MVA - DELIRES
Benchmarking Guided Diffusion Models

Tom LABIAUSSE tom.labiausse@student-cs.fr
CentraleSupélec

Theïlo TERRISSE theilo.terrisse@eleves.enpc.fr
Ecole des Ponts Paristech

1. Introduction

Recent research in the field of deep learning for image restoration leverages diffusion models as generative models conditioned by the observed data. While classical diffusion models sample the prior distribution of clean images by integrating a reverse diffusion process thanks to an iterative denoising process, conditional diffusion models aim at sampling the posterior distribution by incorporating a guidance term into the process. A category of conditional diffusion models, such as [6], propose to train a task-specific denoiser in a supervised way. While these approaches may yield good restoration quality, they require retraining a new model for each task. Instead, zero-shot diffusers build upon unconditional diffusion models and add an separate guidance term to allow tackling any classical restoration problem.

Among these methods, Diffusion Posterior Sampling (DPS) [1] and Pseudoinverse-Guided Diffusion Models (ΠGDM) explicitly approximate the score of the likelihood of the observation by a Dirac distribution and a Gaussian distribution respectively, both centered in the estimate of the clean image $\hat{x}_0$ as delivered by the noise predictor. On the other hand, Denoising Diffusion Models for Plug-and-Play Image Restoration (DiffPIR) use a Plug-and-Play (PnP) denoising approach to construct the noise predictor of the denoising diffusion model, where a pretrained unconditional denoising score-matching function is used as a the underlying prior in the PnP algorithm. As explained by Peng *et al.* [5], these three approaches can be understood under a same framework as leveraging isotropic Gaussian approximations to the posterior distributions over clean images given noisy images.

In this report, we carry out a quantitative and qualitative comparative study of these three methods. All three algorithms were implemented in a common framework and tested on a subset of the Flickr-Faces-HQ (FFHQ) dataset for deblurring and inpainting tasks. Then, the results are compared in terms of quality, fitness to the observation and variability of the generations. This document is organized as follows: Section 2 summarizes the three diffusion models, highlights their similarities and differences and recalls the insight offered by Peng *et al.*. In Section 3, we present the design of our experiments and introduce our metrics. In Section 4, the results are displayed and analysed. Last, we conclude in Section 5. The code related to this project can be found here: **github.com/t0m1ab/MVA_DELIRES_project**

2. Guided diffusion methods

Image restoration Image restoration problems are usually formulated as inverse problems where we wish to recover a clean image $\boldsymbol{x}_0 \in \mathbb{R}^d$, with d the number of pixels, from the observation of a degraded image $\boldsymbol{y} \in \mathbb{R}^m$ of the form

$$\boldsymbol{y} = \mathcal{H}(\boldsymbol{x}_0) + \boldsymbol{n} \tag{1}$$

where $\mathcal{H}$ is the operator that models the degradation, such as blurring or masking, and is typically a linear operator $\mathbf{H}$ in $\mathbb{R}^{m \times d}$, and $\boldsymbol{n}$ is some Gaussian noise of standard deviation $\sigma_{\boldsymbol{y}}$. In practice, we will consider cases where $m = d$.

Denoising diffusion models The methods under scrutiny are based on an unconditional denoising diffusion model, such as Denoising Diffusion Probabilistic Models (DDPM) [4] or more recently Denoising Diffusion Implicit Models (DDIM) [8] that they augment with guidance. For simplicity, in this report we will only use the DDPM framework that we recall. DDPM defines the generative process as the reverse

of a variance-preserving noising process in T steps,

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I}) \tag{2}$$

with $(\beta_t)_t$ chosen to preserve the variance and bring the process to complete Gaussian noise $\boldsymbol{x}_T \sim \mathcal{N}(0, \bar{\beta}_t \mathbf{I})$, where $\bar{\beta}_t = \prod_{s=1}^{t} \beta_s$. Equivalently,

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\boldsymbol{\varepsilon} \tag{3}$$

with $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ where $\alpha_s = 1 - \beta_s$.

DDPM aims at sampling an approximation p_θ of the law p of the reverse process that brings the Gaussian distribution p_T to the prior distribution p_0, by iterating

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}) + \sigma_t \boldsymbol{z} \tag{4}$$

with σ_t a hyperparameter, and $\boldsymbol{\varepsilon}$ would be approximated using a noise predictor $\boldsymbol{\varepsilon}_\theta$ trained to learn the noise that brings $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$ in (3).

Equivalently, using Tweedie's formula, the DDPM update rule (4) can be rewritten in terms of the score function as:

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t + \beta_t \nabla \log(p(\boldsymbol{x}_t))) + \sigma_t \boldsymbol{z}. \tag{5}$$

where the score $\nabla \log(p(\boldsymbol{x}_t))$ would be approximated by a score matching predictor $\boldsymbol{s}_\theta$. The link between the two predictors can be written as $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) = -\frac{\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}$.

DPS and ΠGDM In practice, DPS and ΠGDM aim at sampling the posterior distribution $p(\boldsymbol{x}_0; \boldsymbol{y})$ rather than the prior $p(\boldsymbol{x}_0)$. Therefore, using Bayes formula $p(\boldsymbol{x}_t; \boldsymbol{y}) \propto p(\boldsymbol{y}; \boldsymbol{x_t}) p(\boldsymbol{x_t})$, they replace the update above with

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}'_{t-1} + \frac{\beta_t}{\sqrt{\alpha_t}} \underbrace{\nabla \log p(\boldsymbol{y}; \boldsymbol{x}_t)}_{-\boldsymbol{g}} \tag{6}$$

where $\boldsymbol{x}'_{t-1}$ is the result of (5) and $\boldsymbol{g}$ is coined the "guidance term". Remarking that by marginalization and independence of $\boldsymbol{x}_t$ and $\boldsymbol{y}$ conditionally on $\boldsymbol{x}_0$, $p(\boldsymbol{y}; \boldsymbol{x}_t) = \int p(\boldsymbol{x}_0; \boldsymbol{x}_t) p(\boldsymbol{y}; \boldsymbol{x}_0) \, \mathrm{d}\boldsymbol{x}_0$, the two methods approximate the guidance term by approximating the intractable $p(\boldsymbol{x}_0; \boldsymbol{x}_t)$ using the estimate of $\hat{\boldsymbol{x}}_0 := \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$ derived using the trained score matching predictor:

$$\hat{\boldsymbol{x}}_0 \approx \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \big(1-\bar{\alpha}_t\big)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\big). \tag{7}$$

DPS uses the approximation $p(\boldsymbol{x}_0; \boldsymbol{x}_t) \approx \delta_{\hat{\boldsymbol{x}}_0}$ (and the article [1] details how much of an approximation this constitutes) to obtain the guidance term:

$$g = \nabla_{\boldsymbol{x}_t} \frac{1}{2\sigma_{\boldsymbol{y}}^2} \|\mathcal{H}(\hat{\boldsymbol{x}}_0) - \boldsymbol{y}\|_2^2 \tag{8}$$

which can easily be computed by automatic differentiation. DPS is summarized in Algorithm 1. In this algorithm, $(\zeta_t)_{1 \le t \le T}$ are scaling factors used to stabilize the process, and are set to 1 in our experiments. In blue, we highlight the part of the algorithm that differs from unconditional DDPM sampling.

ΠGDM uses the approximation $p(\boldsymbol{x}_0; \boldsymbol{x}_t) \approx \mathcal{N}(\hat{\boldsymbol{x}}_0, r_t^2 \mathbf{I})$ with r_t a hyperparameter that is set empirically, which in the case of a linear $\mathcal{H}$ yields an estimation in the following vector-Jacobian product:

$$g = -\Big(\underbrace{(\boldsymbol{y} - \mathbf{H}\hat{\boldsymbol{x}}_0)^T \left(r_t^2 \mathbf{H}\mathbf{H}^T + \sigma_{\boldsymbol{y}}^2 \mathbf{I}\right)^{-1} \mathbf{H}}_{\text{vector}} \underbrace{\frac{\partial \hat{\boldsymbol{x}}_0}{\partial \boldsymbol{x}_t}}_{\text{Jacobian}} \Big)^T. \tag{9}$$

In particular, ΠGDM only performs automatic differentiation to compute the Jacobian, but requires computing a pseudo-inverse (which gives the algorithm its name). The method may thus be applied to non-linear degradations to the extent that this pseudo-inverse can be computed. ΠGDM is summarized in Algorithm 2, in the case of a linear operator $\mathbf{H}$. Here again, we highlight the specific part of the algorithm in blue.

DiffPIR While DPS and ΠGDM use an approximation of the likelihood score function, DiffPIR resorts to proximal splitting PnP algorithms. More precisely, drawing inspiration from Half-Quadratic-Splitting (HQS) [2], the authors first derive an estimated sample $\hat{\boldsymbol{x}}_0^{\boldsymbol{y}}$ from the posterior distribution $p(\boldsymbol{x}_0; \boldsymbol{y})$ as the result of the proximal operator associated to the data-fitting term evaluated in $\hat{\boldsymbol{x}}_0$:

$$\hat{\boldsymbol{x}}_0^{\boldsymbol{y}} = \arg\min_{\boldsymbol{x}} \|\mathcal{H}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 + \rho_t \|\boldsymbol{x} - \hat{\boldsymbol{x}}_0\|_2^2 \tag{10}$$

where $\rho_t = \lambda(\frac{\sigma_{\boldsymbol{y}}}{\bar{\sigma}_t})^2$ with λ and $\bar{\sigma}_t$ two hyperparameters, and $\hat{\boldsymbol{x}}_0$ is the result of the subproblem of the HQS algorithm associated with the prior term. Here, $\hat{\boldsymbol{x}}_0 = \mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$ is approximated using the score matching predictor $\boldsymbol{s}_\theta$ as for the case of DPS and ΠGDM.

The estimated sample $\hat{\boldsymbol{x}}_0^{\boldsymbol{y}}$ is then used to get an estimate of the noise $\hat{\boldsymbol{\varepsilon}} = \frac{1}{\sqrt{1-\bar{\alpha}}}(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{x}}_0^{\boldsymbol{y}})$, that was added from the posterior sample. This predicted noise can then be used to apply a DDIM update.

In other words, one step of DiffPIR corresponds to an update of DDIM where, instead of predicting noise added to the sample of the prior distribution, the noise added to the posterior sample is estimated using one step of an HQS PnP denoising algorithm. This PnP step leverages the score-matching predictor of an unconditional DDPM diffusion model. DiffPIR is summarized in Algorithm 3, where the authors introduce a reparameterization of the noise $\boldsymbol{z}$ that is added to get $\boldsymbol{x}_{t-1}$ by introducing a new hyperparameter ζ. The key characteristic step of DiffPIR is here again highlighted in blue.

A unifying framework In [5], Peng *et al.* observe that, although DPS and ΠGDM on the one hand, and DiffPIR on the other hand employ different guidance paradigms as explained above, all three algorithms rely on an isotropic Gaussian approximation $\tilde{p}_t(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0; \hat{\boldsymbol{x}}_0, r_t^2\mathbf{I})$ of the intractable probability $p(\boldsymbol{x}_0, \boldsymbol{x}_t)$. In the case of ΠGDM, r_t is set heuristically, typically to a value $\sqrt{\frac{\sigma_t^2}{\sigma_t^2+1}}$ with σ_t the estimated variance at step t. In the case of DPS, r_t can be understood as approaching 0. In the case of DiffPIR, $r_t = \frac{\bar{\sigma}_t}{\sqrt{\lambda}}$. This can be understood from the fact that step (10) can then be rewritten as

$$\begin{aligned}
\hat{\boldsymbol{x}}_y &= \arg\min_{\boldsymbol{x}} \|\mathcal{H}(\boldsymbol{x}) - \boldsymbol{y}\|^2 + \frac{\sigma_{\boldsymbol{y}}^2}{r_t^2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}_0\|^2 \\
&= \arg\max_{\boldsymbol{x}} [\log p(\boldsymbol{y}; \boldsymbol{x}) + \log \tilde{p}_t(\boldsymbol{x}_0; \boldsymbol{x}_t)] \\
&= \arg\max_{\boldsymbol{x}} \log \tilde{p}_t(\boldsymbol{x}; \boldsymbol{x}_t, \boldsymbol{y})
\end{aligned} \tag{11}$$

with $\tilde{p}_t(\boldsymbol{x}; \boldsymbol{x}_t, \boldsymbol{y})$ defined proportional to $p(\boldsymbol{y}; \boldsymbol{x})\tilde{p}_t(\boldsymbol{x}; \boldsymbol{x}_t, \boldsymbol{y})$ using Bayes rule. In this case, $\hat{\boldsymbol{x}}_0^{\boldsymbol{y}}$ can be understood as the maximum likelihood estimator of the approximated posterior at step t, or equivalently as the Minimum Mean Square Error estimator $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{y}, \boldsymbol{x}_t]$ of this distribution since the approximating probability $\tilde{p}_t(\boldsymbol{x}; \boldsymbol{x}_t, \boldsymbol{y})$ is Gaussian.

Algorithm 1 DPS

Require: $\boldsymbol{y}$, $\mathcal{H}$, T, ζ_t, σ_t, α_t, $\boldsymbol{s}_\theta$.
$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  # Evaluate score matching model
  Approximate $\nabla \log p_t(\boldsymbol{x}_t)$ by $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$
  # Estimate $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}\big(\boldsymbol{x}_t + (1-\bar{\alpha}_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\big)$
  # Apply unguided DDPM update
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{x}'_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t + (1-\alpha_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)) + \sigma_t \boldsymbol{z}$
  # Add guidance term
  $\boldsymbol{g} \leftarrow \zeta_t \nabla_{\boldsymbol{x}_t} \|\mathcal{H}(\hat{\boldsymbol{x}}_0) - \boldsymbol{y}\|_2^2$
  $\boldsymbol{x}_{t-1} \leftarrow \boldsymbol{x}'_{t-1} - \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{g}$
end for
Return $\boldsymbol{x}_0$

Algorithm 2 ΠGDM (linear degradation)

Require: $\boldsymbol{y}$, $\mathbf{H}$, $\sigma_{\boldsymbol{y}}$, T, σ_t, α_t, $\boldsymbol{s}_\theta$.
$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  # Evaluate score matching model
  Approximate $\nabla \log p_t(\boldsymbol{x}_t)$ by $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$
  # Estimate $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}\big(\boldsymbol{x}_t + (1-\bar{\alpha}_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\big)$
  # Apply unguided DDPM update
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{x}'_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t + (1-\alpha_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)) + \sigma_t \boldsymbol{z}$
  # Add guidance term
  $\boldsymbol{v} \leftarrow \mathbf{H}^T\left(r_t^2\mathbf{H}^T\mathbf{H} + \sigma_{\boldsymbol{y}}^2\mathbf{I}\right)^{-1}(\boldsymbol{y} - \mathbf{H}\hat{\boldsymbol{x}}_0)$
  $\boldsymbol{g} \leftarrow -\left(\frac{\partial \hat{\boldsymbol{x}}_0}{\partial \boldsymbol{x}_t}\right)^T \boldsymbol{v}$
  $\boldsymbol{x}_{t-1} \leftarrow \boldsymbol{x}'_{t-1} - \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{g}$
end for
Return $\boldsymbol{x}_0$

Algorithm 3 DiffPIR (DDIM framework)

Require: $\boldsymbol{y}$, $\mathcal{H}$, $\sigma_{\boldsymbol{y}}$, T, ζ, λ, $\bar{\sigma}_t$, α_t, $\boldsymbol{s}_\theta$.

$\rho_t \leftarrow \lambda \frac{\sigma_{\boldsymbol{y}}^2}{\bar{\sigma}_t^2}$
$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  # Evaluate score matching model
  Approximate $\nabla \log p_t(\boldsymbol{x}_t)$ by $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$
  # Estimate $\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]$ (= solve PnP *prior* subproblem)
  $\hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}\big(\boldsymbol{x}_t + (1-\bar{\alpha}_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\big)$
  # Estimate $\mathbb{E}[\boldsymbol{x}_0|y, \boldsymbol{x}_t]$ (= solve PnP *data* subproblem)
  $\hat{\boldsymbol{x}}_0^{\boldsymbol{y}} \leftarrow \arg\min_{\boldsymbol{x}} \|\mathcal{H}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 + \rho_t \|\boldsymbol{x} - \hat{\boldsymbol{x}}_0\|_2^2$
  # Predict effective noise
  $\hat{\boldsymbol{\varepsilon}} \leftarrow \frac{1}{\sqrt{1-\bar{\alpha}_t}}(\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{x}}_0^{\boldsymbol{y}})$
  # Apply DDIM update
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{x}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{x}}_0^{\boldsymbol{y}} + \sqrt{1-\bar{\alpha}_{t-1}}(\sqrt{1-\zeta}\hat{\boldsymbol{\varepsilon}} + \sqrt{\zeta}\boldsymbol{z})$
end for
Return $\boldsymbol{x}_0$

3. Experiments

3.1 Metrics

The three methods introduced in Section 2 are compared both qualitatively and quantitatively. The quantitative study is carried out with the objective to assess four properties of the restored images $\tilde{\boldsymbol{x}}_0$: proximity to the actual clean images $\boldsymbol{x}_0$; fidelity to the observations $\boldsymbol{y}$; overall generation quality; variability of the restored images when running the same algorithm on the same degraded image multiple times. To achieve this, four metrics are used. Note that, when applicable, values for image pixels are expressed in the $[0, 255]$ range to compute the metrics.

Proximity to ground truth To assess the proximity between restoration $\tilde{\boldsymbol{x}}_0$ and ground truth $\boldsymbol{x}_0$, we use the usual Peak Signal to Noise Ratio (PSNR) defined as

$$PSNR = 20 \log_{10}\left(\frac{255}{MSE}\right), \tag{12}$$

where MSE is the Mean Square Error between the restored image $\tilde{\boldsymbol{x}}_0$ and the clean image $\boldsymbol{x}_0$, defined as $MSE = \frac{1}{d}\|\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}_0\|_2^2$ with d the number of pixels. When the metric has to be averaged between N pairs $(\tilde{\boldsymbol{x}}_0^i, \boldsymbol{x}_0^i)$, the average is computed by replacing the MSE in (12) by $\frac{1}{N}\sum_{i=1}^N MSE^i$, with MSE^i the MSE computed on pair i.

Fidelity to the observation As a measure of the fidelity to the degraded image, we use the norm corresponding to the data-fitting term in the restoration problem, or in other words the Root Mean Square Error (RMSE) between the observation and the *degraded restoration*:

$$RMSE = \|\mathcal{H}(\tilde{\boldsymbol{x}}_0) - \boldsymbol{y}\|_2. \tag{13}$$

Here also, when averaging the metric over several pairs, we average the MSE between the $\mathcal{H}(\tilde{\boldsymbol{x}}_0^i)$ and the $\boldsymbol{y}^i$ before taking the root.

Restoration quality To measure restoration quality, we use the Fréchet Inception Distance (FID) [3]. The FID compares two sets of images S and S' by passing all images through a network f (in practice, the *Inception v3* network [9], a classification model trained on ImageNet) deprived of its final classification layer, to get two sets $f(S)$ and $f(S')$ of features that are considered to be representative of high-level characteristics of the image. Two Gaussian distributions $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu', \Sigma')$ are then fitted on the two respective sets $f(S)$ and $f(S')$ to represent them as distributions of images. Lastly, these two distributions are compared using the 2-Wasserstein distance, which in this case writes

$$d_F(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma'))^2 = \|\mu - \mu'\|_2^2 + \operatorname{tr}\left(\Sigma + \Sigma' - 2\left(\Sigma\Sigma'\right)^{\frac{1}{2}}\right). \tag{14}$$

We apply this metric on the sets of restored and clean images. In practice, the number of samples to fit the Gaussian distributions should be greater than the dimension of the coding layer of the network for which the features are extracted. This is to avoid cases where the covariance matrix would not be of full rank. Since the final layer of *Inception v3* has 2048 features and we run experiments on batches of only 100 images, we choose to use an earlier layer of the network instead, namely the second max pooling which has 192 channels. Since this layer has leftover spatial dimensions, we can get several samples per image and reach the required number of samples to estimate the distributions. For this metric, we use the Pytorch implementation of [7].

Variability of generations Last, we measure the pixel-wise standard deviation over multiple restorations $\tilde{\boldsymbol{x}}_0^j$ from the same observation $\boldsymbol{y}$ (averaging the standard deviation over the 3 channels of a coloured pixel) to generate a standard deviation map that we use to assess the variability of generations. To turn this map into a quantitative metric (later simplt referred to as "STD"), we also average the standard deviations of all pixels of a map to get the variability σ for a given observation. Here also, the standard deviations σ^i associated to multiple observations $\boldsymbol{y}^i$ can be averaged to get the overall variability of the generations from a given model.

3.2 Experimental setup

Dataset In order to compare the different guided restoration techniques, we used the Flickr-Faces-HQ (FFHQ) dataset composed of 70,000 high quality PNG images of human faces with variations in terms of age, ethnicity and background details. We selected data from the FFHQ test set composed of images with ids from 69000 to 69999. More specifically, we built two small test sets with images from 69000 to 69099 (see **Figure 1**) and from 69100 to 69199 (see **Figure 3**).

Degradation operators Concerning the restoration tasks, we focused on the deblurring and inpainting. Therefore, we used pre-existing blur kernels presented in **Figure 2** and **Figure 4** which were introduced and used multiple times in the literature including in [10]. These kernels vary in terms of size and aspect: some of them are closer to pure Gaussian blur kernels and others act as motion blur. We also used masks for the inpainting tasks. They can be a simple rectangular area called "box mask" or a random selection of pixels called "random mask" set to zero on the image.

Diffusion networks As diffusion model, we only used a single network across our experiments. It is a classic UNet architecture with residual connections that was trained on the train split of the FFHQ dataset. This network was already used to perform benchmark in the literature and was especially used by the authors of [10] as they mention it here. The network contains approximately 93 millions parameters and was a reasonable choice considering our computing resources and the time we had to run the experiments.

Methods parameters For the DPS algorithm, we use scale factors $\zeta_t = 1$ for simplicity, as this value already gives satisfying performance and stability. For DiffPIR, re-using the parameters proposed by the authors in [10], we use $\lambda = 7$ and $\zeta = 0.3$ for deblurring, and $\lambda = 1$ and $\zeta = 1$ for inpainting. We also use $\bar{\sigma}_t = \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}$, for $t \in [\![1, T]\!]$, for both tasks. Finally, the following table gives the number of steps we chose for each method:

Method	**Steps**
DPS	500
ΠGDM	100
DiffPIR	100

Experimental protocol Our experimental protocol aims to compare DPS, ΠGDM and DiffPIR on identical tasks for a large enough number of images. As we will see, our protocol is divided in two stages. This first stage aims to compare both quantitatively using the metrics but also qualitatively by looking at the results for the three methods applied to the deblurring and inpainting tasks. Here we describe in details the first stage in a sequential manner:

- Selection of 100 clean images represented on **Figure 1.**
- Generate a blurred version of each ground truth image using blur kernels from **Figure 2**. As there are more images than kernels, a single kernel can be applied to multiple images.

- Generate a masked version of each ground truth image by applying a random box mask to each one of them.
- Apply DPS, ΠGDM and DiffPIR once to each degraded image independently and compute the associated PSNR and RMSE for each restoration. The PSNR and RMSE averaged over all images as described in Section 3.1 are also stored.
- For each method and each type of degradation (blurring or masking), compute the FID score between the 100 corresponding restored images and the 100 original images.

The second stage of our experimental protocol aims to study the variability of each method. As it requires to apply each method on the same image multiple times, we reduce the number of tested images to 10. Given that, the second stage is very similar to the first one except that we apply each method 5 times to each of the 10 images to obtain multiple versions of the same reconstruction.

We ran our experiments on *Google Cloud Platform* using a *Nvidia T4* GPU and it took approximately 10 hours to complete. As we had still some time left on the machine, we decided to run the full experiments a second time using ground truth images from **Figure 3**, blur kernels from **Figure 4** and random masks instead of box masks for the inpainting task. In the following, we will make the difference between the results obtained from the first experiments called *first batch* and the second experiments called *second batch* as we purposely didn't use the same kernels and mask in both batches.

4. Results and analysis

4.1 Quantitative analysis

The following **Tables 1, 2** present the resulting averaged PSNR, averaged datafit RMSE and FID score over the 100 corresponding images for each batch of experiments. We first notice that the box inpainting task is unsurprisingly the hardest task for all methods in terms of PSNR, while random inpainting is the easiest. Then, we note that DiffPIR beats the other two methods on both tasks in each batch in terms of PSNR and datafit RMSE. DiffPIR has even a large advantage in terms of datafit RMSE on the inpainting tasks compared to the other two methods. It also beats the other methods in terms of FID for all tasks, except for box inpainting, which may be related to noise artifacts introduced by the method for this task, as mentioned in Section 4.2 below. DiffPIR is also the method that generates restorations with the smallest variability in all cases. On the other hand, DPS yields intermediate performance for all tasks and all metrics, except for the RMSE for which it tends to lag behind the two other approaches. In other words, DPS often lacks fidelity to the observations. It is also the method that yields the highest variability for the two inpainting tasks. Moreover, ΠGDM exhibits very poor performance for the box inpainting task which will be investigated qualitatively in the next section, but it is surprising to observe that the FID does not give account of this poor restoration quality.

Deblurring				
	PSNR	RMSE	FID	STD
DPS	24.85	15.89	5.26	5.65
ΠGDM	23.32	14.56	8.38	8.43
DiffPIR	28.13	14.43	1.63	4.46

Box inpainting				
	PSNR	RMSE	FID	STD
DPS	22.37	15.58	9.92	5.93
ΠGDM	18.41	13.71	7.82	5.51
DiffPIR	23.41	4.47	18.72	3.85

Table 1: Avg PNSR, avg datafit RMSE, avg STD and FID for each task and method with batch 1

Deblurring				
	PSNR	RMSE	FID	STD
DPS	24.12	16.00	6.47	6.26
ΠGDM	24.10	14.49	15.17	9.11
DiffPIR	27.44	14.45	3.47	5.46

Random inpainting				
	PSNR	RMSE	FID	STD
DPS	25.27	11.92	11.86	5.36
ΠGDM	28.79	10.00	10.22	3.38
DiffPIR	29.62	4.55	5.90	2.60

Table 2: Avg PNSR, avg datafit RMSE, avg STD and FID for each task and method with batch 2

We can also compare the results in more details using histograms of the metrics for each task and method as it is done on **Figures 5, 6, 7, 8**. In particular, one can see by looking at the PSNR histograms that DiffPIR seems to produce restored image which are always significanlty closer to the original than the other methods. Conversely, DPS provides restorations that are not as faithful to the degraded image as the other methods as one can see that the datafit RMSE histograms are considerably shifted to higher values for DPS compared to ΠGDM and DiffPIR.

4.2 Qualitative analysis

Deblurring results From **Figure 9 to 16**, we present some results for the deblurring task.

Figures 9, 10 present typical examples outlining the differences between the three methods. First of all, ΠGDM showcases a lot of artefacts even if the underlying reconstruction seems very close to the ground truth image. On the other hand, DPS outputs a likely image but quite different from the ground truth. This reflects the metrics obtained in **Figures 1, 2** where the PSNR is lower for ΠGDM (because of the artefacts) but the datafit RMSE is better than DPS (indicating that artefacts improve the fitting of the blurred image after degradation). On the other hand, DiffPIR gives accurate reconstructions both in terms of image quality but also closeness to the ground truth.

On **Figures 11 and 12**, it is interesting to see that ΠGDM seems to struggle to reconstruct regions that are already a bit blurred such as the man's beard or the the blurred effect on the background. One can see that it creates a kind of texture with small dots. This phenomenon doesn't occur with DPS or DiffPIR.

Another interesting difference between the methods concerns DPS as one can see on **Figures 13, 14**. Indeed, DPS doesn't reconstruct some details of the image like the tiles on the wall or the written digits on the adhesive tape. More generally, it seems that DPS struggles with thin details, which is sometimes more visible when looking at the background or the hair.

At last, **Figures 15, 16**, showcases the variability of the different methods. One can see that ΠGDM sometimes produces artefacts like discussed before which makes the method rather unstable. DPS and DiffPIR produce accurate results for all generations. However, DPS exhibits more variability at each generation whereas DiffPIR is more stable and stays close to the original ground truth for all generations.

Inpainting results From **Figures 17 to 24**, we present some results for the inpainting task mostly for box masks. Indeed, the performance obtained for random masks was rather uniform across all images so we only included a single figure (**Figure 24**) as an example.

Figures 17, 18 are appropriate examples of what we observed concerning the inpainting task for box masks. First of all, it is obvious that ΠGDM suffers catastrophic reconstructions either forcing a hair-like texture in the reconstruction or an unadapted color mix. This explains the very poor PSNR observed for the inpainting task in **Table 1**.

On the other hand, DPS and DiffPIR give reasonable reconstructions. It is however visible that DiffPIR is more accurate with respect to the ground truth image. In general, the datafit RMSE is way lower for DiffPIR on this box inpainting task than it is for DPS as it is shown on **Figure 8**. This indicates that DiffPIR is more bounded to the unmasked regions of the image which logically seems to guide the reconstruction of the mask area to better fit the original ground truth. Interestingly, **Figure 20** gives an example where DPS "adds" glasses to th reconstructed image whereas **Figure 20** shows an example where DiffPIR "removes" glasses from the reconstruction. This phenomenon was observed multiple times across our benchmark though it is hard to analyse it from the algorithmic point of view.

Figure 24 gives an example of random mask inpainting. For this specific task, ΠGDM give results a little bit more accurate than DPS both in terms of PSNR and datafit RMSE. DiffPIR almost always give the best result of the three methods for this task as one can see it is able to reconstruct thin details and color variations.

At last, we noticed that DiffPIR produces noisy images everywhere but at the mask position. This might indicate a flaw in the implementation of the method. Unfortunately, we failed to localize and fix this issue in due time.

Figures 23, 24 compare the variability of the three methods for the box mask inpainting task. One can see that ΠGDM always fail quite dramatically. DiffPIR also seems less stable that DPS when it comes to reconstruct hair regions. However, DPS doesn't always respect the expected approximate symmetry of human faces.

4.3 Synthesis of the results

From the metrics and the qualitative analysis, we were able to extract the main tendencies from the comparison of DPS, ΠGDM and DiffPIR:

- ΠGDM seems unstable both in deblurring and box inpainting tasks. It sometimes produces high quality results but can contain artefacts because of the presence of a specific texture in the ground truth image.
- DPS often produces good quality outputs but smoothes out small details. The restoration can also be quite different from the ground truth image as the method exhibits a non negligible variability in its results for a same input image.
- DiffPIR appears to be stable and with low variability. It is most of the time the best result both in terms of metrics but also qualitatively compared to DPS and ΠGDM.

5. Conclusion

The goal of this project was to study different guided diffusion models for restoration tasks. We chose the tasks of debluring with various blur kernels like Gaussian blur and motion as well as inpainting with box or random masks applied to a subset of the FFHQ dataset. We built a reusable python framework called `delires` which implements the following restoration methods: DPS, ΠGDM and DiffPIR. We ran experiments using a pretrained diffusion network in order to evaluate and compare the three methods both quantitatively using PSNR, RMSE, FID and standard deviation, but also qualitatively with human evaluation of the results.

Overall, we observed that DPS performs reasonably well in all tasks but exhibits a high variability across its generations which can differ from the ground truth data by smoothing details to a certain extent. ΠGDM produces restoration closer to the ground truth compared to DPS but often suffers from important artefacts on the deblurring tasks and dramatic failure cases on the box inpainting task. On the other hand, DiffPIR provides the best results both in terms of data fidelity and stability across all different tasks.

References

[1] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye. Diffusion Posterior Sampling for General Noisy Inverse Problems. In *ICLR*, Sept. 2022.

[2] D. Geman and C. Yang. Nonlinear image recovery with half-quadratic regularization. *IEEE Transactions on Image Processing*, 4(7):932–946, July 1995. Conference Name: IEEE Transactions on Image Processing.

[3] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[4] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models, Dec. 2020. arXiv:2006.11239 [cs, stat].

[5] X. Peng, Z. Zheng, W. Dai, N. Xiao, C. Li, J. Zou, and H. Xiong. Improving Diffusion Models for Inverse Problems Using Optimal Posterior Covariance, Feb. 2024. arXiv:2402.02149 [cs].

[6] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. Image Super-Resolution via Iterative Refinement, June 2021. arXiv:2104.07636 [cs, eess].

[7] M. Seitzer. pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid`, August 2020. Version 0.3.0.

[8] J. Song, C. Meng, and S. Ermon. Denoising Diffusion Implicit Models, Oct. 2022. arXiv:2010.02502 [cs].

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision, Dec. 2015. arXiv:1512.00567 [cs].

[10] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. Van Gool. Denoising Diffusion Models for Plug-and-Play Image Restoration, May 2023. arXiv:2305.08995 [cs, eess].

Appendix

Figure 1: Test set from FFHQ (images 69000-69099)

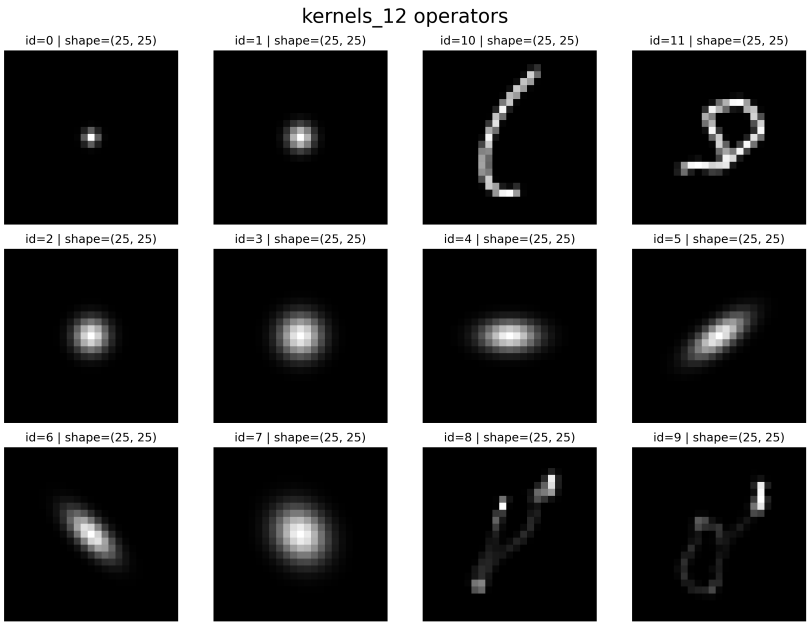

Figure 2: Set of blur kernels for batch 1

Figure 3: Test set from FFHQ (images 69100-69199)

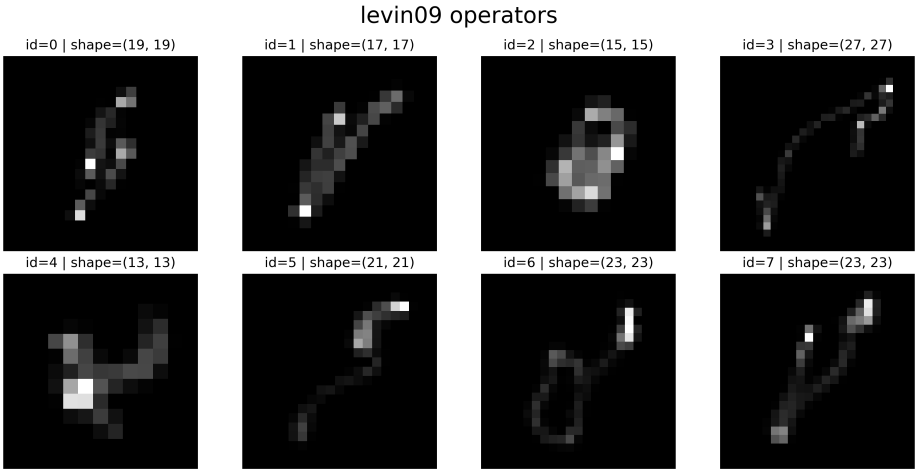

Figure 4: Set of blur kernels for batch 2

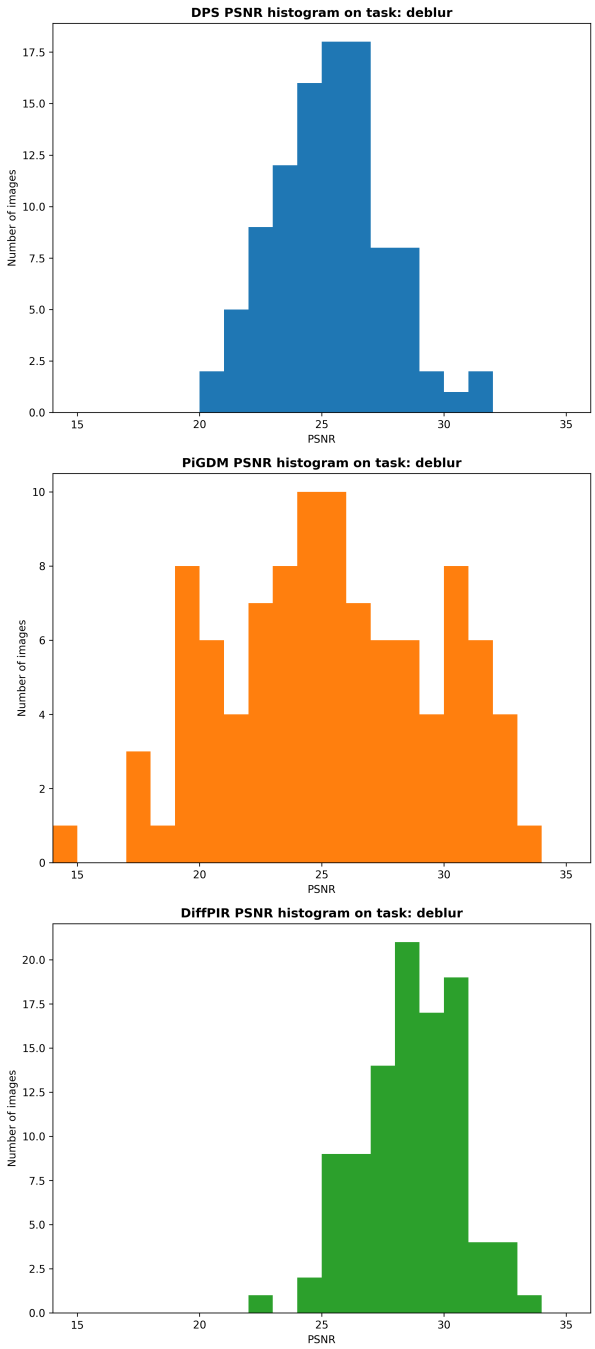

Figure 5: Histogram of PSNR values over 100 images for the debluring task of batch 1

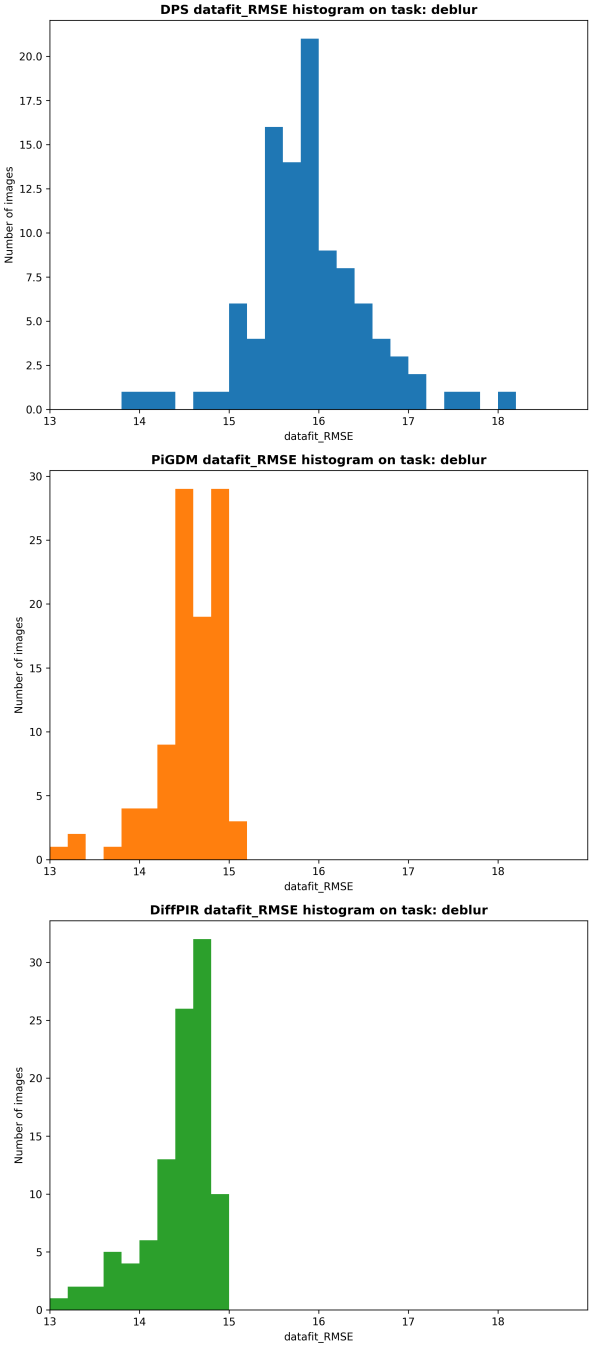

Figure 6: Histogram of datafit RMSE values over 100 images for the debluring task of batch 1

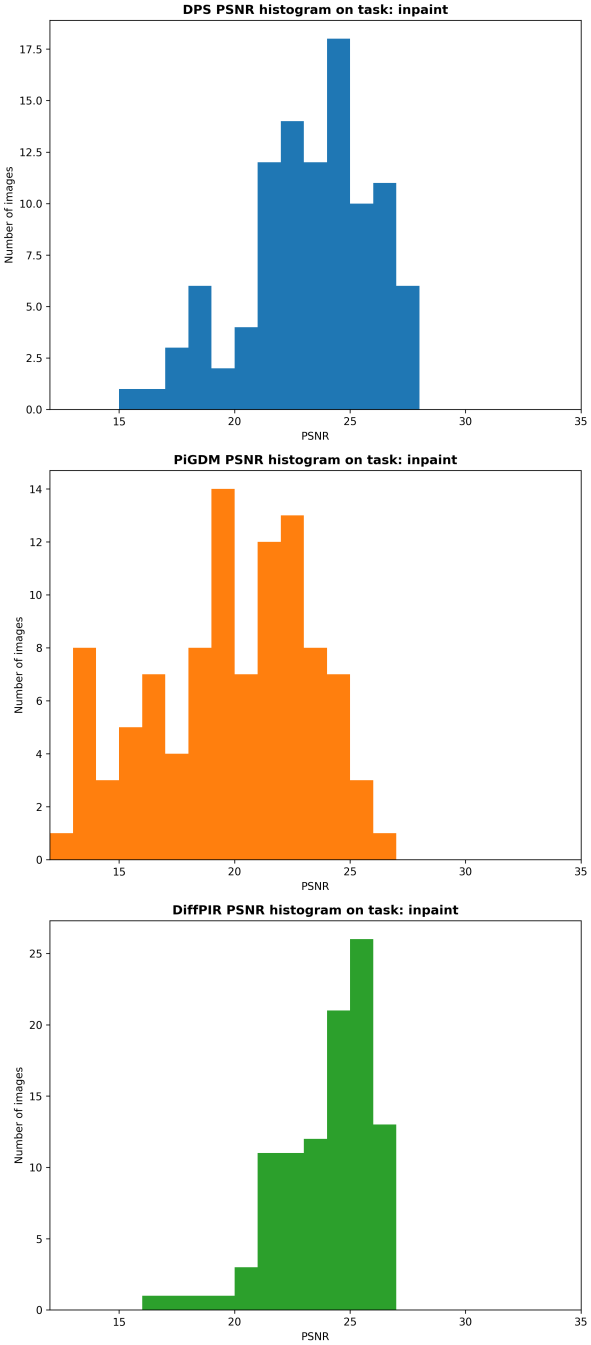

Figure 7: Histogram of PSNR values over 100 images for the inpainting task of batch 1

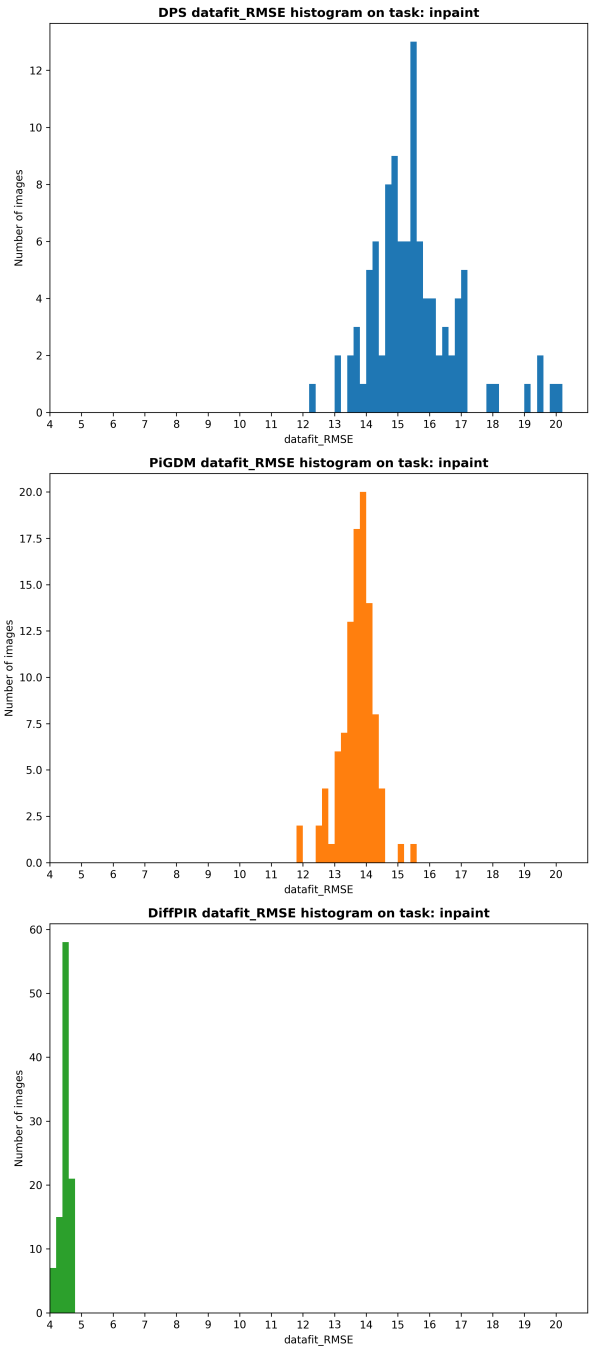

Figure 8: Histogram of datafit RMSE values over 100 images for the inpainting task of batch 1

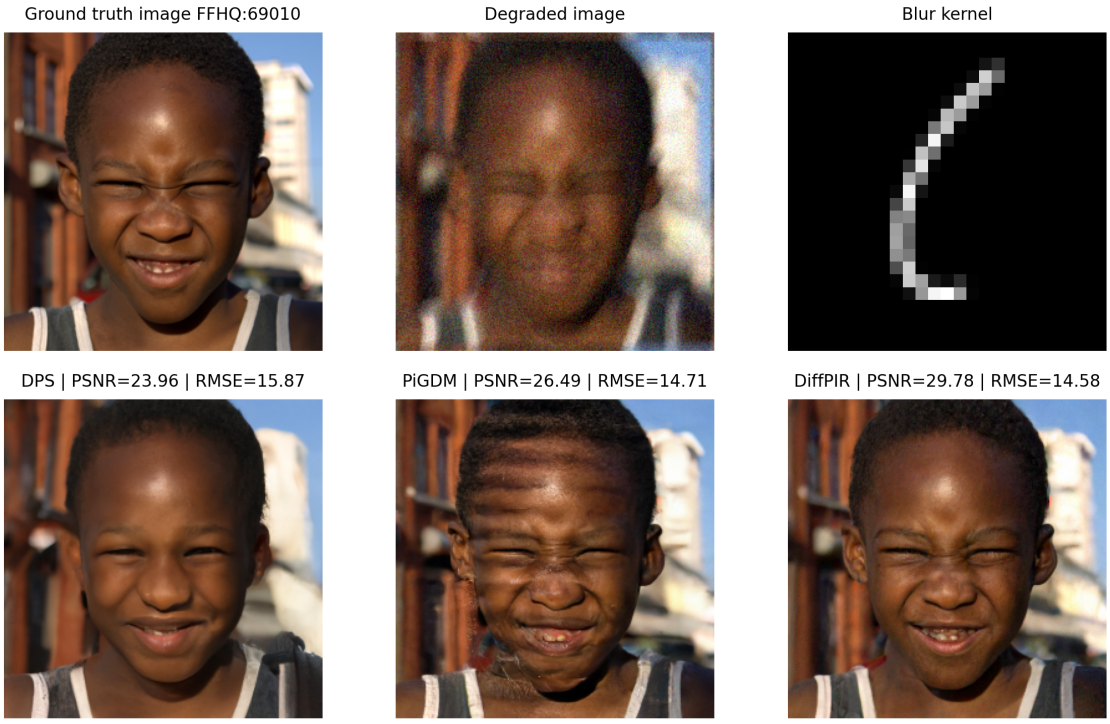

Figure 9: Example of image deblurring (FFHQ:69010)

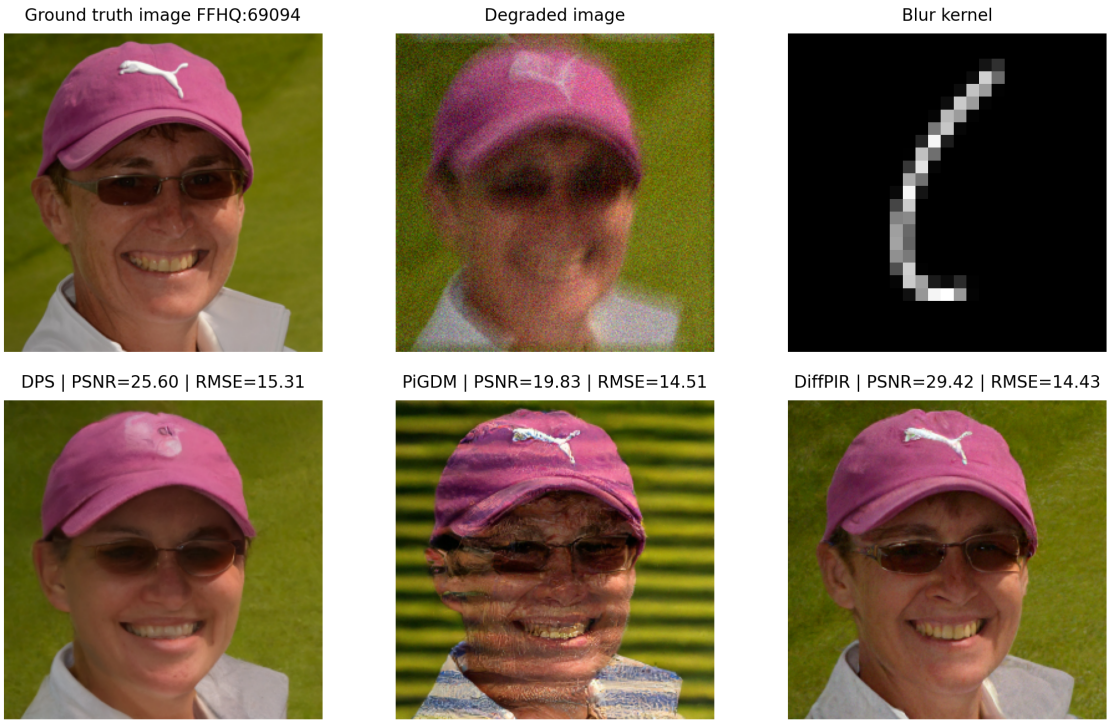

Figure 10: Example of image deblurring (FFHQ:69094)

Figure 11: Example of image deblurring (FFHQ:69025)

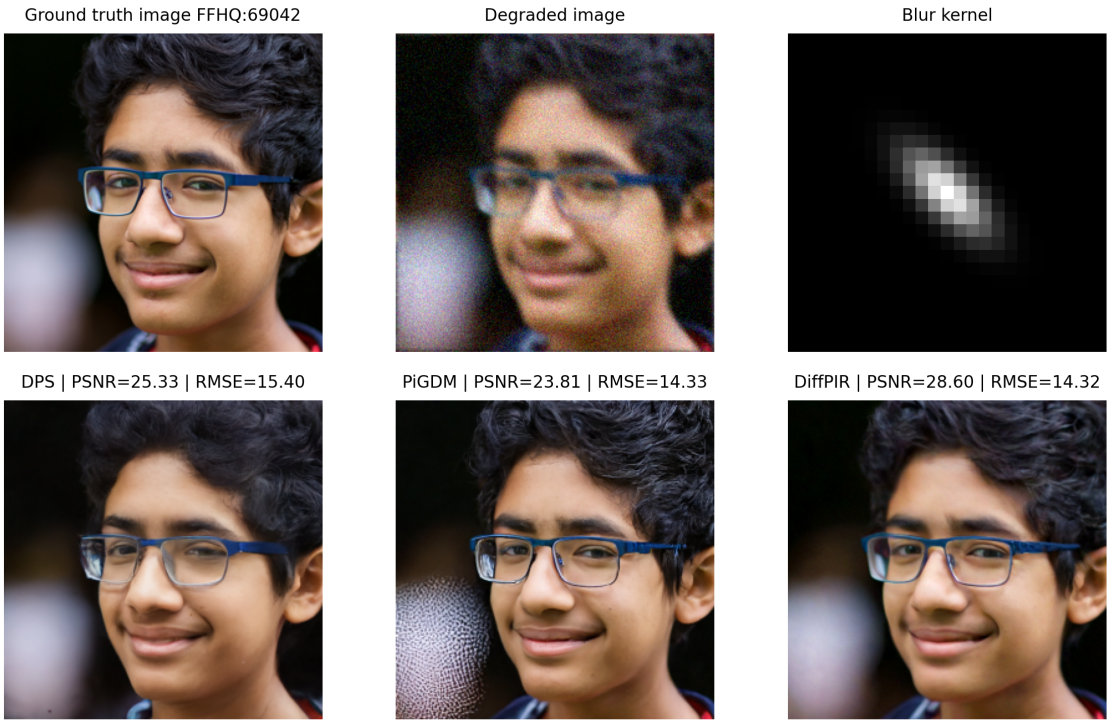

Figure 12: Example of image deblurring (FFHQ:69042)

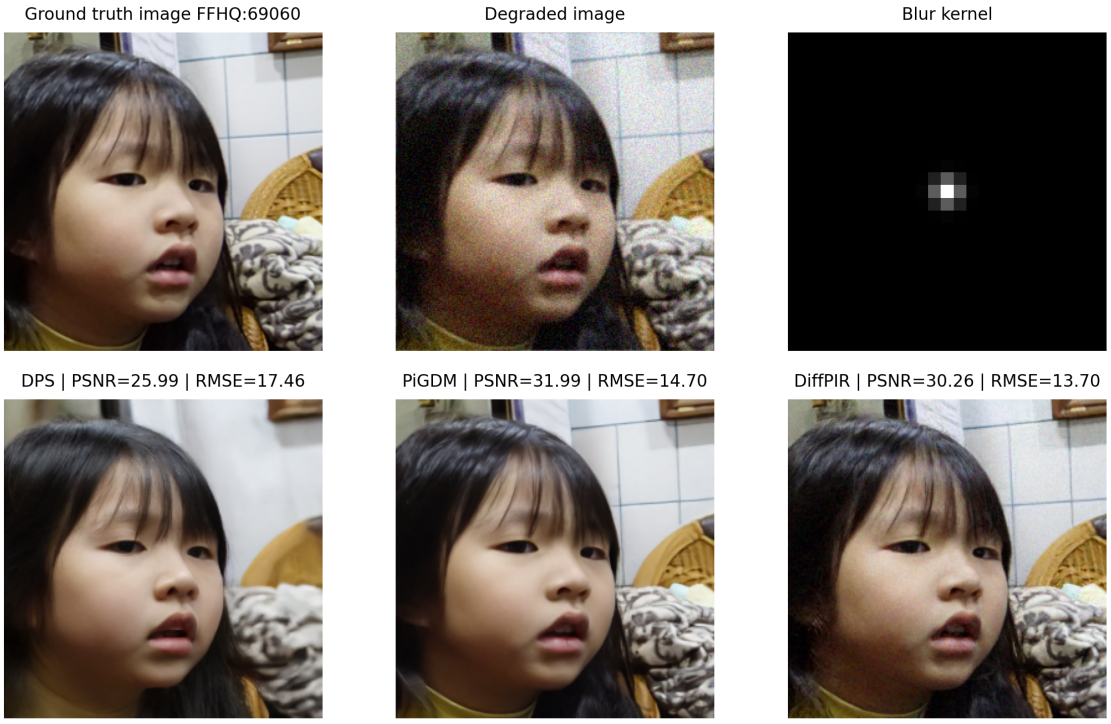

Figure 13: Example of image deblurring (FFHQ:69060)

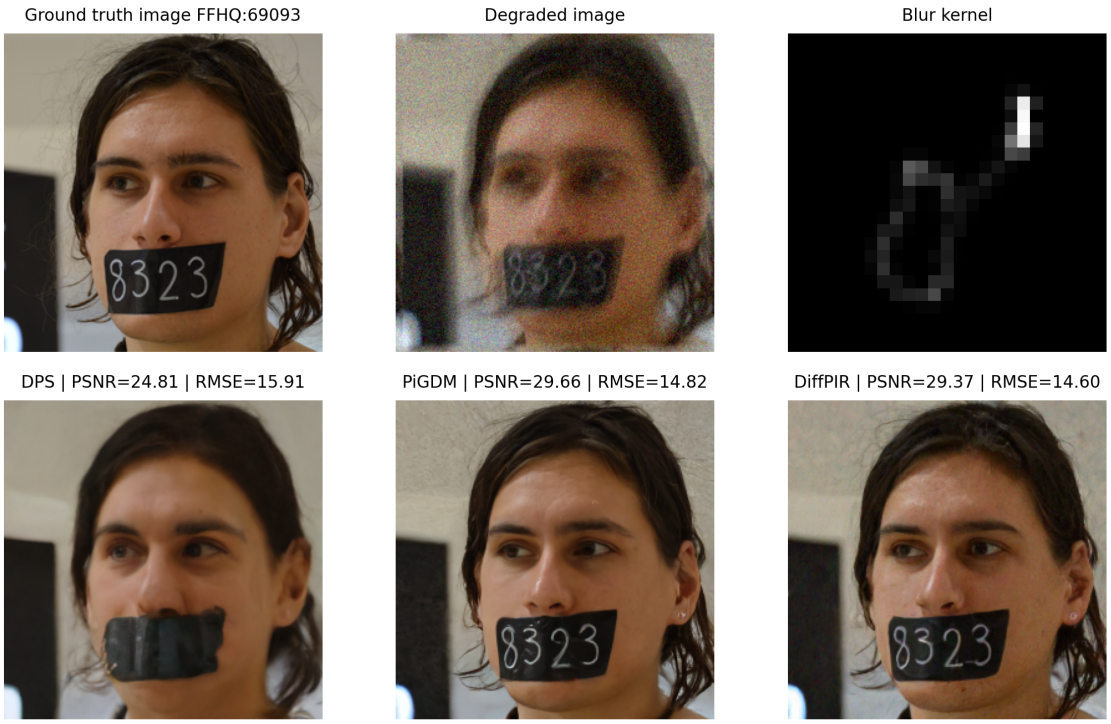

Figure 14: Example of image deblurring (FFHQ:69093)

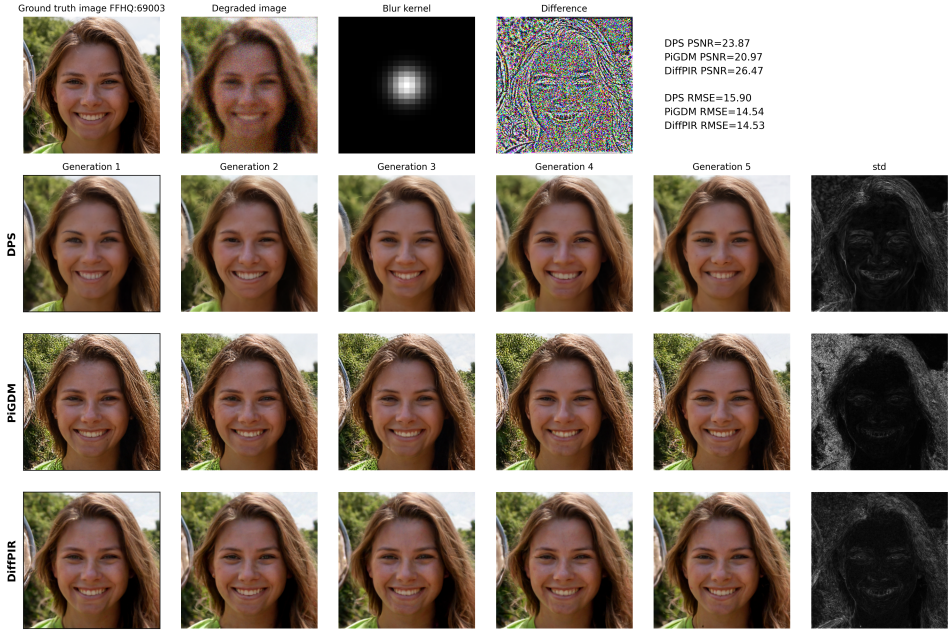

Figure 15: Example of the variability of each method for deblurring (FFHQ:69003)

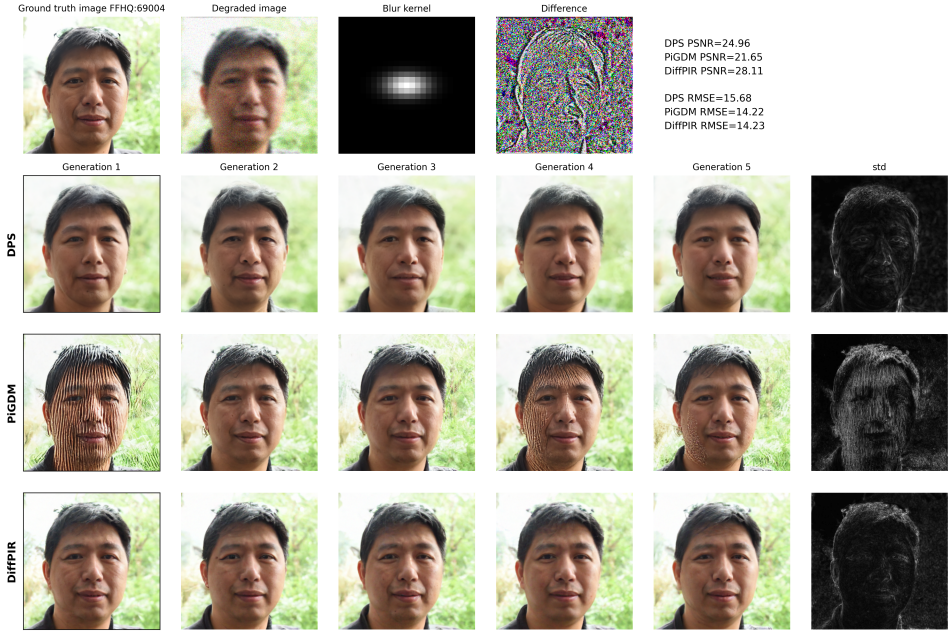

Figure 16: Example of the variability of each method for deblurring (FFHQ:69004)

Figure 17: Example of image inpainting (FFHQ:69019)

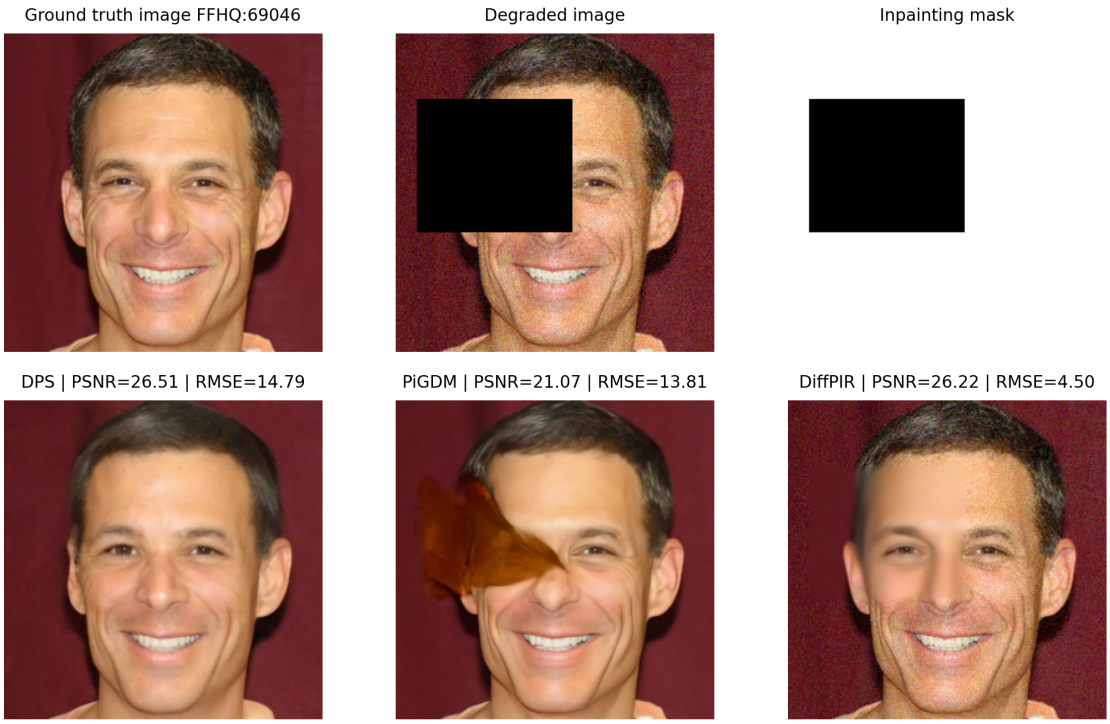

Figure 18: Example of image inpainting (FFHQ:69046)

Figure 19: Example of image inpainting (FFHQ:69037)

Figure 20: Example of image inpainting (FFHQ:69071)

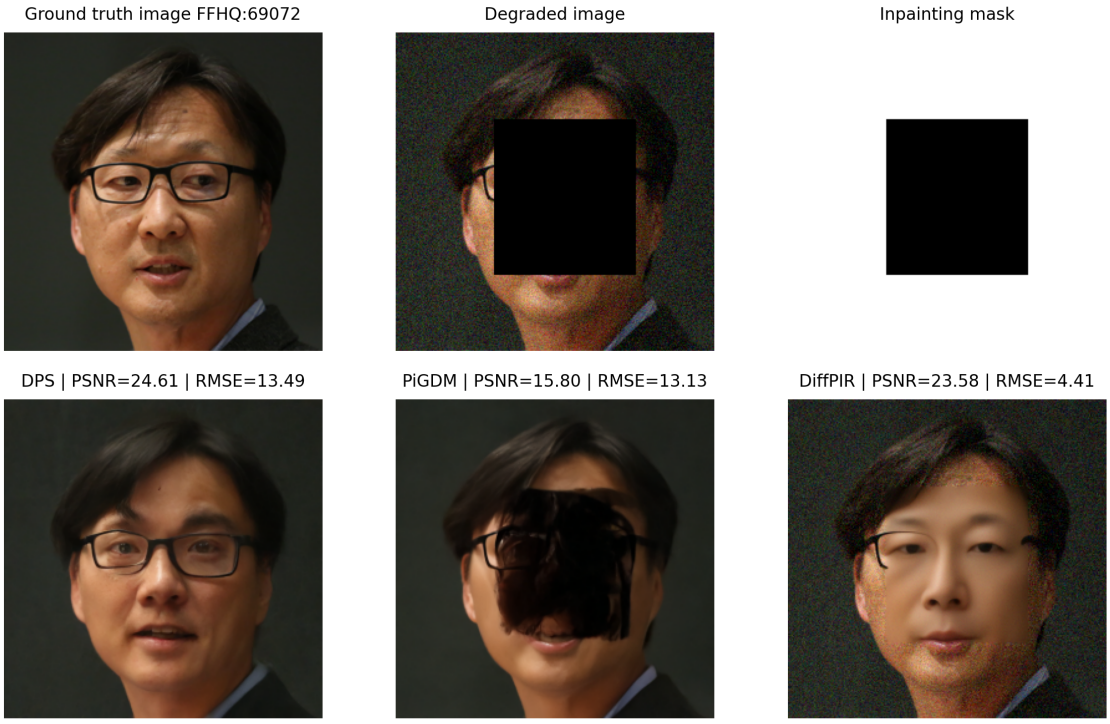

Figure 21: Example of image inpainting (FFHQ:69072)

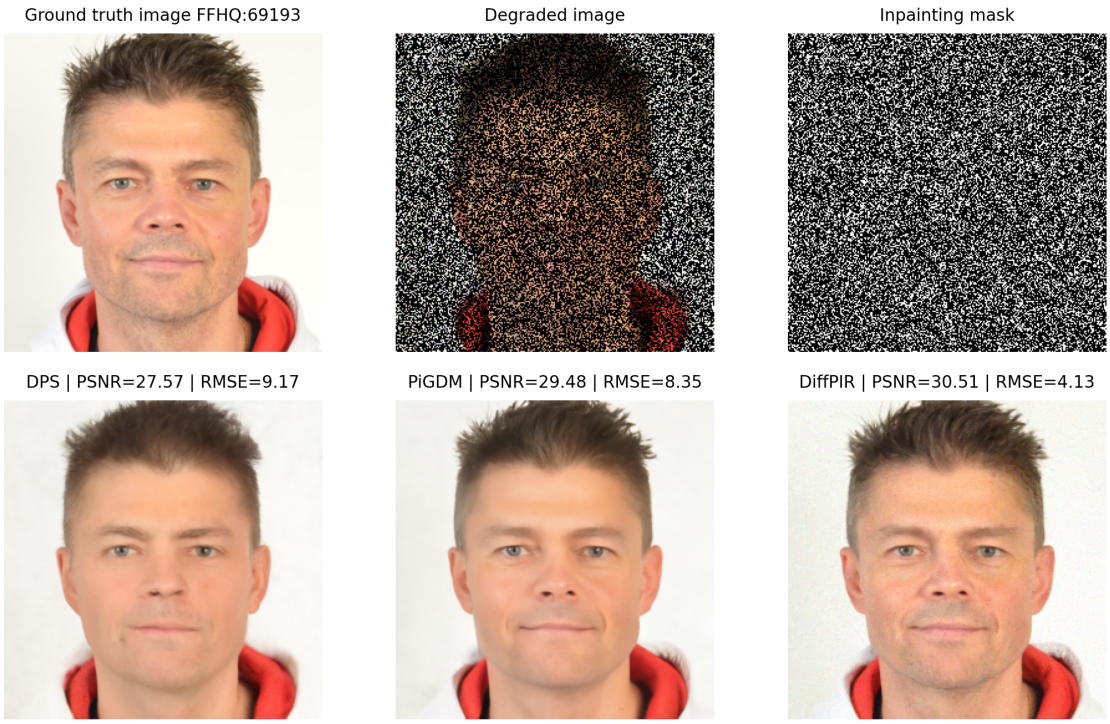

Figure 22: Example of image inpainting (FFHQ:69193)

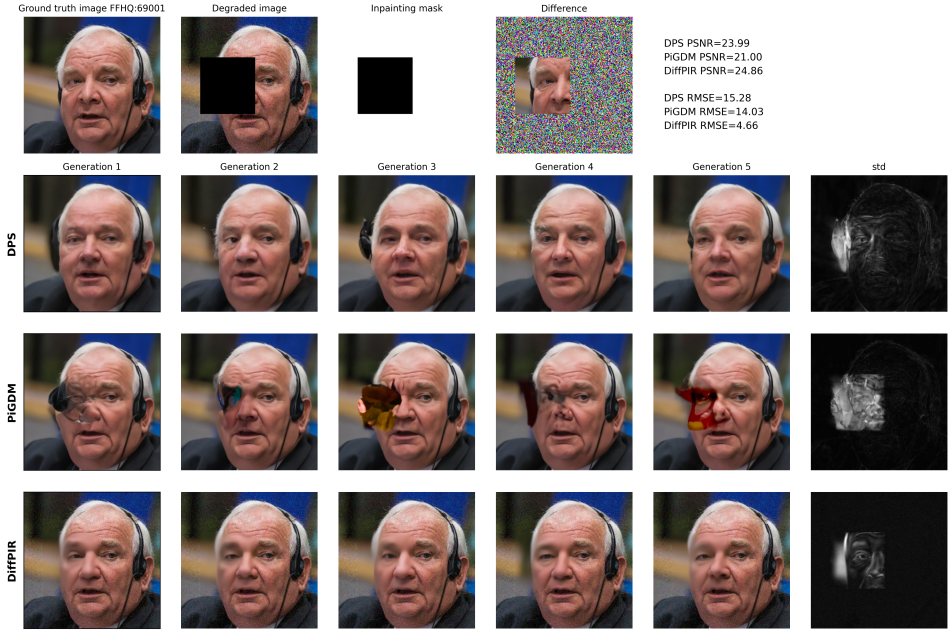

Figure 23: Example of the variability of each method for inpainting (FFHQ:69001)

Figure 24: Example of the variability of each method for inpainting (FFHQ:69008)