark.** Here we use a random chosen single example from the Kodak dataset (left), and added watermark to the Restormer output. This pair was used for stealing the model. At the right we see that we fail to generalize on test images. See the color changes and hues added to the output images of our model.

QF $= 5$, the stealing completely fails. Yet, this comes at the cost of having the API present to the user unacceptably low-quality images. For QF $= 75$, which is a popular setting, the stealing quality is much better than with QF $= 5$, but often still not very good. We conclude that JPEG compression may be an effective defense against the naive stealing approach (1). Overcoming this defense may require modifying the objective in (1) into $\|\text{JPEG}(f(x)) - \text{JPEG}(g_\theta(x))\|^2$, and using differentiable approximations for the quantization within the JPEG method. We leave this for future work.

The results of potentially defending models from stealing using watermark and JPEG compression are summarized in Tables. S16-S17. In Tab. S16, we report the stealing performance when compressing the output of the model using JPEG compression with quality factor of 5 and 75. In Tab. S17, we report the stealing performance when adding a watermark to the output of the model. The watermark defense is effective, however, can be bypassed easily. The JPEG compression is also quite effective especially when the quality factor is low. In such a case, the images look very corrupted.

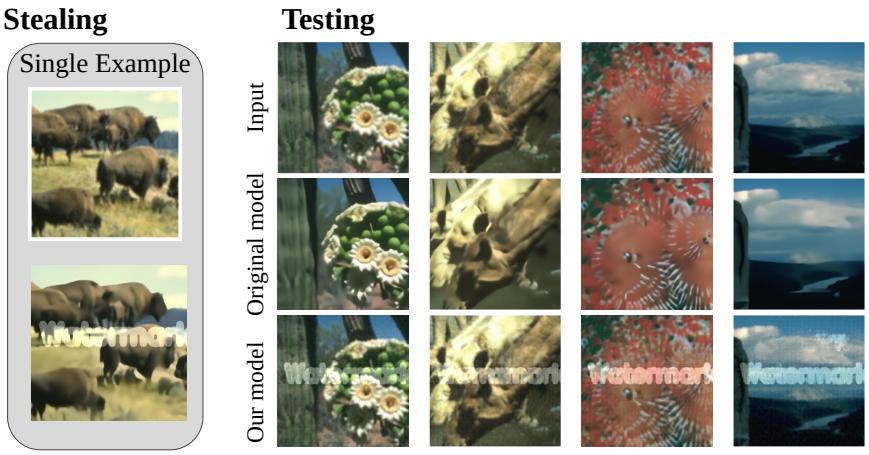

Figure S22. **Protecting the Uformer [40] defocus deblurring model from stealing by adding a watermark.** Here we use a random chosen single example from the BSD100 dataset (left), and added watermark to the Uformer output. This pair was used for stealing the model. At the right we see that we fail to generalize on test images. Through the stealing process, it looks like the Uformer studies also the watermark position and shape and transfer it to the testing images.

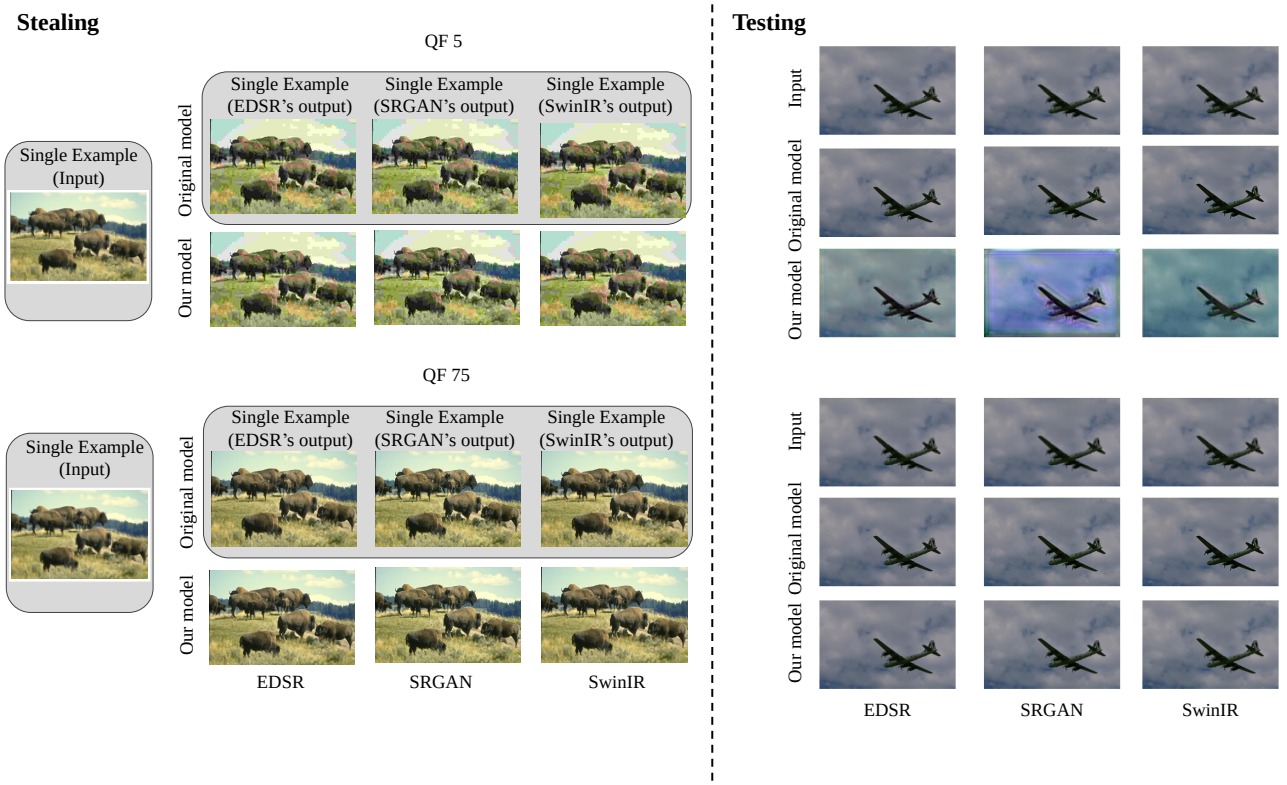

Figure S23. **Defending stealing with JPEG compression.** Similarly to Fig. S19, here examine the possibility of JPEG compression as a defense method. We see that when using the default quality factor (QF $= 75$), we can observe a relatively slight degradation in performance however, visually it is not very dominant. However, when we compress the output images with a very low QF (*e.g.* 5), we totally fail to steal the model

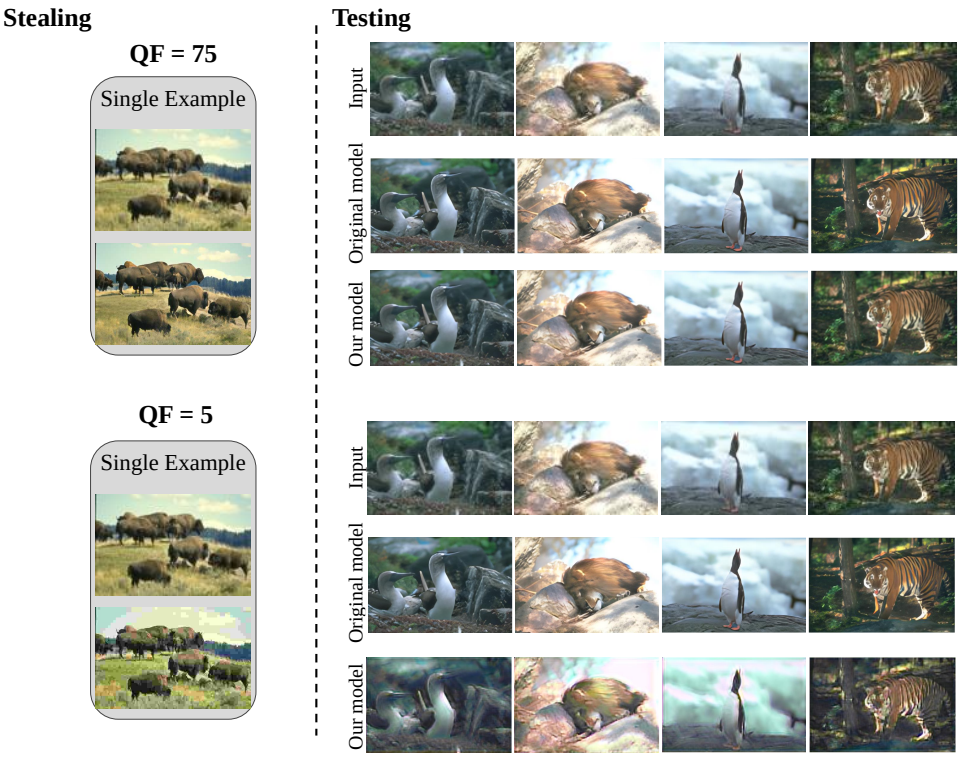

Figure S24. **Protecting the BSRGAN [44] super-resolution model from stealing using JPEG compression** We can see that quality factor of 75 (up) is not so effective as quality factor of 5 (bottom) which also cost a significant corruption of the image coming from the API.

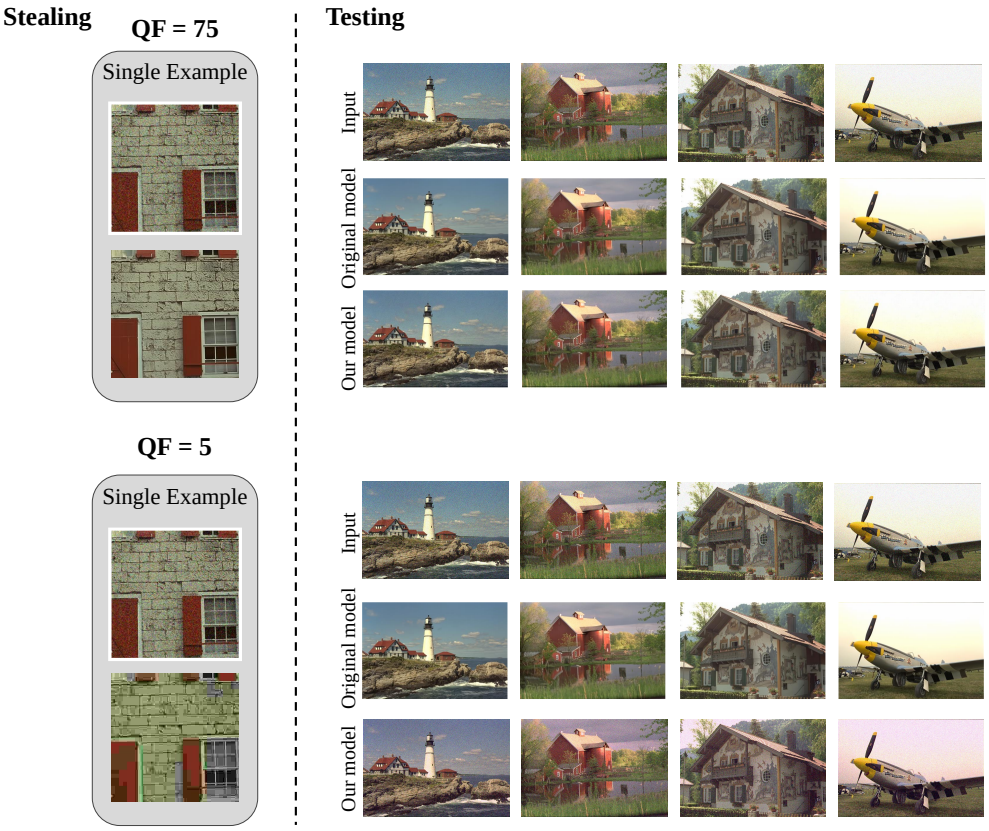

Figure S25. **Protecting the Restormer [42] denoising model from stealing using JPEG compression** We can see that quality factor of 75 (up) is not so effective as quality factor of 5 (bottom) which also cost a significant corruption of the image coming from the API.

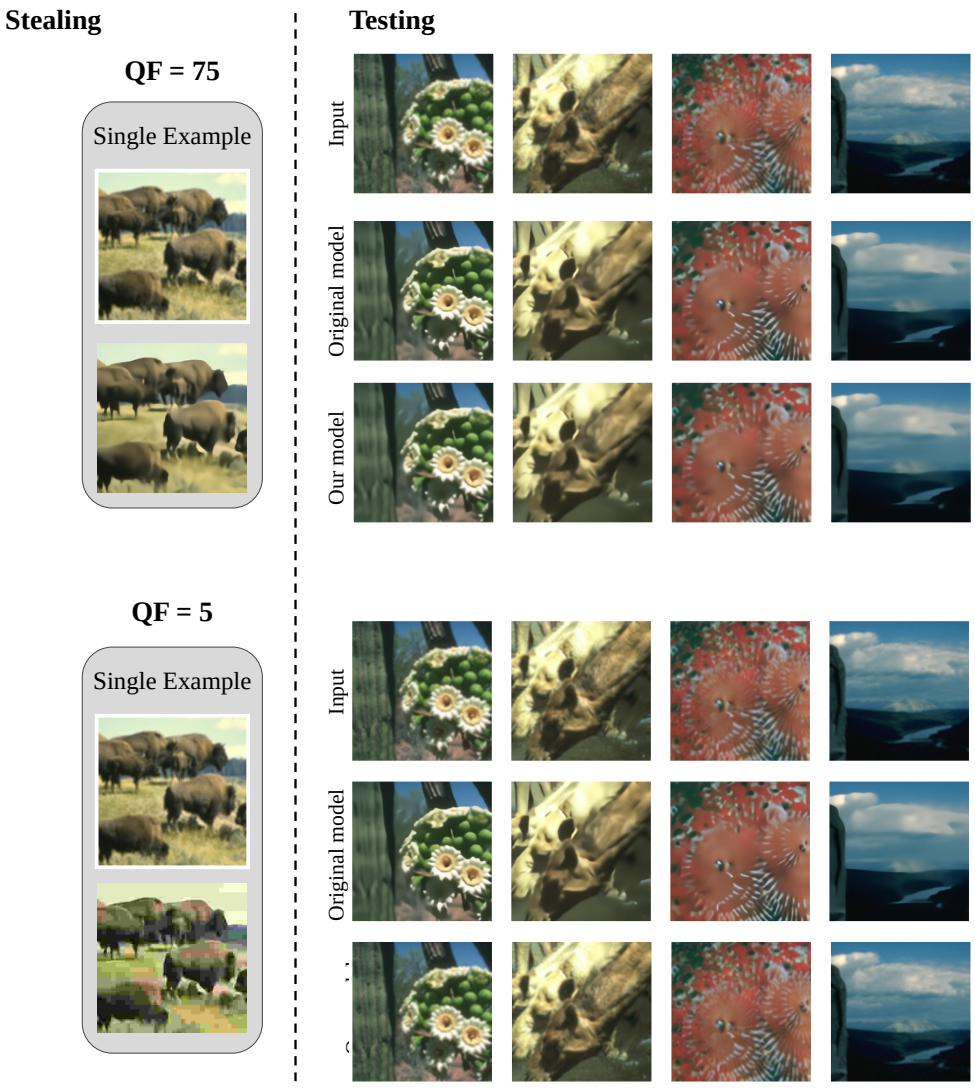

Figure S26. **Protecting the Uformer [40] defocus deblurring model from stealing using JPEG compression** We can see that quality factor of 75 (up) is not so effective as quality factor of 5 (bottom) which also cost a significant corruption of the image coming from the API.

Task	Loss	Architecture	Test set	PSNR (f, g_θ) Protection method		
				W/O	$QF = 10$	$QF = 75$
SR (bicubic kernel)	L_2	SRCNN	BSD100	**34.87**	11.12	**34.64**
		SRCNN	DIV2K	**37.54**	13.05	**36.30**
		SRCNN	Urban100	**38.18**	10.87	**37.11**
	L_1	EDSR	BSD100	**38.85**	13.54	**36.92**
		EDSR	DIV2K	**38.95**	12.05	**36.85**
		EDSR	Urban100	**41.04**	17.98	**37.67**
		SwinIR	BSD100	**38.88**	24.44	**36.68**
		SwinIR	DIV2K	**38.47**	19.87	**36.45**
		SwinIR	Urban100	**39.75**	22.25	**36.88**
	L_2 + Perceptual + Adversarial	SRGAN	BSD100	**35.78**	15.65	**34.17**
		SRGAN	DIV2K	**35.85**	13.11	33.64
		SRGAN	Urban100	**37.85**	12.09	**35.55**
	L_1 + Perceptual + Adversarial	BSRGAN	BSD100	**34.58**	12.21	33.09
		BSRGAN	DIV2K	**38.59**	11.13	**34.20**
		BSRGAN	Urban100	**42.69**	12.01	**36.08**
Denoising (STD $= 25$)	L_2	DnCNN	CBSD68	**36.08**	18.85	**34.94**
		DnCNN	Kodak24	**36.54**	17.87	**35.48**
	L_1	SwinIR	CBSD68	**38.97**	16.11	**36.64**
		SwinIR	Kodak24	**36.24**	18.45	**34.88**
		Restormer	CBSD68	**39.19**	16.77	**37.79**
		Restormer	Kodak24	**37.01**	17.71	**35.08**
		SCUnet	CBSD68	34.13	21.53	33.10
		SCUnet	Kodak24	**36.87**	18.23	33.96
	L_1 + Perceptual + Adversarial	SCUnetG	CBSD68	28.11	19.28	27.48
		SCUnetG	Kodak24	**34.87**	20.18	33.25
Defocus Deblurring ($\sigma = 5$)	Charbonnier	Uformer	BSD-synthetic	**45.23**	18.99	**43.15**
	L_1	Restormer	BSD-synthetic	**42.58**	18.12	**38.56**

Table S16. **Protecting restoration models in the *non-blind restoration* settings.** Here we report the PSNR when protecting the models from stealing by JPEG compression. In all experiments the single example size is 256×256. As can be seen, this defence strategy is effective in preventing stealing only when the quality factor of the compression is very low ($QF = 5$).

Task	Loss	Architecture	Test set	PSNR (f, g_θ) Protection method		
				W/O	WM	Masking WM
SR (bicubic kernel)	L_2	SRCNN	BSD100	**34.87**	10.35	**34.28**
		SRCNN	DIV2K	**37.54**	11.13	**35.97**
		SRCNN	Urban100	**38.18**	10.98	**36.76**
	L_1	EDSR	BSD100	**38.85**	12.19	**35.48**
		EDSR	DIV2K	**38.95**	14.75	**36.79**
		EDSR	Urban100	**41.04**	18.46	**37.11**
		SwinIR	BSD100	**38.88**	26.13	**35.74**
		SwinIR	DIV2K	**38.47**	21.89	34.09
		SwinIR	Urban100	**39.75**	24.97	**35.19**
	L_2 + Perceptual+ Adversarial	SRGAN	BSD100	**35.78**	13.56	33.45
		SRGAN	DIV2K	**35.85**	12.76	31.71
		SRGAN	Urban100	**37.85**	11.34	**34.79**
	L_1 + Perceptual + Adversarial	BSRGAN	BSD100	**34.58**	11.33	32.97
		BSRGAN	DIV2K	**38.59**	13.48	31.70
		BSRGAN	Urban100	**42.69**	13.44	**34.66**
Denoising (STD $= 25$)	L_2	DnCNN	CBSD68	**36.08**	20.51	**34.55**
		DnCNN	Kodak24	**36.54**	19.56	**35.19**
	L_1	SwinIR	CBSD68	**38.97**	18.94	**35.16**
		SwinIR	Kodak24	**36.24**	20.54	34.15
		Restormer	CBSD68	**39.19**	18.56	**37.06**
		Restormer	Kodak24	**37.01**	19.98	**34.88**
		SCUnet	CBSD68	34.13	21.53	31.25
		SCUnet	Kodak24	**36.87**	21.36	33.64
	L_1 + Perceptual + Adversarial	SCUnetG	CBSD68	28.11	21.15	26.08
		SCUnetG	Kodak24	**34.87**	22.38	30.75
Defocus Deblurring (Gaussian blur, $\sigma = 5$)	L_2	Uformer	BSD-synthetic	**45.23**	26.45	**42.58**
	L_1	Restormer	BSD-synthetic	**42.58**	27.04	**40.40**

Table S17. **Protecting restoration models in the *non-blind restoration* settings.** Here we report the PSNR when protecting the models by adding a watermark ("WM" column) to the output of the API. In all experiments the single example size is 256×256. As can be seen, this defence strategy is effective in preventing stealing. However, when training our model with masking the loss inside the watermark, we manage to get better results (column "Masking WM").