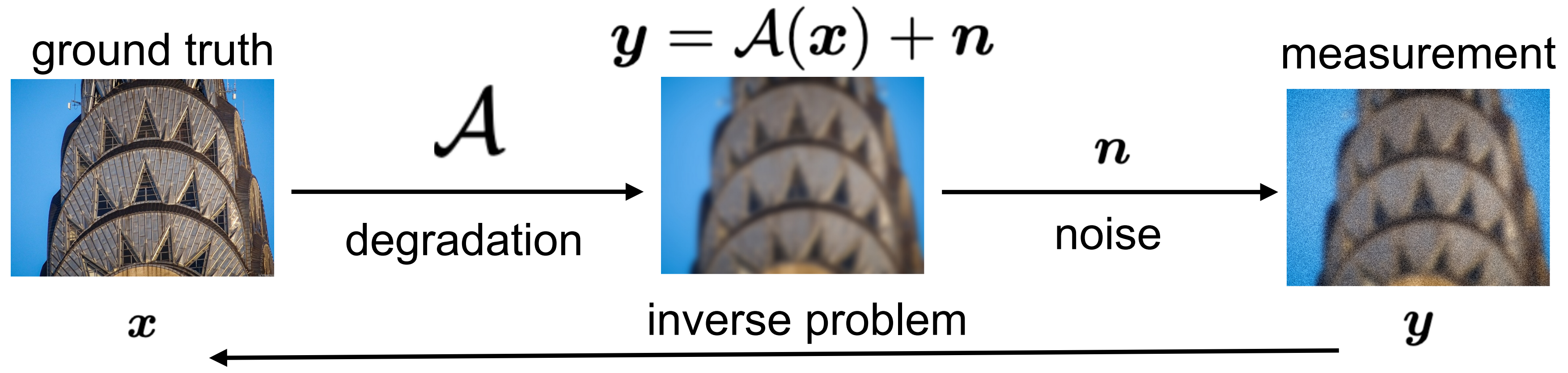
Adapt and Diffuse: Sample-adaptive Reconstruction via Latent Diffusion Models

Zalan Fabian*, Berk Tinaz*, Mahdi Soltanolkotabi

Paper

Code

Motivation


ground truth

$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{n}$

measurement

$\mathcal{A}$

degradation

$\boldsymbol{n}$

noise

$\boldsymbol{x}$

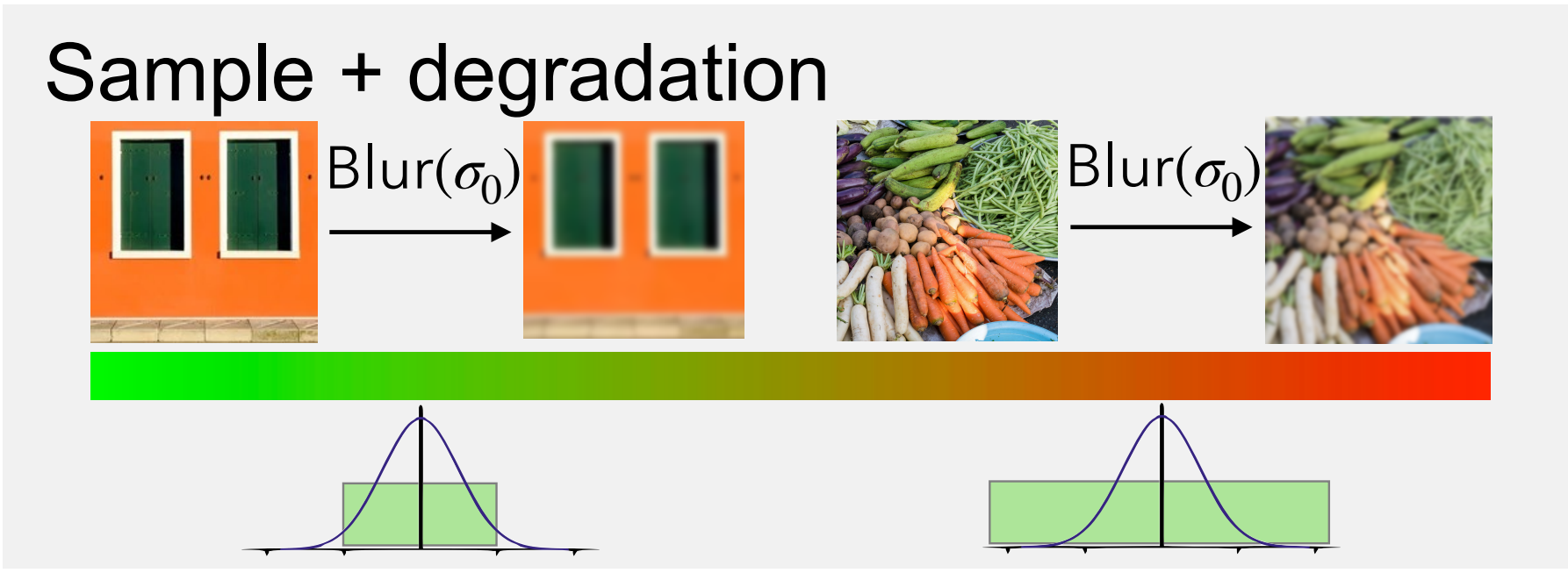
inverse problem

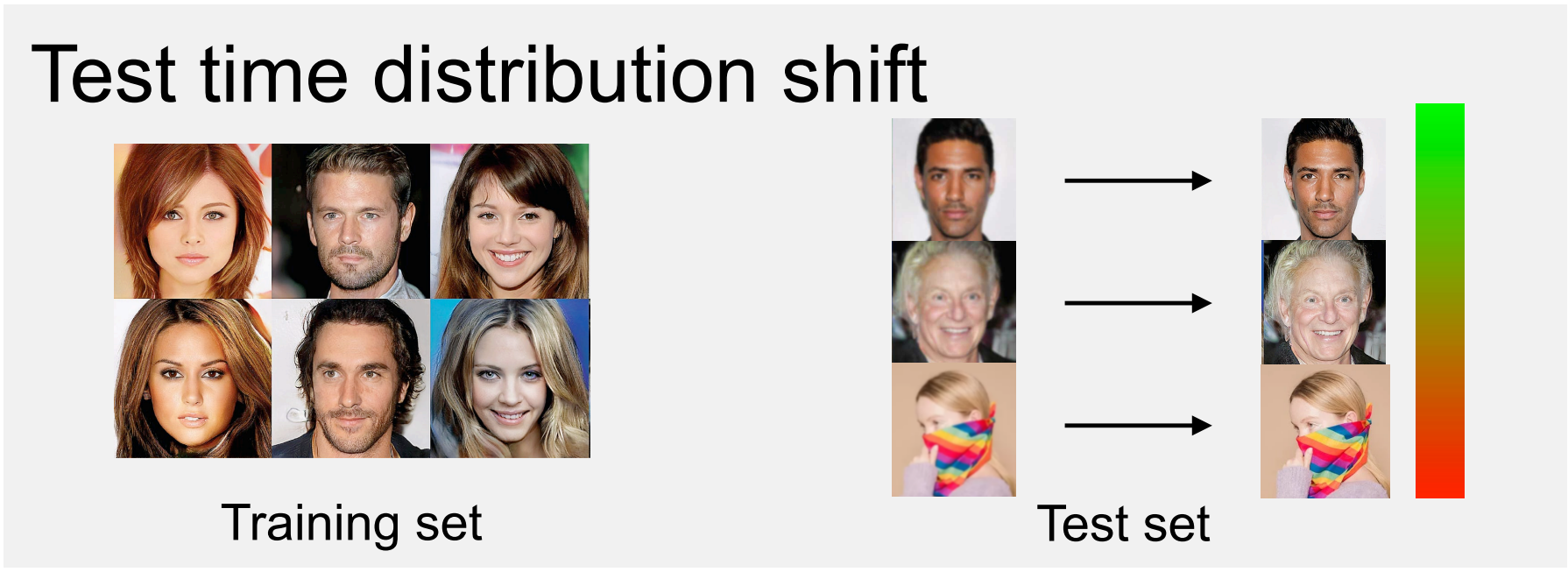
$\boldsymbol{y}$

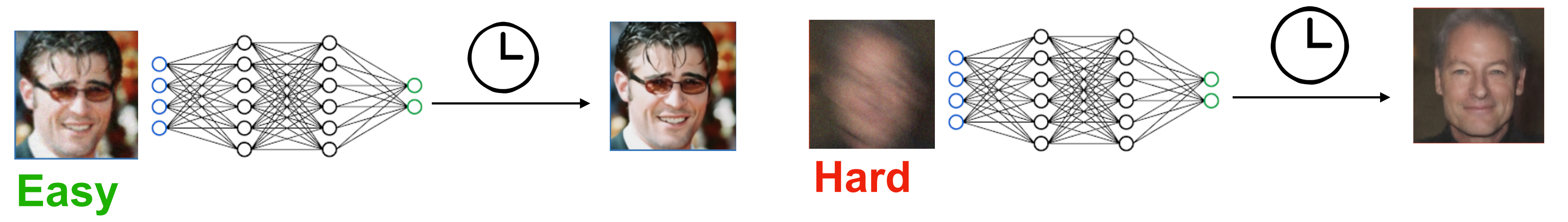
Sample-by-sample variation in reconstruction difficulty

Degradation level

Noise level


Sample + degradation


Test time distribution shift



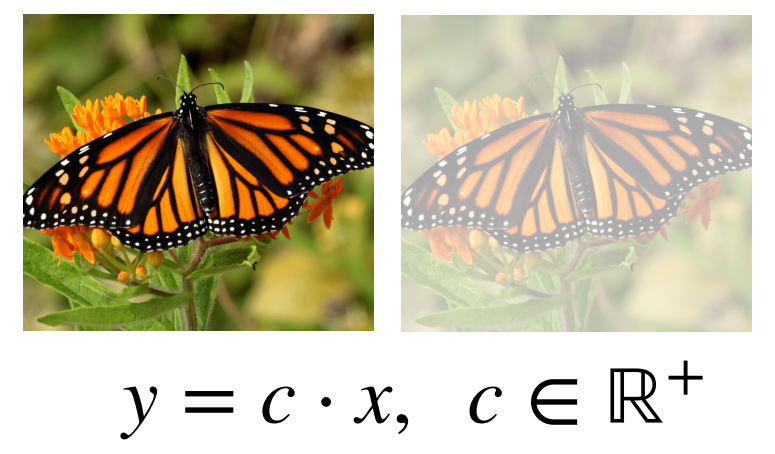
Expending the same amount of resources to reconstruct any sample (easy or hard) is potentially wasteful.

Idea: adapt the compute allocation based on the difficulty of the problem on a **sample-by-sample** basis in test time!

Quantifying Difficulty

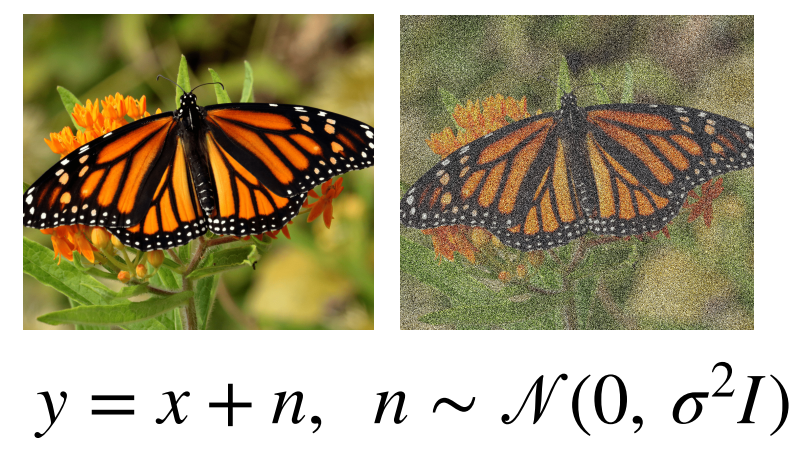
Image space

Autoencoder latents


$y = c \cdot x, \quad c \in \mathbb{R}^+$

- arbitrarily high perturbation
- reconstruction is trivial

- compressed representation of relevant information in image


$y = x + n, \quad n \sim \mathcal{N}(0, \sigma^2 I)$

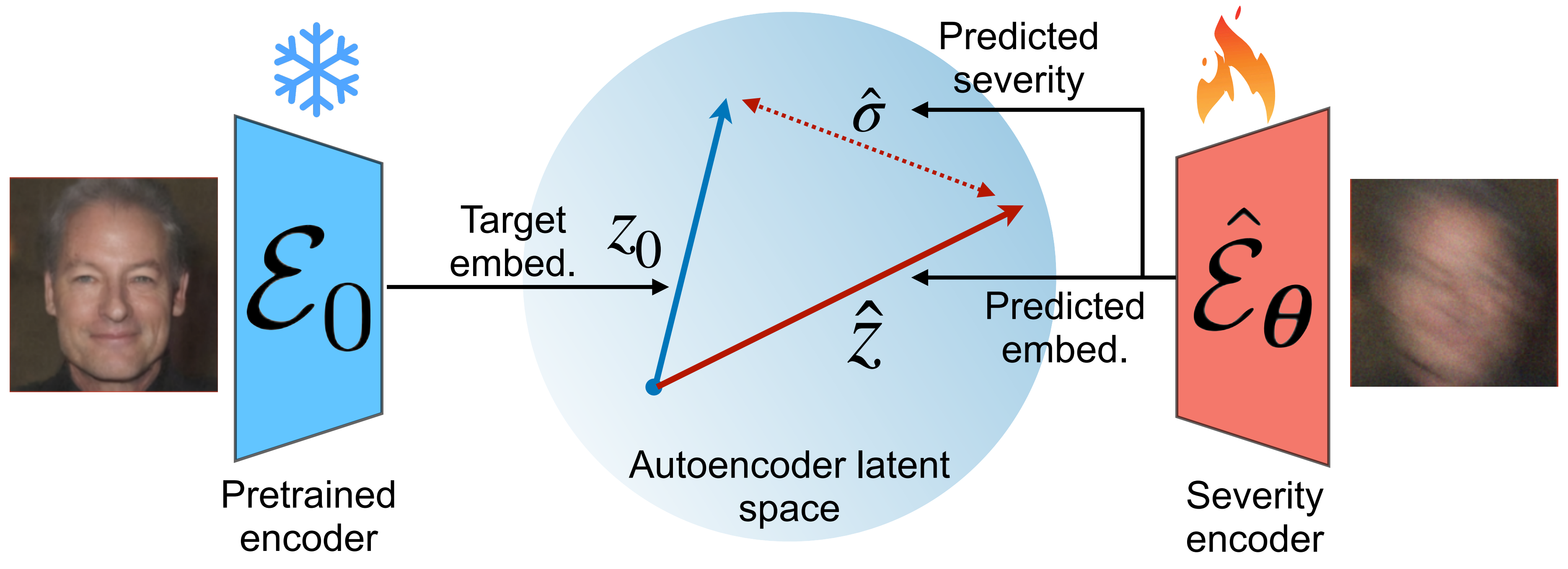
- small perturbation
- reconstruction is challenging

- natural space to quantify loss of information due to corruption

Idea: quantify severity of degradation in the latent space of an autoencoder

Severity Encoding

Estimate degradation severity given corrupted image


Pretrained encoder

Target embed.

z_0

Predicted severity

$\hat{\sigma}$

Predicted embed.

$\hat{z}$

Autoencoder latent space

Severity encoder

Objectives:

1. Predict latent of clean image
2. Estimate prediction error

Intuition: the more degraded the input, the larger the prediction error will be in latent space

We leverage **latent prediction error as a proxy** for degradation severity!

Training the severity encoder

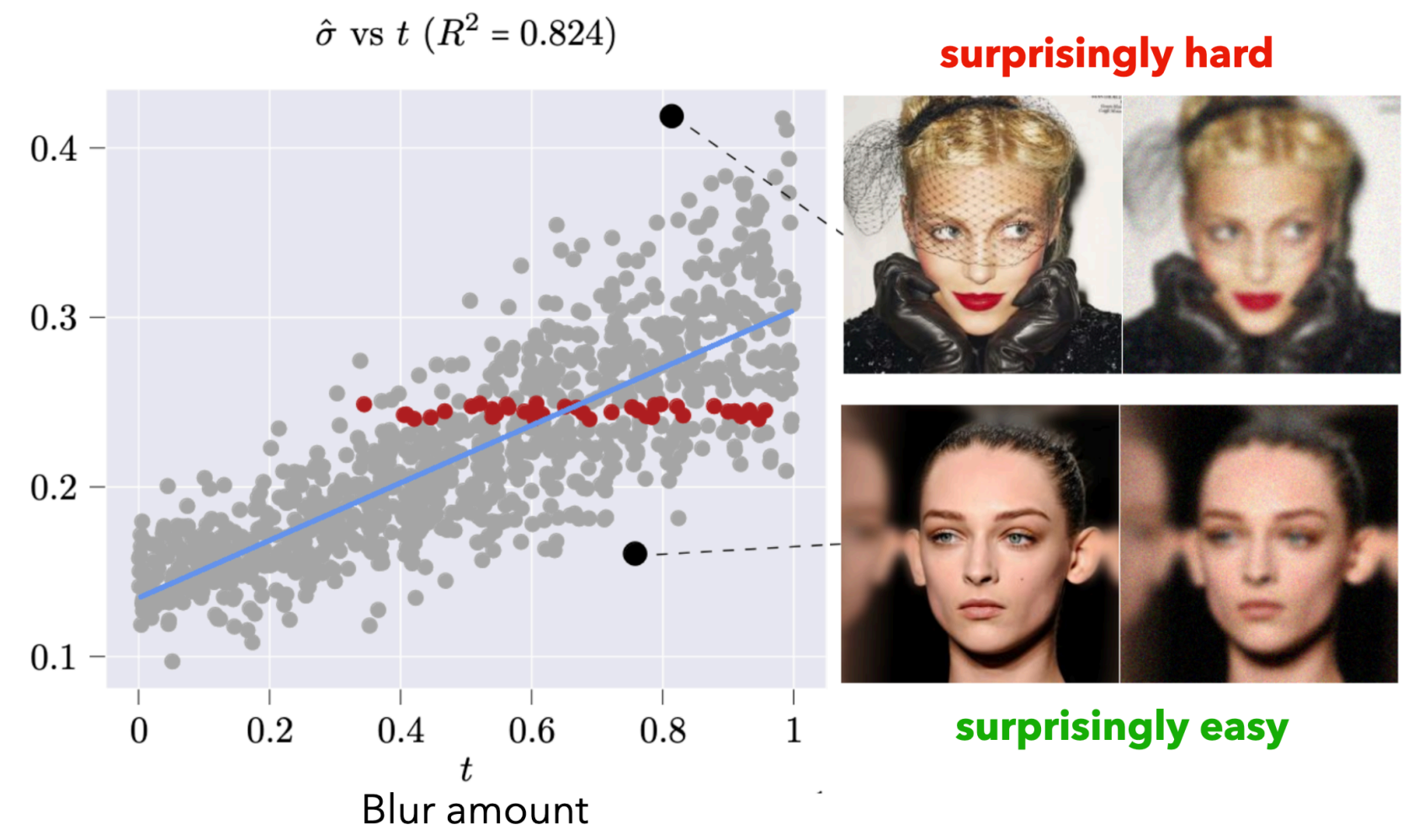
$$\min_{\theta} \mathbb{E}_{\boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0), \boldsymbol{y} \sim \mathcal{N}(\mathcal{A}(\boldsymbol{x}_0), \sigma_y^2 \mathbf{I})} \left[\underbrace{\left\| \boldsymbol{z}_0 - \hat{z}(y;\theta) \right\|^2}_{\text{reconstruction}} + \lambda_\sigma \underbrace{\left\| \bar{\sigma}^2(\boldsymbol{y}, \boldsymbol{z}_0) - \hat{\sigma}(y;\theta) \right\|^2}_{\text{error prediction}} \right]$$

Assumption: prediction error is zero-mean i.i.d. Gaussian: $\boldsymbol{e}(\boldsymbol{y}) = \hat{\boldsymbol{z}} - \boldsymbol{z}_0 \sim \mathcal{N}(\mathbf{0}, \sigma_*^2(\boldsymbol{y})\mathbf{I})$

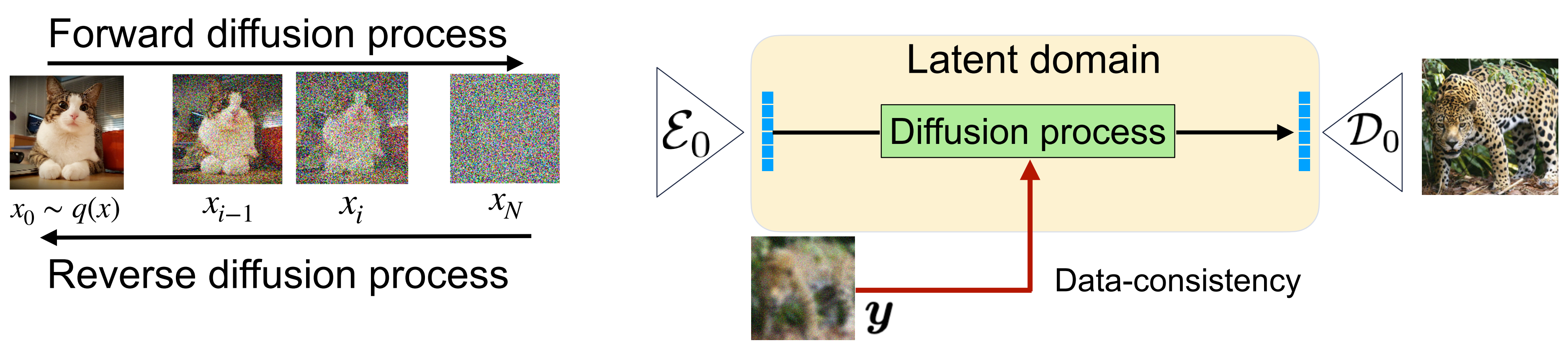
$$\bar{\sigma}^2(\boldsymbol{y}, \boldsymbol{z}_0) = \tfrac{1}{d-1} \sum_{i=1}^{d} (\boldsymbol{e}^{(i)} - \tfrac{1}{d} \sum_{j=1}^{d} \boldsymbol{e}^{(j)})^2$$

Severity encoding experiments

- Predicted severity strongly correlates with blur level
- Outliers indicate the presence of additional contributing factors



Latent Diffusion Solvers


Forward diffusion process

Reverse diffusion process

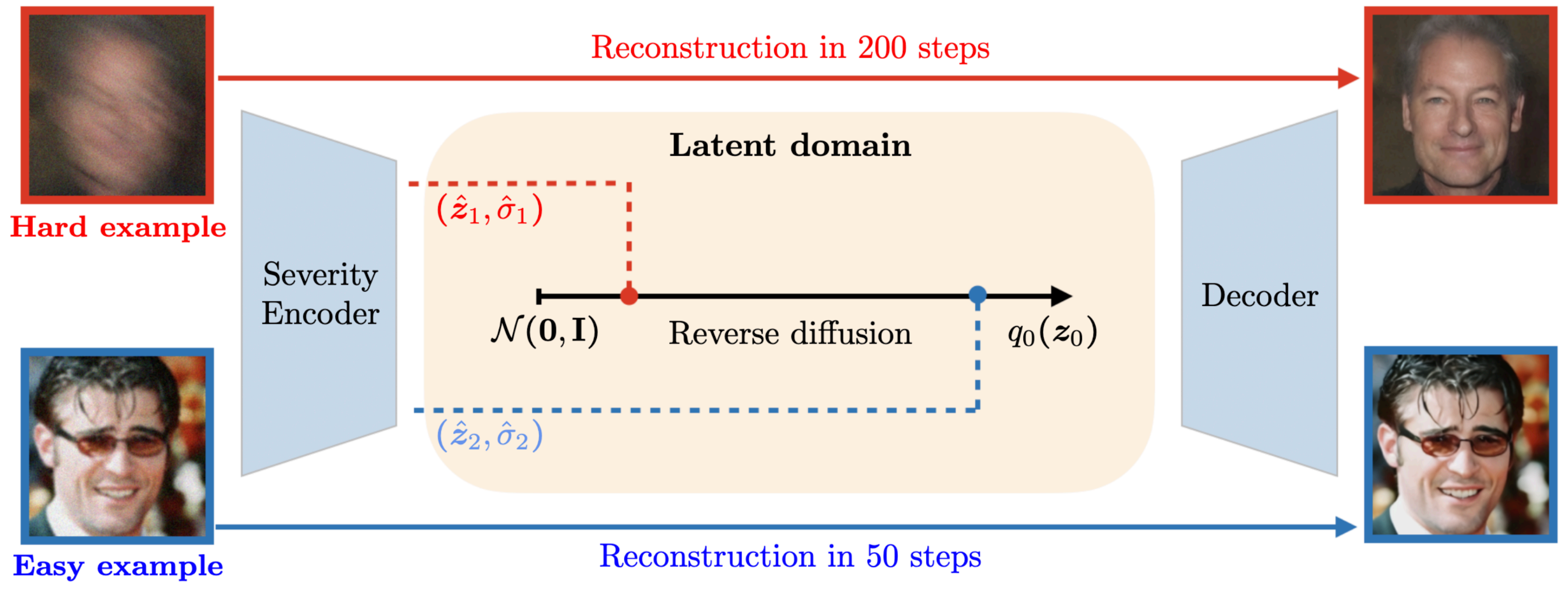
Latent domain

Diffusion process

Data-consistency

FlashDiffusion

Sample-adaptive reconstruction via severity encoding


Reconstruction in 200 steps

Hard example

Severity Encoder

Latent domain

$(\hat{z}_1, \hat{\sigma}_1)$

$\mathcal{N}(\mathbf{0}, \mathbf{I})$

Reverse diffusion

$q_0(z_0)$

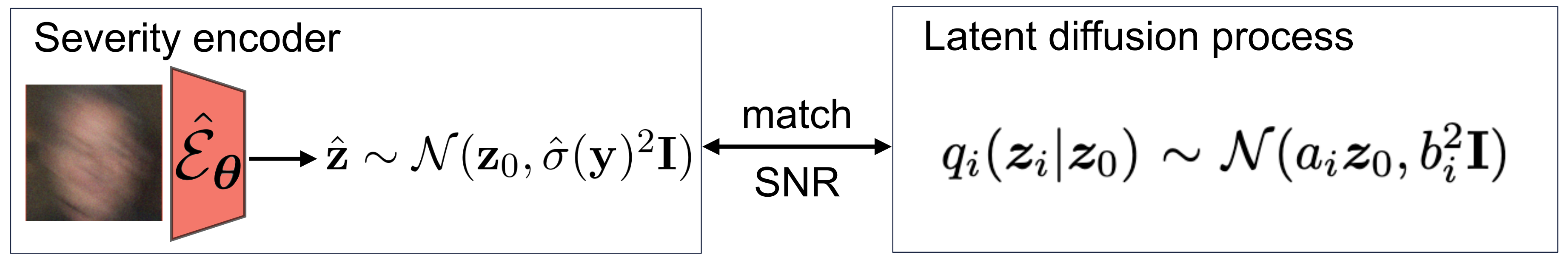
$(\hat{z}_2, \hat{\sigma}_2)$

Decoder

Easy example

Reconstruction in 50 steps

Finding optimal reverse diffusion starting time


Severity encoder

$\hat{\mathcal{E}}_\theta \rightarrow \hat{\mathbf{z}} \sim \mathcal{N}(\mathbf{z}_0, \hat{\sigma}(\mathbf{y})^2\mathbf{I})$

$\xleftrightarrow[\text{SNR}]{\text{match}}$

Latent diffusion process

$q_i(\boldsymbol{z}_i | \boldsymbol{z}_0) \sim \mathcal{N}(a_i \boldsymbol{z}_0, b_i^2 \mathbf{I})$

Adaptive starting time: $i_{start}(\mathbf{y}) = \arg \min_{i \in [1,2,\ldots,N]} \left| \frac{1}{\hat{\sigma}(\mathbf{y})^2} - \frac{a_i^2}{b_i^2} \right|$

FlashDiffusion **acts as a wrapper** around *any* baseline latent diffusion solver, imbuing it with sample-adaptivity.

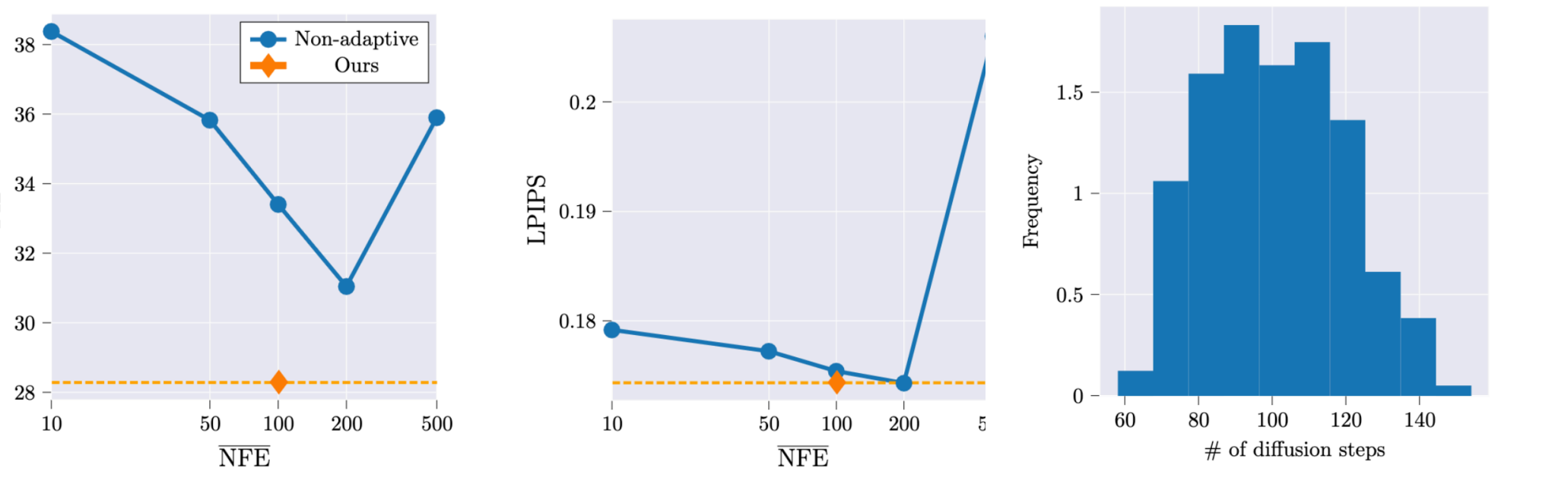
Experiments

Comparison with baseline solvers

FFHQ	**Gaussian Deblurring (Varying)**					**Gaussian Deblurring (Fixed)**					**Nonlinear Deblurring**					**Random Inpainting**				
Method	PSNR(↑)	SSIM(↑)	LPIPS(↓)	FID(↓)	$\overline{\text{NFE}}$	PSNR(↑)	SSIM(↑)	LPIPS(↓)	FID(↓)	$\overline{\text{NFE}}$	PSNR(↑)	SSIM(↑)	LPIPS(↓)	FID(↓)	$\overline{\text{NFE}}$	PSNR(↑)	SSIM(↑)	LPIPS(↓)	FID(↓)	$\overline{\text{NFE}}$
Latent-DPS	23.69	0.6418	0.3579	87.26	1000	22.88	0.6136	0.3690	89.38	1000	22.07	0.5974	0.3814	90.89	1000	23.96	0.6566	0.3666	93.65	1000
`Flash`(Latent-DPS)	29.17	0.8182	0.2240	55.57	100.3	27.44	0.7691	0.2823	80.44	127.7	27.17	0.7659	0.2695	69.78	136.1	29.21	0.8414	**0.1945**	**53.95**	104.7
PSLD (Rout et al., 2024)	25.06	0.6769	0.3194	79.79	1000	23.72	0.6183	0.3324	88.45	1000	-	-	-	-	-	24.94	0.6617	0.3672	85.64	1000
`Flash`(PSLD)	29.26	0.8205	**0.2203**	**53.27**	100.3	27.44	0.7657	**0.2797**	**65.35**	127.7	-	-	-	-	-	27.06	0.8018	0.2185	55.12	104.7
GML-DPS (Rout et al., 2024)	24.98	0.6884	0.3471	100.27	1000	24.01	0.6574	0.3621	102.80	1000	23.00	0.6426	0.3812	108.79	1000	25.20	0.7044	0.3527	103.3	1000
`Flash`(GML-DPS)	29.21	0.8276	0.2274	69.16	100.3	27.47	0.7699	0.2816	69.81	127.7	27.11	0.7640	0.2756	81.93	136.1	28.95	0.8437	0.1957	59.39	104.7
ReSample (Song et al., 2023)	28.77	0.8219	0.2587	81.96	500	27.62	0.7789	0.3148	102.47	500	26.61	0.7318	0.2838	68.57	500	27.51	0.7892	0.2460	63.39	500
`Flash`(ReSample)	29.07	0.8330	0.2383	74.76	49.9	27.77	0.7845	0.3092	100.84	63.6	26.88	0.7660	**0.2667**	**64.57**	67.8	28.13	0.8260	0.2045	56.67	52.1
AE	29.46	0.8358	0.2671	89.29	-	27.69	0.7820	0.3396	110.56	-	27.17	0.7786	0.3364	111.24	-	29.23	0.8432	0.2515	85.87	-
SwinIR (Liang et al., 2021)	**30.69**	**0.8583**	0.2409	87.61	-	**28.41**	**0.8021**	0.3091	108.49	-	**27.60**	**0.7928**	0.3093	99.56	-	**30.08**	**0.8654**	0.2223	78.32	-
DPS (Chung et al., 2022a)	28.34	0.7791	0.2465	81.70	1000	25.49	0.6829	0.3035	97.89	1000	22.77	0.6191	0.3601	109.58	1000	28.30	0.8049	0.2451	82.78	1000

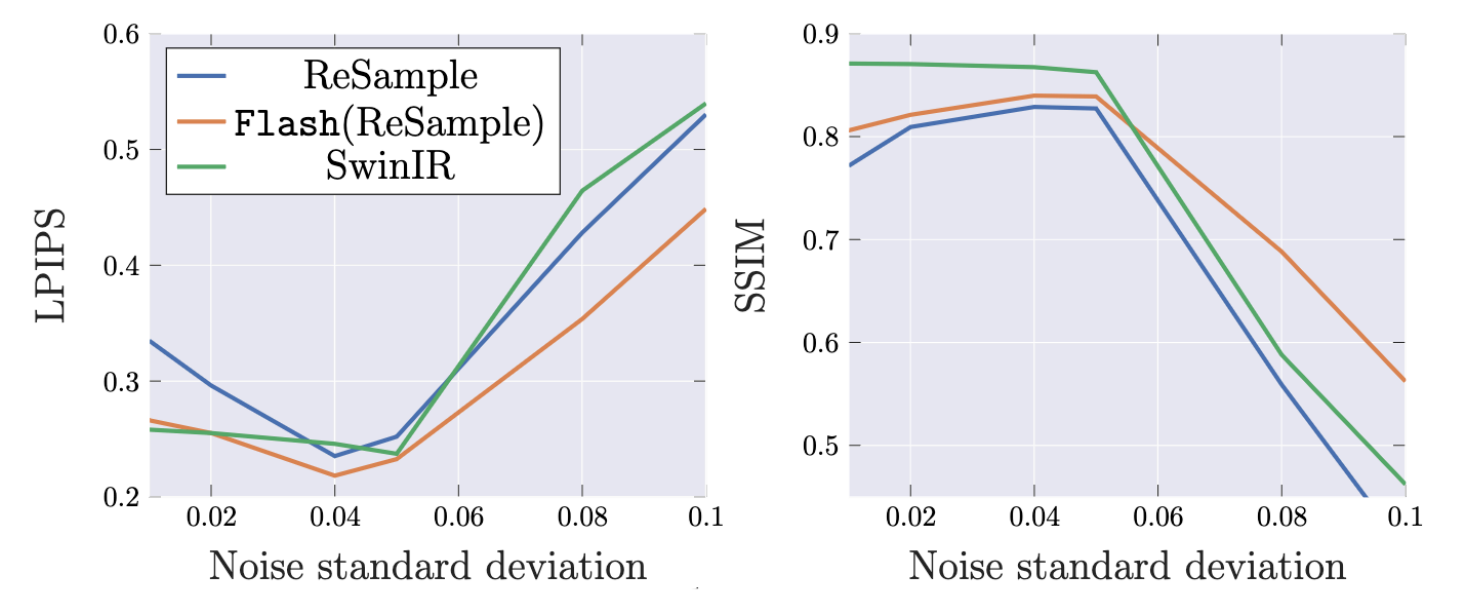
FlashDiffusion accelerates the baseline solver by a factor of up to 10x on average and greatly improves reconstruction quality.

Adaptivity



FlashDiffusion achieves best perceptual quality compared to any non-adaptive starting time.

Robustness



FlashDiffusion performance degrades more gracefully than baseline.