LATINO-PRO: LAtent consisTency INverse sOlver with PRompt Optimization

Supplementary Material

A. Technical details about diffusion ISs

DPS [7]: Diffusion Posterior Sampling (DPS) follows this update rule:

$$\boldsymbol{x}_{t-1} = \mathrm{DDIM}\left(\boldsymbol{x}_t\right) - \eta \nabla_{\boldsymbol{x}_t} \|\boldsymbol{y} - \mathcal{A}\hat{\boldsymbol{x}}_0\|_2^2,$$

where $\mathrm{DDIM}(\cdot)$ represents a single update step of the DDIM sampling [54], defined as:

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\boldsymbol{x}}_0 - (1-\alpha_{t-1}) s_\theta(\boldsymbol{x}_t, t),$$

where $\hat{\boldsymbol{x}}_0$ is estimated from $\boldsymbol{x}_t$, and s_θ is the predicted score at time t. The optimal step size η is dynamically set as $\eta = \frac{1}{\|\boldsymbol{y} - \mathcal{A}\hat{\boldsymbol{x}}_0\|_2^2}$, ensuring adaptive scaling of the likelihood gradient.

LDPS: Latent Diffusion Posterior Sampling (LDPS) can be seen as a direct extension of the image-domain DPS approach proposed by Chung et al. [7]. The update rule for LDPS is given by:

$$\boldsymbol{z}_{t-1} = \mathrm{DDIM}\left(\boldsymbol{z}_t\right) - \rho \nabla_{\boldsymbol{z}_t} \left\|\boldsymbol{y} - \mathcal{A}\mathcal{D}\left(\hat{\boldsymbol{z}}_0\right)\right\|_2$$

where ρ denotes the step size, and $\mathrm{DDIM}(\cdot)$ represents a single step of DDIM sampling. A static step size of $\rho = 1$ is employed, as is commonly adopted in the literature.

LDIR [16] modifies LDPS by introducing a momentum-based gradient update mechanism inspired by Adam. A single iteration of the algorithm follows:

$$\begin{aligned}
\boldsymbol{g}_t &= \nabla_{\boldsymbol{z}_t} \left\|\boldsymbol{y} - \mathcal{A}\mathcal{D}\left(\hat{\boldsymbol{z}}_0\right)\right\| \\
\hat{\boldsymbol{m}}_t &= \left(\beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\, \boldsymbol{g}_t\right) / \left(1-\beta_1\right) \\
\hat{\boldsymbol{v}}_t &= \left(\beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\left(\boldsymbol{g}_t \circ \boldsymbol{g}_t\right)\right) / \left(1-\beta_2\right) \\
\boldsymbol{z}_{t-1} &= \mathrm{DDIM}\left(\boldsymbol{z}_t\right) - \rho \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \varepsilon}
\end{aligned}$$

where $\circ$ denotes element-wise multiplication, and $\beta_1, \beta_2, \varepsilon$ are hyperparameters of the method. The momentum-based approach in LDIR leads to smoother gradient updates. The parameters are set as $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1e-8$. The step size ρ is set to be 0.05.

GML-DPS, PSLD [48]: GML-DPS introduces a constraint to ensure that the estimated clean latent $\hat{\boldsymbol{z}}_0$ remains stable after encoding and decoding. The update rule is:

$$\begin{aligned}
\boldsymbol{z}_{t-1} = \mathrm{DDIM}\left(\boldsymbol{z}_t\right) \\
- \rho \nabla_{\boldsymbol{z}_t} \left(\left\|\boldsymbol{y} - \mathcal{A}\mathcal{D}\left(\hat{\boldsymbol{z}}_0\right)\right\|_2 + \gamma \left\|\hat{\boldsymbol{z}}_0 - \mathcal{E}\left(\mathcal{D}\left(\hat{\boldsymbol{z}}_0\right)\right)\right\|_2\right)
\end{aligned}$$

PSLD refines this approach by incorporating an orthogonal projection step onto the subspace defined by $\mathcal{A}$ between the decoding and encoding stages to enforce fidelity:

$$\begin{aligned}
\boldsymbol{z}_{t-1} = \mathrm{DDIM}\left(\boldsymbol{z}_t\right) - \rho \nabla_{\boldsymbol{z}_t} \left\|\boldsymbol{y} - \mathcal{A}\mathcal{D}\left(\hat{\boldsymbol{z}}_0\right)\right\|_2 \\
- \gamma \nabla_{\boldsymbol{z}_t} \left\|\hat{\boldsymbol{z}}_0 - \mathcal{E}\left(\mathcal{A}^\top \boldsymbol{y} + \left(\mathrm{Id} - \mathcal{A}^\top \mathcal{A}\right) \mathcal{D}\left(\hat{\boldsymbol{z}}_0\right)\right)\right\|_2 .
\end{aligned}$$

A static step size of $\rho = 1$ is applied,and we set $\gamma = 0.1$. These methods aim at guiding latents toward the natural manifold, enforcing their stability after autoencoding.

P2L [8]: The P2L algorithm alternates between two main update steps: optimizing the text embedding c and refining the latent variable $\boldsymbol{z}_t$.

The first step focuses on updating the text embedding c to align it with the measurement $\boldsymbol{y}$ and the current diffusion estimate $\boldsymbol{z}_t$. This is done by maximizing the posterior $p(c \mid \boldsymbol{z}_t, \boldsymbol{y})$, leading to the gradient update:

$$\nabla_c \log p(c \mid \boldsymbol{z}_t, \boldsymbol{y}) \approx \nabla_c \|\mathcal{A}\mathcal{D}(\mathbb{E}[\boldsymbol{z}_0 \mid \boldsymbol{z}_t, c]) - \boldsymbol{y}\|_2^2.$$

This optimization uses stochastic optimizers like Adam [24].

In the second step, the latent variable $\boldsymbol{z}_t$ is refined using the optimized text embedding c_t^* obtained from the first step. This update aims at maximizing $p(\boldsymbol{z}_t \mid \boldsymbol{y}, c_t^*)$, resulting in the following gradient expression:

$$\begin{aligned}
\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z}_t \mid \boldsymbol{y}, c_t^*) \approx & s_\theta^*(\boldsymbol{z}_t, c_t^*) \\
& + \rho_t \nabla_{\boldsymbol{z}_t} \|\mathcal{A}\mathcal{D}(\mathbb{E}[\boldsymbol{z}_0 \mid \boldsymbol{z}_t, c_t^*]) - \boldsymbol{y}\|_2^2,
\end{aligned}$$

where $s_\theta^*(\boldsymbol{z}_t, c_t^*)$ is the score function from the diffusion model and ρ_t is a step size that balances the influence of the likelihood term.

TReg [23]: The TReg algorithm solves the following proximal optimization problem in an ADMM [5] style:

$$\begin{aligned}
\min_{\boldsymbol{x}, \boldsymbol{z}} l_{\text{MAP}}(\boldsymbol{z}) + \gamma l_{\text{TReg}}(\boldsymbol{z}) = l_{\text{MAP}}(\boldsymbol{z}) + \|\boldsymbol{z} - \hat{\boldsymbol{z}}_{0|t}\|_2^2 \\
\text{s.t.} \quad \boldsymbol{x} = \mathcal{D}(\boldsymbol{z})
\end{aligned}$$

where the objective of the maximum a posteriori (MAP) problem is defined as

$$\begin{aligned}
\ell_{\text{MAP}}(\boldsymbol{z}) &= -\log p(\boldsymbol{z} \mid \mathcal{D}(\boldsymbol{z}), \boldsymbol{y}) - \log p(\boldsymbol{y} \mid \mathcal{D}(\boldsymbol{z})) \\
&= \frac{\|\boldsymbol{z} - \mathcal{E}(\mathcal{D}(\boldsymbol{z}))\|_2^2}{2\sigma_{\mathcal{E}}^2} + \frac{\|\boldsymbol{y} - \mathcal{A}(\mathcal{D}(\boldsymbol{z}))\|_2^2}{2\sigma^2},
\end{aligned}$$

where $\sigma_{\mathcal{E}}$ is the encoder variance. First is solved

$$\hat{\boldsymbol{x}}_0(\boldsymbol{y}) = \min_{\boldsymbol{x}} \frac{\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{x})\|_2^2}{2\sigma^2} + \lambda \|\boldsymbol{x} - \mathcal{D}(\hat{\boldsymbol{z}}_{0|t})\|_2^2,$$

where $\boldsymbol{x} = \mathcal{D}(\boldsymbol{z})$, and then:

$$\begin{aligned}\hat{\boldsymbol{z}}_0^{ema} &= \text{argmin}_{\boldsymbol{z}}\, \zeta\|\boldsymbol{z} - \mathcal{E}(\hat{\boldsymbol{x}}_0(\boldsymbol{y}))\|_2^2 + \gamma\|\boldsymbol{z} - \hat{\boldsymbol{z}}_{0|t}\|_2^2 \\ &= \alpha_{t-1}\mathcal{E}(\hat{\boldsymbol{x}}_0(\boldsymbol{y})) + (1-\alpha_{t-1})\hat{\boldsymbol{z}}_{0|t} \qquad (10)\end{aligned}$$

where ζ, γ are empirically chosen to satisfy $\alpha_{t-1} = \zeta/(\zeta + \gamma)$ in order to give the second equality in (10).

After these two steps, a DDIM step is run and eventually the null prompt is optimized through what is called "Adaptive Negation", i.e.:

$$c_\emptyset \leftarrow c_\emptyset - \eta\nabla_\emptyset(\mathcal{T}_{\text{img}}(\hat{\boldsymbol{x}}_0(\boldsymbol{y})), c_\emptyset)$$

where η is a fixed learning rate and $\mathcal{T}_{\text{img}}$ denotes the CLIP image encoder.

B. Hyperparameters tuning

As deeply studied in the theoretical derivation of our method in Section 4, we introduce the hyperparameter δ_k as it represents the implicit Euler step size. We will now show the values of δ_k used for each task.

1. **Gaussian Deblurring:**
   - For $k \geq 5$: $\delta_k = 2 \cdot 10^{-5}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$
   - Otherwise: $\delta_k = 4 \cdot 10^{-5}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$

2. **Motion Deblurring:**
   - For $k \geq 5$: $\delta_k = 4 \cdot 10^{-6}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$
   - Otherwise: $\delta_k = 2 \cdot 10^{-6}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$

3. **Super Resolution $\times 8$:**
   - For $k \geq 6$: $\delta_k = 6 \cdot 10^{-3}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$
   - Otherwise: $\delta_k = 3 \cdot 10^{-3}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$

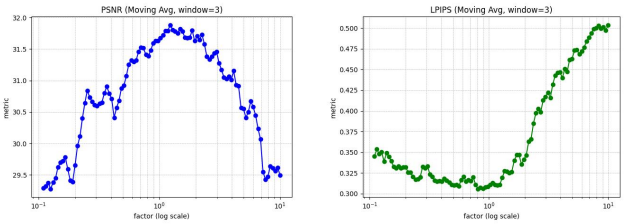
4. **Super Resolution $\times 16$:**
   - For $k \geq 6$: $\delta_k = 2 \cdot 10^{-2}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$
   - Otherwise: $\delta_k = 9 \cdot 10^{-3}(1-\alpha_{t_k})\|\mathcal{A}\boldsymbol{u}^{(k)} - \boldsymbol{y}\|/\sigma_n$

5. **Box Inpainting:**
   - For $k \geq 5$: $\delta_k = (1-\alpha_{t_k})$
   - Otherwise: $\delta_k = 0.5(1-\alpha_{t_k})$

These choices can be motivated in the following way: the normalized L^2 norm acts as a regularizer that strengthens the data-fidelity term when the reconstruction is poor (i.e. big L^2 norm) and gives more freedom to the prior otherwise. In particular, we expect the norm to be big in the first steps, when we need to prevent the prior from deviating from the observation, and small in the final steps when it is more important to be able to generate detailed high-frequency features that cannot be recovered from the noisy observation. A similar reasoning leads to the addition of the $1-\alpha_{t_k}$ term.

The PSNR/LPIPS performance of our method is quite robust to the choice of $\delta_k \in (0.1\delta_k^*, 10\delta_k^*)$ as shown below for gaussian deblurring: **LATINO** uses the optimal δ_k^* as defined above; **LATINO-s** uses $\delta_k = 0.1\delta_k^*$; **LATINO-b** uses $\delta_k = 10\delta_k^*$.

Figure 6. Qualitative comparison on Gaussian deblurring with different δ_k schedules. LATINO uses δ_k^*, LATINO-b uses $10\delta_k^*$, and LATINO-s uses $0.1\delta_k^*$.



PSNR vs. δ_k **LPIPS vs.** δ_k

Figure 7. Robustness of LATINO's choice of δ_k: performance curves on FFHQ-1024 Gaussian deblurring.

C. LATINO as a split-step Langevin sampler

As previously noted, computing exact solutions to the Langevin diffusion process (3) is generally not possible. Therefore, solutions are usually obtained by using a discrete-time numerical integrator whose accuracy and cost are controlled by the size of integration time step. LATINO employs a split-step discretization of the Langevin diffusion process (3) in which the Brownian motion and the drift term associated with the prior density are approximately integrated via the stochastic auto-encoding step (4.a). The likelihood term is handled via an implicit (backwards Euler) or proximal step (4.b), hence the iterate x_{k+1} appears on both sides of the second row, resulting in improved stability properties that permit larger step sizes [1]. The Langevin SDE is a time-homogeneous process, hence $g_y : x \mapsto -\log p(y|x)$ is the exact likelihood or data fidelity term, a key advantage w.r.t. DPS, ΠGDM, DiffPIR, etc., which require approximations. Indeed, the use of the Langevin SDE allows employing the likelihood of y w.r.t. the (noise-less) image x, which is usually tractable. In contrast, strategies such as DPS or ΠGDM seek to embed the likelihood of y w.r.t. a noisy version of x within a time-inhomogeneous reverse diffusion process; such likelihoods are often intractable and require approximations. Also note that the iteration index k is related to the

time of the Langevin diffusion (3)-(4), which goes forward as the algorithm iterations progress. It is not the time of the diffusion SDE (1) which is encapsulated into ((4), top row).

With regards to convergence properties of LATINO, known theoretical convergence results for PnP Langevin sampling suggest that when t is mall, LATINO should converge under a wide class of probability metrics towards a biased approximation of the posterior distribution of interest [26]. Empirically, we observe that LATINO converges very quickly, especially when t is large, allowing to generate samples in very few steps. A theoretical analysis of the convergence of LATINO for large t is a main perspective for future work.

D. LATINO-PRO: gradient computation

As discussed in Section 5, the key step of our LATINO-PRO Algorithm 2 is the computation of the following quantity

$$c_{m+1} = \Pi_C \left[c_m + \gamma_m \nabla_c \log p(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)} | c_m) \right], \tag{11}$$

where $\{\boldsymbol{x}^{(k)}\}_{k=1}^N$ is a Markov chain targeting $p(\boldsymbol{x}|\boldsymbol{y}, c_m)$. This requires running a full iteration of our LATINO algorithm 1, and in particular, we are interested in storing the latent realizations $\{\boldsymbol{z}_{t_k}^{(k)}\}_{k=1}^N$, as this leads to tractable computations by automatic differentiation (to simplify notation, we henceforth use use $\boldsymbol{z}_{t_k} \equiv \boldsymbol{z}_{t_k}^{(k)}$ and $c \equiv c_m$). During the optimization steps in Algorithm 2, we consider $N = 4$, so the computations become:

$$\begin{aligned} &\log p(\boldsymbol{z}_{t_1}, \boldsymbol{z}_{t_2}, \boldsymbol{z}_{t_3}, \boldsymbol{z}_{t_4} \mid c) = \log p(\boldsymbol{z}_{t_4} \mid \boldsymbol{z}_{t_3}, c) + \\ &+ \log p(\boldsymbol{z}_{t_3} \mid \boldsymbol{z}_{t_2}, c) + \log p(\boldsymbol{z}_{t_2} \mid \boldsymbol{z}_{t_1}, c) + \log p(\boldsymbol{z}_{t_1} \mid c). \end{aligned} \tag{12}$$

All the terms can be computed through the definition of the latent part of our stochastic auto-encoder, i.e.,

$$\boldsymbol{z}_{t_{i+1}} = \sqrt{\alpha_{t_{i+1}}} G_\theta(\boldsymbol{z}_{t_i}, t_i, c) + \sqrt{1 - \alpha_{t_{i+1}}} \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{Id}),$$

and hence

$$\boldsymbol{z}_{t_{i+1}} \mid \boldsymbol{z}_{t_i}, c \sim \mathcal{N}(G_\theta(\boldsymbol{z}_{t_i}, t_i, c), (1 - \alpha_{t_{i+1}})\mathrm{Id}),$$

so

$$-\nabla_c \log p(\boldsymbol{z}_{t_{i+1}} | \boldsymbol{z}_{t_i}, c) = \frac{\nabla_c \| \boldsymbol{z}_{t_{i+1}} - \sqrt{\alpha_{t_{i+1}}} G_\theta(\boldsymbol{z}_{t_i}, t_i, c) \|^2}{2(1 - \alpha_{t_{i+1}})}.$$

This holds for all terms in (12), including $\log p(\boldsymbol{z}_{t_1} \mid c)$, for which we simply have a dependence on the starting $\boldsymbol{z}_0 \sim \mathcal{N}(\mathcal{A}^\dagger \boldsymbol{y}, (1 - \alpha_{t_0})\mathrm{Id})$ in the equation. Also, we do not include $p(\boldsymbol{z}_{t_4} \mid \boldsymbol{z}_{t_3}, c)$ as $\boldsymbol{z}_{t_4}$ is deterministically determine from $\boldsymbol{z}_{t_3}$. Instead, $\boldsymbol{z}_{t_4}$ initializes the next iteration of LATINO within the SAPG scheme. In conclusion, (11) becomes

$$c_{m+1} = \Pi_C \left[c_m + \gamma_m \nabla_c \sum_{i=0}^{2} \frac{\nabla_c \| \boldsymbol{z}_{t_{i+1}} - \sqrt{\alpha_{t_{i+1}}} G_\theta(\boldsymbol{z}_{t_i}, t_i, c) \|^2}{2(1 - \alpha_{t_{i+1}})} \right],$$

E. Adaptation to non-linear operators

When the pseudoinverse is not accessible, the proximal operator can be computed with Conjugate Gradient in the linear case. For nonlinear operators, a direct least squares method can be adopted, as already done in [23], using the Adam optimizer with learning rate $1\mathrm{e}{-3}$ and $\beta_1 = 0.9$, $\beta_2 = 0.999$ for 300 iterations to obtain the solution of

$$\min_{\boldsymbol{x}} \frac{\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{x})\|_2^2}{2\sigma^2} + \lambda \|\boldsymbol{x} - \hat{\boldsymbol{x}}_0\|_2^2.$$

In Fig. 9, we tackle a non-linear phase retrieval task

$$\boldsymbol{y} = \left| \mathrm{DFT}(\boldsymbol{x}) \right| + \boldsymbol{n}$$

on FFHQ-512, and compare LATINO-PRO with TReg and LDPS (P2L and PSLD only work for linear problems). Our method is around $\times 4$ faster than TReg and can handle tougher cases like images with complex backgrounds, which cause failures in TReg (bottom row). We stress the fact that a key strength of our method is the possibility to use various discretization schemes in place of the implicit proximal in 4 (implicit–explicit, Runge–Kutta) and even off-the-shelf NN to approximate $\mathrm{prox}_{\delta g_y}$ when a closed form is not available.

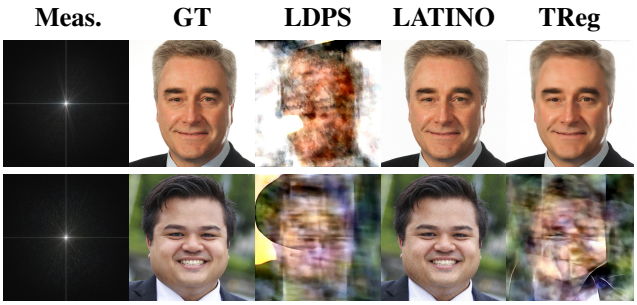


Figure 9. Nonlinear phase retrieval. Top row: Example 1; bottom row: Example 2.

F. Ablation study: prompt choice

Table 5 highlights the robustness of the reconstruction quality to slight semantic variations of the initial prompt. In particular, we observe that less informative prompts often yield better metrics than those that include information about the degradation operator. LATINO-PRO is more robust to variations in the prompt initialization, as it seems that the optimization scheme converges towards an optimal prompt in all three cases.

Prompt	LATINO			LATINO-PRO		
	LPIPS ↓	PSNR ↑	FID ↓	LPIPS ↓	PSNR ↑	FID ↓
"a photo of"	**0.312**	**26.93**	29.24	**0.299**	**27.25**	28.39
"a high resolution photo of"	0.319	26.85	29.22	0.301	27.19	**27.59**
"a sharp photo of"	0.318	26.88	**29.17**	0.301	27.14	27.80

Table 5. Performance of LATINO and LATINO-PRO on FFHQ-1024 1k test dataset, motion blur task, under different prompts.

		Deblur (Gaussian)		SR×16	
Method	**NFE↓**	**FID↓**	**PSNR↑**	**FID↓**	**PSNR↑**
LATINO-PRO	68	**18.37**	**26.82**	**30.40**	**21.52**
LATINO	**8**	20.03	26.25	42.14	20.05
LATINO-LoRA	**8**	57.96	23.02	76.53	17.82

Table 3. Results for Gaussian Deblurring with $\sigma = 5.0$, and $\times 16$ super-resolution, both with noise $\sigma_y = 0.01$ on the AFHQ-512 val dataset. Our LATINO, LATINO-PRO, and LATINO-LoRA models are compared. Prompt: `a sharp photo of a dog` or `a sharp photo of a cat`. **Bold**: best, underline: second best.

		Deblur (Gaussian)			Deblur (Motion)			SR×8		
Method	**NFE↓**	**FID↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**PSNR↑**	**LPIPS↓**
LATINO-PRO	68	**31.98**	**29.11**	**0.292**	**27.80**	**27.14**	**0.301**	40.95	**26.58**	**0.355**
LATINO	**8**	33.94	28.95	0.296	29.17	26.88	0.318	**37.13**	26.22	0.356
LATINO-LoRA	**8**	33.70	28.20	0.340	40.66	24.83	0.407	50.89	25.80	0.428

Table 4. Results for Gaussian deblurring with $\sigma = 3.0$, motion deblurring, and $\times 8$ super-resolution, all with noise $\sigma_y = 0.01$ on the FFHQ-512 val dataset. Our LATINO, LATINO-PRO, and LATINO-LoRA models are compared. Prompt: `a sharp photo of a face`. **Bold**: best, underline: second best.

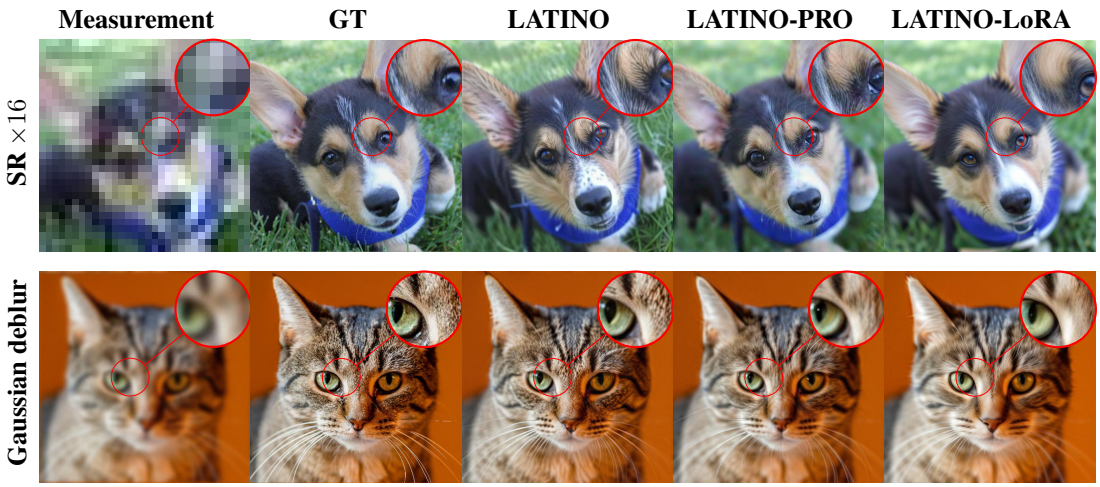


Figure 8. Qualitative comparison of image restoration results. Samples taken from AFHQ-512. Prompt: `a sharp photo of a dog` or `a sharp photo of a cat`.

G. Ablation study: prior choice

Alongside DMD2 [64], other distilled models can be found in the literature, some of which are based on the SD1.5 [47] backbone. [31] introduces a LoRA fine-tuning of SD1.5 that allows a few-step sampling following the CM scheme. We decided to try also this model to show the universal adaptability of our model.

We consider the 8-step version of SD1.5-LoRA, and we tried to solve the same inverse problems as done in Section 6: Gaussian deblurring with $\sigma = 5.0$ and SR×16 on the AFHQ-512 1k val dataset, Gaussian deblurring with $\sigma = 3.0$ and SR×8 on the FFHQ-512 1k val dataset; in all cases $\sigma_n = 0.01$. Table 3 and Table 4 sum up these results, while Figure 8 and Figure 12 show an extended visual comparison. We call this version of our algorithm LATINO-LoRA.

To better understand the difference in performances between the SD1.5-LoRA [31] and the DMD2 [64], we provide in Figure 10 some prior-generated images from the same prompts used during the reconstructions, focusing on the faces case. It is evident how SD1.5-LoRA tends to generate cartoonish features that are good-looking but unrealistic and that this increases the perceptual distance during the reconstruction process. At the same time, we show how the original SD1.5 can generate quite realistic faces as well, comparable to the DMD2 ones.

To further show that the performances are not related

to the improved capacities of the LCM prior, we compare LATINO with LDPS, PSLD, and P2L using SDXL, the LDM from which DMD2 [64] was distilled, as the prior. Since SDXL uses 50 time steps, while the other methods use 1000, we report in Figure 11 results on a Gaussian deblurring example on FFHQ-1024 for both settings.

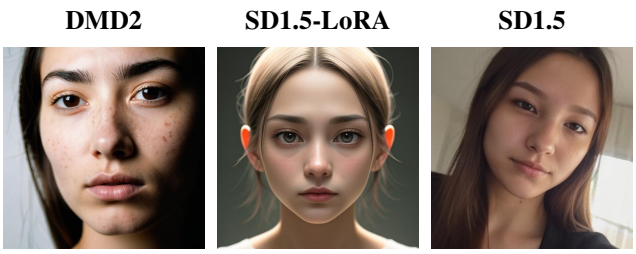


Figure 10. Prior comparisons. Prompt: `a photo of a face`.

H. Equivalent HR problems

H.1. The deblurring case

In order to precisely compare our deblurring results with the ones of methods that work with lower resolutions we need to transform our low-resolution problem into an equivalent high-resolution one, and go back to lower resolution.

LR problem: Our original problem is the following deblurring one: Find x from measurement y obtained via (1)

$$(1) \quad y = x * h + n$$

where $*$ denotes convolution, and h is a low-resolution blur kernel.

HR problem: Our proxy high-resolution problem is a joint deblurring and super-resolution problem: Find X from measurements y obtained via (2)

$$(2) \quad y = S_s(X * H) + n$$

where H is a high-resolution blur kernel. The output of our algorithm is $x = S_s(X)$, where S_s is a downsampling operator to be defined below. In our case $s = 2$ since our method works at resolution 1024×1024 and the other algorithms work at resolution 512×512.

If we want problems (1) and (2) to be equivalent, we need to find a high-resolution kernel H such that:

$$S_s(X) * h = S_s(X * H). \tag{13}$$

The following definition establishes a family of subsampling operators S_s for which condition (13) is satisfied, as long as the HR kernel H is chosen as shown in proposition 1.

Definition 2. *An alias-free subsampling operator S_s is defined as:*

$$S_s(X) = \downarrow_s (h_s * X)$$

where $\downarrow_s$ is the decimation operator (which takes one sample every s pixels without any filtering) and h_s is a convolution kernel with spectral support in $[-\pi/s, \pi/s]$.

The following choices provide alias-free subsampling operators (or approximations thereof)

1. $h_s = \text{sinc}(\cdot/s) = \text{sinc}_s$, i.e. standard Shannon subsampling.
2. $h_s = \mathcal{F}^{-1}(\varphi)$ where φ is a smooth function with support in $[-\pi/s\, \pi/s]$. Avoids Gibbs artifacts that are commonly associated with Shannon subsampling.
3. h_s = kernel implicit in spline downsampling of order k. This is not exactly alias-free, but a good approximation of Shannon subsampling for sufficiently large k. In practice we use bicubic downsampling for $k = 3$, which is a sufficiently good approximation.

Proposition 1. *If S_s is an alias-free subsampling operator, then any H satisfying*

$$h = \downarrow_s (\text{sinc}_s * H) \tag{14}$$

satisfies the equivalence condition (13).

Proof. Let N^2 be the size of image X. Let P_s denote the periodization operator with period N/s. Then using the fact that convolution (respectively s-sampling) becomes a product (respectively N/s-periodization) we have that

$$\begin{aligned}
&\mathcal{F}(S_s(X * H)) \\
=&\mathcal{F}(\downarrow_s (h_s * X * H)) \\
=&P_s\left[\mathcal{F}(h_s)\mathcal{F}(X)\mathcal{F}(H)\mathcal{F}(\text{sinc}_s)\right] \\
=&P_s\left[\mathcal{F}(h_s)\mathcal{F}(X)\right] P_s\left[\mathcal{F}(H)\mathcal{F}(\text{sinc}_s)\right] \\
=&\mathcal{F}(\downarrow_s (h_s * X))\mathcal{F}(\downarrow_s (H * \text{sinc}_s)) \\
=&\mathcal{F}(S_s(X))\mathcal{F}(\downarrow_s (H * \text{sinc}_s)).
\end{aligned}$$

The third line is true because both $\mathcal{F}(h_s)\mathcal{F}(X)$ and $\mathcal{F}(H)\mathcal{F}(\text{sinc}_s)$ are supported in $[-\pi/s, \pi/s]$. Taking inverse fourier transform we have

$$S_s(X * H) = S_s(X) * \downarrow_s (H * \text{sinc}_s) = S_s(X) * h$$

and the equivalence is established. □

Practical Considerations. When S_s is Shannon subsampling, any H such that $\hat{H}|_{[-\pi/s,\pi/s]^2} = \hat{h}$ satisfies condition (14). In particular we can choose Shannon (zero-padding) upsampling, or a kernel H that satisfies (14) and minimizes the total variation (to minimize Gibbs artifacts).

Similarly when S_s is bicubic downsampling, we choose H as bicubic upsampling of h. In this case we have an approximation of conditions (13) and (14) that is good enough for our purposes.

Figure 11. Comparison on a sample from FFHQ-1024 using SDXL as an LDM prior. Prompt: `a sharp photo of a face`.

Figure 12. Qualitative comparison of image restoration results. Samples taken from FFHQ-512. Prompt: `a sharp photo of a face`.

H.2. The super-resolution case

To precisely compare our super-resolution results with the ones of methods that work with lower resolutions, we need to transform our low-resolution problem into an equivalent high-resolution one, and go back to lower resolution.

LR problem: Our original problem is the following super-resolution problem: Find x from the measurements y obtained via (1)

$$(1) \quad y = S_a(x) + n,$$

where S_s is a downsampling operator of factor $s > 0$.

HR problem: Our proxy high-resolution problem is a super-resolution problem: Find X from measurements y obtained via (2):

$$(2) \quad y = S_{ab}(X) + n.$$

The output of our algorithm is $x = S_b(X)$. In our examples $a = 8$ and $b = 2$ so that $ab = 16$, or $a = 16$ and $b = 2$ so that $ab = 32$.

If we want problems (1) and (2) to be equivalent, we need to find a subsampling operator such that:

$$S_{ab}(X) = S_a(S_b(X)). \tag{15}$$

Any subsampling operator derived from a wavelet transform (including average pooling) satisfies condition (15). So does Shannon sub-sampling. Bicubic downsampling approximately satisfies condition (15) since it is a good approximation of Shannon subsampling.

I. Additional results: FFHQ1024

In Figures 13, 14 and 15 we show more tests at the 1024×1024 resolution. We also provide in Table 6 the metrics for the FFHQ-1024 1k val dataset, to allow comparisons with future works. The tasks considered are the same of Section 6, adapted to the new resolution, as discussed in H.

		Deblur (Gaussian)			Deblur (Motion)			SR×16		
Method	**NFE↓**	**FID↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**PSNR↑**	**LPIPS↓**
LATINO-PRO	68	31.98	28.76	0.372	27.80	26.89	0.423	40.95	25.81	0.445
LATINO	8	33.94	28.41	0.382	29.17	26.58	0.445	37.13	25.71	0.450

Table 6. Results for Gaussian deblurring with $\sigma = 6.0$, motion deblurring, and $\times 16$ super-resolution, all with noise $\sigma_y = 0.01$ on the FFHQ-1024 val dataset. Our LATINO and LATINO-PRO are compared. Prompt: `a sharp photo of a face`.

J. Memory and time consumption

In Table 7 we provide an exhaustive comparison of our models with respect to current SOTA in terms of memory consumption and time needed. We implemented versions of P2L and TReg starting from the official codebases as described in [8, 23]. For estimating the memory consumption and speed in the XL versions of these two methods, we adapted such codes to the SDXL prior. As an important remark, we highlight that SD provides float16 implementations (half precision) to speed up and reduce memory usage. However, this implementation does not allow taking gradients with respect to the score and decoder network, as it results in an integer overflow error during the `torch.autograd()` call. This forces all the implementations below (except for TReg, as it does not require such gradients) to make use of the float32 (full precision) implementation of SDXL, which explains the even bigger overhead in GPU usage and time.

Method	**GPU (Gb)**	**Time (s)**	**Resolution**
LATINO	13.6	5.53	1024^2
LATINO-PRO	23.4	48.8	1024^2
LATINO-LoRA	4.16	2.89	512^2
TReg	7.75	60.5	512^2
P2L	8.18	402	512^2
LDPS	8.16	279	512^2
PSLD	9.44	326	512^2
LDPS-XL	56.6	1670	1024^2
PSLD-XL	69.5	2200	1024^2
TReg-XL	33.5	240	1024^2
P2L-XL	57.1	6800	1024^2

Table 7. GPU Memory and Time consumption comparison

The NFEs considered for each algorithm are the same as shown in Table 1. As a reference, we also provide in Table 8 the GPU memory consumption of the respective priors, which can be considered as lower bounds. The times are those obtained by running the algorithms on a single Nvidia A100 GPU, averaging the times of the different inverse problems considered.

Prior Method	**GPU (Gb)**	**Resolution**
DMD2	10.7	1024^2
SD1.5	3.25	512^2
SD1.5LoRA	3.84	512^2

Table 8. GPU Memory consumption when sampling an image with different generative models priors).

K. Comparison with TReg

One of the main strengths of SOTA algorithms such as TReg is the possibility of shifting the semantic domain of the reconstruction through the prompt c. As we described in Section 5, our algorithm can obtain the same type of results, providing a useful fast and light tool for image editing. To prove this, we performed experiments on the Food-101 dataset [4] as done in [23] as shown in Figure 16. The degradations used are Gaussian Deblurring of intensity $\sigma = 5.0$ and super-resolution $\times 16$, both with an additional white noise of $\sigma_n = 0.01$. All the images are at 512×512 resolution, meaning that, as for the AFHQ dataset, we have to rescale the problems to their equivalent 1024×1024 resolution versions as seen in H.1,H.2. In particular, the images obtained are at a higher resolution than the original ones, i.e. 1024×1024.

To further show the improvements in the reconstruction quality with respect to both TReg and P2L, we provide a comparison of the results obtained on the AFHQ val dataset. Together with Table 1, we show in Figure 17 and 18 visual examples.

L. Prompt Tuning: experimental results

In the Experimental section 6, we explored prompt tuning as a way to improve the reconstructions when the prompt is already aligned with the image semantic (e.g., `a sharp photo of a face` for the FFHQ dataset). We will now show the capabilities of LATINO-PRO to significantly improve the reconstructions in cases where the given prompt is not aligned.

In Table 9, we can see how when we try to reconstruct images of dogs with the prompt `a sharp photo of`

Figure 13. Gaussian deblur FFHQ-1024 LATINO-PRO.

Figure 14. SR×16 FFHQ-1024 LATINO-PRO.

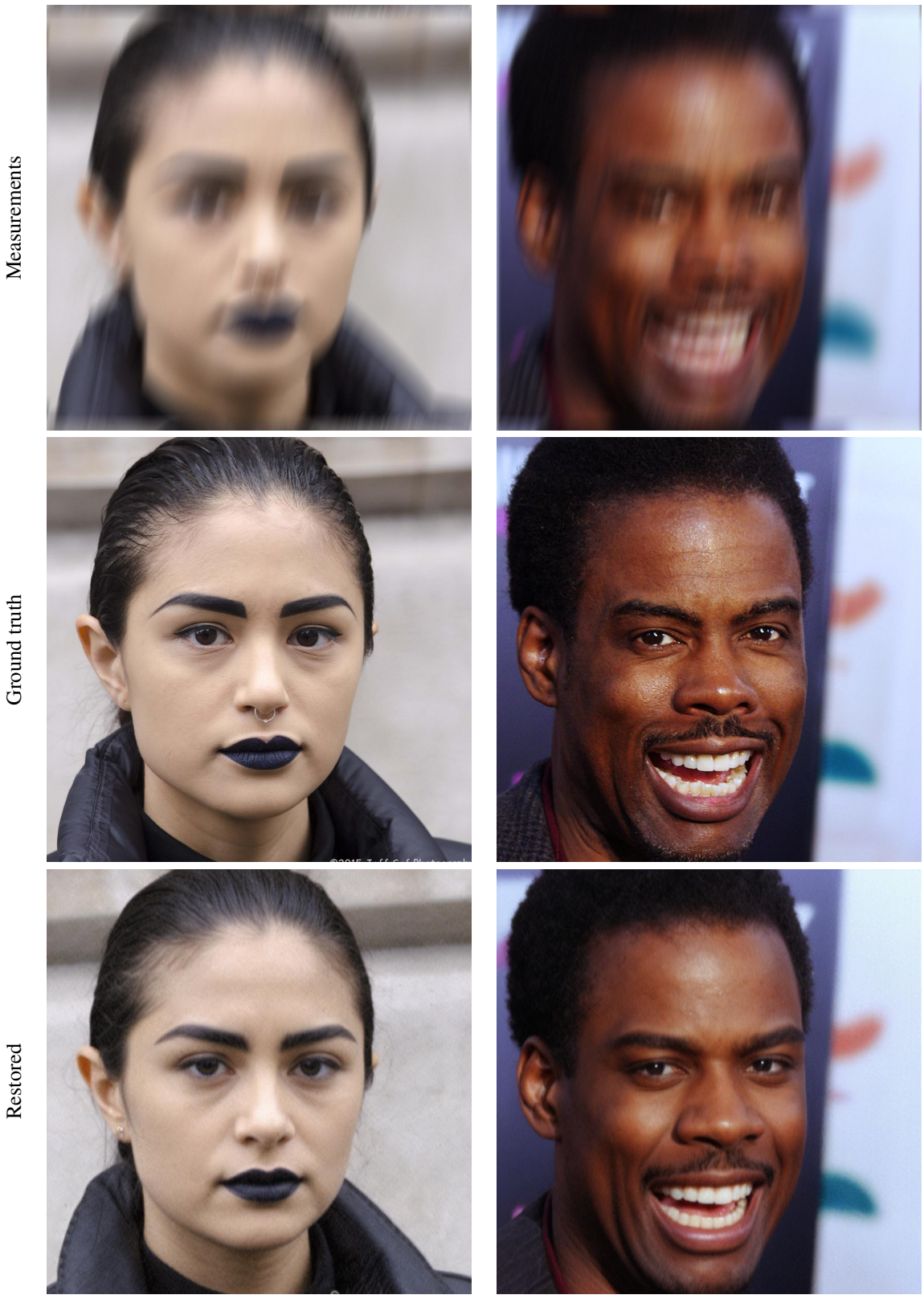


Figure 15. Motion Deblurring FFHQ-1024 LATINO-PRO.

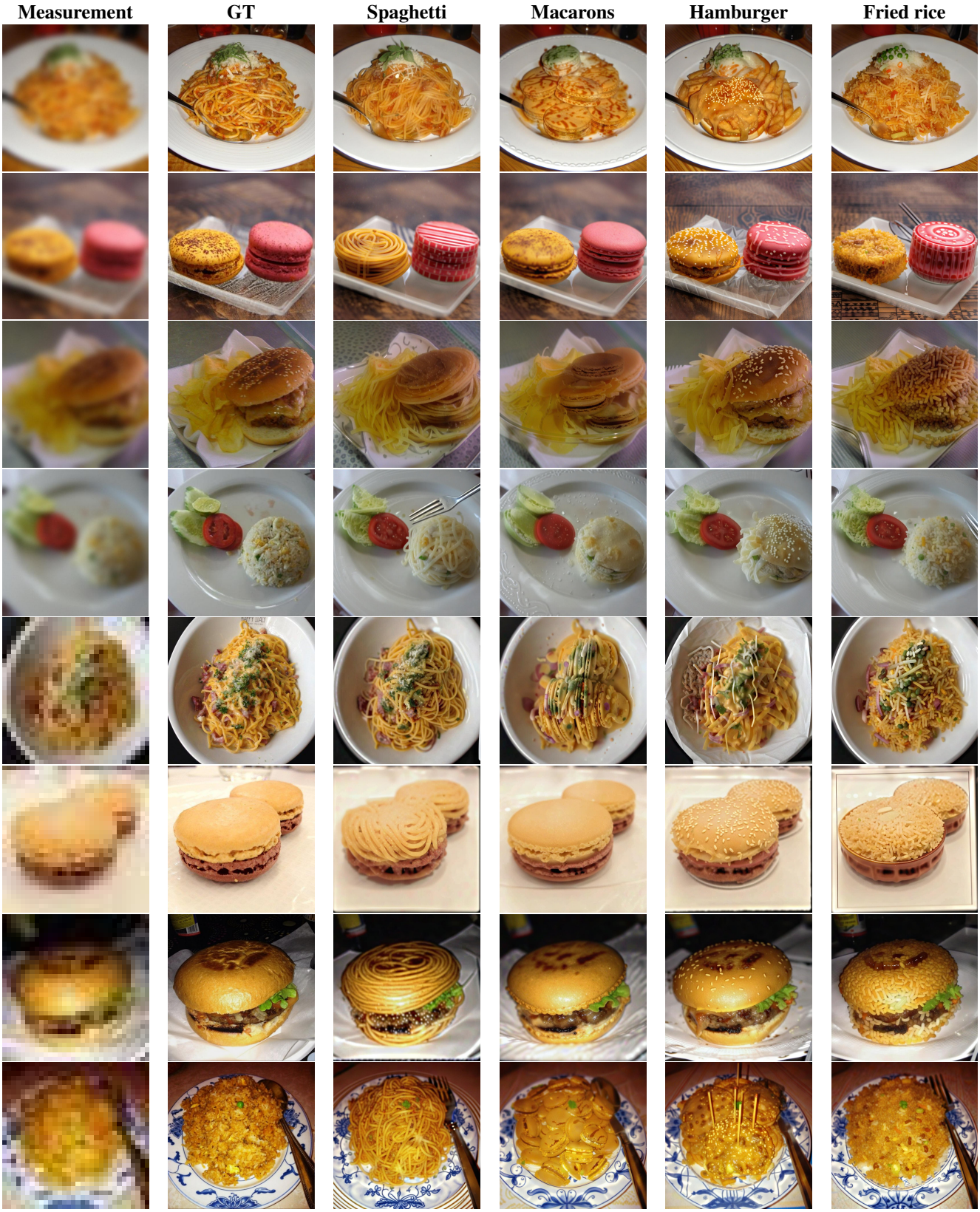


Figure 16. Visual results of the 8-steps LATINO on Food101 dataset for semantic shift task.

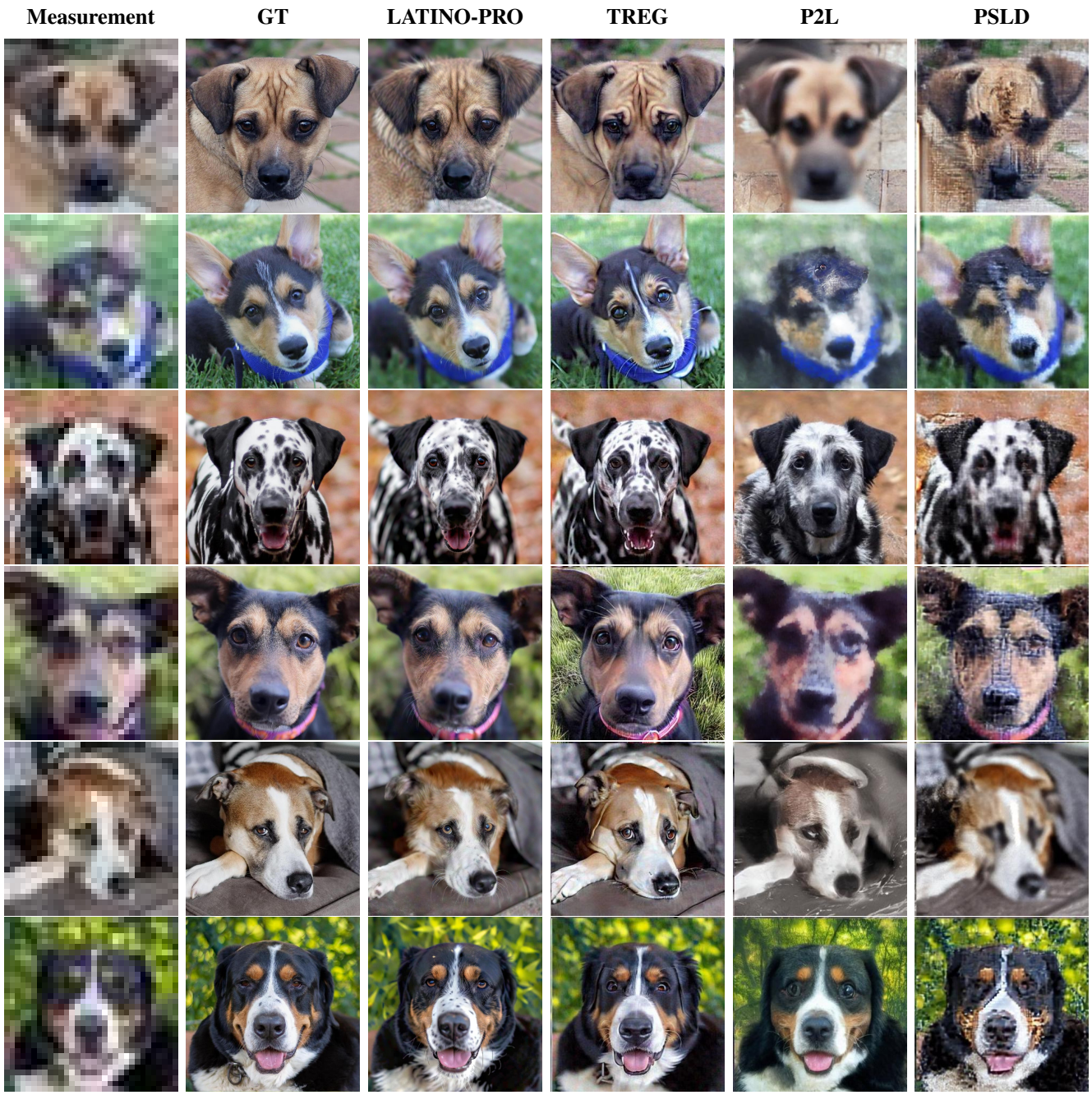


Figure 17. Comparisons between LATINO-PRO, TReg and P2L.

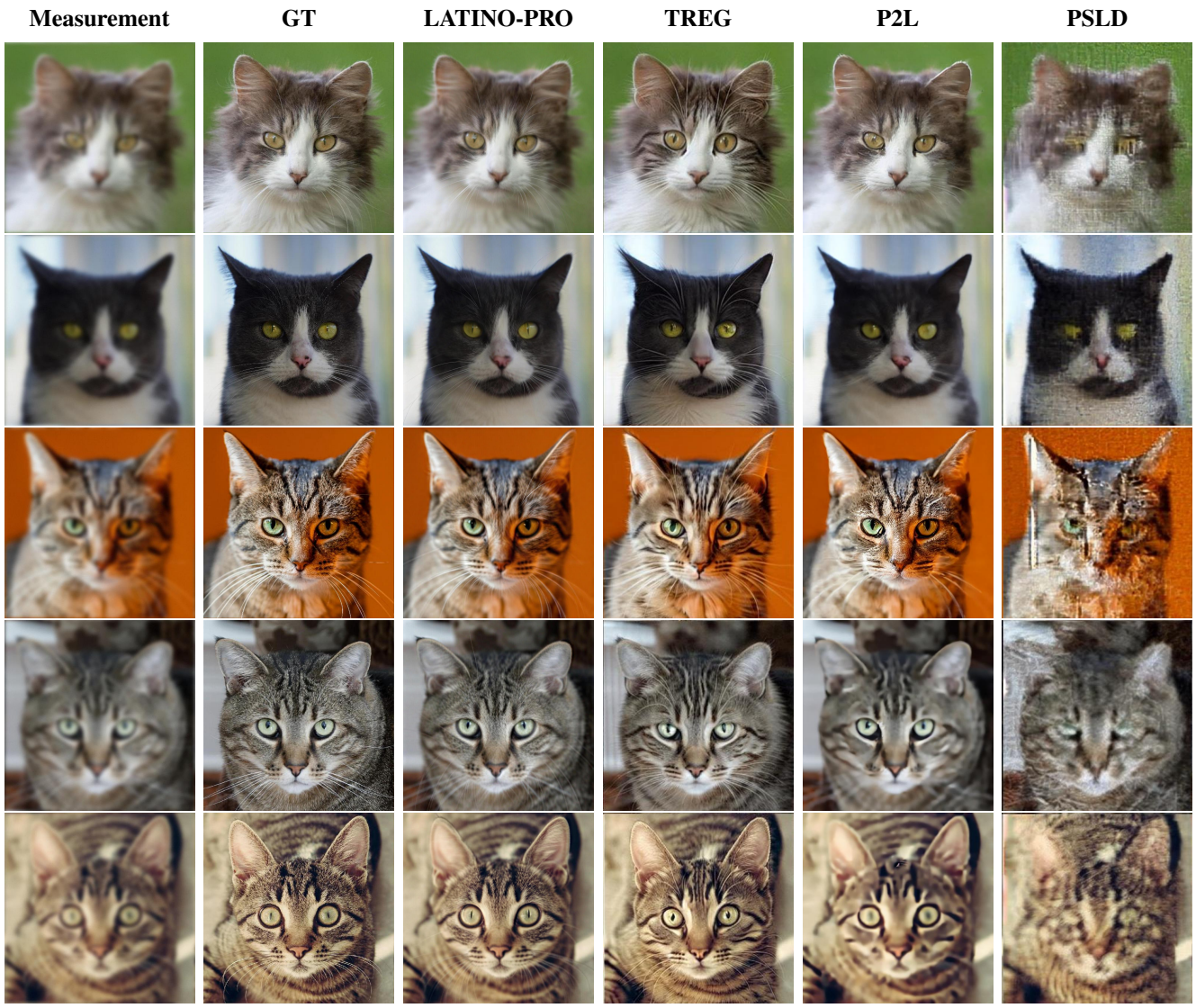


Figure 18. Comparisons between LATINO-PRO, TReg and P2L.

		Deblur (Gaussian)			SR×16		
Method	**NFE↓**	**FID↓**	**PSNR↑**	**LPIPS↓**	**FID↓**	**PSNR↑**	**LPIPS↓**
LATINO-PRO	68	**24.56**	**27.77**	**0.347**	**39.85**	**22.28**	**0.463**
LATINO	**8**	28.02	27.21	0.360	59.03	20.84	0.496
LATINO-PRO "a cat"	68	48.93	27.42	0.382	125.63	22.23	0.502
LATINO "a cat"	**8**	82.45	26.34	0.420	190.04	20.09	0.547
LDPS	1000	81.18	24.86	0.502	154.3	18.26	0.667
PSLD [48]	1000	41.04	26.12	0.455	92.35	22.20	0.585

Table 9. Results for Gaussian Deblurring with $\sigma = 5.0$, and $\times 16$ super-resolution, both with noise $\sigma_y = 0.01$ on the AFHQ-512 dogs val dataset. Base prompt when not specified: `a sharp photo of a dog`. **Bold**: best, underline: second best.

Figure 19. Effect of prompt optimization on the AFHQ-dogs val dataset. Initial prompt: `a sharp photo of a cat`.

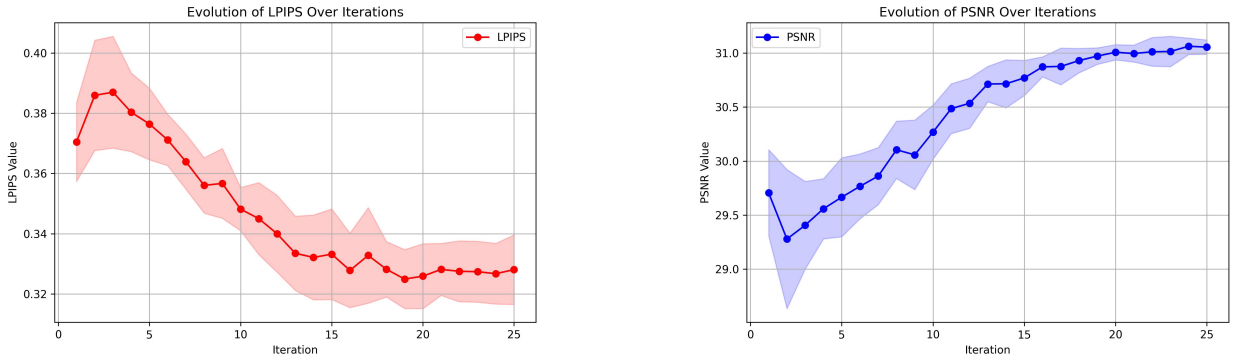


Figure 20. Metrics evolution during LATINO-PRO iterations for the example in Figure 19. Initial prompt: `a sharp photo of a cat`.

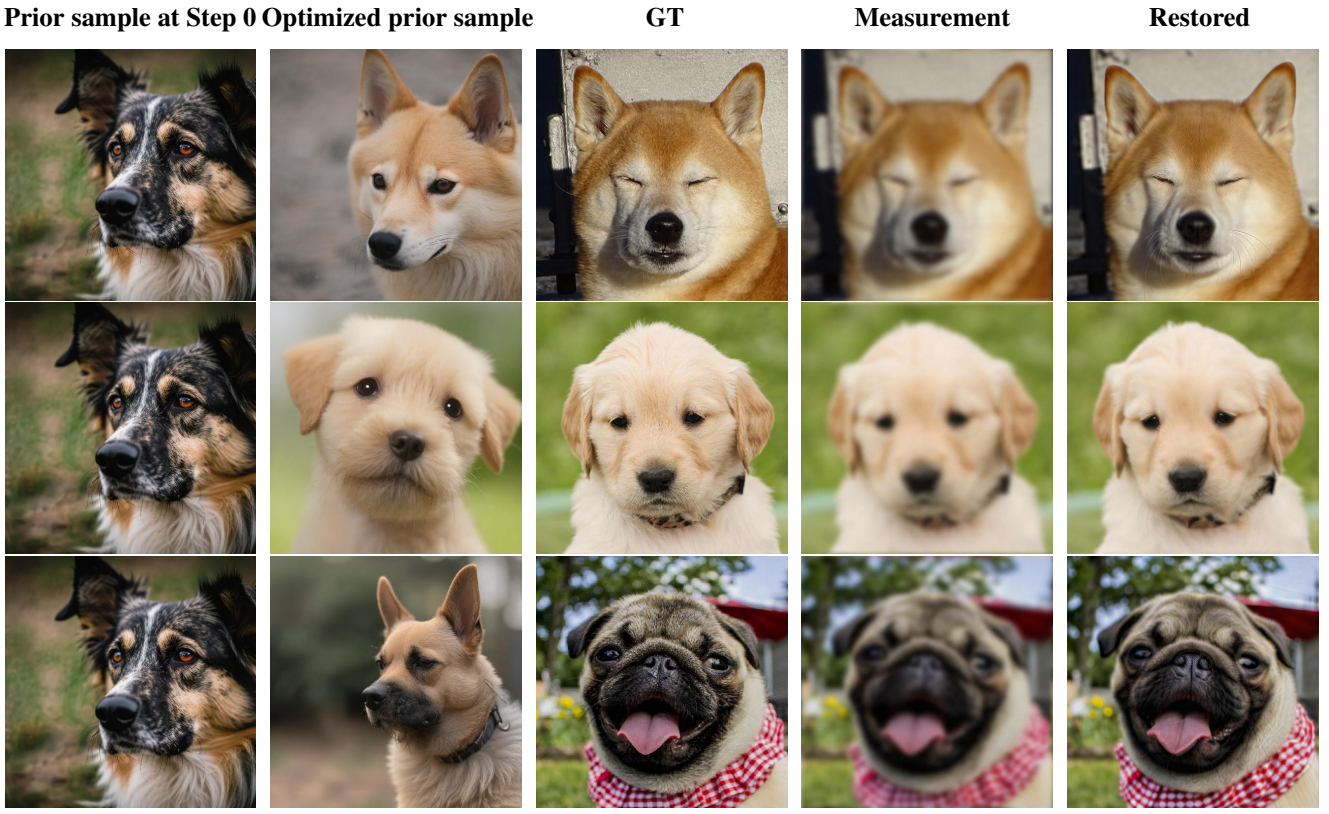


Figure 21. Effect of prompt optimization on the AFHQ-dogs val dataset. Initial prompt: `a sharp photo of a dog`.

`a cat`, the PSNR, SSIM and LPIPS metrics are much worse than with the prompt `a sharp photo of a dog`. However, LATINO-PRO brings the results closer to the optimal case. In particular we can appreciate the effectiveness in the Gaussian deblurring case, while the SR $\times 16$ seems harder to re-align. This is in fact in accordance with what we would expect, since the amount of information contained in the degraded observation $\boldsymbol{y}$ is small, and thus the prior has more influence on the reconstruction. In pracice an high resolution image of a cat could be compatible with a low resolution one of a dog, and thus the prompt is not able to learn the actual ground truth.

In Figure 19 we can see how the prior can learn from the measurement the main features. The semantic shift from cat $\rightarrow$ dog is appreciable, as well as how the main colors are learned by the prior. Figure 20 shows the trend of LPIPS and PSNR during the LATINO-PRO iterations, averaged over 20 different seeds. We decided to run the algorithm for 25 steps to show why it is preferable to early-stop it at around $15-20$ steps, the LPIPS metric indeed tends to usually rise after this interval, and the PSNR does not show any significant improvement.

A similar experiment has been conducted on less extreme cases, as shown in Figure 21, where the prompt given was `a sharp photo of a dog`. We can observe the ability of the prior to learn characteristics like the breed of the dog or whether it is a puppy or an adult.

M. Inpainting

While in Section 6, we focused on the deblurring and super-resolution tasks, we here apply our model to the inpainting case. We consider box impainting tasks where we cover the eyes of the animals in the AFHQ-512 dataset and both the eyes and the mouth for the faces in the FFHQ-512 dataset, as done in TReg [23]. We show the FID and PSNR metrics for the FFHQ and AFHQ 1k validation datasets in Table 10.

In Figure 22 and Figure 23 we show a comparison of the available methods, . For the TReg algorithm for which the code is unavailable, the restored image is not available in the original paper/website, while the entries in the table are those advertised in the paper [23]. The last rows were obtained using the LATINO-PRO model, giving as prompt `a photo of` + `a face` or `a dog` + the specific caption. The results can be interpreted in the following way: the reduced number of steps of LATINO makes the inpainting task more challenging, especially for more complex images such as faces. The prompt optimization done through SAPG helps to mitigate this phenomenon with better visual results, especially for the AFHQ case. The high performance of the proposed method on the averaged metrics show that similar problems are present in current SOTA methods.

	FFHQ		AFHQ	
Method	**FID↓**	**PSNR↑**	**FID↓**	**PSNR↑**
LATINO-PRO	67.79	**20.55**	**19.91**	**21.05**
LATINO	87.78	<u>20.01</u>	<u>27.01</u>	<u>19.92</u>
P2L [8]	85.32	16.84	138.4	16.07
TReg [23]	<u>66.93</u>	19.95	51.97	17.39
PSLD [48]	**60.97**	19.76	104.7	16.93

Table 10. Box Inpainting results on FFHQ (left) and AFHQ (right). **Bold**: best, <u>underline</u>: second best.

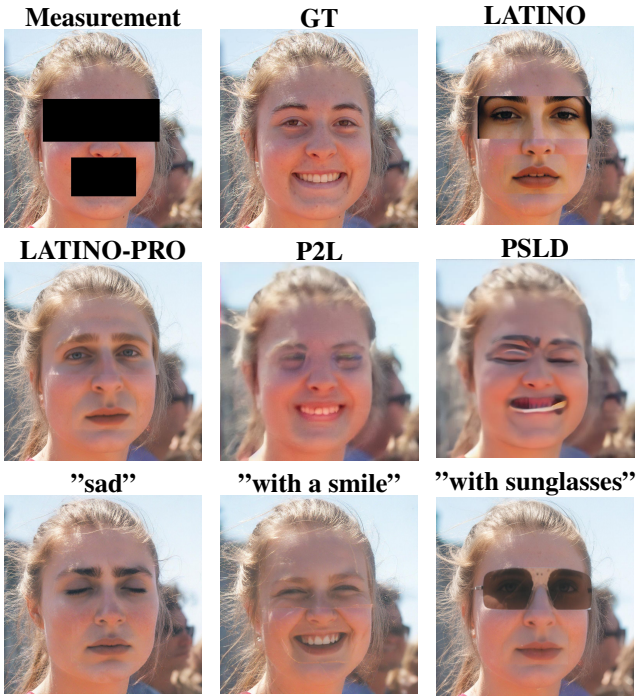


Figure 22. Box inpainting results on FFHQ-512. Middle row: LATINO-PRO with the prompt `a sharp photo of a face`, P2L and PSLD. Bottom row: LATINO-PRO with different prompts.

N. Harder cases

As shown in the the preview Figure 1, LATINO-PRO is able to solve also harder restoration tasks. Here we provide in Figure 24 more visual results on the FFHQ1024 in this direction. In particular we observe an high level of consistency for aggressive tasks like Gaussian Deblurring with $\sigma = 20.0$ pixels and $\times 32$ super-resolution. We identify as a limitation the inability of our algorithm to keep the same level of consistency when the noise level is increased to $\sigma_n = 0.1$. The prior tendency to dominate is not contrasted enough by the proximity operator, and the results tend to deviate more from the actual ground truth.

Figure 23. Box inpainting (AFHQ-512). Second row: LATINO-PRO with the prompt `a sharp photo of a dog`, P2L and PSLD. Last row: LATINO-PRO with various prompt initializations.

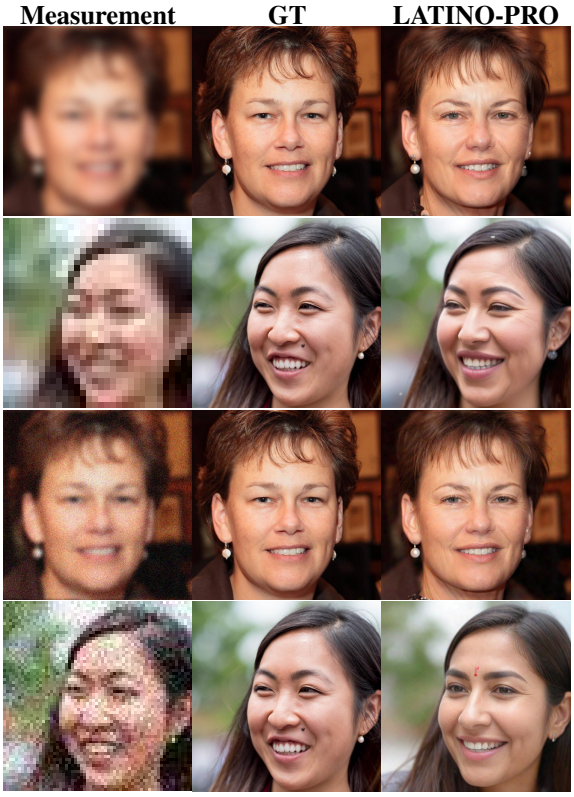


Figure 24. Qualitative comparison of LATINO-PRO on hard image restoration on FFHQ-1024. Tasks: Gaussian deblur $\sigma = 20.0$ and $\times 32$ super-resolution with noise $\sigma_n = 0.01$. Gaussian deblur $\sigma = 10.0$ and $\times 16$ super-resolution with noise $\sigma_n = 0.1$.