Towards Unsupervised Blind Face Restoration using Diffusion Prior Supplementary Material

Tianshu Kuai[2,†], Sina Honari[1], Igor Gilitschenski[2,3], Alex Levinshtein[1]
[1]Samsung AI Center Toronto, [2]University of Toronto, [3]Vector Institute for AI

Acknowledgements. This work was done at Samsung AI Center Toronto. It was funded by Mitacs and Samsung Research, Samsung Electronics Co., Ltd.

A. Summary

In Appendix B, we provide more details of our synthetic datasets. Appendix C contains additional implementation details of pre-trained models and fine-tuning. We present additional qualitative and quantitative results along with more ablation studies in Appendix D. We provide more discussions on other relevant components of our approach in Appendix E. Finally, Appendix F and Appendix G discuss the limitations and the potential negative impact of our method.

B. Details on Synthetic Datasets

B.1. Pre-training Dataset

To perform the pre-training, we adopt the low-quality data synthesis pipeline from other literature [12, 13, 26] to create input-GT training data pairs:

$$\mathcal{I}_{LQ} = \left\{ \left[(\mathcal{I}_{HQ} * \mathbf{k}_\sigma)_{\downarrow_r} + \mathbf{n}_\delta \right]_{JPEG_q} \right\}_{\uparrow_r}, \qquad (1)$$

where a high-quality image $\mathcal{I}_{HQ}$ is first convolved with a Gaussian blur kernel $\mathbf{k}_\sigma$ of kernel size σ and downsampled by a factor of r. Gaussian noise with standard deviation of δ is added to the downsampled image. Then JPEG compression of quality factor q and upsampling by a factor of r are applied to synthesize the low-quality counterpart I_{LQ}. For pre-training the SwinIR [14] model, we use randomly sampled σ, r, δ, and q from $[0.1, 15]$, $[0.8, 32]$, $[0, 20]$, and $[30, 100]$, respectively. We use the pre-trained CodeFormer [33] from the authors. This model is pre-trained with randomly sampled σ, r, δ, and q from $[1, 15]$, $[1, 30]$, $[0, 20]$, and $[30, 90]$.

B.2. Our Synthetic Dataset

As mobile phones become more accessible, significant number of images are captured with phone cameras nowadays. We can simulate more realistic low-quality images by taking the camera ISP and RAW noise models into consideration. Specifically, we generate the low-quality images from high-quality ones as:

$$\mathcal{I}_{LQ} = \left\{ ISP\left[ISP^{-1}((\mathcal{I}_{HQ})_{\downarrow_r}) + \mathbf{n}_c \right] \right\}_{\uparrow_r}, \qquad (2)$$

where (ISP^{-1}) is an image unprocessing pipeline [24] that converts a sRGB image to RAW, (ISP) is the reverse procedure of image unprocessing to render the sRGB image from RAW image [24], r is the downsampling and upsampling factor, and n_c is the simulated RAW camera noise. For the simulated noise, we apply the commonly used Heteroscedastic Gaussian models [3, 16, 20] to generate realistic camera noise. We construct our synthetic datasets at *moderate* and *severe* noise levels, at ISO levels of ($\sim$ 1600) and ($\sim$ 3200), respectively. For each noise level, we also generate two separate datasets at 4$\times$ downsampling and 8$\times$ downsampling, which gives us four datasets in total. For each dataset, we use the same 3,000 high-quality images from the CelebA-HQ dataset [10]. We take 2,500 of them for generating pesudo targets and fine-tuning the pre-trained models, and the rest of the 500 images for evaluation. We do not use the GT images during pseudo target generation and fine-tuning. For fine-tuning, we start with the same pre-trained checkpoint for all four dataset setups.

C. Implementation Details

C.1. Pre-trained Models

In our work, both SwinIR [14] and CodeFormer [33] are pre-trained on the entire FFHQ dataset [11], which contains 70,000 high-quality face images with resolution of 512$\times$512. The training data pairs are generated by following the degradation pipeline described in Eq. (1).

We train the SwinIR from scratch using the AdamW optimizer [19] with initial learning rate of $1e{-}4$ and update the learning rate following cosine annealing [18]. We use batch size of 16 and train the model for 800,000 iterations in total with image-level $L1$ loss, perceptual (LPIPS) loss [32], and adversarial (GAN) loss [5]. For losses, we use the same set

[†]Work done during an internship at Samsung AI Center Toronto.

Encoder	Trans.	CFT	PSNR↑	SSIM↑	LPIPS↓	FID↓	MANIQA↑	MUSIQ↑
✓			22.38	0.5816	0.4391	**40.59**	0.6568	75.60
✓	✓		22.85	0.6036	0.4258	41.21	**0.6581**	**75.76**
✓	✓	✓	**23.01**	**0.6556**	**0.4225**	47.67	0.6295	71.13

Table 1. **Different fine-tuning setups for CodeFormer.** We compare the three setups of fine-tuning CodeFormer on 4× downsampling data at *severe* noise level.

of losses and the weights as the fine-tuning setup described in the main paper.

For CodeFormer, we directly adopt the pre-trained checkpoint from the CodeFormer authors [33]. It is pre-trained following their 3-stage training scheme. Please refer to their paper for more details on CodeFormer pre-training.

For the unconditional face diffusion model used in our pipeline, we follow the model architecture from [2] and the adopt the model weights from [31]. This model is also trained on FFHQ dataset [11]. The total number of timesteps is $T = 1000$ and we follow techniques from [22] to accelerate the denoising process by reducing the total number discrete timesteps down to 250.

C.2. Fine-tuning

For fine-tuning the pre-trained SwinIR model, we use the same optimizer, initial learning rate, and batch size as in pre-training (Appendix C.1). We fine-tune the SwinIR model with input and pseudo target pairs for 20,000 iterations. The setup for the loss functions has been described in the main paper.

For CodeFormer, we only fine-tune its encoder and transformer module while keeping the codebook, decoder, and the controllable feature transformation module (CFT) fixed. We use the code-level loss $\mathcal{L}_{code}^{feat}$ and the cross-entropy loss on the code prediction $\mathcal{L}_{code}^{token}$ to fine-tune the encoder and transformer module (same set of losses used in stage-II training of CodeFormer):

$$\mathcal{L} = \mathcal{L}_{code}^{feat} + \lambda_{token}\mathcal{L}_{code}^{token}, \tag{3}$$

where λ_{token} is the weight for the cross-entropy loss on the code prediction.

Note that no image-level loss is used since we found empirically that only fine-tuning the encoder and transformer module provides the best performance, and there is no need for image-level losses when only fine-tuning these two parts of the model. We compare the results of different fine-tuning setups in Tab. 1. Fine-tuning the encoder and the transformer module provides the best balance between fidelity and quality among all the setups. Fine-tuning all three components obtains better PSNR, SSIM, and LPIPS. However, there is a compromise in the image quality as reflected in FID, MANIQA, and MUSIQ metrics.

D. Additional Results

D.1. Qualitative Results

We show additional qualitative results on our synthetic datasets in Figs. 7 and 8. We also provide qualitative results on our Wider-Test-200 set in Fig. 6.

D.2. Quantitative Results

We provide detailed results for both SwinIR and CodeFormer on each synthetic dataset setup. The results of the pre-trained, fine-tuned SwinIR along with the SwinIR-based targets are included in Tab. 10. For CodeFormer, we provide the results in a similar manner as in the main paper, but for each noise level separately. That is, we show the CodeFormer results in Tabs. 11 and 12 for *moderate* and *severe* noise levels, respectively. Note that for all the models, we adopt the scripts and checkpoints from their authors, and run the models with their suggested parameters and setups for blind face restoration. The pre-trained CodeFormer achieves the best results on 4× *moderate* noise levels because it is robust enough to handle such inputs, however it cannot completely get rid of all artifacts (See Fig. 7). Fine-tuning this model effectively removes the remaining artifacts, but would slightly degrade its quantitative performance due to small content distortion in pseudo targets. However, on more severe degradations where the pre-trained model's performance degrades, our approach consistently improves upon the pre-trained model. In addition, we provide the results of fine-tuning the pre-trained model using GT clean images as pseudo targets (i.e. supervised fine-tuning) for both model architectures in Tabs. 13 and 14. They serve as the performance upper bound for our problem setup.

D.2.1 Mild degradations

For completeness, we follow the same approach as *medium* and *severe* degradations to construct a fine-tuning dataset with weaker degradation using ISO level of ($\sim$ 800). We refer it as *mild* degradation and report the results in Tab. 22. The results show consistent improvement of our fine-tuned model compared to the pre-trained ones.

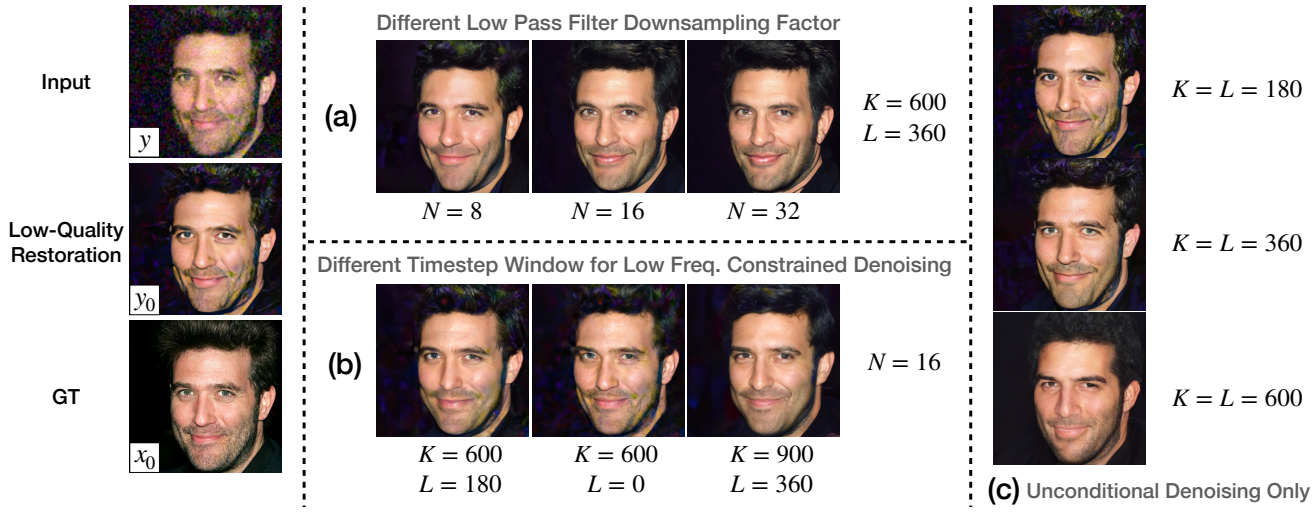

Figure 1. **Effects of low pass downsampling factor and timestep choices on SwinIR pseudo target.** In (a) we show the effects of adjusting the low pass filter downsampling factor (N); In (b) we show the effects of different timestep windows for low frequency constrained denoising (K and L); In (c) we show the pseudo targets if only unconditional denoising is applied (**zoom in for details**).

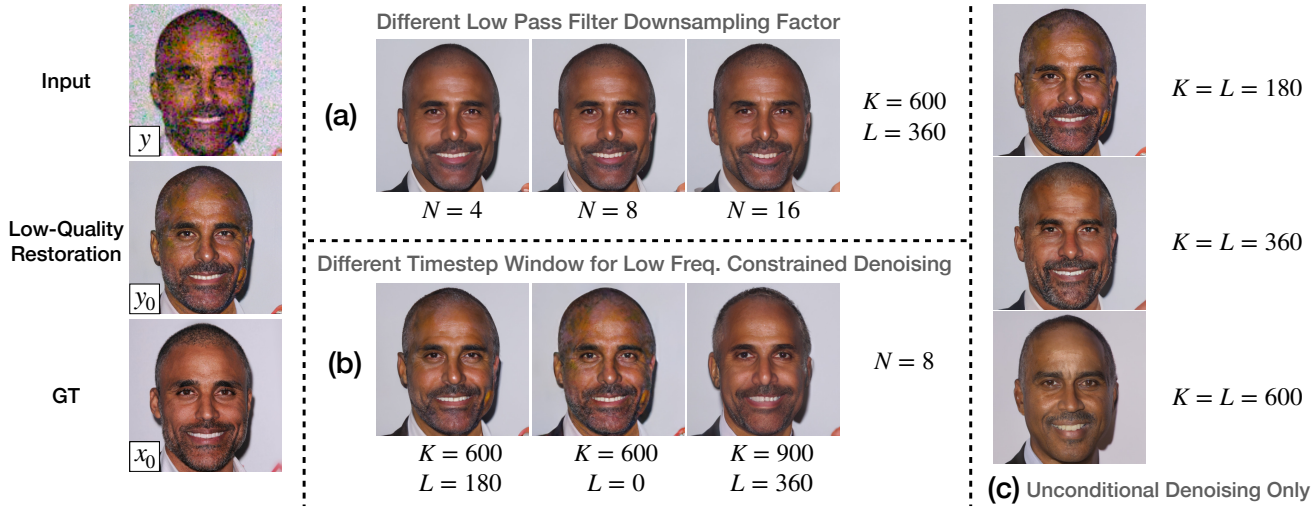

Figure 2. **Effects of low pass downsampling factor and timestep choices on CodeFormer pseudo target.** In (a) we show the effects of adjusting the low pass filter downsampling factor (N); In (b) we show the effects of different timestep windows for low frequency constrained denoising (K and L); In (c) we show the pseudo targets if only unconditional denoising is applied (**zoom in for details**).

D.2.2 Comparison on face recognition metrics

We report two face recognition metrics: Deg. [27] and LMD [7] in Tab. 21. Our pseudo targets obtain the best LMD and the second best Deg. Among the diffusion-free models, our model obtains the best results on PSNR, SSIM, LPIPS, and FID, while obtaining lower performance on LMD and Deg. We observe that the models have to make a compromise between fidelity and quality. While the facial lankmark is deteriorated slightly, the reference-based metrics of PSNR, SSIM, and LPIPS still show superior performance of our proposed approach.

D.2.3 Runtime and memory efficiency

In Tab. 23, we compare the memory usage and inference time of our model with others on a Nvidia RTX 3090 GPU. For diffusion-based models, even if we reduce the number of time-steps to only one step, our method is more efficient in terms of both run-time and memory. This is still regardless of the drop in accuracy of diffusion-based models that one-step diffusion would incur, as shown in [21, 25].

D.3. Additional Ablation Studies

D.3.1 Effects of timesteps and low pass filter choices

The choices for timesteps of when to start and stop the low frequency content constraint and the low pass filter's downsampling factor have significant impact on the quality and fidelity of the pseudo targets. In Figs. 1 and 2, we show the effects of the varying these parameters on SwinIR and CodeFormer pseudo targets. We provide quantitative ablation studies on the pseudo targets for both SwinIR [14] and CodeFormer [33] in Tabs. 15 and 16, as well as the results of fine-tuning using the corresponding targets in Tabs. 17 and 18. Our selection of the parameters provide the best balance between perceptual quality and fidelity.

D.3.2 Loss functions setup for fine-tuning

The performance of our approach depends on the fine-tuning of the pre-trained model. In Tab. 2, we provide the results of a fine-tuned SwinIR model with different combinations of the fine-tuning losses. We use the same set of input and pseudo target data pairs for all the setups in this table and fine-tune the model for the same number of iterations. Using $\mathcal{L}_{L1}$ and $\mathcal{L}_{LPIPS}$ together obtains better PSNR, however, it obtains worse results on perceptual metrics. When applying all the three losses, the model's results are in the best perceptual quality, achieving the best LPIPS and FID among all the setups. We provide the ablation on the fine-tuning loss functions for CodeFormer in the supplementary material.

D.3.3 Weights of the loss functions

In Tab. 3, we compare the results of fine-tuning SwinIR with different weights for the perceptual loss ($\mathcal{L}_{LPIPS}$) and adversarial loss ($\mathcal{L}_{GAN}$). In our experiments, we set the weights of these two losses to be 0.1, as this setup gives the best perceptual quality among all the combinations while maintaining good fidelity. We also conduct ablation study on the weight of the cross-entropy loss ($\mathcal{L}_{code}^{token}$) for CodeFormer fine-tuning. As shown in Tab. 4, the results of the fine-tuned model do not vary much with different weights.

D.3.4 Number of images used in fine-tuning

We investigate the effects of the fine-tuning dataset size on both SwinIR and CodeFormer. In Tab. 4 in the main paper and Tab. 5, we compare the results of the fine-tuning models with pre-trained models using different sizes of the fine-tuning dataset. We gain consistent improvements with our approach on CodeFormer even with 20 images, thanks to the discrete codebook that prevents potential over-fitting.

In Fig. 9, we evaluate the impact of gradually increasing the number of images used in fine-tuning on the fine-tuned restoration model performance. As can be observed, more images help improve the restoration model beyond the pseudo targets. In particular, training on such images adds a prior to the restoration model on how to handle the observed artifacts, allowing the model to learn on the ensemble of target degradations, which is not the case for pseudo-targets.

E. More Analysis and Discussions

E.1. Pseudo Target Generation vs. Other Diffusion-based Methods

Here we provide more detailed comparison between our pseudo target generation process and other relevant diffusion-based methods. Specifically, we compare the detailed procedure of our pseudo target generation with DifFace [31], ILVR [1], DDA [4], PG-Diff [30], DiffBIR [15], and DR2 [29]. We summarize the detailed procedures in Algorithms 1 to 7, respectively.

To highlight the main differences, DifFace [31] adds Gaussian noise to the output of a preprocessing model [14], and applies standard unconditional denoising to get the clean image. ILVR [1] and DDA [4] apply similar low frequency content guidance, but for all the denoising timesteps. Note that despite ILVR targeting tasks of conditional image generation and image editing, and DDA targeting domain adaptation for image classification, their denoising diffusion process also utilizes low frequency content as guidance, which in theory could be applied to the task of blind image restoration. However, as they apply such guidance for all denoising timesteps, the results would be blurry and still contain artifacts due to inaccurate guidance from degraded input image. PG-Diff [30] and DiffBIR [15] use the output of a pre-processing model [14, 28] on the input image to guide the denoising process. PG-Diff uses an unconditional face diffusion model [2], while DiffBIR uses a fine-tuned stable-diffusion model [23]. DR2 [29] stops the low frequency content constraining at a pre-defined timestep, performs a one-step project to timestep $t = 0$, and relies on a pre-trained restoration model [6] as a post-processing step to restore the high-frequency details. It also downsamples the input by a factor of 2 before applying a 256×256 face diffusion model, and upsamples the resulting image from the diffusion model to restore the original resolution before applying the pre-trained post-processing model. This downsampling procedure helps reducing the amount of artifacts in the image before passing it to the diffusion model, at the expense losing finer details.

Since DR2 [29] achieves comparable performance as our targets on Wider-Test-200 set (better on MANIQA and MUSIQ while worse on FID), we perform experiments of fine-tuning pre-trained models with DR2's results as pseudo targets. We compare the results of fine-tuning SwinIR and CodeFormer with our pseudo targets and DR2 outputs

$\mathcal{L}_1$	$\mathcal{L}_{LPIPS}$	$\mathcal{L}_{GAN}$	PSNR↑	SSIM↑	LPIPS↓	FID↓	MANIQA↑	MUSIQ↑
✓			25.06	**0.7207**	0.4587	92.04	0.3151	41.72
✓	✓		**25.30**	0.7106	0.3908	47.94	0.5545	59.30
✓		✓	24.52	0.6587	0.4226	50.70	0.5869	72.88
✓	✓	✓	24.75	0.6676	**0.3853**	**41.42**	**0.6023**	**73.36**

Table 2. **Different setups of fine-tuning losses for SwinIR.** We compare different loss functions setup of fine-tuning SwinIR on 4× downsampling data at *moderate* noise level.

$\mathcal{L}_1$	$\mathcal{L}_{LPIPS}$	$\mathcal{L}_{GAN}$	PSNR↑	SSIM↑	LPIPS↓	FID↓	MANIQA↑	MUSIQ↑
1.0	1.0	1.0	24.18	0.6656	0.3913	42.57	0.6022	72.99
1.0	0.1	1.0	23.83	0.6362	0.3881	41.80	0.6107	73.47
1.0	1.0	0.1	24.89	0.6899	0.3873	46.37	0.6000	69.83
1.0	**0.1**	**0.1**	24.75	0.6676	0.3853	41.42	0.6023	73.36
1.0	0.01	1.0	23.16	0.6133	0.4034	46.48	0.5943	73.82
1.0	0.01	0.1	24.39	0.6531	0.3889	41.81	0.6247	74.77
1.0	0.01	0.01	24.87	0.6801	0.3890	44.36	0.6020	71.10
1.0	0.1	0.01	25.16	0.6943	0.3850	45.82	0.6002	69.54
1.0	1	0.01	25.17	0.7041	0.3890	46.73	0.5863	65.11

Table 3. **Ablation study on the weights of the fine-tuning losses for SwinIR.** We compare the results of fine-tuning pre-trained SwinIR with different loss weights on 4× downsampling data at *moderate* noise level. Red and blue indicate the best and the second best results. **Bold** indicates our selection.

in Tab. 6. With our pseudo targets, both fine-tuned models can produce more realistic results.

E.2. CodeFormer Fidelity Weight

CodeFormer [33] has a unique controllable features transformation (CFT) module that controls how much information from the encoder is fused into the decoder. It allows users to control the balance between the quality ($w = 0$) and fidelity ($w = 1$) of the restoration, by simplying modifying this fidelity weight $w \in [0, 1]$. We set the $w = 0.5$ (default value suggested by CodeFormer) when we run the pre-trained and the fine-tuned CodeFormer in this paper for a fair comparison between the two. We also show the comparison of our fine-tuned CodeFormer with its pre-trained counterpart at different fidelity weights in Fig. 3. Our fine-tuned model is consistently better than the pre-trained model. In addition, the fine-tuned model achieves the best results at $w = 1.0$. We believe that fine-tuning improves the quality of the extracted features from the encoder, thus more information from the encoder flowing into the decoder is helpful in terms of both restoration quality and fidelity.

E.3. Pseudo Target Generation Without Pre-trained Restoration Model

As shown in the qualitative results in the main paper and in Fig. 7 in this supplementary material, a pre-trained model effectively removes some artifacts from the inputs, while preserving good fidelity of the image content. Better initial images for pseudo target generation leads to better pseudo target quality, as shown in Fig. 4. The pre-trained CodeFormer is able to produce a cleaner image for the pseudo target generation process, despite that its output still contains artifacts. These artifacts are cleaned up by our pseudo target generation process. Using the better images also benefit the low-frequency constrained denoising process as we will have a more accurate and less noisy constraint applied in target generation. We also provide quantitative results for this behaviour in Tab. 19, where the pseudo targets generated from a pre-trained CodeFormer's outputs are consistently better than the ones generated directly from degraded inputs, and ultimately make the fine-tuned CodeFormer models better (see Tab. 20) at all noise levels and downsampling levels.

E.4. Pseudo Targets Fidelity - Quality Trade-off

Our pseudo targets may not be perfectly aligned with inputs in terms of fidelity, because the unconditional diffusion model is trained for generation rather than for restoration. The low-frequency constraint during the diffusion denoising process does not ensure exact matching in high-frequency details. Such process improves the image quality at the expense of fidelity. The property of an unconditional diffusion model is that, starting the denoising process at higher timesteps (more Gaussian noise injected into

$\mathcal{L}_{code}^{feat}$	$\mathcal{L}_{code}^{token}$	PSNR↑	SSIM↑	LPIPS↓	FID↓	MANIQA↑	MUSIQ↑
1.0	1.0	**22.31**	**0.5856**	**0.4282**	42.16	0.6574	**75.84**
1.0	**0.5**	22.28	0.5848	0.4290	**41.72**	**0.6580**	**75.84**
1.0	0.1	22.26	0.5833	0.4303	42.18	0.6578	75.81

Table 4. **Ablation study on the weights of the fine-tuning losses for CodeFormer.** We compare the results of fine-tuning pre-trained CodeFormer with different weights on 4× downsampling data at *severe* noise level. **Bold** indicates our selection.

Number of images	PSNR↑	SSIM↑	LPIPS↓	FID↓	MANIQA↑	MUSIQ↑
Pre-trained	22.90	0.5810	0.4420	53.02	0.6371	74.96
20	23.42	0.6288	0.4276	46.08	0.6535	74.93
100	23.28	0.6189	0.4259	44.25	0.6562	75.40
500	22.99	0.6080	0.4262	42.34	0.6590	75.63
1000	22.88	0.6038	0.4266	41.81	0.6590	75.76
2500	22.85	0.6036	0.4258	41.21	0.6581	75.76

Table 5. **Ablation Study on the fine-tuning dataset size on CodeFormer.** We compare the results of fine-tuning pre-trained CodeFormer with different numbers of images used in fine-tuning on 4× downsampling data at *severe* noise level.

Model	Targets	MANIQA↑	MUSIQ↑	FID↓
Fine-tuned SwinIR	DR2	0.5740	69.27	89.48
	Ours	**0.6093**	**74.15**	**88.21**
Fine-tuned CodeFormer	DR2	**0.6386**	72.63	90.59
	Ours	0.6343	**73.02**	**84.65**

Table 6. **Our pseudo targets vs. DR2 [29] as targets.** We compare the results of fine-tuning models with our pseudo targets and with DR2 outputs as targets on the Wider-Test-200 set.

Setup	PSNR↑	SSIM↑	LPIPS↓	FID-GT↓	FID-FFHQ↓
$T = 180$	22.83	0.5981	0.4498	47.41	71.17
$T = 600$	20.11	0.5746	0.4857	41.94	59.43
$T = 900$	12.96	0.4066	0.7017	66.98	49.49
Ours	23.15	0.6507	0.4364	37.42	67.40

Table 7. **Pseudo targets fidelity vs. quality.** We compare the pseudo targets in different target generation setups on 4× downsampling data at *severe* noise level. The first three rows correspond to unconditional denoising with different starting timesteps, and the last row corresponds to our target generation setup in this work. Note that specifically in this table, FID-GT refers to the FID score against the statistics of the ground-truth image set (measures a combination of fidelity and quality, while FID-FFHQ refers to the FID score against the FFHQ dataset statistics (measures overall face image quality and not fidelity). As described in the Metrics section of the main paper, we report FID-GT as the FID score for evaluations on synthetic datasets in all the other tables of this work, which aligns better with the targets of interest when available.

the image) will apply more distortion to the content of the denoised image, which leads to worse fidelity but higher overall image quality (higher FID-FFHQ score in Tab. 7). This means that one can generate pseudo targets with great image quality while completely losing the identity information of the face to be restored, as shown in Fig. 5. Starting at timestep of 900 for a 1,000 steps diffusion model provides pseudo target of another person. Such case will be catastropic for the later fine-tuning because the networks are optimized with image-level losses ($L1$ and LPIPS losses), which has strict requirements in terms of fidelity between target and input. As shown in Tab. 8, fine-tuning with inconsistent targets ($T = 900$) lead to very poor results. Our setup with constrained denoising process finds a good balance between fidelity and quality for the pseudo targets, which is an essential part of its success.

E.5. Improved DifFace and DiffBIR

DifFace [31] and DiffBIR [15] are two diffusion-based approaches that apply a pre-processing restoration model first to remove some artifacts on the images before passing images to the diffusion models. Both methods use the same pre-trained restoration model, a SwinIR [14] that is pre-trained on the FFHQ dataset [11] with MSE loss only. The motivation behind this choice of loss is that this model would output blurry faces, but with good fidelity. The diffusion models are then responsible for generating high-frequency details. However, these two methods would fail if this pre-trained SwinIR fails on inputs with out-of-distribution degradation. In Tab. 9, we show the improvements of DifFace and DiffBIR after using our fine-tuning pipeline. Note that this fine-tuned SwinIR is based on the pre-trained SwinIR with MSE loss only, not from the pre-trained SwinIR we used for other experiments in this work.

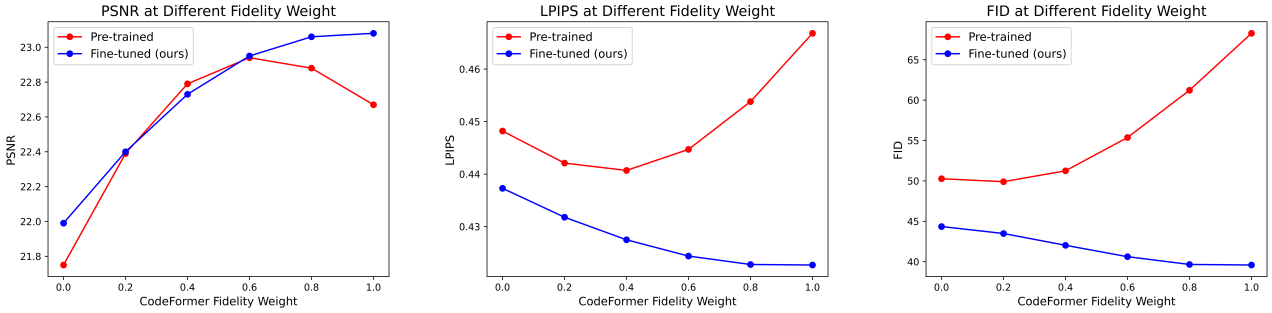

Figure 3. **CodeFormer results using different fidelity weights.** We plot the results of pre-trained and fine-tuned CodeFormer when using different fidelity weights at testing time on 4× downsampling data at *severe* noise level.

Figure 4. **Pseudo target quality without pre-trained restoration model.** We show the pseudo targets with and without running a pre-trained CodeFormer before applying target generation.

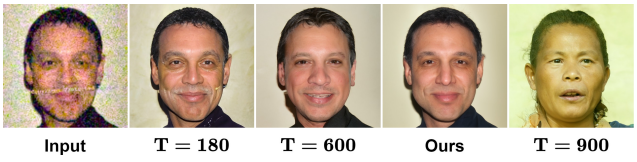

Figure 5. **Pseudo target quality vs. fidelity.** We show the pseudo targets generated with unconditional denosing up to timestep of 900 and with our optimal timestep setup (combination of low frequency constrained denoising and unconditional denoising). When the timestep is large enough, the generated target becomes a completely different face.

The results show that with our fine-tuning approach, one can improve methods that rely on a restoration model for pre-processing.

F. Limitations

Our work relies on a robust pre-trained diffusion model, and the diffusion model must be pre-trained to generate images of the same category as the low-quality inputs. This similarity requirement is only in terms of the image categories (e.g. faces, animals, natural images), while the types of degradations in the low-quality inputs are independent from the pre-trained diffusion model. As part of the inputs to our pipeline, a set of unpaired low-quality images is needed for the purpose of fine-tuning the pre-trained model. As shown in Tab. 4 in the main paper, despite being able to improve the pre-trained model's performance, having a small set of images with only ∼20 images can potentially lead to over-fitting for certain models [14], which deteriorates restoration quality. In addition, one may need to manually modify the downsampling factor N for low pass filter based on the quality of the pre-trained model's restoration (more severe artifacts require larger N) although we've shown that using timesteps of $K = 600$ and $L = 360$ are robust to various quality of restoration. If the pre-trained restoration model fails completely, our pseudo target generation will not output feasible target either as it depends on the restoration model's output. We suggest the users to directly feed input image to our pseudo target generation pipeline in this extreme case.

Pseudo Target Setup	PSNR↑	SSIM↑	LPIPS↓	FID↓
$T = 180$	22.81	0.5849	0.4294	46.00
$T = 600$	21.45	0.5487	0.4581	48.72
$T = 900$	16.90	0.3591	0.5673	104.54
Ours	22.85	0.6036	0.4258	41.21

Table 8. **Effects of pseudo targets fidelity vs. quality in fine-tuning.** We compare the results of fine-tuning using pseudo targets in different target generation setups on 4× downsampling data at *severe* noise level. The first three rows correspond to targets with unconditional denoising with different starting timesteps, and the last row corresponds to our targets generation setup in this work.

G. Potential Negative Impact

In this work, we use the FFHQ dataset [11] to pre-train and use a subset of the CelebA-HQ [10] to fine-tune our models. The two datasets are publicly available, where FFHQ consists of 70,000 high-quality face images crawled from Flickr and CelebA-HQ contains high-resolution version of the face images of different celebrities from CelebA

Model	With which SwinIR?	PSNR↑	SSIM↑	LPIPS↓	FID↓
DifFace [31]	Pre-trained	21.21	0.5645	0.5504	84.99
	Fine-tuned	**23.11**	**0.6210**	**0.4438**	**60.38**
DiffBIR [15]	Pre-trained	21.59	0.5371	0.6172	100.81
	Fine-tuned	**22.89**	**0.5769**	**0.4750**	**55.78**

Table 9. **Improved DifFace [31] and DiffBIR [15] with fine-tuned SwinIR.** We show the improvements gained for DifFace and DiffBIR by replacing their pre-trained SwinIR with the fine-tuned counterpart on our synthetic datasets.

dataset [17]. Our trained models will inherit the existing bias in these face datasets, particularly the imbalance in the distribution of gender, ethnicity, and age [9]. This could potentially limit the model's performance on certain under-represented groups of population. It also leads to perpetuation and amplification of societal biases within the current datasets. A large-scale face image dataset that is more balanced and diverse is needed for future research. In our Wider-Test-200 dataset, we make the effort to improve the diversity of the faces in our test sets by including testing images from different gender, ethnicity, and age groups. In addition, the misuse of our pipeline will pose ethical issues on potential personal privacy breach and illegal manipulation of face images.

	Noise Level	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
Pre-trained	moderate	21.28	0.5744	0.5446	74.12	21.28	0.5744	0.5446	99.44
Pseudo targets	moderate	22.89	0.6317	0.4444	**41.13**	21.23	0.5819	0.5168	77.89
Fine-tuned	moderate	**24.75**	**0.6676**	**0.3853**	41.42	**23.28**	**0.6206**	**0.4348**	**68.25**
Pre-trained	severe	20.92	0.5444	0.5842	120.44	19.00	0.4813	0.6435	152.90
Pseudo targets	severe	21.24	0.5871	0.4950	**47.47**	19.53	0.5300	0.5822	**86.01**
Fine-tuned	severe	**23.41**	**0.6284**	**0.4156**	49.46	**21.91**	**0.5720**	**0.4821**	115.97

Table 10. **SwinIR's improvements after fine-tuning.** We show the effectiveness of our approach on a pre-trained SwinIR on all four of our synthetic data setups.

	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
DifFace [31]	23.32	0.6399	0.4467	43.48	21.05	0.5649	0.5564	87.91
DiffBIR [15]	**23.95**	0.6185	0.5217	64.33	21.30	0.5356	0.6134	92.02
PG-Diff [30]	23.41	**0.6515**	0.4235	41.30	20.68	0.5799	0.4992	87.53
DR2 [29]	22.01	0.6168	0.4400	55.78	21.35	0.5962	0.4548	51.58
Our pseudo targets	23.08	0.6407	**0.4241**	**36.45**	**22.17**	**0.6185**	**0.4493**	**42.00**
GFPGAN [26]	24.12	**0.6454**	0.4385	45.76	21.90	0.5629	0.5013	67.94
VQFR [6]	21.78	0.5372	0.4711	83.84	20.22	0.4919	0.5184	104.12
CodeFormer [33]	**24.31**	0.6335	**0.4007**	**40.66**	22.19	0.5716	0.4420	51.91
CodeFormer + Ours	23.20	0.6138	0.4117	41.74	**22.28**	**0.5848**	**0.4290**	**41.72**

Table 11. **CodeFormer results on data with moderate noise level.** We compare our pseudo targets and fine-tuned results with pre-trained CodeFormer and other baselines. Top rows: diffusion-dependent models at test time. Bottom rows: diffusion-free models at test time.

	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
DifFace [31]	21.46	0.5731	0.5396	75.78	18.99	0.4800	0.6589	132.81
DiffBIR [15]	21.83	0.5380	0.6262	86.05	19.26	0.4562	0.7076	160.85
PG-Diff [30]	21.82	0.6012	0.4923	64.82	18.31	0.4879	0.5976	152.10
DR2 [29]	20.13	0.5631	0.4818	53.16	19.32	0.5353	0.5082	58.30
Our pseudo targets	**23.15**	**0.6507**	**0.4364**	**37.42**	**21.14**	**0.5902**	**0.4918**	**50.59**
GFPGAN [26]	22.89	0.5999	0.4807	56.07	20.27	0.4937	0.5582	97.86
VQFR [6]	20.16	0.4709	0.5318	114.00	18.54	0.4161	0.5950	142.11
CodeFormer [33]	**22.90**	0.5810	0.4420	53.02	20.60	0.5108	0.4938	72.16
CodeFormer + Ours	22.85	**0.6036**	**0.4258**	**41.21**	**21.38**	**0.5514**	**0.4589**	**46.66**

Table 12. **CodeFormer results on data with severe noise level.** We compare our pseudo targets and fine-tuned results with pre-trained CodeFormer and other baselines. Top rows: diffusion-dependent models at test time. Bottom rows: diffusion-free models at test time.

Noise Level	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
moderate	26.79	0.7201	0.3302	32.69	24.65	0.6618	0.3722	37.52
severe	25.78	0.6894	0.3522	34.99	23.75	0.6447	0.3908	40.16

Table 13. **SwinIR results using GT images as pseudo targets.** We show the results of fine-tuning a pre-trained SwinIR using GT clean images on all four of our synthetic data setups (supervised fine-tuning).

Noise Level	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
moderate	24.53	0.6565	0.3598	36.57	23.38	0.6169	0.3881	39.06
severe	24.02	0.6369	0.3755	37.00	22.73	0.5978	0.4063	40.98

Table 14. **CodeFormer results using GT images as pseudo targets.** We show the results of fine-tuning a pre-trained CodeFormer using GT clean images on all four of our synthetic data setups (supervised fine-tuning).

Low Freq. Constraint	Low Pass Filter's N	Timestep K	Timestep L	PSNR↑	SSIM↑	LPIPS↓	FID↓
✓	8	600	360	23.47	0.6564	0.4381	43.01
✓	**16**	**600**	**360**	22.89	0.6317	0.4444	41.13
✓	32	600	360	22.05	0.6127	0.4504	40.92
✓	16	600	0	22.73	0.6114	0.4991	58.10
✓	16	600	180	22.85	0.6204	0.4786	55.03
✓	16	900	360	22.59	0.6277	0.4512	43.20
✗	-	600	-	20.53	0.5874	0.4717	43.51
✗	-	360	-	23.04	0.6319	0.4568	51.84
✗	-	180	-	23.52	0.6446	0.4872	71.18

Table 15. **Quantitative ablation study on the low pass downsampling factor and timestep choices for SwinIR pseudo targets.** We compare the results of the SwinIR pseudo targets with different timesteps choices (K and L) and low pass filter downsampling parameters (N) on 4× downsampling data at *moderate* noise level. Red and blue indicate the best and the second best results. **Bold** indicates our selection.

Low Freq. Constraint	Low Pass Filter's N	Timestep K	Timestep L	PSNR↑	SSIM↑	LPIPS↓	FID↓
✓	4	600	360	23.37	0.6694	0.4346	40.85
✓	**8**	**600**	**360**	23.15	0.6507	0.4364	37.42
✓	16	600	360	22.59	0.6285	0.4458	37.81
✓	8	600	0	23.54	0.6409	0.4506	54.58
✓	8	600	180	23.59	0.6543	0.4368	41.28
✓	8	900	360	22.40	0.6266	0.4511	38.99
✗	-	600	-	20.11	0.5746	0.4857	41.94
✗	-	360	-	22.65	0.6170	0.4440	37.45
✗	-	180	-	22.83	0.5981	0.4498	47.41

Table 16. **Quantitative ablation study on low pass downsampling factor and timestep choices for CodeFormer pseudo targets.** We compare the results of CodeFormer pseudo targets with different timesteps choices (K and L) and low pass filter downsampling parameters (N) on 4× downsampling data at *severe* noise level. Red and blue indicate the best and the second best results. **Bold** indicates our selection.

Low Freq. Constraint	Low Pass Filter's N	Timestep K	Timestep L	PSNR↑	SSIM↑	LPIPS↓	FID↓
✓	8	600	360	25.06	0.6782	0.3803	40.71
✓	**16**	**600**	**360**	24.75	0.6676	0.3853	41.42
✓	32	600	360	24.33	0.6580	0.3943	42.56
✓	16	600	0	24.68	0.6657	0.4183	57.17
✓	16	600	180	24.70	0.6668	0.4080	54.01
✓	16	900	360	24.26	0.6567	0.3919	42.54
✗	-	600	-	23.72	0.6422	0.4047	45.40
✗	-	360	-	25.09	0.6763	0.3896	43.72
✗	-	180	-	25.21	0.6832	0.4146	54.63

Table 17. **Quantitative ablation study on the low pass downsampling factor and timestep choices for SwinIR finetuning.** We compare the results of the SwinIR pseudo targets with different timesteps choices (K and L) and low pass filter downsampling parameters (N) on 4× downsampling data at *moderate* noise level. Red and blue indicate the best and the second best results. **Bold** indicates our selection.

Low Freq. Constraint	Low Pass Filter's N	Timestep K	Timestep L	PSNR↑	SSIM↑	LPIPS↓	FID↓
✓	4	600	360	22.87	0.6055	0.4249	40.71
✓	**8**	**600**	**360**	22.85	0.6036	0.4258	41.21
✓	16	600	360	22.66	0.5941	0.4312	42.30
✓	8	600	0	22.96	0.5927	0.4296	46.73
✓	8	600	180	22.94	0.6017	0.4256	42.27
✓	8	900	360	22.88	0.6049	0.4260	41.61
✗	-	600	-	21.45	0.5487	0.4581	48.72
✗	-	360	-	22.64	0.5869	0.4291	42.40
✗	-	180	-	22.81	0.5849	0.4294	46.00

Table 18. **Quantitative ablation study on low pass downsampling factor and timestep choices for CodeFormer finetuning.** We compare the results of CodeFormer pseudo targets with different timesteps choices (K and L) and low pass filter downsampling parameters (N) on 4× downsampling data at *severe* noise level. Red and blue indicate the best and the second best results. **Bold** indicates our selection.

	Noise Level	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
without	moderate	22.96	0.6316	0.4499	42.44	21.39	0.5808	0.5406	70.88
with	moderate	**23.08**	**0.6407**	**0.4241**	**36.45**	**22.17**	**0.6185**	**0.4493**	**42.00**
without	severe	21.75	0.5941	0.5228	59.71	19.90	0.5323	0.6188	98.49
with	severe	**23.15**	**0.6507**	**0.4364**	**37.42**	**21.14**	**0.5902**	**0.4918**	**50.59**

Table 19. **Pseudo targets with vs. without running a pre-trained CodeFormer.** We compare the pseudo targets generated with and without running a pre-trained CodeFormer before the pseudo target generation process.

	Noise Level	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
without	moderate	23.10	0.6033	0.4170	41.23	21.87	0.5442	0.4556	52.62
with	moderate	**23.20**	**0.6138**	**0.4117**	**41.74**	**22.28**	**0.5848**	**0.4290**	**41.72**
without	severe	22.27	0.5629	0.4505	46.32	20.63	0.4749	0.5041	72.73
with	severe	**22.85**	**0.6036**	**0.4258**	**41.21**	**21.38**	**0.5514**	**0.4589**	**46.66**

Table 20. **CodeFormer fine-tuning results using pseudo targets with vs. without running a pre-trained CodeFormer.** We compare the results of fine-tuning using pseudo targets generated with and without running a pre-trained CodeFormer before the pseudo target generation process.

	PSNR↑	SSIM↑	LPIPS↓	FID↓	Deg.↓	LMD↓
DifFace [31]	20.23	0.5266	0.5993	104.30	60.30	4.88
DiffBIR [15]	20.55	0.4971	0.6669	123.45	**52.15**	4.80
PG-Diff [30]	20.07	0.5445	0.5450	108.46	57.95	4.87
DR2 [29]	19.73	0.5492	0.4950	55.73	67.01	6.37
Our pseudo targets	**21.87**	**0.6094**	**0.4688**	**44.20**	57.38	**4.35**
GFPGAN [26]	19.35	0.4435	0.5634	128.06	52.48	4.37
VQFR [6]	21.58	0.5468	0.5195	76.97	51.28	4.14
CodeFormer [33]	21.75	0.5459	0.4679	62.59	**49.48**	**3.83**
CodeFormer + Ours	**22.12**	**0.5775**	**0.4424**	**43.94**	53.88	4.11

Table 21. **CodeFormer results of facial recognition based metrics on data with severe noise level.** We compare our pseudo targets and fine-tuned results with pre-trained CodeFormer and other baselines on metrics including Deg. [26] and LMD [6]. Top rows: diffusion-dependent models at test time. Bottom rows: diffusion-free models at test time.

	4× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓	8× Downsampling PSNR↑	SSIM↑	LPIPS↓	FID↓
Pre-trained	25.32	0.6909	0.4173	46.34	23.12	0.6341	0.4752	66.43
Fine-tuned	**25.37**	**0.6872**	**0.3658**	**39.03**	**23.75**	**0.6412**	**0.4063**	**48.87**

Table 22. **SwinIR's results on data with *mild* degradations.** We show the effectiveness of our approach on a pre-trained SwinIR on inputs with mild degradations.

	Number of Parameters ↓	Inference Time ↓
DifFace [31]	159.59M	4.0053s
DiffBIR [15]	1666.75M	9.1807s
PG-Diff [30]	159.59M	15.9534s
DR2 [29]	93.56M	1.0812s
GFPGAN [26]	76.21M	0.0249s
VQFR [6]	76.56M	0.1392s
One step of diffusion model [8]	159.59M	0.0402s
SwinIR [14] + Ours	15.79M	0.0311s
CodeFormer [33] + Ours	94.11M	0.0274s

Table 23. **Memory and inference time comparison.** We compare the memory and inference time of the baseline methods with the model architectures we used in our pipeline. Note that the one step of the diffusion model refers to the time it performs one denoising step.

Algorithm 1 Generating pseudo targets (ours)

Input: low-quality restoration output $y_0 = \mathcal{R}(y)$, low-pass filter ϕ_N, pre-defined timesteps L and K where $L < K < T$
Output: pseudo target $\bar{x}_0$ for low-quality input y
$\bar{x}_K \leftarrow$ sample from $\mathcal{N}(y_K; \sqrt{\bar{\alpha}_K}y_0, (1-\bar{\alpha}_K)\mathbf{I})$
for t from K to 1 **do**
  $\bar{x}_{t-1} \leftarrow$ sample from $p_\theta(\bar{x}_{t-1}|\bar{x}_t)$ ▷ unconditional denoising
  if $t > L$ **then**
    $y_{t-1} \leftarrow$ sample from $\mathcal{N}(y_{t-1}; \sqrt{\bar{\alpha}_{t-1}}y_0, (1-\bar{\alpha}_{t-1})\mathbf{I})$
    $\bar{x}_{t-1} \leftarrow \bar{x}_{t-1} - \phi_N(\bar{x}_{t-1}) + \phi_N(y_{t-1})$ ▷ low frequency content constraint
  end if
end for
return $\bar{x}_0$

Algorithm 2 DifFace [31]

Input: output of a pre-trained restoration model $y_0 = \mathcal{R}(y)$, a pre-defined timestep K where $K < T$
Output: clean image x_0 for low-quality input y
$x_K \leftarrow$ sample from $\mathcal{N}(y_K; \sqrt{\bar{\alpha}_K}y_0, (1-\bar{\alpha}_K)\mathbf{I})$
for t from K to 1 **do**
  $x_{t-1} \leftarrow$ sample from $p_\theta(x_{t-1}|x_t)$ ▷ unconditional denoising
end for
return x_0

Algorithm 3 ILVR [1]

Input: low-quality input y, low-pass filter ϕ_N
Output: clean image x_0 for low-quality input y
$x_T \leftarrow$ sample from $\mathcal{N}(\mathbf{0}; \mathbf{I})$
for t from T to 1 **do**
  $x_{t-1} \leftarrow$ sample from $p_\theta(x_{t-1}|x_t)$ ▷ unconditional denoising
  $y_{t-1} \leftarrow$ sample from $\mathcal{N}(y_{t-1}; \sqrt{\bar{\alpha}_{t-1}}y, (1-\bar{\alpha}_{t-1})\mathbf{I})$
  $x_{t-1} \leftarrow x_{t-1} - \phi_N(x_{t-1}) + \phi_N(y_{t-1})$
end for
return x_0

Algorithm 4 DDA [4]

Input: low-quality input y, low-pass filter ϕ_N, a pre-defined timestep K where $K < T$, diffusion model's noise prediction network $\boldsymbol{\epsilon}_\theta$, guidance weight $\boldsymbol{s}$
Output: clean image x_0 for low-quality input y
$x_K \leftarrow$ sample from $\mathcal{N}(y_K; \sqrt{\bar{\alpha}_K}y, (1-\bar{\alpha}_K)\mathbf{I})$
for t from K to 1 **do**
  $\hat{x}_{t-1} \leftarrow$ sample from$p_\theta(x_{t-1}|x_t)$ ▷ unconditional denoising
  $\hat{x}_0 \leftarrow (x_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(x_t, t))/\sqrt{\bar{\alpha}_t}$ ▷ Estimate x_0 from x_t directly
  $x_{t-1} \leftarrow \hat{x}_{t-1} - \boldsymbol{s}\nabla_{x_t}||\phi_N(y) - \phi_N(\hat{x}_0)||_2$ ▷ gradient guidance
end for
return x_0

Algorithm 5 PG-Diff [30]

Input: output of a pre-trained restoration model $y_0 = \mathcal{R}(y)$, pre-defined timesteps τ and K where $\tau < K < T$, unconditional denoising $p_\theta(\hat{x}_{t-1}|\hat{x}_t) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$, diffusion model's noise prediction network $\boldsymbol{\epsilon}_\theta$, guidance weight $\boldsymbol{s}$, number of gradient steps G
Output: clean image x_0 for low-quality input y
$x_T \leftarrow$ sample from $\mathcal{N}(\mathbf{0}; \mathbf{I})$
for t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)$
  $\hat{x}_0 \leftarrow (x_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(x_t, t))/\sqrt{\bar{\alpha}_t}$ ▷ Estimate x_0 from x_t directly
  if $\tau \leq t \leq K$ **then** ▷ multiple guidance steps
    repeat
      $x_t \leftarrow$ sample from $\mathcal{N}(\mu - \boldsymbol{s}\nabla_{\hat{x}_0}||y_0 - \hat{x}_0||_2^2, \Sigma)$
      $\hat{x}_0 \leftarrow (x_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(x_t, t))/\sqrt{\bar{\alpha}_t}$
    until $G-1$ times
  end if
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu - \boldsymbol{s}\nabla_{\hat{x}_0}||y_0 - \hat{x}_0||_2^2, \Sigma)$ ▷ gradient guidance
end for
return x_0

Algorithm 6 DiffBIR [15]

Input: output of a pre-trained restoration model $y_0 = \mathcal{R}(y)$, unconditional denoising of a latent diffusion model $p_\theta(\hat{z}_{t-1}|\hat{z}_t) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$, a fine-tuned latent diffusion model's noise prediction network $\boldsymbol{\epsilon}_\theta$ with text prompt set to empty, latent diffusion model's encoder E and decoder D, guidance weight $\boldsymbol{s}$
Output: clean image x_0 for low-quality input y
$z_{y_0} \leftarrow E(y_0)$
$z_T \leftarrow$ sample from $\mathcal{N}(\mathbf{0}; \mathbf{I})$
for t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)$
  $\hat{z}_0 \leftarrow (z_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(z_t, t))/\sqrt{\bar{\alpha}_t}$ ▷ Estimate z_0 from z_t directly
  $z_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu - \boldsymbol{s}\nabla_{\hat{x}_0}||z_{y_0} - \hat{z}_0||_2^2, \Sigma)$ ▷ gradient guidance
end for
$x_0 \leftarrow D(z_0)$
return x_0

Algorithm 7 DR2 [29]

Input: low-quality input y, low-pass filter ϕ_N, a pre-trained face restoration model f for post-processing, predefined timesteps τ and K where $\tau < K < T$, diffusion model's noise prediction network $\boldsymbol{\epsilon}_\theta$, downsampling factor $r = 2$
Output: clean image x_0 for low-quality input y
$y_0 \leftarrow (y)_{\downarrow_r}$ ▷ Downsampling by a factor of r
$\hat{x}_K \leftarrow$ sample from $\mathcal{N}(y_K; \sqrt{\bar{\alpha}_K}y_0, (1-\bar{\alpha}_K)\mathbf{I})$
for t from K to $(\tau+1)$ **do**
  $\hat{x}_{t-1} \leftarrow$ sample from $p_\theta(\hat{x}_{t-1}|\hat{x}_t)$ ▷ unconditional denoising
  $y_{t-1} \leftarrow$ sample from $\mathcal{N}(y_{t-1}; \sqrt{\bar{\alpha}_{t-1}}y_0, (1-\bar{\alpha}_{t-1})\mathbf{I})$
  $\hat{x}_{t-1} \leftarrow \hat{x}_{t-1} - \phi_N(\hat{x}_{t-1}) + \phi_N(y_{t-1})$ ▷ low frequency content constraint
end for
$\hat{x}_0 \leftarrow (x_\tau - \sqrt{1-\bar{\alpha}_\tau}\boldsymbol{\epsilon}_\theta(x_\tau, \tau))/\sqrt{\bar{\alpha}_\tau}$ ▷ Estimate x_0 from x_τ directly
$\hat{x}_0 \leftarrow (\hat{x}_0)_{\uparrow_r}$
$x_0 \leftarrow f(\hat{x}_0)$ ▷ Run post-processing restoration model
return x_0

Figure 6. Qualitative comparison on testing samples from our Wider-Test-200 (**zoom in for details**).

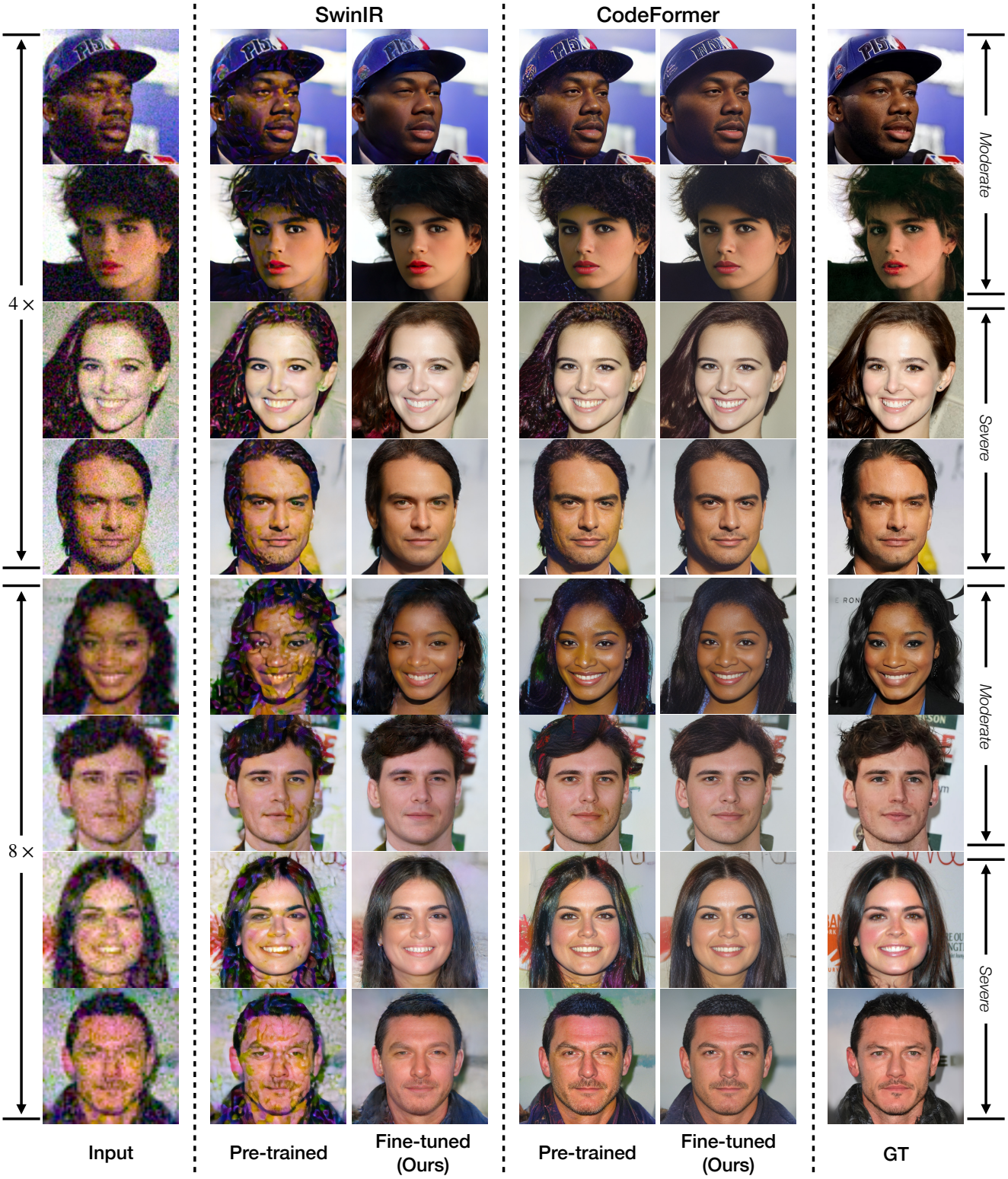

Figure 7. Additional qualitative comparison between pre-trained and fine-tuned models at different degradation levels (**zoom in for details**).

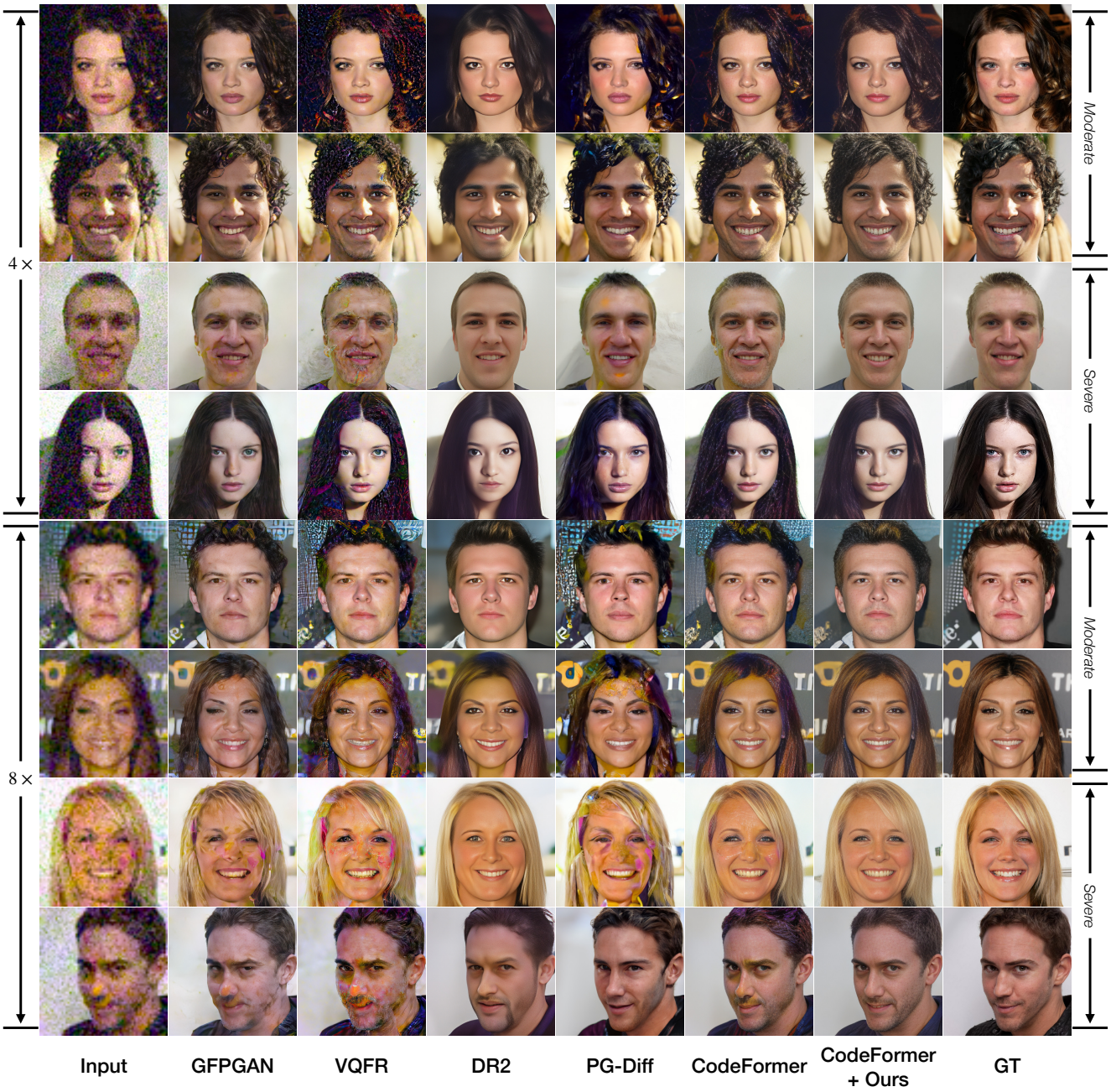

Figure 8. Additional qualitative comparison with other baselines at different degradation levels (**zoom in for details**).

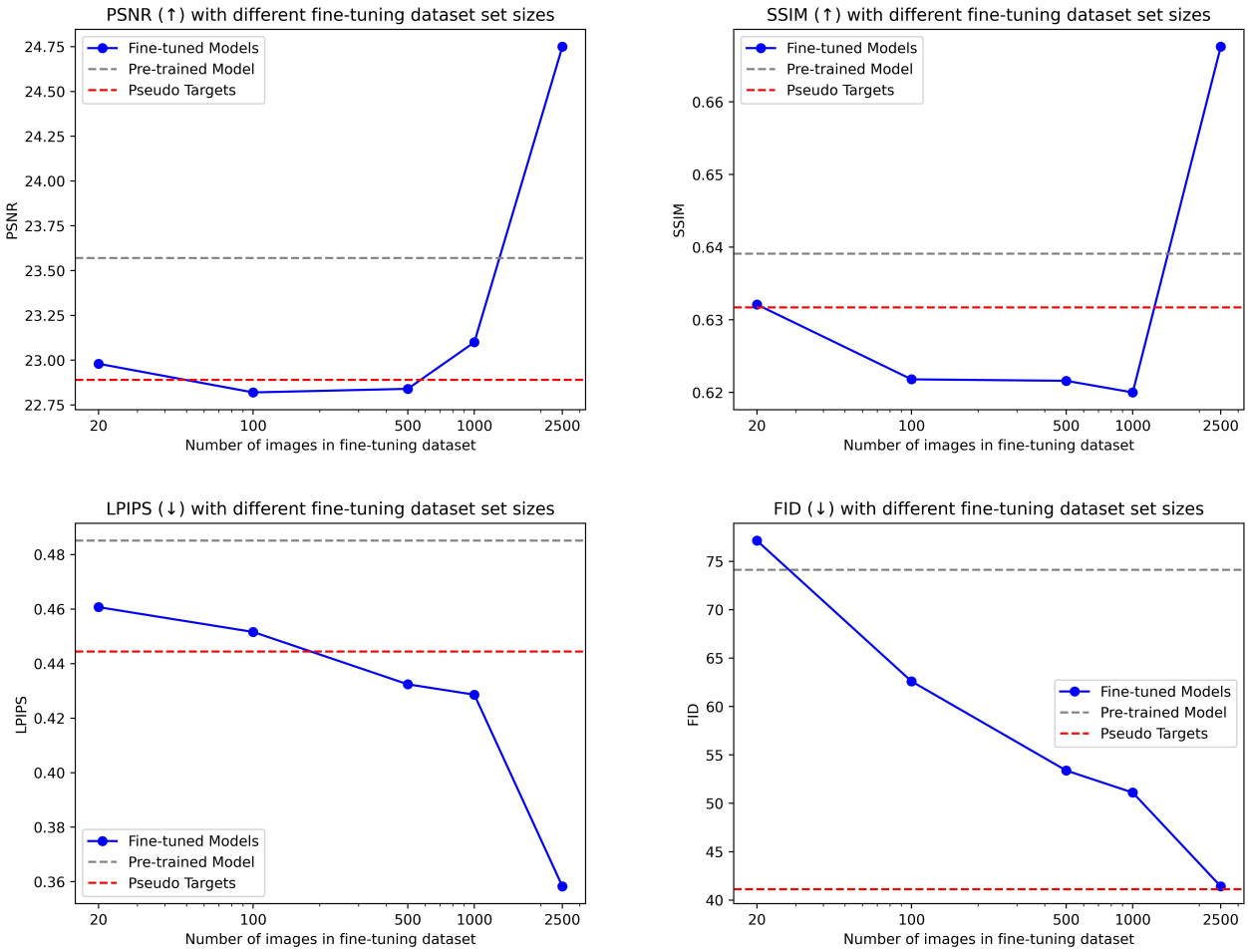

Figure 9. Fine-tuned SwinIR's performance when using different sizes of fine-tuning datasets on 4× downsampling data at *moderate* noise level.

References

[1] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *ICCV*, 2021. 4, 13

[2] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 2, 4

[3] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. In *TIP*, 2008. 1

[4] Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. Back to the source: Diffusion-driven adaptation to test-time corruption. In *CVPR*, 2023. 4, 13

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014. 1

[6] Yuchao Gu et al. Vqfr: Blind face restoration with vector-quantized dictionary and parallel decoder. In *ECCV*, 2022. 4, 9, 12

[7] Yuchao Gu, Xintao Wang, Liangbin Xie, Chao Dong, Gen Li, Ying Shan, and Ming-Ming Cheng. Vqfr: Blind face restoration with vector-quantized dictionary and parallel decoder. In *ECCV*, 2022. 3

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 12

[9] Marco Huber, Anh Thi Luu, Fadi Boutros, Arjan Kuijper, and Naser Damer. Bias and diversity in synthetic-based face recognition. In *WACV*, 2024. 8

[10] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv*, 2017. 1, 7

[11] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 2, 6, 7

[12] Xiaoming Li, Chaofeng Chen, Shangchen Zhou, Xianhui Lin, Wangmeng Zuo, and Lei Zhang. Blind face restoration via deep multi-scale component dictionaries. In *ECCV*, 2020. 1

[13] Xiaoming Li, Ming Liu, Yuting Ye, Wangmeng Zuo, Liang Lin, and Ruigang Yang. Learning warped guidance for blind face restoration. In *ECCV*, 2018. 1

[14] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ICCV*, 2021. 1, 4, 6, 7, 12

[15] Xinqi Lin, Jingwen He, Ziyan Chen, Zhaoyang Lyu, Ben Fei, Bo Dai, Wanli Ouyang, Yu Qiao, and Chao Dong. Diffbir: Towards blind image restoration with generative diffusion prior. *arXiv*, 2023. 4, 6, 8, 9, 12, 14

[16] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Practical signal-dependent noise parameter estimation from a single noisy image. In *TIP*, 2014. 1

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 8

[18] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv*, 2016. 1

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

[20] Markku Makitalo and Alessandro Foi. Optimal inversion of the generalized anscombe transformation for poisson-gaussian noise. In *TIP*, 2012. 1

[21] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023. 3

[22] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 2

[23] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 4

[24] Donghwan Seo, Abhijith Punnappurath, Luxi Zhao, Abdelrahman Abdelhamed, Sai Kiran Tedla, Sanguk Park, Jihwan Choe, and Michael S Brown. Graphics2raw: Mapping computer graphics images to sensor raw images. In *ICCV*, 2023. 1

[25] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023. 3

[26] Xintao Wang et al. Towards real-world blind face restoration with generative facial prior. In *CVPR*, 2021. 1, 9, 12

[27] Xintao Wang, Yu Li, Honglun Zhang, and Ying Shan. Towards real-world blind face restoration with generative facial prior. In *CVPR*, 2021. 3

[28] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *ICCVW*, 2021. 4

[29] Zhixin Wang, Ziying Zhang, Xiaoyun Zhang, Huangjie Zheng, Mingyuan Zhou, Ya Zhang, and Yanfeng Wang. Dr2: Diffusion-based robust degradation remover for blind face restoration. In *CVPR*, 2023. 4, 6, 9, 12, 14

[30] Peiqing Yang, Shangchen Zhou, Qingyi Tao, and Chen Change Loy. Pgdiff: Guiding diffusion models for versatile face restoration via partial guidance. In *NeurIPS*, 2023. 4, 9, 12, 13

[31] Zongsheng Yue and Chen Change Loy. Diffface: Blind face restoration with diffused error contraction. *arXiv*, 2022. 2, 4, 6, 8, 9, 12, 13

[32] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 1

[33] Shangchen Zhou, Kelvin C.K. Chan, Chongyi Li, and Chen Change Loy. Towards robust blind face restoration with codebook lookup transformer. In *NeurIPS*, 2022. 1, 2, 4, 5, 9, 12