Towards Unsupervised Blind Face Restoration using Diffusion Prior

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

Abstract

Blind face restoration methods have shown remarkable performance, particularly when trained on large-scale synthetic datasets with supervised learning. These datasets are often generated by simulating low-quality face images with a handcrafted image degradation pipeline. The models trained on such synthetic degradations, however, cannot deal with inputs of unseen degradations. In this paper, we address this issue by using only a set of input images, with unknown degradations and without ground truth targets, to fine-tune a restoration model that learns to map them to clean and contextually consistent outputs. We utilize a pre-trained diffusion model as a generative prior through which we generate high quality images from the natural image distribution while maintaining the input image content through consistency constraints. These generated images are then used as pseudo targets to fine-tune a pre-trained restoration model. Unlike many recent approaches that employ diffusion models at test time, we only do so during training and thus maintain an efficient inference-time performance. Extensive experiments show that the proposed approach can consistently improve the perceptual quality of pre-trained blind face restoration models while maintaining great consistency with the input contents. Our best model also achieves the state-of-the-art results on both synthetic and real-world datasets. Project Page.

1. Introduction

Image restoration is a fundamental task in computational photography that aims to recover a high-quality image from its degraded low-quality counterpart. Blind image restoration is a more challenging task, where the degradation process is unknown. One needs to find a good balance between maintaining the fidelity of the image content and the output's perceptual quality. This is particularly important in the case of blind face restoration, as both fidelity and quality are important when restoring face images.

Most of the existing blind face restoration methods [19, 70, 75, 82, 89] are trained in a supervised manner using a paired dataset of low-quality inputs and high-quality target images. The training pairs are often constructed by manually designing a degradation process [33, 35, 70], where a high-quality image is synthetically degraded to form the corresponding low-quality input. Supervised learning achieves great performance on test data that aligns with the training degradations (Fig. 1(a)). However, this produces results with severe artifacts when tested on images that do not fall under the training degradation distribution (Fig. 1(b)). In addition, ground-truth data is not always available for the supervised learning setup. We commonly have access to only the low-quality observations in a real-world setting. In this paper, we address such blind unsupervised setup with neither access to the paired ground truth images nor to the degradation process of the inputs.

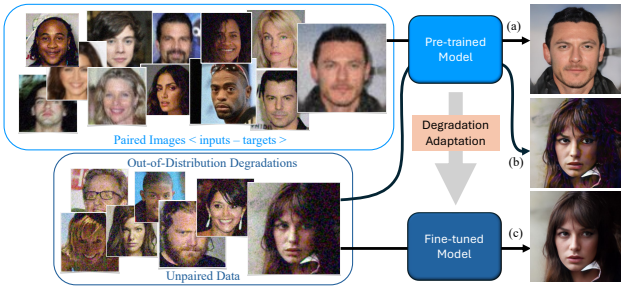

Figure 1. **Overview.** Given a restoration model pre-trained on synthetic datasets in a supervised fashion, it can produce high-quality restoration on low-quality images that are aligned with the degradation distribution used in training (a). However, it often fails on inputs of out-of-distribution degradations (b). We propose an unsupervised pipeline to adapt a pre-trained model to unpaired degraded images of the target degradation with a much smaller data size. This addresses the domain gap in degradation types without paired ground-truth images or the knowledge of the target data's degradation type (c). (**zoom in for details**).

Image diffusion models [23, 56] have recently shown remarkable performance in image generation. Due to their powerful modeling of the natural image manifold, pre-trained diffusion models can be used as priors for image restoration tasks in a zero-shot manner. Some blind image restoration methods [9, 39, 76, 79, 85] have achieved great results on severely degraded data. However, they require

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

running the diffusion model's sampling process during inference, which results in a significant computational cost and extremely slow runtime. On the other hand, a series of works [6, 13, 28, 44, 45, 73, 92] involve designing handcrafted denoising process for known degradation types in a supervised setup, which limits the applicable scenarios.

In this work, we tackle the problem of unsupervised blind image restoration by taking the advantages from the above two groups of works. Given a pre-trained restoration model that fails on inputs with some out-of-distribution degradations (target degradations), our approach consists of two stages: pseudo target generation and model fine-tuning. To generate pseudo targets, we design a denoising diffusion process that cleans up the restoration model's outputs, where it preserves the input image content while considerably enhancing high frequency details. In the second step, the cleaned images are treated as pseudo targets to fine-tune the pre-trained restoration model using input and pseudo target pairs. The fine-tuned model is then able to handle inputs with the same target degradations (Fig. 1(c)). The proposed approach requires only a small set of unpaired low-quality observations for training, and does not require running the diffusion model at test time, which is a much more practical setting for real-world applications. To the best of our knowledge, this is the first approach to use pseudo-targets to adapt a pre-trained restoration model to unknown degradations for blind face restoration.

In summary, our contributions are as follows: (i) An unsupervised pipeline for adaptation of face restoration models to unknown unpaired degradations; (ii) a method to obtain content-preserving pseudo targets from a diffusion model that achieves better fidelity and perceptual quality than previous zero-shot diffusion-based restoration methods; (iii) our approach consistently improves the pre-trained models, and our best fine-tuned model achieves state-of-the-art performance on both synthetic and real-world datasets without the need for running a diffusion model at inference time.

2. Related Work

2.1. Supervised Blind Face Restoration

Most of the existing methods in blind face restoration involve supervised learning with simulated training data pairs, combined with various types of priors. Due to the structured nature of facial images, many works explore face-specific priors such as geometry priors [2, 3, 32, 64, 83, 84, 90, 91] and reference priors [10, 34–36] to retain natural and faithful restoration for the given low-quality faces. To further improve the perceptual quality of the restoration, several models [1, 37, 50, 70, 82] employ GAN-based priors with perceptual and adversarial losses during supervised training, or use pre-trained GAN models [26, 27] as priors directly. Some works [33, 36] explore facial component dictionaries as a more robust prior for higher quality restoration and identity preservation.

Following the high-quality codebook learning approaches [12,69], more recent methods [19,75,88,89] show great performance on real-world degraded faces by first pre-training on large-scale clean face data to obtain high-quality discrete dictionaries or codebooks as priors during restoration. These methods achieve great perceptual quality, but do not generalize to the degradations that are outside of the training degradation distribution. The reliance on the large-scale training data pairs also limit their practicality for real-world applications. Different from the previous supervised approaches, we aim to address the generalization problem of pre-trained face restoration models on out-of-distribution degradations in an unsupervised manner.

2.2. Diffusion Priors for Image Restoration

Diffusion models [23,56] have recently become the most powerful generative models in image synthesis. Despite not being initially designed for low-level imaging tasks, some works modify diffusion model's architecture and train them by conditioning on either the degraded image or its features for tasks such as super-resolution (SR) [17, 46, 59, 61, 78], shadow-removal [21, 46], deblurring [55, 78], inpainting and uncropping [60, 78], face-restoration [87], and adverse weather restoration [46, 53]. Some works [45, 54, 77] explore different sampling and training procedure of diffusion models for better restoration performance. However, all of these models require supervised training with large amount of data pairs and computational resources, while still suffering from lack of adaptability to generalize to out-of-distribution degradations.

A large group of the methods utilize pre-trained diffusion models for zero-shot image restoration tasks [5–7, 11, 13, 14, 28–30, 51, 58, 67, 73, 92] including super-resolution, inpainting, deblurring, denoising, and JPEG artifact correction. However, they require the knowledge of the degradation type to design custom denoising process, which cannot be applied to blind image restoration directly. To tackle the blind restoration problem, where the degradation is unknown, the conditioning or guidance during the denoising process needs to be generalized enough to handle a variety of degradation types. On this line of research, some papers [4, 16, 76] use the low frequency content of the images to guide the denoising process in order to preserve the content of the image. Some other papers [39, 79, 85] use a simple pre-trained restoration model to first reduce the amount of artifacts in the image while preserving the smoothed semantics, and then use a pre-trained or fine-tuned diffusion model to inject sharp textures to the restoration model's output. Although these zero-shot approaches achieve great level of perceptual quality, they share the common problem

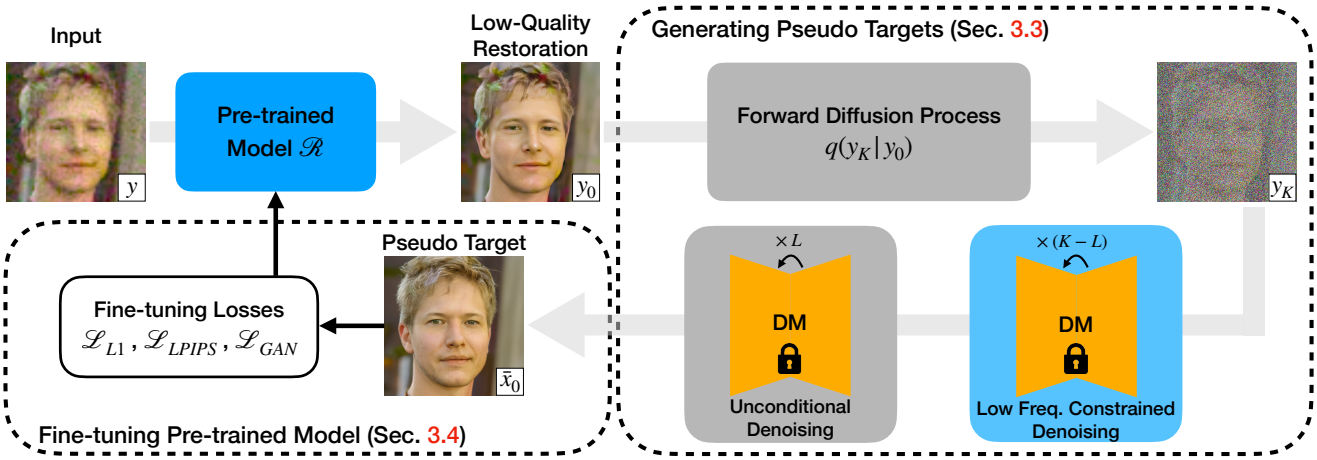

Figure 2. **Overview of our unsupervised fine-tuning pipeline.** Given a pre-trained restoration model that produces low-quality restoration outputs (severe artifacts on hair and over-smoothed skin) on samples with unknown and out-of-distribution degradations, we generate pseudo targets using a pre-trained unconditional diffusion model with a combination of low frequency content constrained denoising and unconditional denoising. The generated clean images can be used as pseudo GT to fine-tune the pre-trained restoration model without the need for real GT images.

of long inference time due to running the diffusion model for every input. There have been efforts to make the diffusion models faster by reducing the number of sampling using DDIM [66] or progressively distilling it into fewer steps [62] at the expense of image quality. Our approach elimiates the burden of running the diffusion model altogether at inference time. It only uses the diffusion model's outputs during training as pseudo targets to improve a pre-trained restoration model by injecting its prior to restore unknown degradations.

3. Method

3.1. Preliminaries on Diffusion Models

Denoising Diffusion Probabilistic Model (DDPM) [23, 65] has been one of the most used and powerful generative models in computer vision. An unconditional diffusion model learns a natural image manifold from large-scale image datasets. It follows a Markov forward process to gradually corrupt an image x_0 with a pre-defined Gaussian noise variance schedule β_t for each timestep $t \in \{1, 2, ..., T\}$. Thanks to the Markovian formulation of the diffusion process, one can write the expression for the noisy image x_t at any timestep t, given the clean image x_0 as

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{1}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

An unconditional diffusion model generates natural images by reversing the forward diffusion process. Specifically, the process can be written as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \sigma_t^2\mathbf{I}), \tag{2}$$

where σ_t^2 is set to a time-dependent constant, and the mean of the denoised image $\boldsymbol{\mu}_\theta(x_t, t)$ is computed as:

$$\boldsymbol{\mu}_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(x_t, t)\Big), \tag{3}$$

where the noise $\boldsymbol{\epsilon}_\theta(x_t, t)$ at timestep t is predicted by a trained timestep conditioned U-Net [57]. To perform unconditional image generation, we can start with a sample from the standard Gaussian distribution $x_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and gradually denoise it using the predicted noise at each timestep. Note that one can use techniques from DDIM [66] or simply uniformly skip timesteps during the reverse diffusion process to accelerate the denoising process.

3.2. Method Overview

Given a restoration model that is pre-trained on synthetic data pairs, its performance on out-of-distribution inputs is often heavily degraded. To bridge the gap without the need for paired ground-truth high-quality images, we use a pre-trained unconditional diffusion model [23] to clean up the artifacts in restoration model's output via a low frequency constrained denoising process. As shown in Fig. 2, our pipeline consists of two stages: 1) generating pseudo targets using a diffusion model (Section 3.3) and 2) using the generated targets to fine-tune the pre-trained restoration model (Section 3.4).

3.3. Generating Pseudo Targets

Consider a pre-trained restoration model, $\mathcal{R}$, and a real-world low-quality image observation y. Due to the domain gap between the synthetic data and the real-world data, the output from a pre-trained restoration model, $y_0 = \mathcal{R}(y)$,

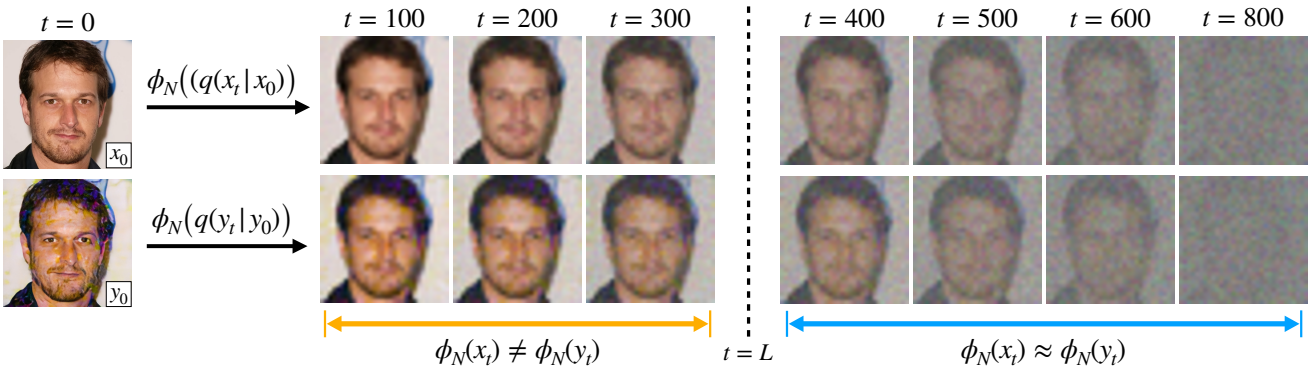

Figure 3. **Visualization of low frequency contents at different timesteps.** We show low frequency contents of the low-quality restoration from a pre-trained SwinIR [38] and its GT counterparts at different timesteps of the forward diffusion process (**zoom in for details**).

still contains a lot of artifacts. Following Eq. (1), we can apply the forward diffusion process on y_0, to get a noisy version of the low-quality restoration y_t at timestep t as:

$$y_t = \sqrt{\bar{\alpha}_t} y_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (4)$$

If we directly perform an unconditional denoising on the noisy image y_t as in [85], the structured content will not be preserved well, which yields inconsistent restoration. Similar to the observations from [9, 76], we found that as more Gaussian noise (larger timestep) is added to the low-quality restoration, the low frequency content of y_t is getting closer to the low frequency content of the noisy image x_t as if we start with the clean image counterpart x_0. Specifically, given a low pass filter ϕ_N and if t is large enough ($t > L$):

$$\phi_N(y_t) \approx \phi_N(x_t) = \phi_N(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}), \quad (5)$$

where x_t is the noisy version of the clean image, and $\boldsymbol{\epsilon}$ is the same sampled noise in Eq. (4). We visualize and compare the low frequency contents of the two images at different timesteps in Fig. 3. The noisy images become closer as timestep increases, and eventually indistinguishable visually after $t = 400$. With this critical observation, we can constrain the denoising process by regularizing the low frequency content at each denoising timestep when $t > L$, in order to preserve the structural information. At lower timesteps ($t <= L$), the low frequency property in Eq. 5 no longer holds and applying such regularization will deteriorate the denoising process. In addition, since we are using an unconditional diffusion model, going all the way to $t = T$ would completely destroy all the information in the image. Hence, we start the low frequency constrained denoising process at a smaller timestep $t = K$, where the low frequency content is not yet destroyed by the injected Gaussian noise.

Combined with the insights above, we describe our pseudo target generation process as follows: we take the restoration model's output, y_0, on an image from the target dataset, and follow the pre-defined noise schedule to inject Gaussian noise into the image up to timestep K. We then pass it to the diffusion model to clean up the image. Similar to [4, 16, 76], we guide the denoising process by constraining the low frequency content to be consistent with the input. This is done by replacing the low frequency content of the denoised image with the corresponding part from the noisy copy of the input image at each time step. Some methods [4, 16] apply such guidance on all denoising steps, which would lead to blurry outputs with artifacts due to over-constraining the denoised images on information that can be a mixture of signal and noise.

Different from previous methods, we only apply this low frequency content constraint for timesteps when $t > L$. This is because the low frequency property does not hold anymore for small timesteps ($t \leq L$) as shown in Fig. 3. Moreover, we observe that the denoised images at these timesteps already have reasonably good structure. There-

Algorithm 1 Generating pseudo targets using a pre-trained diffusion model

Input: low-quality restoration output $y_0 = \mathcal{R}(y)$, low-pass filter ϕ_N
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 $\bar{x}_0$

fore, we perform unconditional denoising for the remaining L timesteps since unconditional denoising steps contribute to high-frequency details at small timesteps [48]. With this approach, there is no need for directly estimating x_0 from x_L in one step and running another enhancement model on the generated image [76]. We summarize the detailed procedure of generating pseudo targets in Algorithm 1.

3.4. Fine-tuning the Pre-trained Models

After obtaining the pseudo targets from the diffusion model, one can fine-tune the pre-trained restoration model with the low-quality inputs and pseudo targets data pairs. We apply the widely used image-level $L1$ loss, perceptual (LPIPS) loss [86], and adversarial (GAN) loss [18]:

$$\mathcal{L}_{L1} = ||\mathcal{R}(y) - \bar{x}_0||_1, \quad \mathcal{L}_{LPIPS} = \text{LPIPS}\big(\mathcal{R}(y), \bar{x}_0\big), \\ \mathcal{L}_{GAN} = \log\big(1 - \mathcal{D}(\mathcal{R}(y))\big), \tag{6}$$

where $\mathcal{R}$ is our restoration model, and $\mathcal{D}$ is a discriminator that outputs the probability of its input coming from the distribution of real natural faces. This discriminator is optimized from scratch along with the restoration model, with the following cross-entropy training objective [18]:

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}_{x\sim\mathcal{R}(y)}\big[-(1 - \log\mathcal{D}(x))\big] + \mathbb{E}_{x\sim\mathbb{P}_r}\big[-\log\mathcal{D}(x)\big], \tag{7}$$

where $\mathbb{P}_r$ represents the distribution of real high-quality face images. In practice we treat the images in FFHQ dataset [26] as our real data distribution. Therefore, we use randomly sampled images from the FFHQ dataset as clean references for optimizing the discriminator. This ensures that the discriminator is robust enough to provide useful gradient signals for optimizing the restoration model. The complete training objective is as follows:

$$\mathcal{L} = \mathcal{L}_{L1} + \lambda_{LPIPS}\mathcal{L}_{LPIPS} + \lambda_{GAN}\mathcal{L}_{GAN}, \tag{8}$$

where λ_{LPIPS} and λ_{GAN} are hyperparameters for the weights of the losses.

4. Experiments

4.1. Implementation and Evaluation Settings

Pre-trained Models. We use a pre-trained unconditional face diffusion model with image resolution of 512×512. This diffusion model is trained by following the training procedure from [8,85] on the entire FFHQ dataset [26] with 70,000 images. We demonstrate the effectiveness of our approach on two model architectures: SwinIR [38] and CodeFormer [89] (current state-of-the-art non-diffusion based method for blind face restoration). For SwinIR, we follow training setup from [70] to pre-train the SwinIR model on the FFHQ [26] dataset following the losses in Eq. (8). For CodeFormer, we use the pre-trained checkpoint from the authors. Both models are trained using the synthetic degradation function as in [33, 35, 70]:

$$\mathcal{I}_{LQ} = \Big\{\big[(\mathcal{I}_{HQ} * \mathbf{k}_\sigma)_{\downarrow_r} + \mathbf{n}_\delta\big]_{JPEG_q}\Big\}_{\uparrow_r}, \tag{9}$$

where the high-quality image $\mathcal{I}_{HQ}$ is first convolved with a Gaussian blur kernel $\mathbf{k}_\sigma$ of kernel size σ and downsampled by factor of r. Then Gaussian noise of standard deviation δ is added, followed by JPEG compression of quality factor q and upsampling by a factor of r to obtain the low-quality image I_{LQ}.

Low Pass Filter and Timesteps. We adopt the low pass filter from [4], where it is implemented as a sequence of downsampling and upsampling with factor of N. For the SwinIR pseudo targets, we set N to be 16. For the CodeFormer pseudo targets, we set N to be 8 for all the 4× data setup, and to be 16 for all the 8×. We set the starting timestep to be $K = 600$ and apply the low frequency constrained denoising process for 240 timesteps. As a result, we start the unconditional denoising at the timestep of $L = 360$.

Fine-tuning Setup. For fine-tuning the SwinIR model, we set the weights of the losses to be $\lambda_{LPIPS} = 0.1$ and $\lambda_{GAN} = 0.1$ for all the experiments. For CodeFormer [89], we follow their training setup and empirically found that only adopting their code-level losses to optimize the code prediction module and the VQ-GAN encoder gives better fine-tuning performance than the image-level losses in Eq. (8). Note that this fine-tuning procedure is specific to CodeFormer architecture and cannot be generalized to all model architectures. Please refer to the supplementary material for more details on our fine-tuning experiments.

Testing Datasets. We evaluate our pipeline on both synthetic and real-world datasets. For the synthetic dataset, we generate low-quality testing inputs using 3,000 high-quality face images from the CelebA-HQ [25] dataset. To simulate more realistic degradations instead of the the commonly used pipeline (Eq. (9)) in [33, 35, 70], we apply the following degradation model:

$$\mathcal{I}_{LQ} = \Big\{ISP\big[ISP^{-1}((\mathcal{I}_{HQ})_{\downarrow_r}) + \mathbf{n}_c\big]\Big\}_{\uparrow_r}, \tag{10}$$

where we first unprocess (ISP^{-1}) the downsampled high-quality images $\mathcal{I}_{HQ}$ to RAW [63], and then apply the widely used camera noise models [15,40,47] $\mathbf{n}_c$ to simulate noisy RAW images. We then render (ISP) the images back to sRGB [63] and upsample the images to the same resolution as the high-quality ones. We construct datasets at **4**× and **8**× downsampling factors $\downarrow_r$. For each downsampling factor, we generate degraded images at *moderate* and *severe* noise levels. This yield four sets with each set containing 3,000 input-GT pairs. We use 2,500 images to generate the pseudo targets for fine-tuning, and use the remaining 500 pairs for evaluation. Note that during fine-tuning, we do not

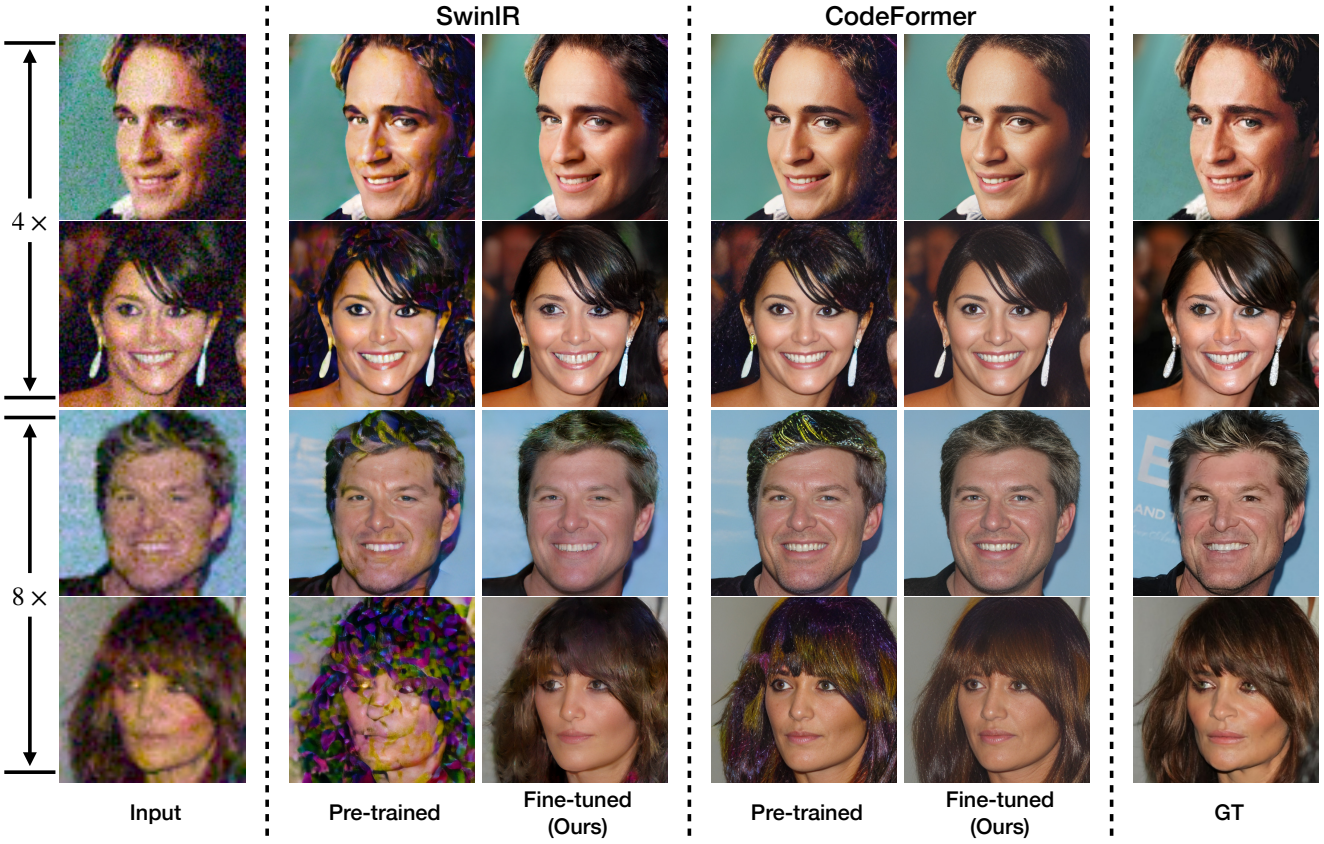

Figure 4. **Qualitative comparison of pre-trained versus fine-tuned models.** We show the effectiveness of our proposed approach to a pre-trained SwinIR [38] and a pre-trained CodeFormer [89] models on 4× and 8× downsampled data at *moderate* noise level. The fine-tuned models are able to produce realistic restoration (**zoom in for details**).

	4× Downsampling				8× Downsampling			
	PSNR↑	SSIM↑	LPIPS↓	FID↓	PSNR↑	SSIM↑	LPIPS↓	FID↓
SwinIR [38]	21.28 / 20.92	0.5744 / 0.5444	0.5446 / 0.5842	74.12 / 120.44	21.28 / 19.00	0.5744 / 0.4813	0.5446 / 0.6435	99.44 / 152.90
SwinIR + Ours	**24.75 / 23.41**	**0.6676 / 0.6284**	**0.3853 / 0.4156**	**41.42 / 49.46**	**23.28 / 21.91**	**0.6206 / 0.5720**	**0.4348 / 0.4821**	**68.25 / 115.97**
CodeFormer [89]	**24.31 / 22.90**	**0.6335** / 0.5810	**0.4007** / 0.4420	**40.66** / 53.02	22.19 / 20.60	0.5716 / 0.5108	0.4420 / 0.4938	51.91 / 72.16
CodeFormer + Ours	23.20 / 22.85	0.6138 / **0.6036**	0.4117 / **0.4258**	41.74 / **41.21**	**22.28 / 21.38**	**0.5848 / 0.5514**	**0.4290 / 0.4589**	**41.72 / 46.66**

Table 1. Improvements gained on at 4× downsampling and 8× downsampling test sets at *moderate* and *severe* noise levels for pre-trained SwinIR [38] and CodeFormer [89]. Numbers presented as: [*moderate* / *severe*].

use the GT images and each set is fine-tuned independently. Due to limited space, we average the results on the *moderate* and *severe* settings when presenting results for each one of **4**× and **8**× downsampling factors. More details on our dataset synthesis pipeline and the detailed results on each one of the four sets are provided in the supplementary material. For the real-world dataset, we use the Wider-Test set from [89] for fine-tuning the pre-trained model, and we select another 200 severely degraded images from the testing set of the Wider-Face dataset [80] as **Wider-Test-200** for evaluating fine-tuned models. Note that our Wider-Test-200 has no overlap with the set we obtained from [89].

Evaluation Metrics. For synthetic datasets, we report PSNR, SSIM [74], and LPIPS [86], as we have access to the ground-truth images. For our Wider-Test-200 set, we report the non-reference image quality metrics (MANIQA [81] and MUSIQ [31]). In addition, we report the commonly used FID scores [22] for all datasets, where we use the distribution of the ground-truth images as the reference statistics for synthetic dataset. Since the Wider-Test-200 does not contain the ground-truth images, we use statistics of the FFHQ dataset [26] as reference to measure the FID score.

4.2. Results

Pre-trained vs. fine-tuned models. We first compare the performance of the pre-trained models with the fine-tuned

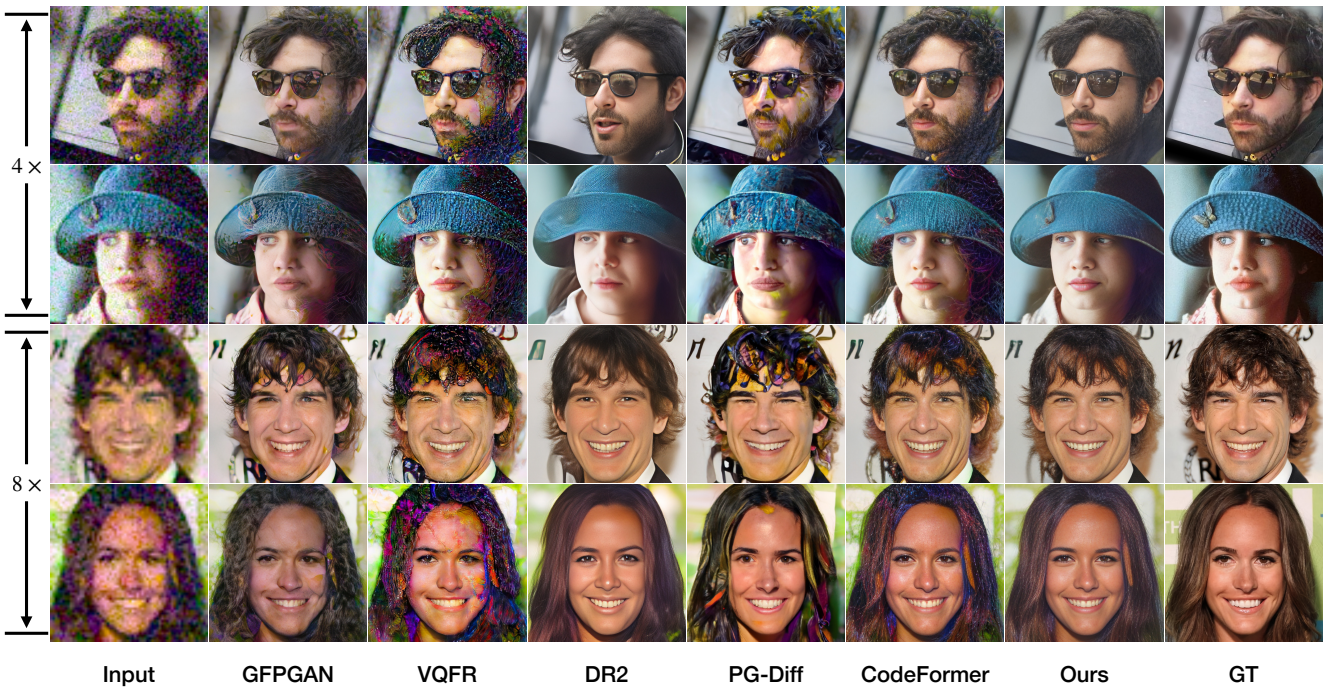

Figure 5. **Qualitative comparison with SOTA baselines on synthetic datasets.** Our fine-tuned CodeFormer model outperforms all other baselines and its pre-trained counterparts on severely degraded inputs from both 4× downsampling and 8× downsampling inputs (**zoom in for details**).

	PSNR↑	SSIM↑	LPIPS↓	FID↓	Diffusion at inference time?
DifFace [85]	20.02	0.5225	0.6077	110.36	✓
DiffBIR [39]	20.28	0.4959	0.6605	126.44	✓
PG-Diff [79]	19.50	0.5339	0.5484	119.82	✓
DR2 [76]	20.34	0.5658	0.4815	54.94	✓
Our targets	**21.66**	**0.6044**	**0.4706**	**46.29**	✓
GFPGAN [70]	21.09	0.5283	0.5298	82.90	✗
VQFR [19]	19.38	0.4540	0.5567	123.12	✗
CodeFormer [89]	21.40	0.5412	0.4679	62.04	✗
CodeFormer + Ours	**21.83**	**0.5681**	**0.4440**	**44.19**	✗

Table 2. **Results on data at 8× downsampling test set.** Top rows: diffusion-dependent models at test time. Bottom rows: diffusion-free models at test time.

	MANIQA↑	MUSIQ↑	FID↓	Diffusion at inference time?
DifFace [85]	0.5252	55.10	89.19	✓
DiffBIR [39]	0.5994	62.42	92.33	✓
PG-Diff [79]	0.5643	61.69	97.82	✓
DR2 [76]	**0.6007**	**68.05**	90.45	✓
Ours targets	0.5772	62.56	**80.60**	✓
GFPGAN [70]	0.5864	67.14	87.71	✗
VQFR [19]	0.5929	69.19	91.76	✗
CodeFormer [89]	0.6082	66.46	87.60	✗
CodeFormer + Ours	**0.6343**	**73.02**	**84.65**	✗

Table 3. **Results on Wider-Test-200 set.** Top rows: diffusion-dependent models at test time. Bottom rows: diffusion-free models at test time.

ones (obtained following our proposed approach in Section 3). We show the qualitative comparison in Fig. 4.

For SwinIR [38], the pre-trained model produces lots of artifacts. We achieve significant improvements on the perceptual quality of the restoration after fine-tuning. We observe that the pre-trained CodeFormer [89] already outputs faces in relatively good quality at lower degradation levels, but still produces noticeable artifacts on the 8× downsampling data, especially in dark and hair regions. Our fine-tuned model is able to remove such artifacts.

We show quantitative comparison for the two models in Tab. 1. For SwinIR, the pre-trained model receives consistent improvements after fine-tuning. The perceptual improvements are reflected in terms of the large boost in FID and LPIPS scores. For CodeFormer, we found it to be more resistant to smaller degradations (4× *moderate*), and hence the fine-tuning mostly makes the images more realistic by removing the remaining artifacts (as observed in Fig. 4 top two rows). On inputs with larger degradations (4× *severe* and 8×), our approach consistently improves the pre-trained model.

Comparison with state-of-the-art models. We benchmark our best fine-tuned model (fine-tuned CodeFormer) against other blind face restoration baselines in Tab. 2. Our approach (bottom row) outperforms all baseline methods. In top rows we provide results of the zero-shot diffusion-based baselines that use a diffusion model at test time and compare them with our pseudo targets. Our targets improve on all

Number of images	PSNR↑	SSIM↑	LPIPS↓	FID↓	MANIQA↑	MUSIQ↑
Pre-trained	23.57	0.6391	0.4851	74.12	0.4623	64.38
20	22.98	0.6321	0.4607	77.14	0.5239	59.05
100	22.82	0.6218	0.4516	62.60	0.5748	68.56
500	22.84	0.6216	0.4324	53.39	0.5932	72.17
1000	23.10	0.6200	0.4286	51.11	0.5918	72.80
2500	24.75	0.6676	0.3853	41.42	0.6023	73.36

Table 4. **Effects of varying fine-tuning dataset size for SwinIR.**

metrics compared to these models, showing our approach can better preserve the content in the input image (higher PSNR and SSIM) while being more realistic according to LPIPS and FID scores.

The bottom rows in Tab. 2 show test-time non-diffusion based models, compared to which we improve consistently on all metrics. This group contains closer baseline models to ours, as none of them require running a diffusion model at test time, and hence are much more efficient. An interesting observation is that on some metrics our fine-tuned results are better than our pseudo targets. We believe a mixture of inductive bias from the pre-trained model together with distilled information from the diffusion model is injected into the parameters of the restoration model. In addition, directly learning from samples of the target (fine-tuning) dataset helps better generalize to target degradations. As shown in the first ablation study (Sec. 4.3), using more samples of the target dataset leads to better fine-tuned models.

Evaluation on real-world dataset. In Tab. 3, we compare our fine-tuned model with baselines on real-world degradations (Wider-Test-200). We improve on all non-reference based metrics compared to all baselines. We provide qualitative results on Wider-Test-200 in the supplementary material. One observation is that DR2 obtains higher MANIQA and MUSIQ while our approach gets higher FID. We believe that it is due to compromising fidelity for quality, as observed in examples in the visual results of Wider-Test-200 in the supplementary. Our fine-tuned model, however, improves on all metrics on this real-world dataset.

4.3. Ablation Study

Effects of fine-tuning dataset size. We investigate the effects of varying fine-tuning dataset size. In Tab. 4, we compare the results of the fine-tuning pre-trained SwinIR fine-tuning datasets of different sizes (i.e. different number of low-quality and pseudo target pairs). We starts to obtain consistent improvements even with only 100 fine-tuning images. The fine-tuned SwinIR becomes worse with 20 fine-tuning images. We believe that the SwinIR has over-fitted to this extremely small fine-tuning dataset in this case, which causes slight performance degradation. We also provide the same ablation on fine-tuned CodeFormer in Tab. 9.

Effects of timesteps and low pass filter choices. In the pseduo target generation stage, our approach involves applying a forward diffusion process on low-quality restoration to timestep $t = K$, and then utilizing low frequency constrained denoising process from timestep $t = K$ down to $t = L$, followed by the standard unconditional denoising for the rest of the timesteps. Having downsampling factor of $N = 8$ for the low pass filter gives better fidelity to the inputs, but with artifacts carried over, while larger N produces targets with higher perceptual quality. Overall, $N = 16$ gives the best balance between fidelity and quality (Fig. 6 (a)). As shown in Fig. 6 (b), applying low frequency constraint down to $L = 180$ and to $L = 0$ would also produce lower quality images due to artifacts in the low frequency content at small timesteps, while our approach is robust to the starting timestep ($K = 900$). We also show the results of unconditional denoising for all timesteps in (c), where small timesteps of ($K = 180$) and ($K = 360$) produces artifacts, and a large timestep of ($K = 600$) distorts the image content due to the lack of regularization during denoising.

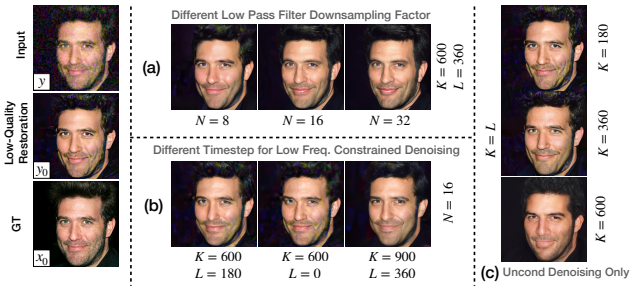

Figure 6. **Effects of low pass downsampling factor and timestep choices on pseudo target.** In (a) we show the effects of adjusting the low pass filter downsampling factor (N); In (b) we show the effects of different timestep windows for low frequency constrained denoising (K and L); In (c) we show the pseudo targets if only unconditional denoising is applied (**zoom in for details**). A large version of this figure is included in the supplementary material for better visualization of the details.

5. Conclusion

In this paper, we propose an unsupervised approach for blind face restoration that addresses the problem of pre-trained restoration models failing on out-of-distribution degradations. Our method requires neither paired ground-truth high-quality images nor knowledge of the inputs' degradation process. Our two-stage pipeline starts by generating pseudo targets using a pre-trained diffusion model with a combination of low frequency constrained denoising and unconditional denoising. We then fine-tune the pre-trained models with pairs of low-quality inputs and pseudo targets. Our approach can consistently improve a pre-trained model's performance on out-of-distribution degradations, producing realistic restoration with satisfactory balance of fidelity and perceptual quality. Our best fine-tuned model achieves state-of-the-art blind face restoration on both synthetic and real-world datasets.

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 [33, 35, 70] to create input-GT training data pairs:

$$\mathcal{I}_{LQ} = \Big\{ \big[(\mathcal{I}_{HQ} * \mathbf{k}_\sigma)_{\downarrow_r} + \mathbf{n}_\delta \big]_{JPEG_q} \Big\}_{\uparrow_r}, \tag{11}$$

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 [38] 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 [89] 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:

$$\mathcal{I}_{LQ} = \Big\{ ISP\big[ISP^{-1}((\mathcal{I}_{HQ})_{\downarrow_r}) + \mathbf{n}_c \big] \Big\}_{\uparrow_r}, \tag{12}$$

where (ISP^{-1}) is an image unprocessing pipeline [63] that converts a sRGB image to RAW, (ISP) is the reverse procedure of image unprocessing to render the sRGB image from RAW image [63], 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 [15, 40, 47] 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 [25]. 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 [38] and CodeFormer [89] are pre-trained on the entire FFHQ dataset [26], which contains 70,000 high-quality face images with resolution of 512×512. The training data pairs are generated by following the degradation pipeline described in Eq. (11).

We train the SwinIR from scratch using the AdamW optimizer [43] with initial learning rate of $1e{-}4$ and update the learning rate following cosine annealing [42]. We use batch size of 16 and train the model for 800,000 iterations in total with image-level $L1$ loss, perceptual (LPIPS) loss [86], and adversarial (GAN) loss [18]. For losses, we use the same set 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 [89]. 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 [8] and the adopt the model weights from [85]. This model is also trained on FFHQ dataset [26]. The total number of timesteps is $T = 1000$ and we follow techniques from [52] 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.

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

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):

$$\mathcal{L} = \mathcal{L}_{code}^{feat} + \lambda_{token}\mathcal{L}_{code}^{token}, \tag{13}$$

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. 5. 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. 13 and 14. We also provide qualitative results on our Wider-Test-200 set in Fig. 12.

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. 14. 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. 15 and 16 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. 13). 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. 17 and 18. 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. 26. The results show consistent improvement of our fine-tuned model compared to the pre-trained ones.

D.2.2 Comparison on face recognition metrics

We report two face recognition metrics: Deg. [71] and LMD [20] in Tab. 25. 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. 27, 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 [49, 68].

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 down-

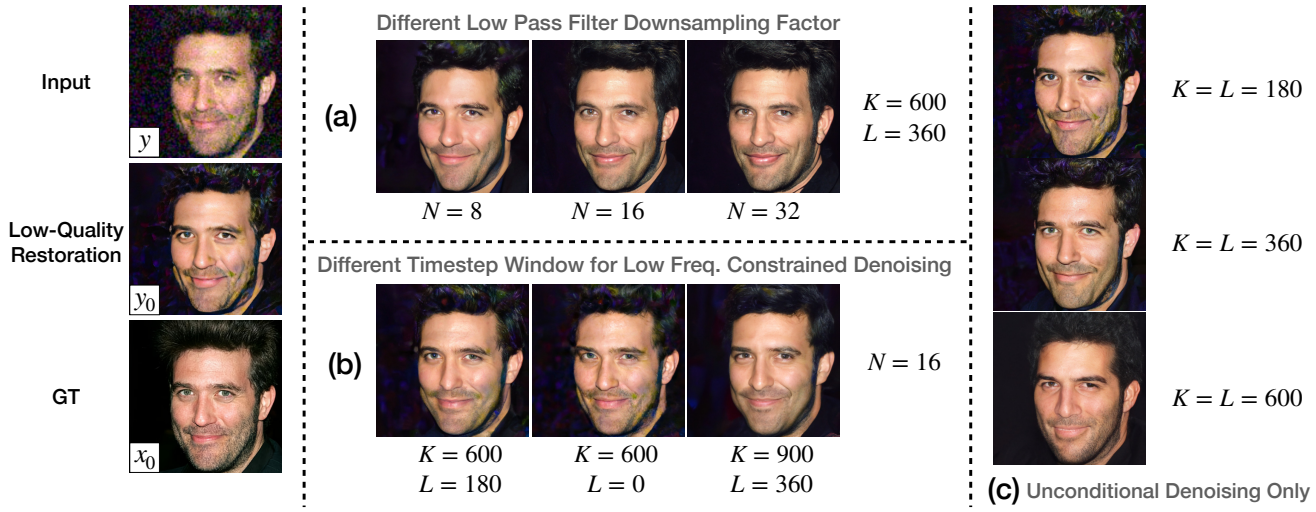

Figure 7. **Effects of low pass downsampling factor and timestep choices on SwinIR pseudo target.** In (a) we show the effects of adjusting the low pass filter downsampling factor (N); In (b) we show the effects of different timestep windows for low frequency constrained denoising (K and L); In (c) we show the pseudo targets if only unconditional denoising is applied (**zoom in for details**).

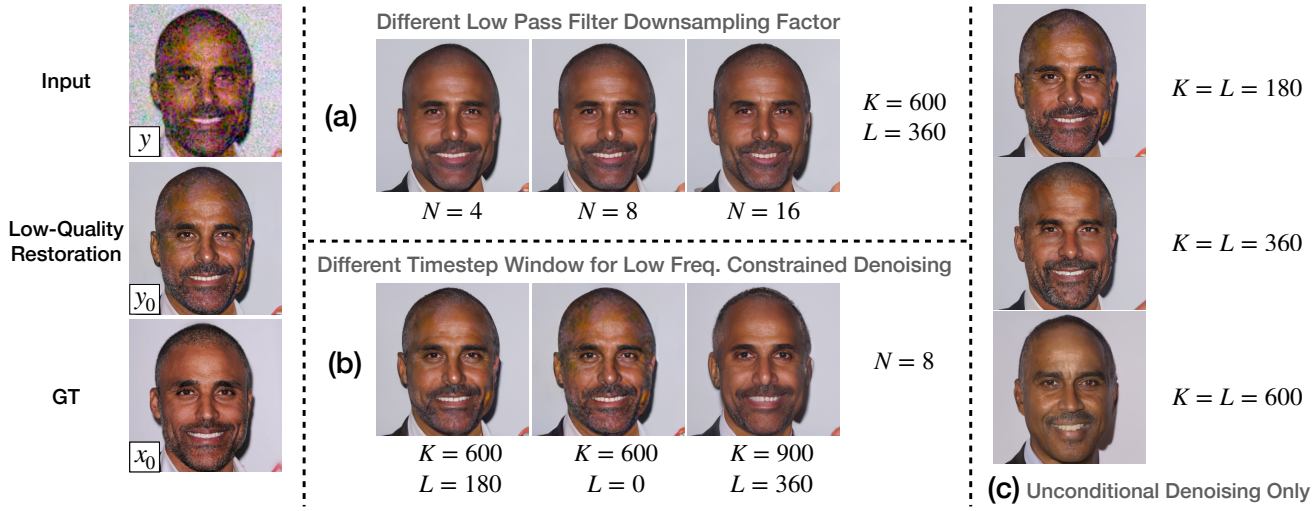

Figure 8. **Effects of low pass downsampling factor and timestep choices on CodeFormer pseudo target.** In (a) we show the effects of adjusting the low pass filter downsampling factor (N); In (b) we show the effects of different timestep windows for low frequency constrained denoising (K and L); In (c) we show the pseudo targets if only unconditional denoising is applied (**zoom in for details**).

sampling factor have significant impact on the quality and fidelity of the pseudo targets. In Figs. 7 and 8, 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 [38] and CodeFormer [89] in Tabs. 19 and 20, as well as the results of fine-tuning using the corresponding targets in Tabs. 21 and 22. 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. 6, 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

$\mathcal{L}_1$	$\mathcal{L}_{LPIPS}$	$\mathcal{L}_{GAN}$	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 6. **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 7. **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.

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. 7, 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. 8, 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 Tabs. 4 and 9, 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 the potential over-fitting.

In Fig. 15, 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 [85], ILVR [4], DDA [16], PG-Diff [79], DiffBIR [39], and DR2 [76]. We summarize the detailed procedures in Algorithms 2 to 8, respectively.

To highlight the main differences, DifFace [85] adds Gaussian noise to the output of a preprocessing model [38], and applies standard unconditional denoising to get the clean image. ILVR [4] and DDA [16] 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

$\mathcal{L}_{code}^{feat}$	$\mathcal{L}_{code}^{token}$	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 8. **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 9. **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.

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 [79] and DiffBIR [39] use the output of a pre-processing model [38, 72] on the input image to guide the denoising process. PG-Diff uses an unconditional face diffusion model [8], while DiffBIR uses a fine-tuned stable-diffusion model [56]. DR2 [76] 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 [19] 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 [76] 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 in Tab. 10. With our pseudo targets, both fine-tuned models can produce more realistic results.

E.2. CodeFormer Fidelity Weight

CodeFormer [89] 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. 9. 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.

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 10. **Our pseudo targets vs. DR2 [76] 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.

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

As shown in the qualitative results in the main paper and in Fig. 13 in this supplementary material, a pre-trained model effectively removes some artifacts from the inputs,

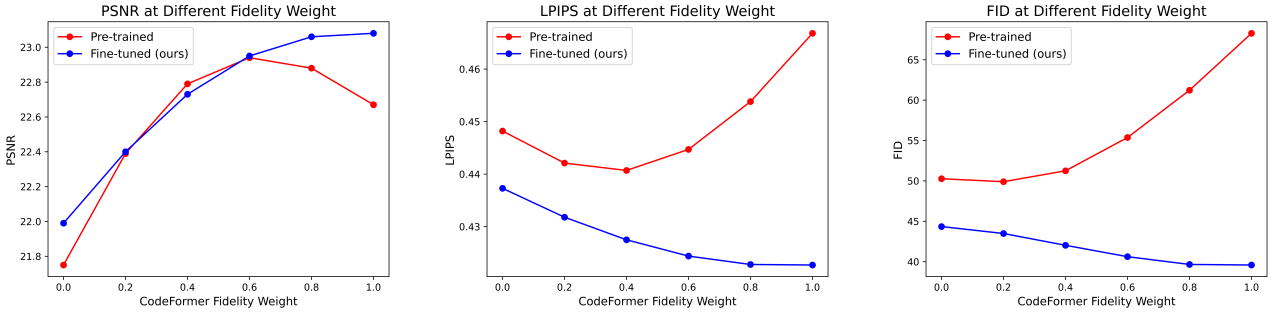

Figure 9. **CodeFormer results using different fidelity weights.** We plot the results of pre-trained and fine-tuned CodeFormer when using different fidelity weights at testing time on 4× downsampling data at *severe* noise level.

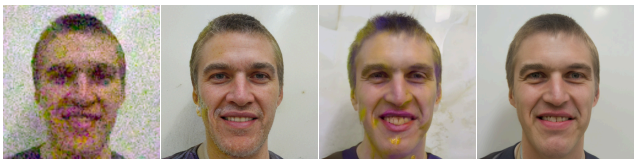

Figure 10. **Pseudo target quality without pre-trained restoration model.** We show the pseudo targets with and without running a pre-trained CodeFormer before applying target generation.

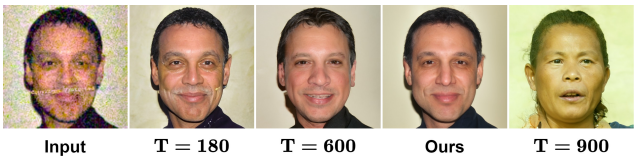

Figure 11. **Pseudo target quality vs. fidelity.** We show the pseudo targets generated with unconditional denosing up to timestep of 900 and with our optimal timestep setup (combination of low frequency constrained denoising and unconditional denoising). When the timestep is large enough, the generated target becomes a completely different face.

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. 10. 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. 23, 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. 24) 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 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. 11). This means that one can generate pseudo targets with great image quality while completely losing the identity information

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 11. **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.

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 12. **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.

of the face to be restored, as shown in Fig. 11. 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. 12, 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 [85] and DiffBIR [39] 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 [38] that is pre-trained on the FFHQ dataset [26] 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. 13, 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. 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

Model	With which SwinIR?	PSNR↑	SSIM↑	LPIPS↓	FID↓
DifFace [85]	Pre-trained	21.21	0.5645	0.5504	84.99
	Fine-tuned	**23.11**	**0.6210**	**0.4438**	**60.38**
DiffBIR [39]	Pre-trained	21.59	0.5371	0.6172	100.81
	Fine-tuned	**22.89**	**0.5769**	**0.4750**	**55.78**

Table 13. **Improved DifFace [85] and DiffBIR [39] 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.

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, 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 [38], 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.

G. Potential Negative Impact

In this work, we use the FFHQ dataset [26] to pre-train and use a subset of the CelebA-HQ [25] 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 dataset [41]. Our trained models will inherit the existing bias in these face datasets, particularly the imbalance in the distribution of gender, ethnicity, and age [24]. 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
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
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 14. **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 [85]	23.32	0.6399	0.4467	43.48	21.05	0.5649	0.5564	87.91
DiffBIR [39]	**23.95**	0.6185	0.5217	64.33	21.30	0.5356	0.6134	92.02
PG-Diff [79]	23.41	**0.6515**	0.4235	41.30	20.68	0.5799	0.4992	87.53
DR2 [76]	22.01	0.6168	0.4400	55.78	21.35	0.5962	0.4548	51.58
Ours targets	23.08	0.6407	**0.4241**	**36.45**	**22.17**	**0.6185**	**0.4493**	**42.00**
GFPGAN [70]	24.12	**0.6454**	0.4385	45.76	21.90	0.5629	0.5013	67.94
VQFR [19]	21.78	0.5372	0.4711	83.84	20.22	0.4919	0.5184	104.12
CodeFormer [89]	**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 15. **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 [85]	21.46	0.5731	0.5396	75.78	18.99	0.4800	0.6589	132.81
DiffBIR [39]	21.83	0.5380	0.6262	86.05	19.26	0.4562	0.7076	160.85
PG-Diff [79]	21.82	0.6012	0.4923	64.82	18.31	0.4879	0.5976	152.10
DR2 [76]	20.13	0.5631	0.4818	53.16	19.32	0.5353	0.5082	58.30
Ours targets	**23.15**	**0.6507**	**0.4364**	**37.42**	**21.14**	**0.5902**	**0.4918**	**50.59**
GFPGAN [70]	22.89	0.5999	0.4807	56.07	20.27	0.4937	0.5582	97.86
VQFR [19]	20.16	0.4709	0.5318	114.00	18.54	0.4161	0.5950	142.11
CodeFormer [89]	**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 16. **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 17. **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 18. **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 19. **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 20. **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 21. **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 22. **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 23. **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 24. **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 [85]	20.23	0.5266	0.5993	104.30	60.30	4.88
DiffBIR [39]	20.55	0.4971	0.6669	123.45	**52.15**	4.80
PG-Diff [79]	20.07	0.5445	0.5450	108.46	57.95	4.87
DR2 [76]	19.73	0.5492	0.4950	55.73	67.01	6.37
Ours targets	**21.87**	**0.6094**	**0.4688**	**44.20**	57.38	**4.35**
GFPGAN [70]	19.35	0.4435	0.5634	128.06	52.48	4.37
VQFR [19]	21.58	0.5468	0.5195	76.97	51.28	4.14
CodeFormer [89]	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 25. **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. [70] and LMD [19]. 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 26. **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 [85]	159.59M	4.0053s
DiffBIR [39]	1666.75M	9.1807s
PG-Diff [79]	159.59M	15.9534s
DR2 [76]	93.56M	1.0812s
GFPGAN [70]	76.21M	0.0249s
VQFR [19]	76.56M	0.1392s
One step of diffusion model [23]	159.59M	0.0402s
SwinIR [38] + Ours	15.79M	0.0311s
CodeFormer [89] + Ours	94.11M	0.0274s

Table 27. **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 2 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 3 DifFace [85]

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 4 ILVR [4]

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 5 DDA [16]

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 6 PG-Diff [79]

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 7 DiffBIR [39]

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 8 DR2 [76]

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

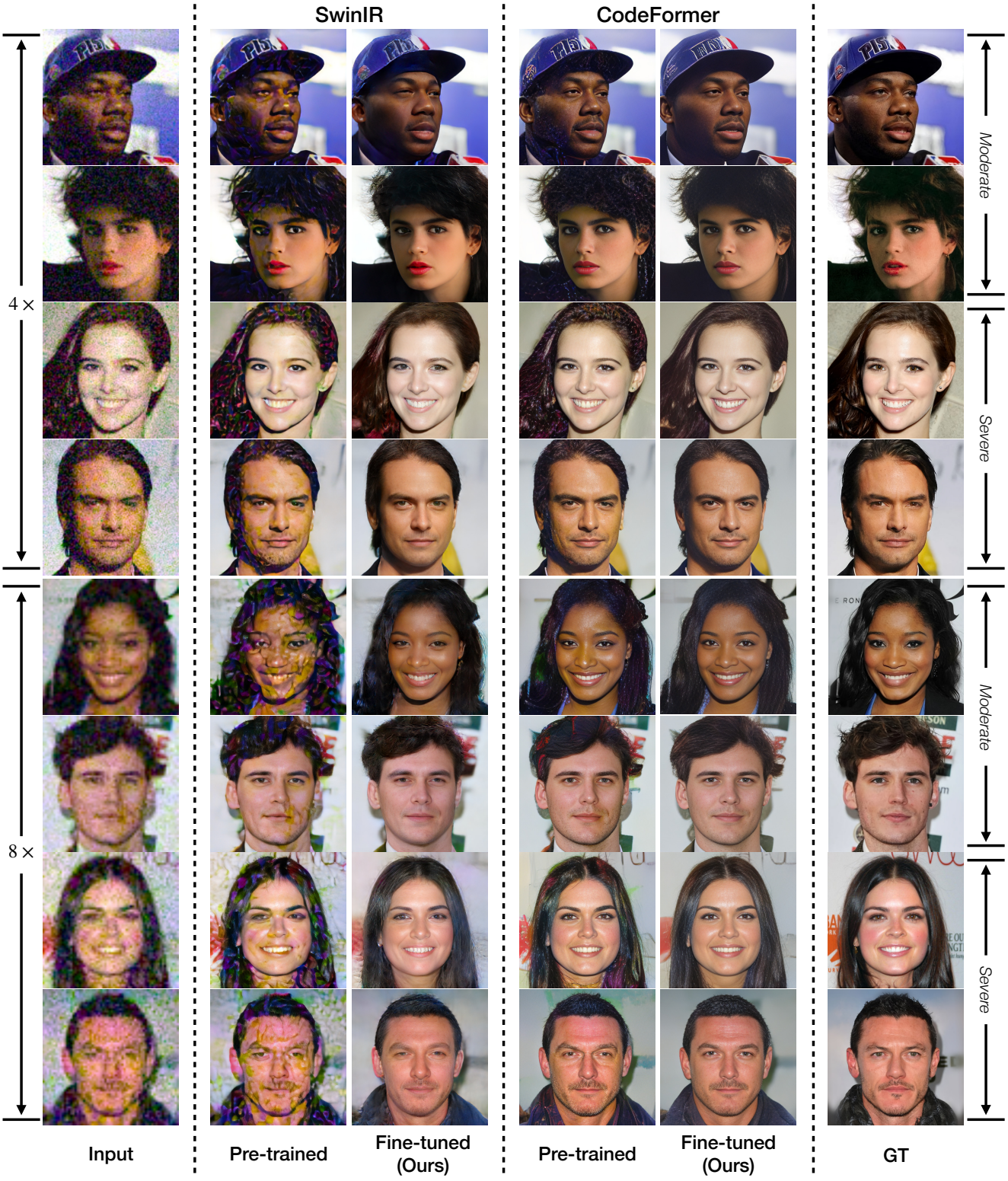

Figure 13. Additional qualitative comparison between pre-trained and fine-tuned models at different degradation levels (**zoom in for details**).

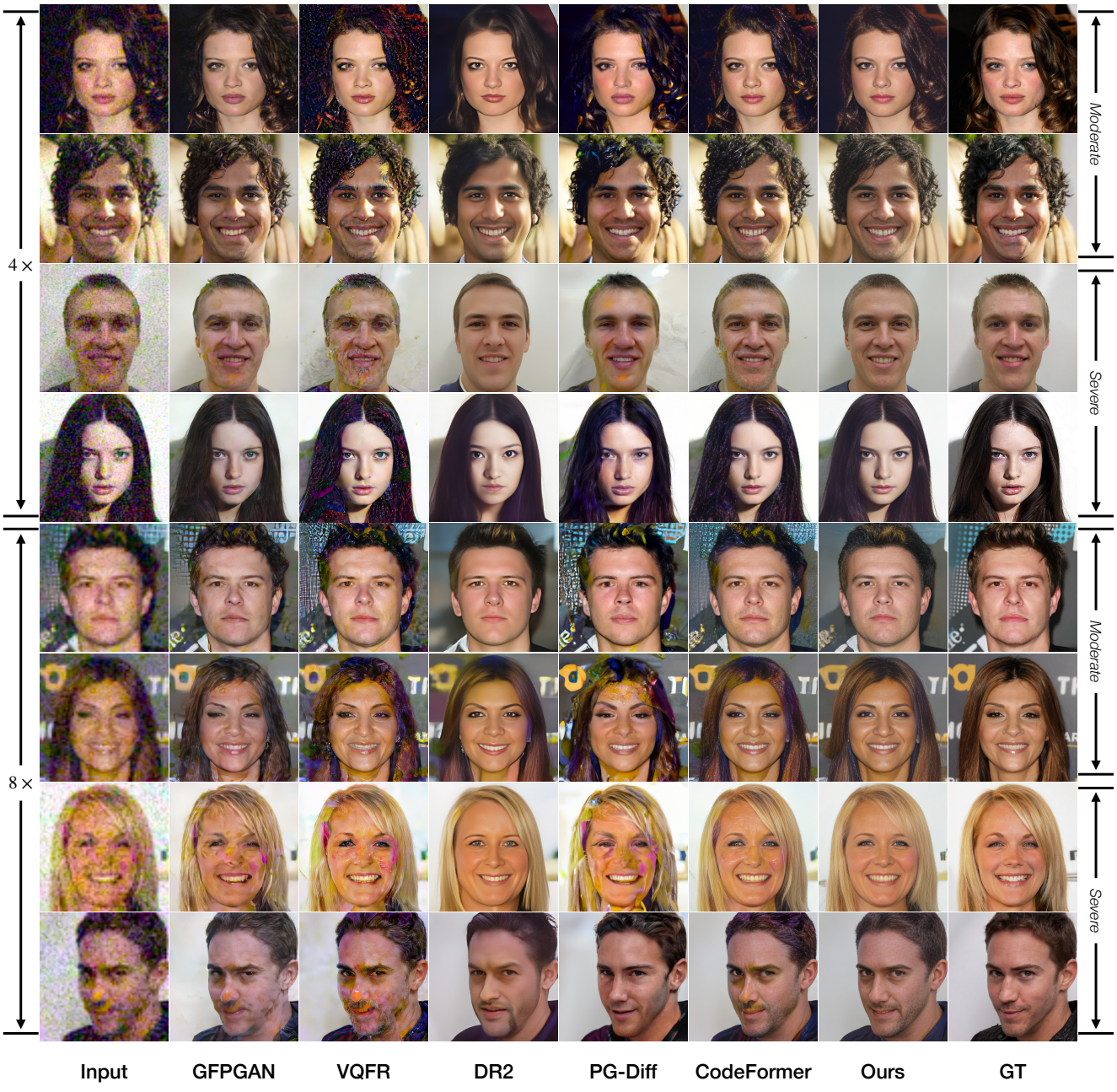

Figure 14. Additional qualitative comparison with other baselines at different degradation levels (**zoom in for details**).

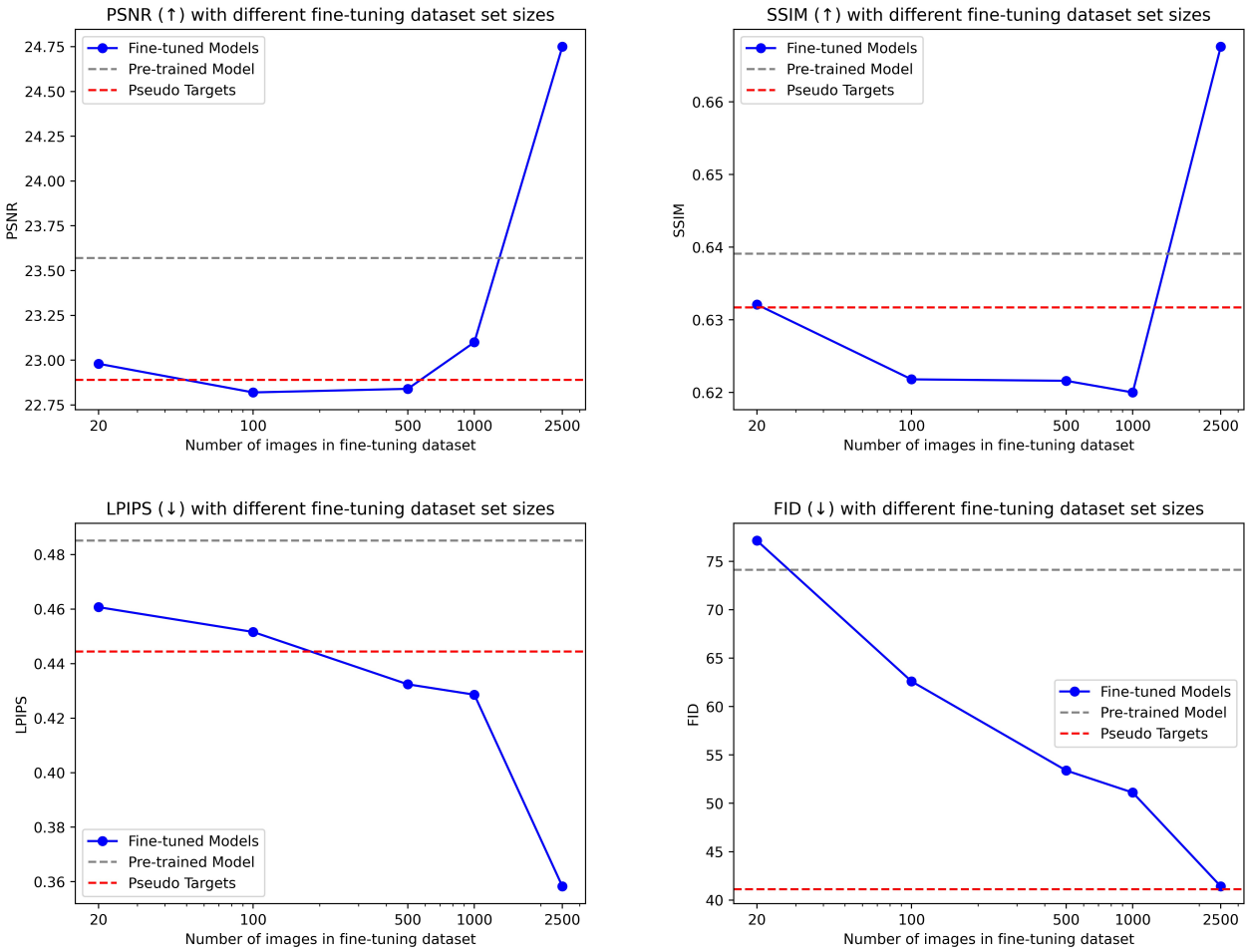

Figure 15. Fine-tuned SwinIR's performance when using different sizes of fine-tuning datasets on 4× downsampling data at *moderate* noise level.

References

[1] Kelvin CK Chan, Xintao Wang, Xiangyu Xu, Jinwei Gu, and Chen Change Loy. Glean: Generative latent bank for large-factor image super-resolution. In *CVPR*, 2021. 2

[2] Chaofeng Chen, Xiaoming Li, Lingbo Yang, Xianhui Lin, Lei Zhang, and Kwan-Yee K Wong. Progressive semantic-aware style transformation for blind face restoration. In *CVPR*, 2021. 2

[3] Yu Chen, Ying Tai, Xiaoming Liu, Chunhua Shen, and Jian Yang. Fsrnet: End-to-end learning face super-resolution with facial priors. In *CVPR*, 2018. 2

[4] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *ICCV*, 2021. 2, 4, 5, 12, 20

[5] Hyungjin Chung, Jeongsol Kim, Sehui Kim, and Jong Chul Ye. Parallel diffusion models of operator and image for blind inverse problems. In *CVPR*, 2023. 2

[6] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *ICLR*, 2023. 2

[7] Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *CVPR*, 2022. 2

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 5, 9, 13

[9] Zheng Ding, Xuaner Zhang, Zhuowen Tu, and Zhihao Xia. Restoration by generation with constrained priors. In *CVPR*, 2024. 1, 4

[10] Berk Dogan, Shuhang Gu, and Radu Timofte. Exemplar guided face image super-resolution without facial landmarks. In *CVPRW*, 2019. 2

[11] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *ICLR*, 2024. 2

[12] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 2

[13] Ben Fei, Zhaoyang Lyu, Liang Pan, Junzhe Zhang, Weidong Yang, Tianyue Luo, Bo Zhang, and Bo Dai. Generative diffusion prior for unified image restoration and enhancement. In *CVPR*, 2023. 2

[14] Berthy T Feng, Jamie Smith, Michael Rubinstein, Huiwen Chang, Katherine L Bouman, and William T Freeman. Score-based diffusion models as principled priors for inverse imaging. In *ICCV*, 2023. 2

[15] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. In *TIP*, 2008. 5, 9

[16] 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. 2, 4, 12, 20

[17] Sicheng Gao, Xuhui Liu, Bohan Zeng, Sheng Xu, Yanjing Li, Xiaoyan Luo, Jianzhuang Liu, Xiantong Zhen, and Baochang Zhang. Implicit diffusion models for continuous super-resolution. In *CVPR*, 2023. 2

[18] 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. 5, 9

[19] Yuchao Gu et al. Vqfr: Blind face restoration with vector-quantized dictionary and parallel decoder. In *ECCV*, 2022. 1, 2, 7, 13, 16, 19

[20] 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. 10

[21] Lanqing Guo, Chong Wang, Wenhan Yang, Siyu Huang, Yufei Wang, Hanspeter Pfister, and Bihan Wen. Shadowdiffusion: When degradation prior meets diffusion model for shadow removal. In *CVPR*, 2023. 2

[22] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 6

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1, 2, 3, 19

[24] Marco Huber, Anh Thi Luu, Fadi Boutros, Arjan Kuijper, and Naser Damer. Bias and diversity in synthetic-based face recognition. In *WACV*, 2024. 15

[25] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv*, 2017. 5, 9, 15

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 2, 5, 6, 9, 15

[27] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 2

[28] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *NeurIPS*, 2022. 2

[29] Bahjat Kawar, Jiaming Song, Stefano Ermon, and Michael Elad. Jpeg artifact correction using denoising diffusion restoration models. In *NeurIPS Workshop*, 2022. 2

[30] Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. In *NeurIPS*, 2021. 2

[31] Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *ICCV*, 2021. 6

[32] Deokyun Kim, Minseon Kim, Gihyun Kwon, and Dae-Shik Kim. Progressive face super-resolution via attention to facial landmark. *arXiv*, 2019. 2

[33] 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, 2, 5, 9

[34] Xiaoming Li, Wenyu Li, Dongwei Ren, Hongzhi Zhang, Meng Wang, and Wangmeng Zuo. Enhanced blind face restoration with multi-exemplar images and adaptive spatial feature fusion. In *CVPR*, 2020. 2

[35] Xiaoming Li, Ming Liu, Yuting Ye, Wangmeng Zuo, Liang Lin, and Ruigang Yang. Learning warped guidance for blind face restoration. In *ECCV*, 2018. 1, 2, 5, 9

[36] Xiaoming Li, Shiguang Zhang, Shangchen Zhou, Lei Zhang, and Wangmeng Zuo. Learning dual memory dictionaries for blind face restoration. In *TPAMI*, 2022. 2

[37] Zelin Li, Dan Zeng, Xiao Yan, Qiaomu Shen, and Bo Tang. Analyzing and combating attribute bias for face restoration. In *AAAI*, 2023. 2

[38] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ICCV*, 2021. 4, 5, 6, 7, 9, 11, 12, 13, 15, 19

[39] 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. 1, 2, 7, 12, 13, 15, 16, 19, 21

[40] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Practical signal-dependent noise parameter estimation from a single noisy image. In *TIP*, 2014. 5, 9

[41] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 15

[42] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv*, 2016. 9

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 9

[44] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022. 2

[45] Ziwei Luo, Fredrik K Gustafsson, Zheng Zhao, Jens Sjölund, and Thomas B Schön. Image restoration with mean-reverting stochastic differential equations. In *ICML*, 2023. 2

[46] Ziwei Luo, Fredrik K Gustafsson, Zheng Zhao, Jens Sjölund, and Thomas B Schön. Refusion: Enabling large-size realistic image restoration with latent-space diffusion models. In *CVPRW*, 2023. 2

[47] Markku Makitalo and Alessandro Foi. Optimal inversion of the generalized anscombe transformation for poisson-gaussian noise. In *TIP*, 2012. 5, 9

[48] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 5

[49] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023. 10

[50] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *CVPR*, 2020. 2

[51] Naoki Murata, Koichi Saito, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, and Stefano Ermon. Gibbsddrm: A partially collapsed gibbs sampler for solving blind inverse problems with denoising diffusion restoration. In *ICML*, 2023. 2

[52] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 9

[53] Ozan Özdenizci and Robert Legenstein. Restoring vision in adverse weather conditions with patch-based denoising diffusion models. In *TPAMI*, 2023. 2

[54] Xinmin Qiu, Congying Han, ZiCheng Zhang, Bonan Li, Tiande Guo, and Xuecheng Nie. Diffbfr: Bootstrapping diffusion model towards blind face restoration. *arXiv*, 2023. 2

[55] Mengwei Ren, Mauricio Delbracio, Hossein Talebi, Guido Gerig, and Peyman Milanfar. Multiscale structure guided diffusion for image deblurring. In *ICCV*, 2023. 2

[56] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2, 13

[57] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[58] Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alex Dimakis, and Sanjay Shakkottai. Solving linear inverse problems provably via posterior sampling with latent diffusion models. In *NeurIPS*, 2023. 2

[59] Hshmat Sahak, Daniel Watson, Chitwan Saharia, and David Fleet. Denoising diffusion probabilistic models for robust image super-resolution in the wild. *arXiv*, 2023. 2

[60] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *SIGGRAPH*, 2022. 2

[61] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. In *TPAMI*, 2022. 2

[62] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022. 3

[63] 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. 5, 9

[64] Ziyi Shen, Wei-Sheng Lai, Tingfa Xu, Jan Kautz, and Ming-Hsuan Yang. Deep semantic face deblurring. In *CVPR*, 2018. 2

[65] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3

[66] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ICLR*, 2021. 3

[67] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *ICLR*, 2023. 2

[68] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023. 10

[69] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017. 2

[70] Xintao Wang et al. Towards real-world blind face restoration with generative facial prior. In *CVPR*, 2021. 1, 2, 5, 7, 9, 16, 19

[71] Xintao Wang, Yu Li, Honglun Zhang, and Ying Shan. Towards real-world blind face restoration with generative facial prior. In *CVPR*, 2021. 10

[72] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *ICCVW*, 2021. 13

[73] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. In *ICLR*, 2023. 2

[74] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 6

[75] Zhouxia Wang, Jiawei Zhang, Runjian Chen, Wenping Wang, and Ping Luo. Restoreformer: High-quality blind face restoration from undegraded key-value pairs. In *CVPR*, 2022. 1, 2

[76] 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. 1, 2, 4, 5, 7, 12, 13, 16, 19, 21

[77] Jay Whang, Mauricio Delbracio, Hossein Talebi, Chitwan Saharia, Alexandros G Dimakis, and Peyman Milanfar. Deblurring via stochastic refinement. In *CVPR*, 2022. 2

[78] Bin Xia, Yulun Zhang, Shiyin Wang, Yitong Wang, Xinglong Wu, Yapeng Tian, Wenming Yang, and Luc Van Gool. Diffir: Efficient diffusion model for image restoration. In *ICCV*, 2023. 2

[79] Peiqing Yang, Shangchen Zhou, Qingyi Tao, and Chen Change Loy. Pgdiff: Guiding diffusion models for versatile face restoration via partial guidance. In *NeurIPS*, 2023. 1, 2, 7, 12, 13, 16, 19, 20

[80] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *CVPR*, 2016. 6

[81] Sidi Yang, Tianhe Wu, Shuwei Shi, Shanshan Lao, Yuan Gong, Mingdeng Cao, Jiahao Wang, and Yujiu Yang. Maniqa: Multi-dimension attention network for no-reference image quality assessment. In *CVPR*, 2022. 6

[82] Tao Yang, Peiran Ren, Xuansong Xie, and Lei Zhang. Gan prior embedded network for blind face restoration in the wild. In *CVPR*, 2021. 1, 2

[83] Rajeev Yasarla, Federico Perazzi, and Vishal M Patel. Deblurring face images using uncertainty guided multi-stream semantic networks. In *TIP*, 2020. 2

[84] Xin Yu, Basura Fernando, Bernard Ghanem, Fatih Porikli, and Richard Hartley. Face super-resolution guided by facial component heatmaps. In *ECCV*, 2018. 2

[85] Zongsheng Yue and Chen Change Loy. Difface: Blind face restoration with diffused error contraction. *arXiv*, 2022. 1, 2, 4, 5, 7, 9, 12, 15, 16, 19, 20

[86] 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. 5, 6, 9

[87] Yang Zhao, Tingbo Hou, Yu-Chuan Su, Xuhui Jia, Yandong Li, and Matthias Grundmann. Towards authentic face restoration with iterative diffusion models and beyond. In *ICCV*, 2023. 2

[88] Yang Zhao, Yu-Chuan Su, Chun-Te Chu, Yandong Li, Marius Renn, Yukun Zhu, Changyou Chen, and Xuhui Jia. Rethinking deep face restoration. In *CVPR*, 2022. 2

[89] 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, 5, 6, 7, 9, 11, 13, 16, 19

[90] Feida Zhu, Junwei Zhu, Wenqing Chu, Xinyi Zhang, Xiaozhong Ji, Chengjie Wang, and Ying Tai. Blind face restoration via integrating face shape and generative priors. In *CVPR*, 2022. 2

[91] Shizhan Zhu, Sifei Liu, Chen Change Loy, and Xiaoou Tang. Deep cascaded bi-network for face hallucination. In *ECCV*, 2016. 2

[92] Yuanzhi Zhu, Kai Zhang, Jingyun Liang, Jiezhang Cao, Bihan Wen, Radu Timofte, and Luc Van Gool. Denoising diffusion models for plug-and-play image restoration. In *CVPRW*, 2023. 2