Supplementary Material: Self-Supervised Effective Resolution Estimation with Adversarial Augmentations

Manuel Kansy[1,2] *, Julian Balletshofer[2], Jacek Naruniec[2], Christopher Schroers[2], Graziana Mignone[2], Markus Gross[1,2], and Romann M. Weber[2]

[1]Department of Computer Science, ETH Zurich, [2]DisneyResearch|Studios, Zurich

{manuel.kansy, grossm}@inf.ethz.ch, {<first name>.<last name>}@disneyresearch.com

1. Training details and hyperparameters

Our network consists of a ResNet50 [3] pretrained on ImageNet [1] as a backbone, followed by an average pooling and a fully connected layer with no activation function to get a single output. During inference, we clip the output to $[0, 1]$. We use a batch size of 4 patches but accumulate the gradients to simulate a batch size of 512 to stabilize the training process. We train the model for 10 epochs with the Adam [7] optimizer and a learning rate of 10^{-3} that is decayed by a factor of 0.9 every epoch. Training takes around one or two days on a TITAN X GPU. To determine the best model, we use a validation data set constructed similarly to the evaluation data but smaller and with fewer test subjects. As a loss function, we use mean absolute percentage error (MAPE).

We calculate face masks using an implementation [20] of a modified BiSeNet [17]. The following classes are considered background during masking: *background*, *neck*, *neck_l*, *cloth*, *hat*, and *invalid*. To generate training samples, we sample the downscaling factor uniformly in $\frac{1}{[\frac{1}{16}, 1]}$ and the interpolation method uniformly from: *area*, *bicubic*, *bilinear*, *gaussian*, *lanczos3*, *lanczos5*, *mitchellcubic*, and *nearest neighbor*. For 10% of the samples, we do not perform downscaling, so that the network sees images without synthetic degradations sometimes. To be more robust to different face sizes, we prescale the training images to a random resolution in $[384, 2048]$ for 80% of the samples, adjusting the regression target y accordingly. Afterwards, we extract patches of size 256×256 (or 128×128) with a patch stride of 128 and a random start offset. Patches that contain less than 50% foreground pixels are filtered out during training.

To generate adversarial augmentations, we use projected gradient descent (PGD) attacks [10] with 10 PGD steps of size 30 in the L_2 norm (where the image values are in the range $[0, 255]$). We further project the adversarial noise to the L_∞ ball of size 10 to avoid any large changes of the input image that could affect the perceived sharpness. We perform regular and adversarial training steps at the same frequency and weigh them equally for the loss.

During inference, we calculate the patch stride dynamically to obtain about 100 patches per image. We then filter out patches that contain less than 90% foreground. Lastly, we take the median of the patch scores as final output.

Table 1 lists the hyperparameters used in the paper.

*Corresponding author.

Hyperparameter	Value
Model parameters	
Backbone	ResNet50 [3]
Final activation function	None / linear
Pretraining	ImageNet [1]
Training parameters	
Number of epochs	10
Batch size (real)	4 | 16
Batch size (simulated through gradient accumulation)	512
Loss parameters	
Loss function	Mean absolute percentage error (MAPE)
Optimizer	Adam [7] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$
Learning rate	10^{-3} with staircase decay factor 0.9 every epoch
Adversarial noise parameters	
Ratio number of regular vs. adversarial training steps	1
Ratio weight of regular vs. adversarial training step losses	1
Method for generating adversarial noise	Projected gradient descent (PGD)
Number of PGD steps	10
Step size	$L_2 = 30 \mid 15$
Epsilon ball to project to after each step	$L_\infty = 10$
Random initialization within epsilon ball	False
Range of values to clip perturbed images	$[0, 255]$
Preprocessing parameters	
Patch size	$256 \times 256 \mid 128 \times 128$
Patch stride	128
Use random patch offset when generating patches	True
Background masking	True
Minimum foreground percentage	50%
Interpolation methods	area, bicubic, bilinear, gaussian, lanczos3, lanczos5, mitchellcubic, nearest neighbor
Interpolation sampling	Uniform
Maximum downscale factor df_m	16
Downscale factor sampling	Uniform $(\frac{1}{df_m}, 1)$
Range prescale height	$[384, 2048] \mid [256, 2048]$
Prescaling frequency	80%
Downscaling frequency	90%
Use antialiasing for downscaling	True
Postprocessing parameters	
Number of patches	100
Background masking	True
Minimum foreground percentage	90%
Patch score aggregation	Median

Table 1. List of hyperparameters and their corresponding values. If multiple values are listed, the value left of | refers to patch size 256×256 and the value right of | to patch size 128×128.

2. Comparison with state-of-the-art methods for unmasked images

Table 2 shows the results of the state-of-the-art methods with unmasked evaluation images. Note that the performance of most methods degrades tremendously for generated images, likely due to the strong and unnatural background degradations occurring in them. Classical approaches (such as Δ DOM [8]) suffer especially from this whereas deep learning approaches handle unmasked images fairly well. Similar to the masked evaluation, our method outperforms all other methods by more than 2%.

Method Type	Method	Generated		Real		All	
		SRCC↑	PRA↑	SRCC↑	PRA↑	SRCC↑	PRA↑
Baselines	Frequency baseline	−0.0965	0.4907	0.7305	0.7795	0.3170	0.6351
	Compression baseline	−0.1106	0.4850	0.6121	0.7128	0.2508	0.5989
Classic general	BRISQUE [11]	0.1288	0.5678	0.5561	0.7091	0.3424	0.6385
	NIQE [12]	0.5030	0.6790	0.6985	0.7671	0.6008	0.7230
	IL-NIQE [18]	0.5152	0.6990	0.4636	0.6959	0.4894	0.6974
Deep learning general	NIMA [15]	0.5848	0.7020	0.4606	0.6518	0.5227	0.6769
	PaQ-2-PiQ [16]	0.5652	0.7207	0.7455	0.8090	0.6553	0.7648
	DB-CNN [19]	0.4909	0.7175	0.7273	0.7864	0.6091	0.7520
	MUSIQ (PaQ-2-PiQ) [6, 16]	<u>0.8803</u>	<u>0.8806</u>	<u>0.8667</u>	<u>0.8716</u>	<u>0.8735</u>	<u>0.8761</u>
	MUSIQ (SPAQ) [6, 2]	0.5288	0.7102	0.8015	0.8195	0.6652	0.7649
	MUSIQ (KonIQ-10k) [6, 4]	<u>0.8318</u>	<u>0.8469</u>	0.8576	0.8635	<u>0.8447</u>	<u>0.8552</u>
Classic blur-specific	CPBD [13]	−0.2894	0.4312	0.2288	0.5857	−0.0303	0.5084
	Δ DOM [8]	0.4500	0.6781	0.7833	0.8051	0.6167	0.7416
Deep learning blur-specific	SFA [9]	0.6500	0.7385	0.8045	0.8209	0.7273	0.7797
	MT-A [2]	0.5500	0.7139	0.8394	0.8490	0.6947	0.7815
	KonIQ++ [14]	0.7015	0.7786	**0.9242**	**0.9127**	0.8129	0.8457
Ours	Ours (patch size 256)	**0.8985**	**0.9091**	<u>0.8970</u>	<u>0.8845</u>	**0.8977**	**0.8968**
Ground truth	Mean	0.9407	0.9250	0.9466	0.9317	0.9436	0.9284
	Standard deviation	0.0256	0.0213	0.0186	0.0178	0.0172	0.0151

Table 2. Evaluation results (unmasked). SRCC denotes the Spearman rank correlation coefficient, and PRA denotes the pairwise ranking accuracy. The best score per column is marked in bold, the second- and third-best are underlined.

3. Complete ablation study

Table 3 shows the complete ablation of the most important training parameters. Note that all ablations are based on patch size 256×256 because it resulted in the best validation performance and allows to visualize the adversarial noise better than a patch size of 128×128.

Category	Setting	Generated		Real		All	
		SRCC↑	PRA↑	SRCC↑	PRA↑	SRCC↑	PRA↑
Training data set	One person	0.7697	0.8070	0.8864	0.8822	0.8280	0.8446
	Small	0.8652	0.8721	0.8955	0.8844	0.8803	0.8783
	Adding blurry images	0.9076	0.9004	0.8970	0.8787	0.9023	0.8896
	Subset of FFHQ [5]	0.9242	0.9176	0.8864	0.8672	0.9053	0.8924
Adversarial method	None	-0.1606	0.4494	0.5773	0.7036	0.2083	0.5765
	Step size $L_2 = 3$ (*/10)	**0.9500**	**0.9263**	0.8258	0.8394	0.8879	0.8828
	Step size $L_2 = 300$ $(* \cdot 10)$	0.8000	0.8442	0.8197	0.8175	0.8098	0.8309
	Clip to $L_\infty = 1$ (*/10)	0.8894	0.8810	0.8530	0.8578	0.8712	0.8694
	Clip to $L_\infty = 100$ $(* \cdot 10)$	0.9121	0.9062	$\underline{0.9045}$	$\underline{0.8902}$	0.9083	$\underline{0.8982}$
	FGSM 1 $L_\infty = 0.05$ step	$\underline{0.9470}$	$\underline{0.9231}$	0.6955	0.7508	0.8212	0.8370
	FGSM 1 $L_\infty = 0.5$ step	$\underline{0.9364}$	0.9177	0.7955	0.8253	0.8659	0.8715
	PGD 10 $L_\infty = 0.005$ steps	0.9303	0.9119	0.6182	0.7259	0.7742	0.8189
	PGD 10 $L_\infty = 0.05$ steps	0.8667	0.8777	0.8833	0.8725	0.8750	0.8751
	PGD 10 $L_\infty = 0.5$ steps	0.8545	0.8726	0.8500	0.8535	0.8523	0.8631
Unmasked	Unmasked training	0.8379	0.8751	0.8894	0.8729	0.8636	0.8740
	Unmasked inference	0.9258	0.9091	0.9000	0.8817	$\underline{0.9129}$	0.8954
	Unmasked training + inference	0.8985	0.9091	0.8970	0.8845	0.8977	0.8968
Pre-processing	Only bicubic interp.	0.9076	0.9064	0.9030	0.8848	0.9053	0.8956
	Only nearest neighbor interp.	0.7567	0.1108	0.7434	0.7916	0.7461	0.4512
	W/o nearest neighbor interp.	0.9091	0.9093	0.8894	0.8703	0.8992	0.8898
	Max downscale factor 8 (*/2)	0.9091	0.9032	0.8470	0.8498	0.8780	0.8765
	Max downscale factor 32 $(* \cdot 2)$	0.8364	0.8608	**0.9258**	**0.9161**	0.8811	0.8885
	W/o prescaling	0.8939	0.8805	$\underline{0.9106}$	$\underline{0.9023}$	0.9023	0.8914
	Foreground 10% $(* - 40\%)$	0.9333	$\underline{0.9232}$	0.8864	0.8731	$\underline{0.9098}$	$\underline{0.8981}$
	Foreground 90% $(* + 40\%)$	0.9212	0.9091	0.8833	0.8641	0.9023	0.8866
Post-processing	Mean	0.9167	0.9007	0.8970	0.8787	0.9068	0.8897
	Foreground 10% $(* - 80\%)$	0.9167	0.9007	0.8682	0.8576	0.8924	0.8791
	Foreground 50% $(* - 40\%)$	0.9242	0.9120	0.8939	0.8810	0.9091	0.8965
Model	W/o pretraining	0.8258	0.8439	0.8955	0.8779	0.8606	0.8609
Loss function	MAE	0.8924	0.8834	0.8682	0.8675	0.8803	0.8755
	MSE	0.8121	0.8406	0.8606	0.8611	0.8364	0.8509
	MSLE	0.8500	0.8633	0.8318	0.8314	0.8409	0.8473
Optimization	W/o gradient accumulation	0.8712	0.8976	0.8697	0.8702	0.8705	0.8839
	W/o learning rate decay	0.9227	0.9119	0.8924	0.8698	0.9076	0.8908
Ours	Ours (patch size 256)	0.9258	0.9120	$\underline{0.9045}$	0.8847	**0.9152**	**0.8984**

Table 3. Ablation study results (complete). SRCC denotes the Spearman rank correlation coefficient, and PRA denotes the pairwise ranking accuracy. The best score per column is marked in bold, the second- and third-best are underlined. * indicates the value of the parameter in "Ours (patch size 256)".

Training data set

Refer to the main paper for a discussion about the effects of the training data set.

Adversarial method

Refer to the main paper for a discussion about the effects of not using adversarial augmentations or having too small / large L_2 steps when generating the adversarial noise.

Similar to the step size, clipping the adversarial noise to a given L_∞ ball can be used to control the strength of the adversarial noise, where the noise must be sufficiently strong to make the model robust but not too strong where it changes the perceived sharpness of a training image significantly. Since the adversarial noise is also constrained by the step size, the influence of the L_∞ ball clipping parameter is rather small.

When performing projected gradient descent (PGD) steps in the L_∞ norm rather than the L_2 norm, the best results are obtained when performing 10 PGD steps with $L_\infty = 0.05$. Taking only a single, but 10 times larger, step using the fast gradient sign method (FGSM) leads to only slightly worse results but speeds up the training tremendously. We hypothesize that adversarial augmentations with L_2 steps are more effective than L_∞ steps because it enables the noise to change some pixel regions (*e.g*. edges) more than others (*e.g*. uniformly colored regions). This can also be seen in Figures 6 and 7.

Unmasked

Refer to the main paper for a discussion about the effects of not masking out the background during training and/or inference.

Preprocessing

Refer to the main paper for a discussion about the effects of using only bicubic or only nearest neighbor interpolation during training. Interestingly, while the experiment using only nearest neighbor interpolation suggests that the domain gap between the blocky nearest neighbor artifacts and real degradations is too large, removing nearest neighbor (and area) interpolation during training also degrades the performance slightly, even more than if only using bicubic interpolation (likely by chance).

There is a trade-off for the maximum downscale factor to create training samples. A larger factor leads to relatively better results on the real images whereas a smaller factor leads to relatively better results on generated images. A downscaling factor of 16 (default setting) leads to the most balanced and best overall performance.

Changing other preprocessing settings such as not prescaling images during training (prescaling should make the model more robust to different face sizes) or changing the foreground percentage threshold degrade the performance but only slightly.

Postprocessing

Using the mean instead of the median to aggregate the patch scores during inference leads to slightly worse performance, likely because the mean is less robust to outliers. Similarly, decreasing the foreground percentage threshold for considering patches during inference decreases the performance as more of the background, which is irrelevant for the face's quality, is considered.

Model

Training a model from scratch rather than using a model pretrained on ImageNet [1] degrades the performance quite significantly and causes the training to take longer to reach a reasonable performance.

Loss function

Other loss functions including the mean absolute error (MAE), mean squared error (MSE), and mean squared logarithmic error (MSLE) all lead to worse performance. Our intuition for the superiority of the mean absolute percentage error (MAPE) is that it considers the error relative to the regression target. For example, predicting 0.1 when the actual value is 0.2 should be punished more than predicting 0.7 when the actual value is 0.8.

Optimization

Not using the techniques aimed to stabilize the training, *i.e*. gradient accumulation and learning rate decay, both degrade the performance slightly on average.

4. Training data set

Figure 4 shows two images of each subject from the training data set. Note that the absolute resolution of each image differs, but the figure shows them with equal width.

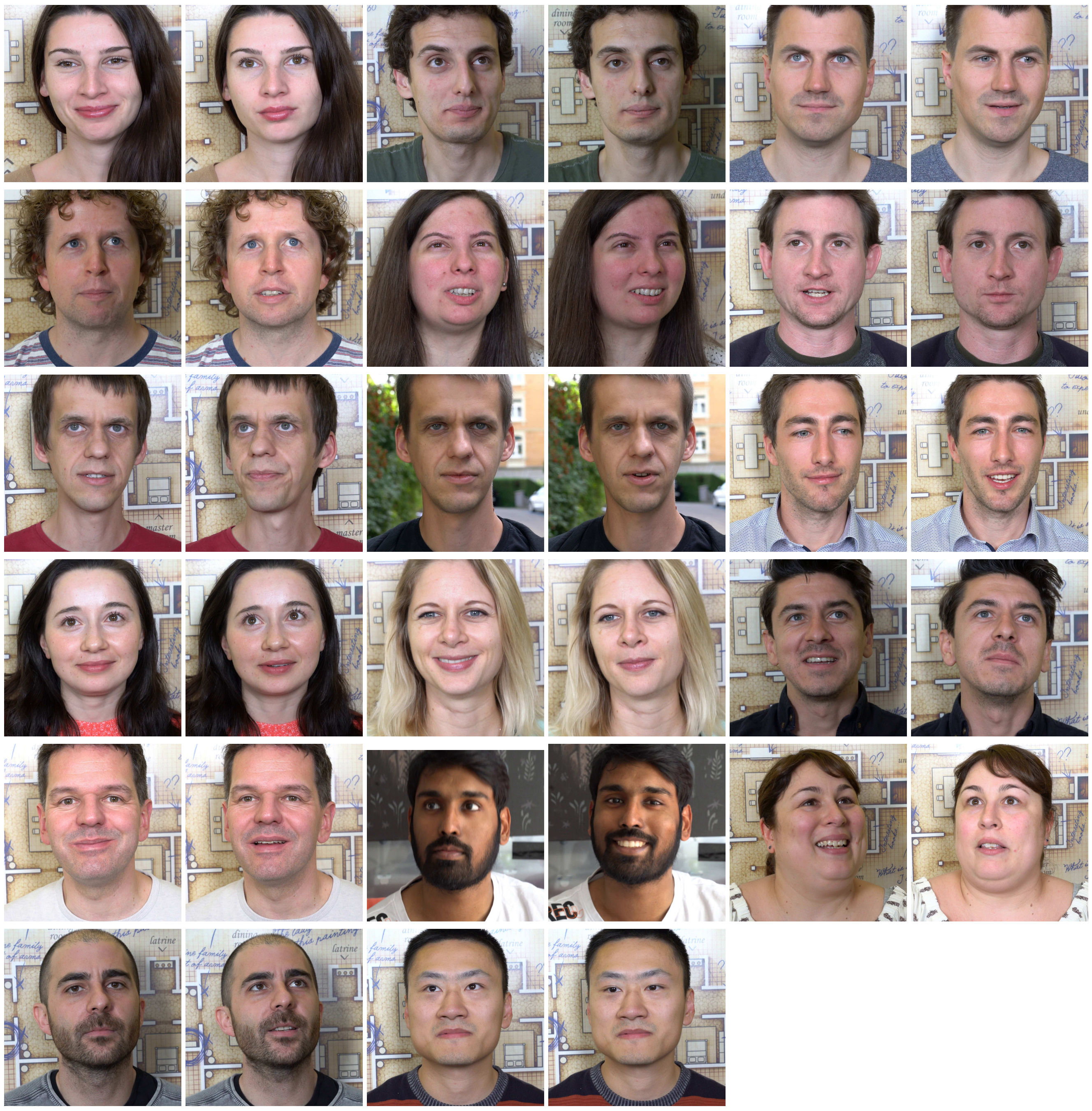
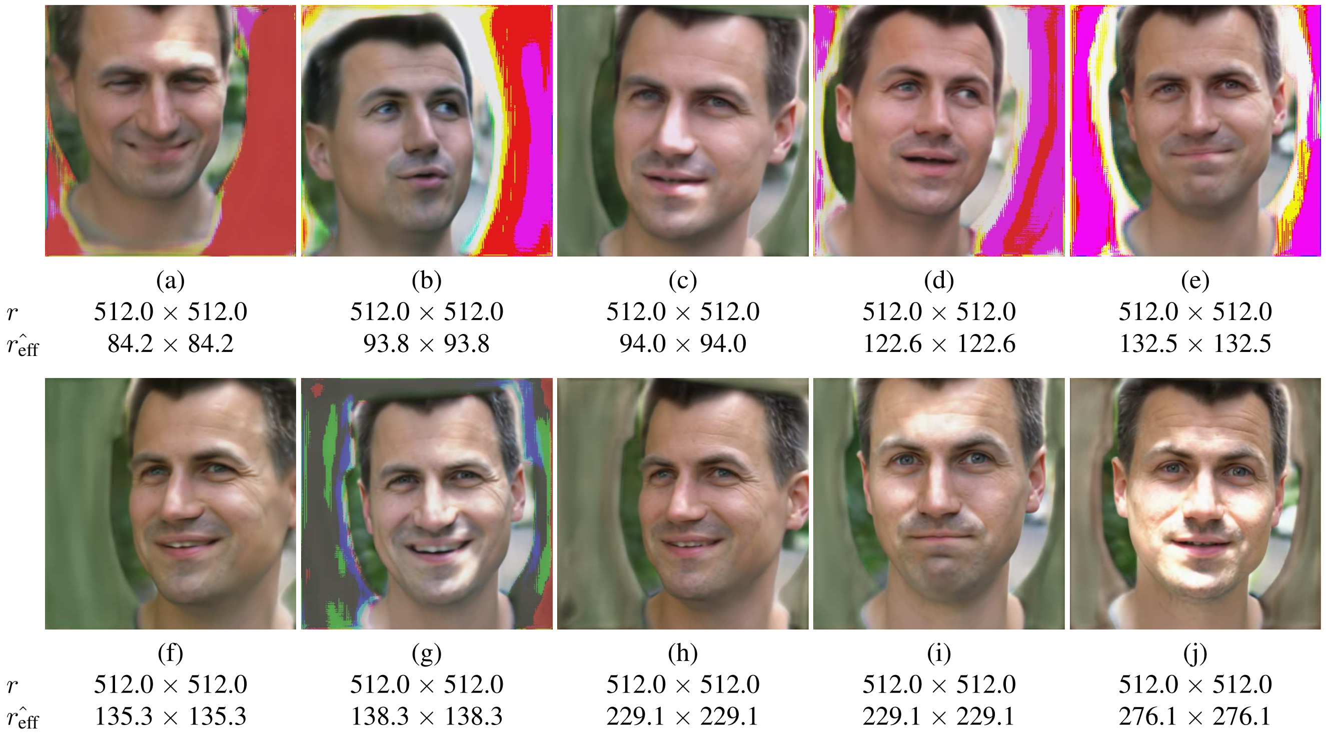


Figure 1. Examples from the training data set.

5. Visualization of the ranking order of our proposed method and human labels

Figures 2 and 3 show the sort order of our method and the human labels for one evaluation folder containing 10 generated images of the same person. In this context, (a) refers to the blurriest image and (j) to the sharpest image according to our method or human labels. The ranking of the human labels and our proposed method is almost identical. Merely, images (e) and (f) are swapped compared to the human labels, but the scores as well as the perceived blurriness are very similar. Note that the background, the lighting and the color changes are not taken into account for the human ranking order.

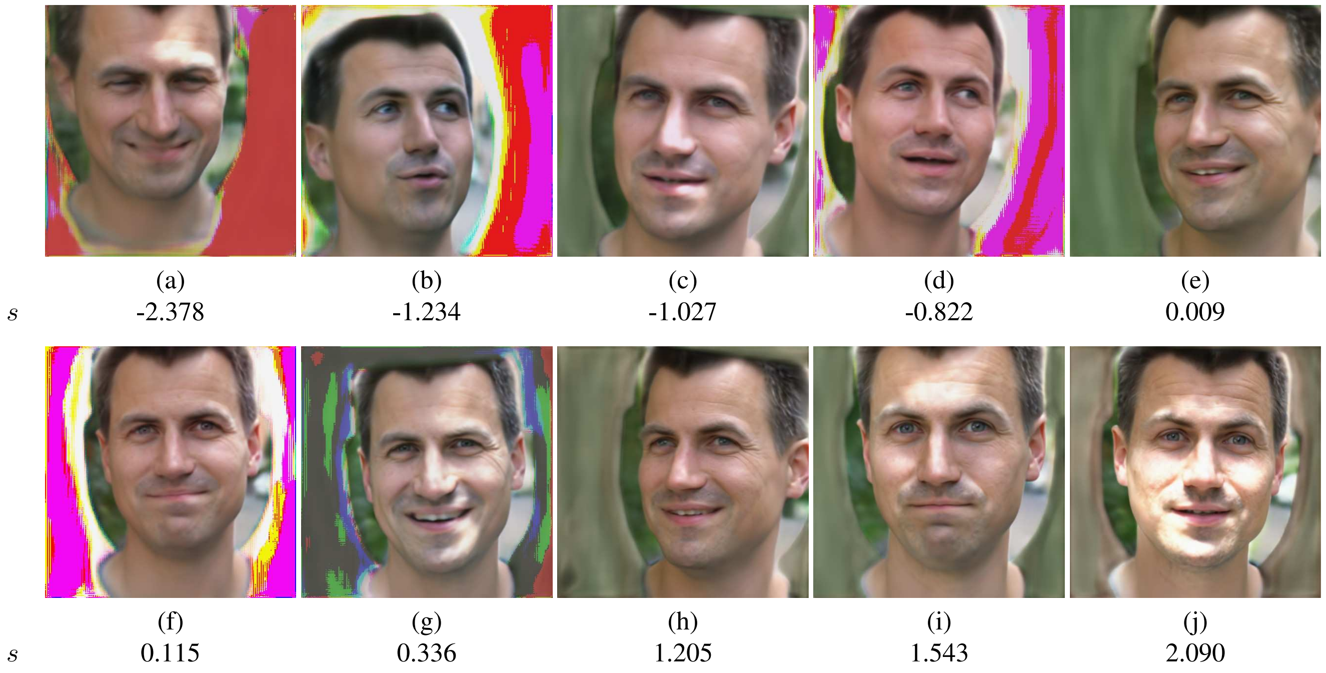
Figures 4 and 5 show the sort orders for one evaluation folder containing 10 real images of the same person. Here, similar conclusions can be drawn as for the generated images. The rank order mostly differs for images with almost the same sharpness level ((c)-(e), and (g) and (h)), but the overall tendency is consistent between our method and the human labels.



	(a)	(b)	(c)	(d)	(e)
r	512.0×512.0	512.0×512.0	512.0×512.0	512.0×512.0	512.0×512.0
$\hat{r_{\text{eff}}}$	84.2×84.2	93.8×93.8	94.0×94.0	122.6×122.6	132.5×132.5

	(f)	(g)	(h)	(i)	(j)
r	512.0×512.0	512.0×512.0	512.0×512.0	512.0×512.0	512.0×512.0
$\hat{r_{\text{eff}}}$	135.3×135.3	138.3×138.3	229.1×229.1	229.1×229.1	276.1×276.1

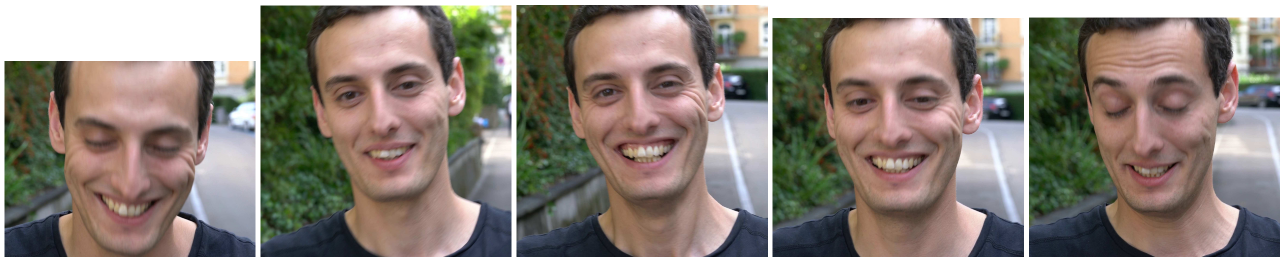
Figure 2. Example of the sort order for generated images according to our method. (a) refers to the lowest sharpness, (j) to the highest.



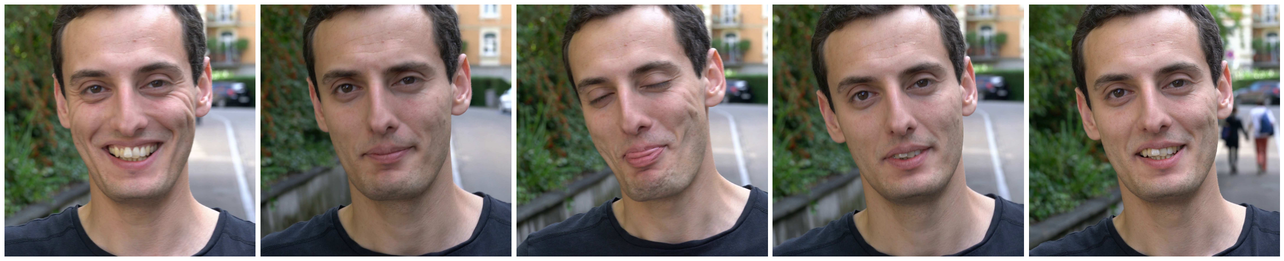
	(a)	(b)	(c)	(d)	(e)
s	-2.378	-1.234	-1.027	-0.822	0.009

	(f)	(g)	(h)	(i)	(j)
s	0.115	0.336	1.205	1.543	2.090

Figure 3. Example of the sort order for generated images according to human labels. (a) refers to the lowest sharpness, (j) to the highest. The human label score s has no intuitive meaning, but a larger score implies a sharper image.

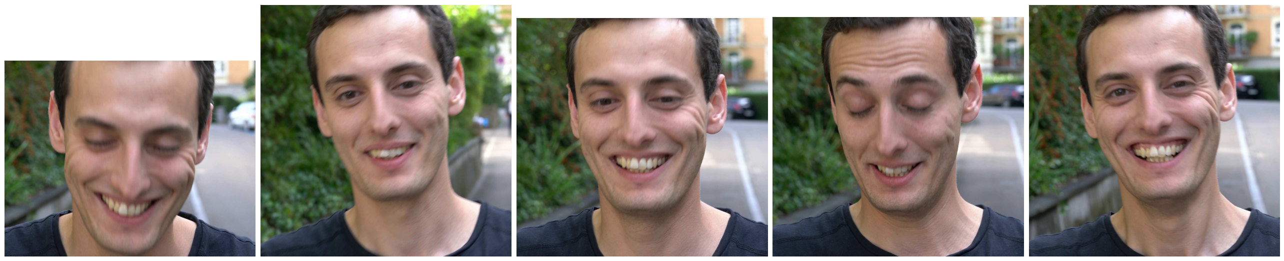


	(a)	(b)	(c)	(d)	(e)
r	1945.0 × 1514.0	1831.0 × 1831.0	1691.0 × 1691.0	1883.0 × 1786.0	1849.0 × 1759.0
$\hat{r_{\text{eff}}}$	217.1 × 217.1	279.6 × 279.6	302.2 × 302.2	350.5 × 350.5	356.2 × 356.2

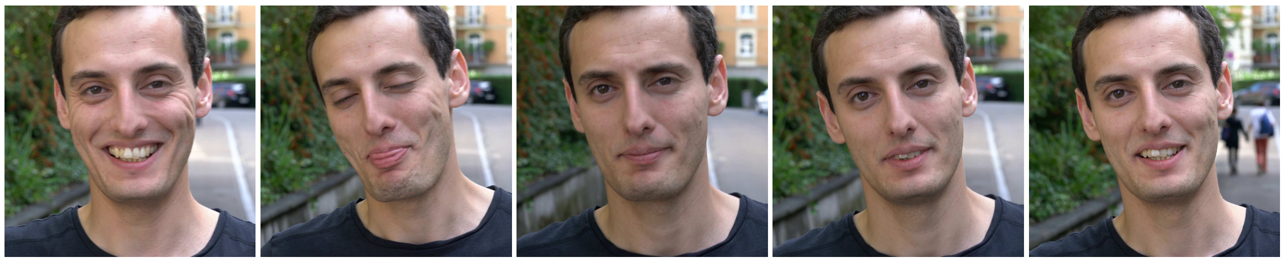


	(f)	(g)	(h)	(i)	(j)
r	1697.0 × 1697.0	1775.0 × 1775.0	1823.0 × 1823.0	1709.0 × 1709.0	1751.0 × 1751.0
$\hat{r_{\text{eff}}}$	700.3 × 700.3	788.6 × 788.6	829.1 × 829.1	921.1 × 921.1	1016.1 × 1016.1

Figure 4. Example of the sort order for real images according to our method. (a) refers to the lowest sharpness, (j) to the highest.



	(a)	(b)	(c)	(d)	(e)
s	-2.946	-1.510	-0.955	-0.805	-0.568



	(f)	(g)	(h)	(i)	(j)
s	0.408	0.725	0.795	1.723	2.726

Figure 5. Example of the sort order for real images according to human labels. (a) refers to the lowest sharpness, (j) to the highest. The human label score s has no intuitive meaning, but a larger score implies a sharper image.

6. Image enhancement and degradation using strong adversarial noise

As stated in the main paper, the adversarial noise of our proposed model is actually not really adversarial anymore but meaningful instead. Thus, if we apply strong adversarial noise, we can actually change the perceived sharpness of an image by optimizing the noise towards increasing or decreasing the predicted effective resolution. Figures 6 and 7 show the difference between the adversarial noise when performing 10 PGD steps with a step size of $L_\infty = 0.5$ and 10 PGD steps with a step size of $L_2 = 100$. Decreasing the predicted effective resolution leads to realistic results for both methods whereas the L_∞ norm sometimes produces unrealistic patterns when increasing the predicted effective resolution. In contrast, the L_2 norm produces realistic skin and hair patterns that improve the perceived sharpness of the image.

Figures 8 and 9 show further examples of the L_2 norm for decreasing or increasing the predicted effective resolution of an input image. Notably, the effect on skin and hair structures appears very promising. This demonstrates that our proposed self-supervised training procedure enables the model to learn features that are especially useful for face images. An interesting future direction would be to further investigate how the features and gradient of the network can be used for downstream tasks. Furthermore, the robustness obtained from the adversarial augmentations during training paves the way towards exploring the use of no-reference image quality assessment methods as loss functions during training to improve the output quality of generative models.

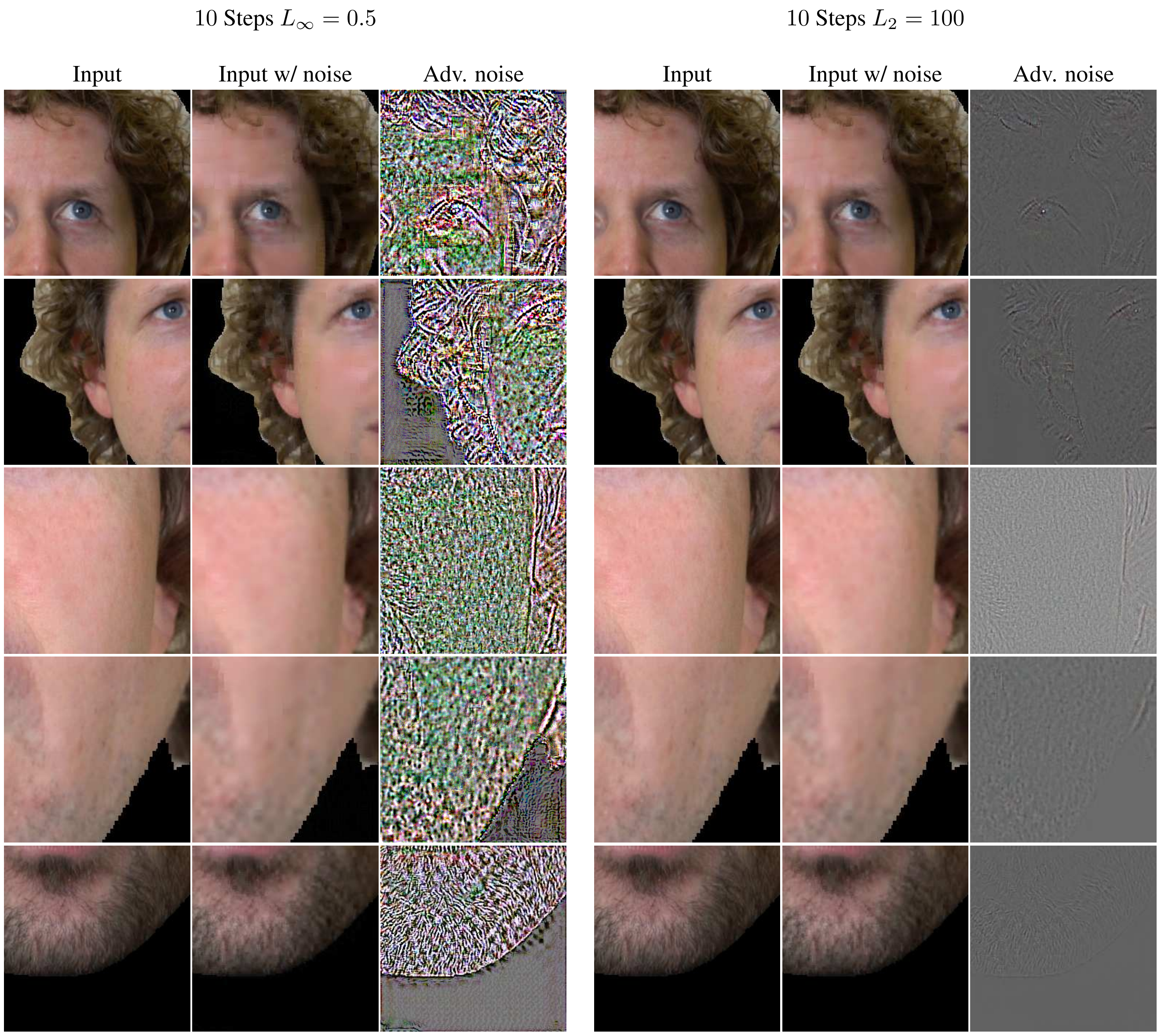


Figure 6. Examples of images where the adversarial noise leads to blurrier results. Comparison L_∞ vs. L_2 norm.

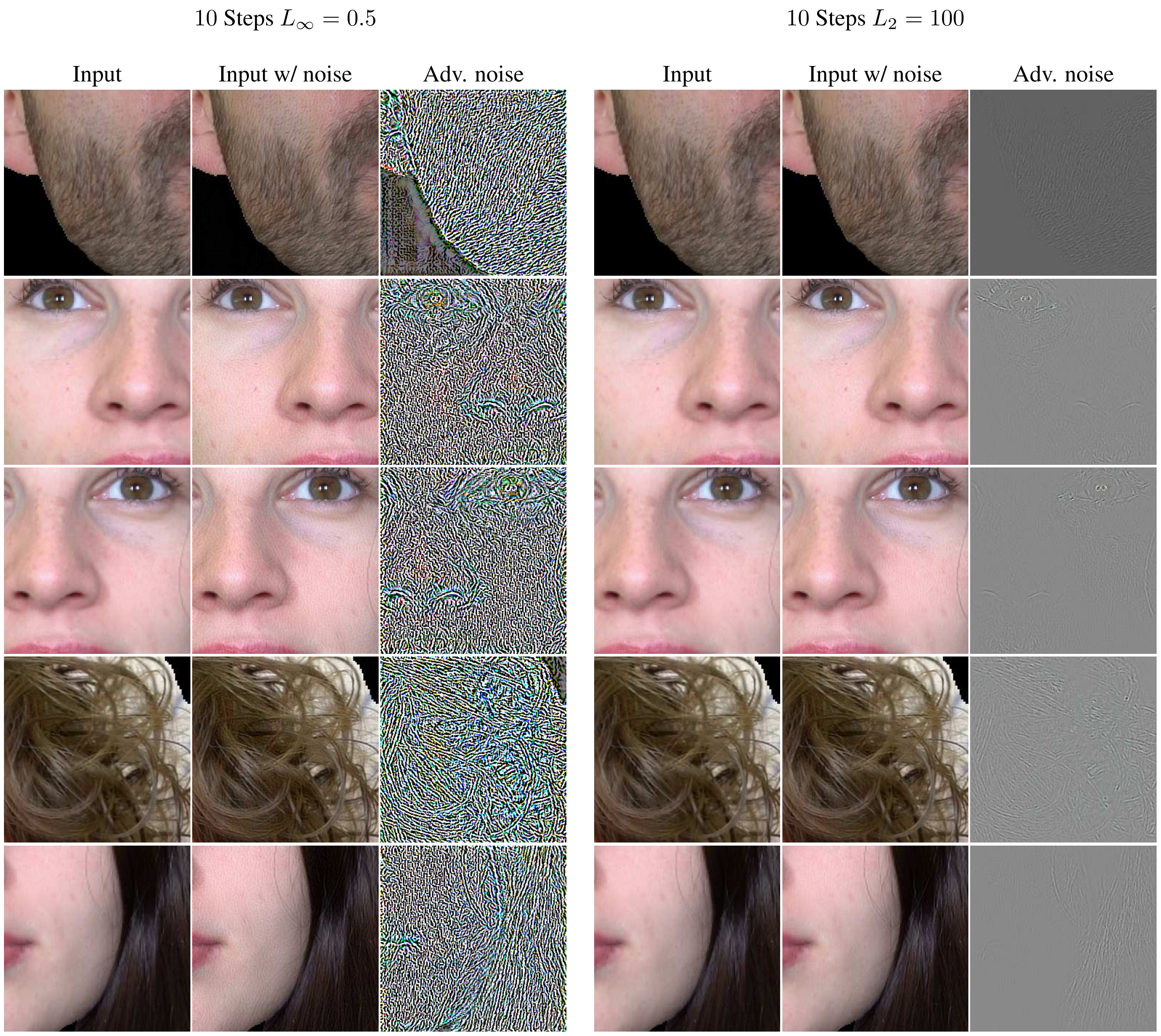


Figure 7. Examples of images where the adversarial noise leads to sharper results. Comparison L_∞ vs. L_2 norm.

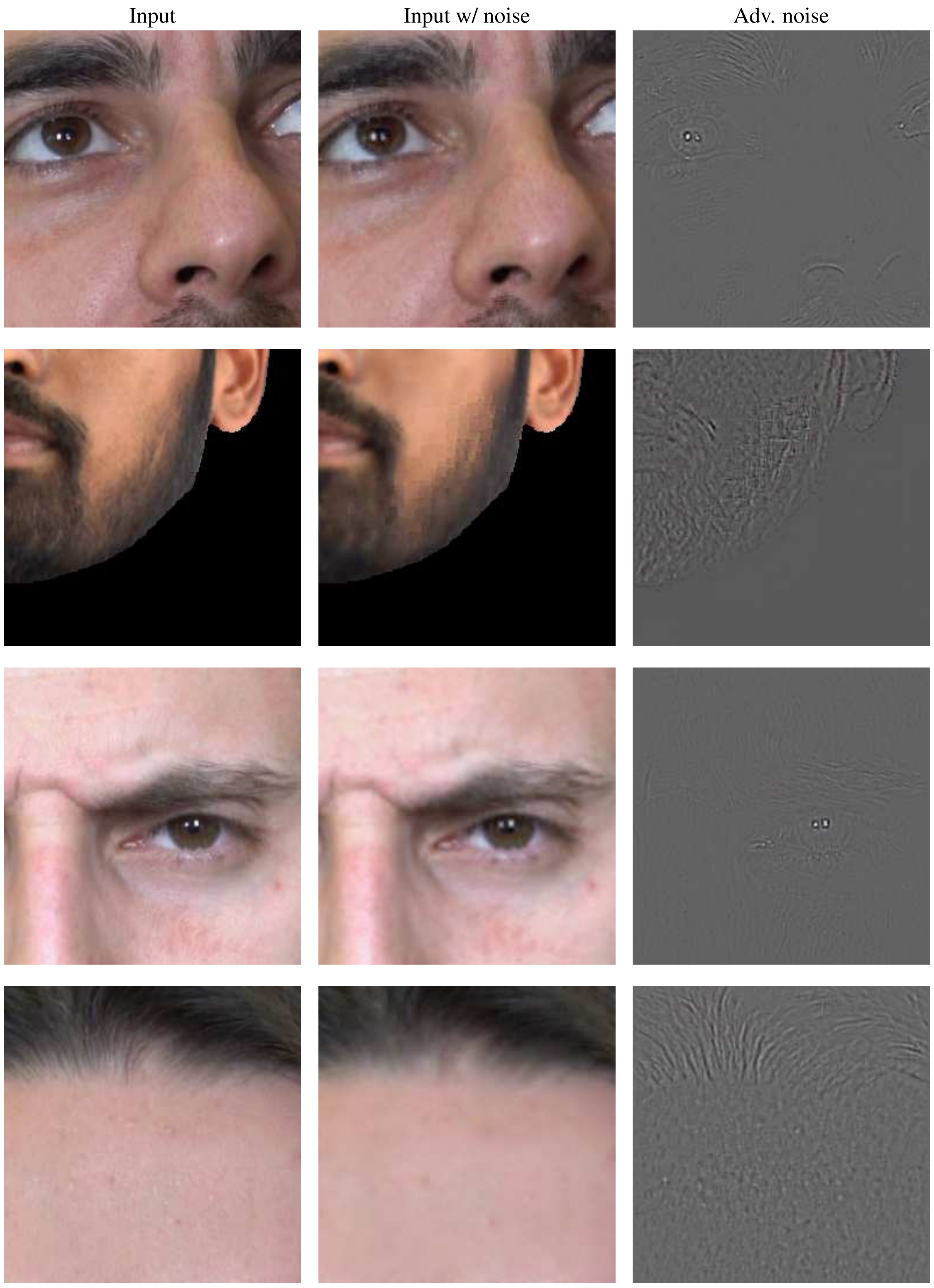


Figure 8. Additional examples of images where L_2 adversarial noise leads to blurrier results.

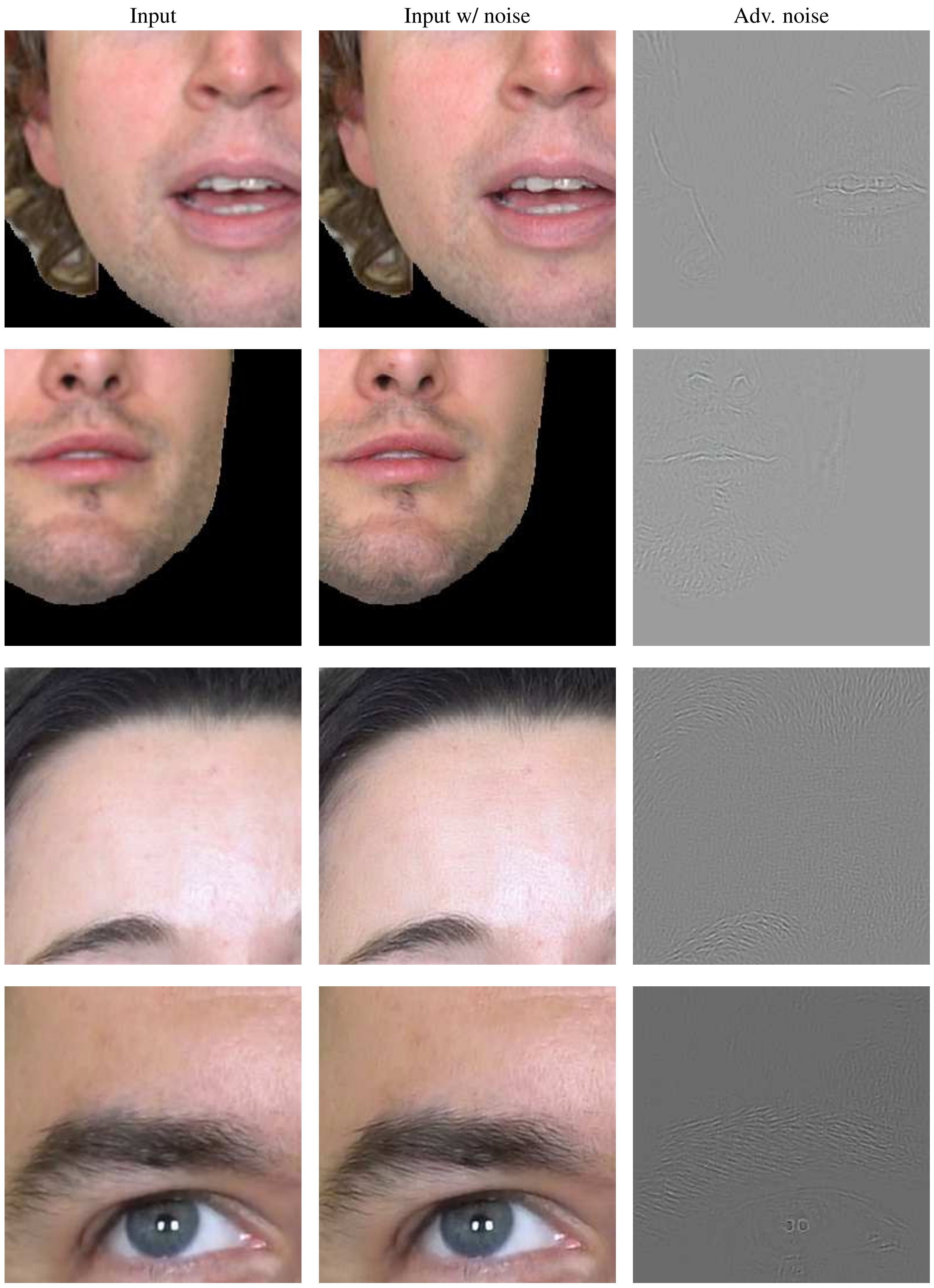


Figure 9. Additional examples of images where L_2 adversarial noise leads to sharper results.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Yuming Fang, Hanwei Zhu, Yan Zeng, Kede Ma, and Zhou Wang. Perceptual quality assessment of smartphone photography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3677–3686, 2020.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Vlad Hosu, Hanhe Lin, Tamas Sziranyi, and Dietmar Saupe. Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment. *IEEE Transactions on Image Processing*, 29:4041–4056, 2020.

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[6] Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5148–5157, 2021.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Jayant Kumar, Francine Chen, and David Doermann. Sharpness estimation for document and scene images. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3292–3295. IEEE, 2012.

[9] Dingquan Li, Tingting Jiang, Weisi Lin, and Ming Jiang. Which has better visual quality: The clear blue sky or a blurry animal? *IEEE Transactions on Multimedia*, 21(5):1221–1234, 2018.

[10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[11] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12):4695–4708, 2012.

[12] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012.

[13] Niranjan D Narvekar and Lina J Karam. A no-reference image blur metric based on the cumulative probability of blur detection (cpbd). *IEEE Transactions on Image Processing*, 20(9):2678–2683, 2011.

[14] Shaolin Su, Vlad Hosu, Hanhe Lin, Yanning Zhang, and Dietmar Saupe. Koniq++: Boosting no-reference image quality assessment in the wild by jointly predicting image quality and defects. In *The 32nd British Machine Vision Conference*, volume 2, 2021.

[15] Hossein Talebi and Peyman Milanfar. Nima: Neural image assessment. *IEEE transactions on image processing*, 27(8):3998–4011, 2018.

[16] Zhenqiang Ying, Haoran Niu, Praful Gupta, Dhruv Mahajan, Deepti Ghadiyaram, and Alan Bovik. From patches to pictures (paq-2-piq): Mapping the perceptual space of picture quality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3575–3585, 2020.

[17] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

[18] Lin Zhang, Lei Zhang, and Alan C Bovik. A feature-enriched completely blind image quality evaluator. *IEEE Transactions on Image Processing*, 24(8):2579–2591, 2015.

[19] Weixia Zhang, Kede Ma, Jia Yan, Dexiang Deng, and Zhou Wang. Blind image quality assessment using a deep bilinear convolutional neural network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(1):36–47, 2018.

[20] zllrunning. face-parsing.pytorch, 10 2019.