Bayesian imaging with deep generative priors

Marcelo Pereyra

Heriot-Watt University & Maxwell Institute for Mathematical Sciences

Joint work with Alessio Spagnoletti, Jean Prost, Andres Almansa and Nicolas Papadakis

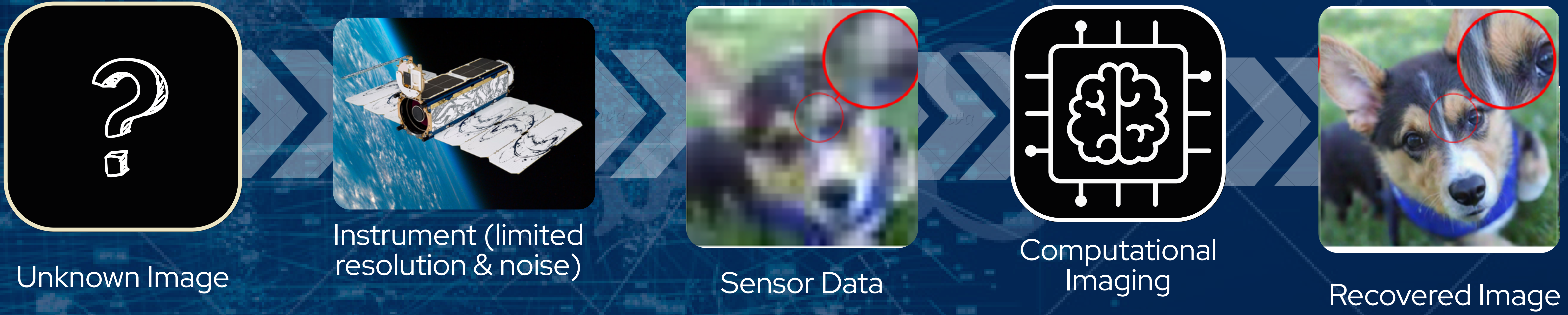



24 June 2025, Glasgow
30th Biennial Numerical Analysis Conference

Background

<u>Problem:</u>

- Image data often not useful in raw form (limited resolution & noisy, or accurate but too expensive).
- Evidence-based decision-making needs accurate solutions and reliable uncertainty quantification.



<u>Vision:</u>

Use mathematics to upgrade imaging instruments into smart decision-making support systems.

<u>Approach:</u>

A probabilistic computational imaging framework integrating physical and generative AI models, Bayesian statistical decision-theory and fast (exa)scalable stochastic algorithms.

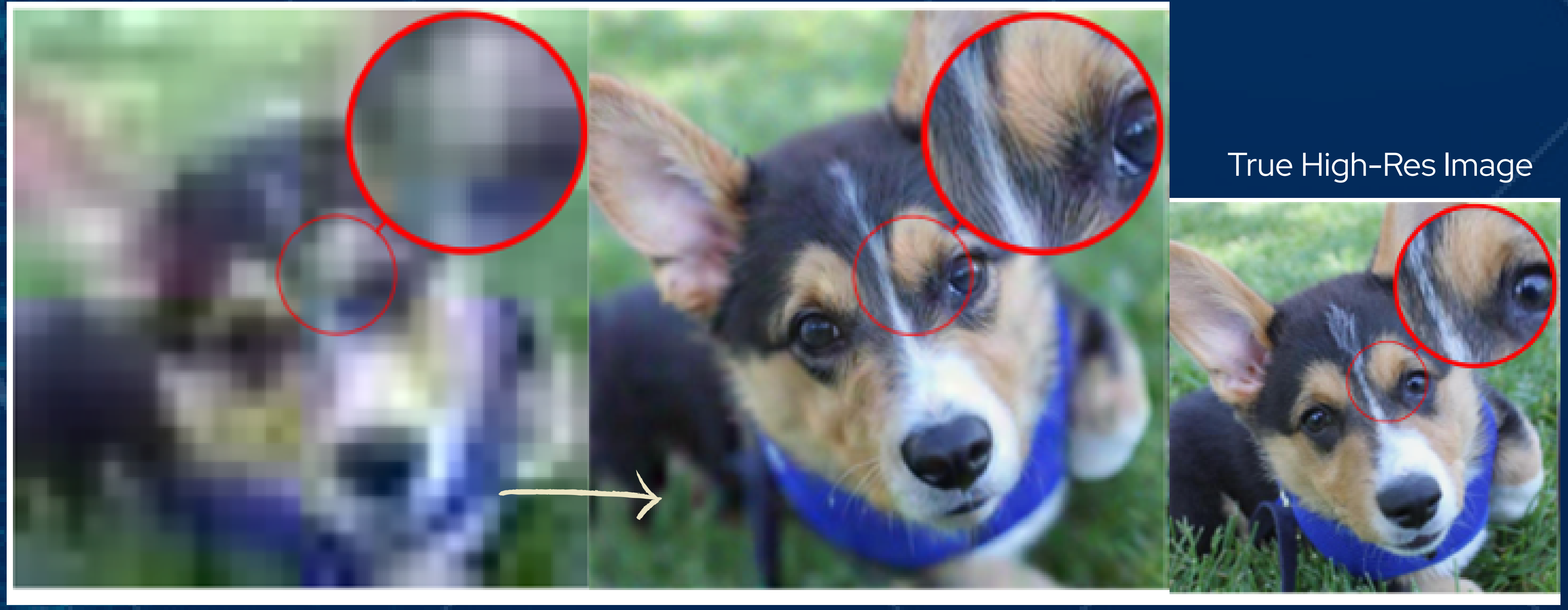
Today's talk: Generative AI-based Bayesian Imaging

Example of image generated by a Vision Language Model (VLM). These are probabilistic generative models represented by massive deep neural nets.

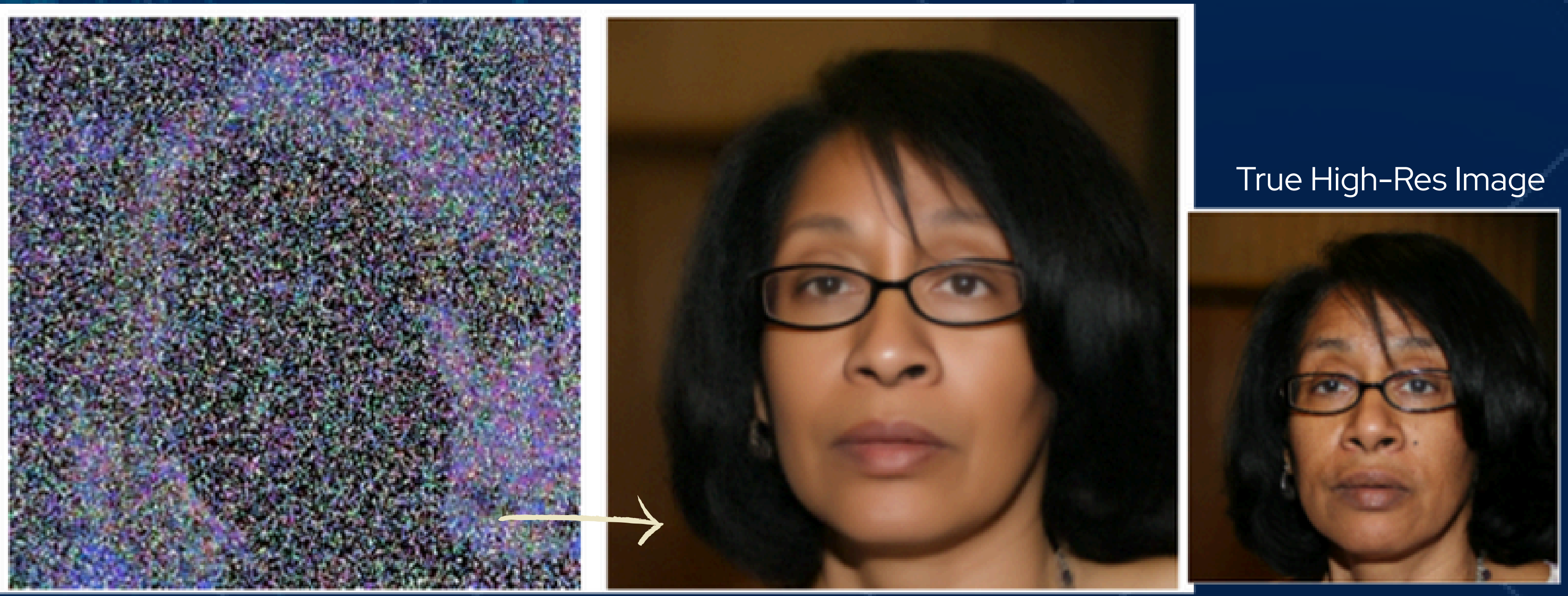
Prompt: Beautiful white Mediterranean outdoor courtyard, decorated with string lights and candles...Credit: Midjourney.com

Key breakthrough: new mathematical underpinning allows embedding physical models into VLMs and *prompting* with physical measurements, while self-adjusting text prompts.

Exterme Zoom



Extreme noise



Problem Statement

We are interested in an unknown image $x^\star \in \mathbb{R}^d$

We measure $y = Ax^\star + w$

Recovering $x^\star$ from y is not well posed.

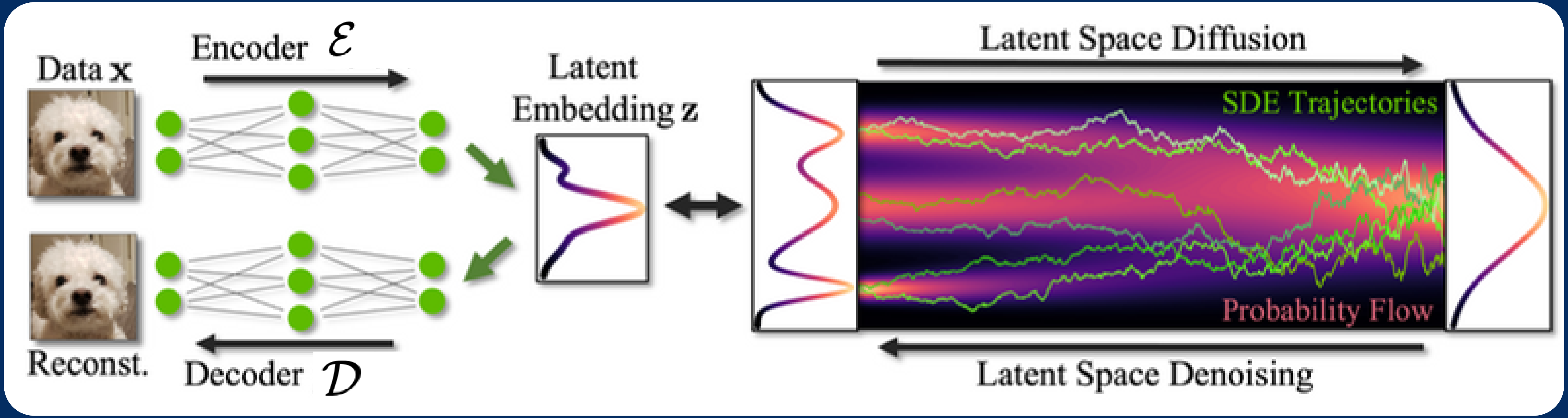
Bayesian Statistical Framework

Model $x^\star$ as a realisation of $\mathbb{x}$ and y as a realisation of $(\mathbb{y}|\mathbb{x} = x^\star)$

We draw inferences about $\mathbb{x}$ having observed $\mathbb{y} = y$ by using Bayes' theorem to combine observed and prior information

$$p(x|y) = \frac{p(y|x)p(x)}{\int_{\mathbb{R}^d} p(y|\tilde{x})p(\tilde{x})\mathrm{d}\tilde{x}}$$

Latent Diffusion Models



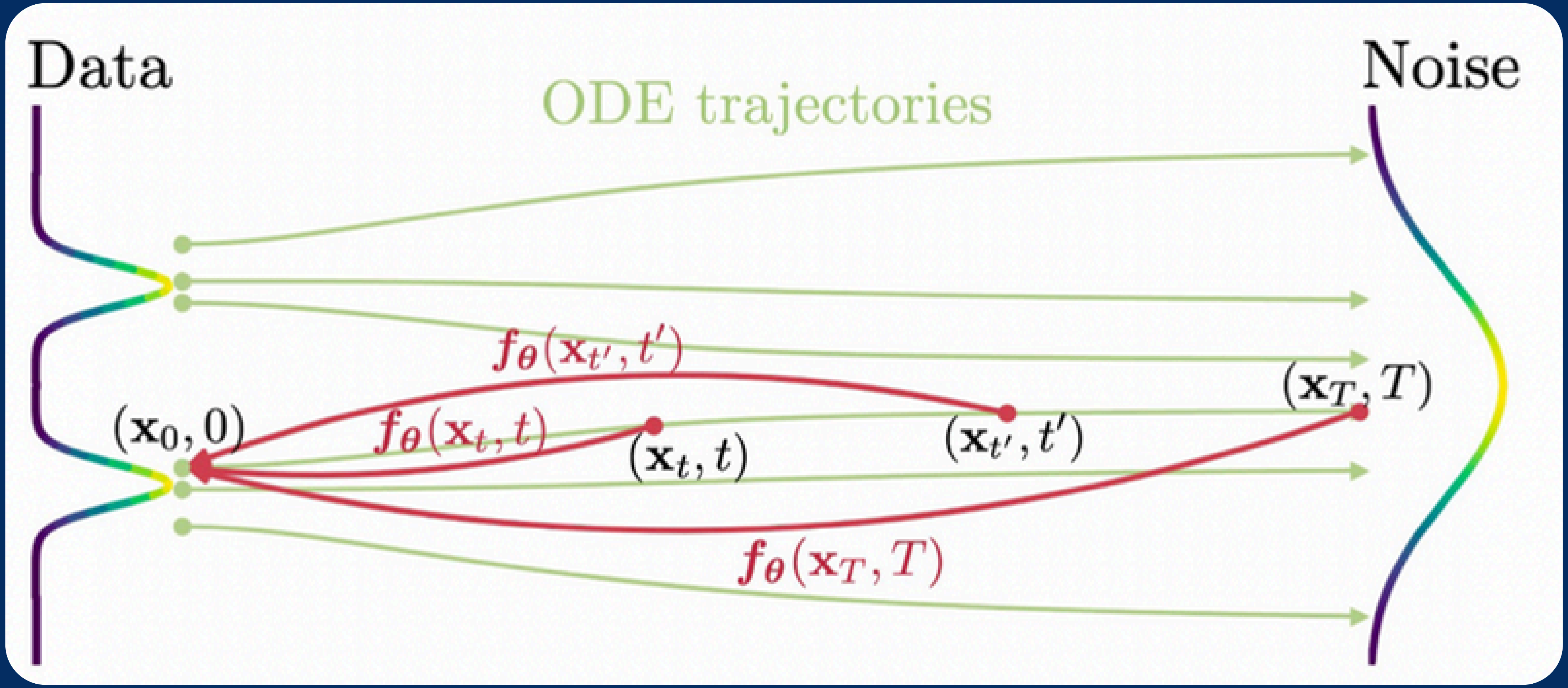
Latent Diffusion scheme (Source NeurIPS 2023 Tutorial)

$$d\mathbf{z}_t = -\frac{\beta_t}{2}\mathbf{z}_t dt + \sqrt{\beta_t} d\mathbf{w},$$

$$d\mathbf{z}_t = \left[-\frac{\beta_t}{2}\mathbf{z}_t - \beta_t \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t) \right] dt + \sqrt{\beta_t} d\bar{\mathbf{w}},$$

$$\mathcal{E} : \mathbb{R}^n \mapsto \mathbb{R}^d, \quad \mathcal{D} : \mathbb{R}^d \mapsto \mathbb{R}^n, \quad \mathbf{x} \approx \mathcal{D}(\mathcal{E}(\mathbf{x})),$$

Probability Flow ODE & Consistency Models



Consistency Models:

A distilled diffusion model obtained by training a deep neural network to transport $\mathbf{x}_t$ to $\mathbf{x}_0$ by mapping any point on the ODE's trajectory back to the origin. CMs are one-step samplers.

Posterior Sampling
Overdamped Langevin diffusion

$$\mathrm{d}\boldsymbol{x}_s = \nabla \log p(\boldsymbol{y}|\boldsymbol{x}_s)\mathrm{d}s + \nabla \log p(\boldsymbol{x}_s|c)\mathrm{d}s + \sqrt{2}\mathrm{d}\boldsymbol{w}_s$$

Key observations:

- Converges exponentially fast to the posterior **p(x|y,c)** as the time **s** increases.
- Modular structure with explicit likelihood (data fidelity) and regularisation terms.
- No need to embed likelihood within reverse SDE/ODE through approximations.
- How do we replace $\nabla \log p(\boldsymbol{x}_s|c)$ by a generative model, e.g., stable diffusion ?

Proposed discrete-time approximation

$$\boldsymbol{u} = \boldsymbol{x}_k + \int_0^\delta \nabla \log p(\tilde{\boldsymbol{x}}_s|c)\mathrm{d}s + \sqrt{2}\mathrm{d}\boldsymbol{w}_s\,,\ \tilde{\boldsymbol{x}}_0 = \boldsymbol{x}_k\,,$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{u} + \delta\nabla \log p(\boldsymbol{y}|\boldsymbol{x}_{k+1})\,,$$

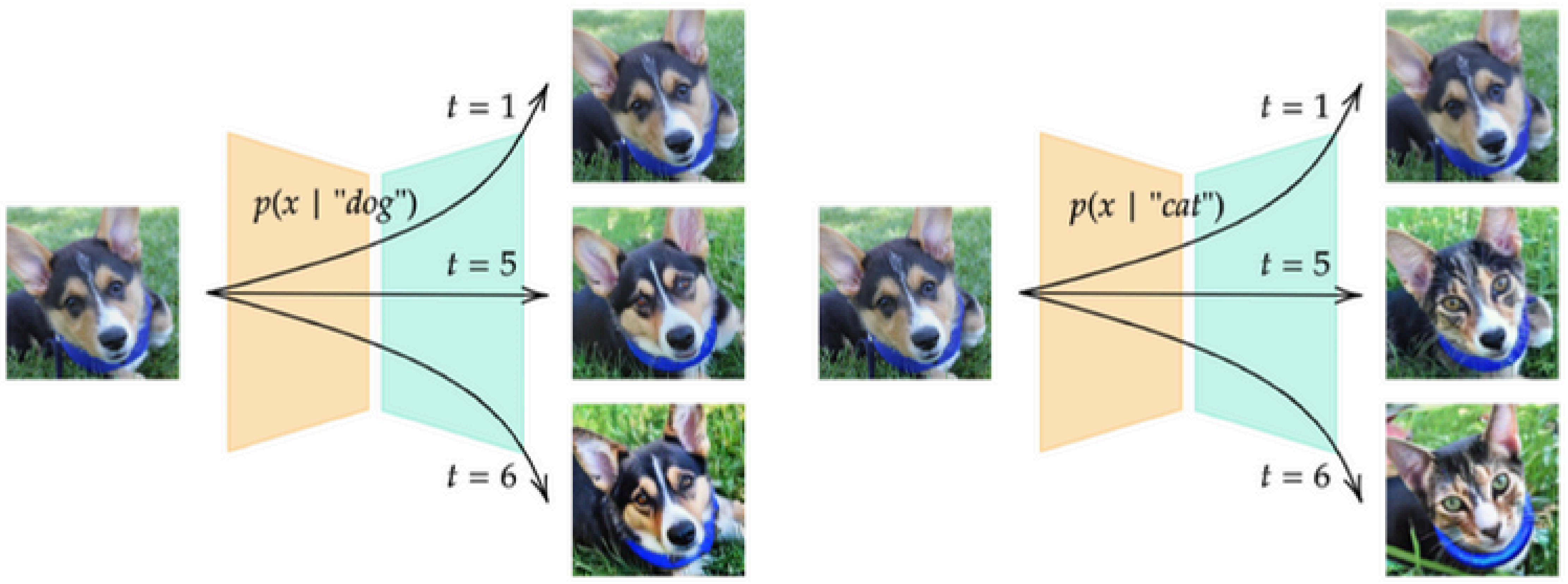
Main observations:

- The first line corresponds to a Langevin SDE targeting the prior **p(x|c)**.
  - It admits **p(x|c)** as unique invariant distribution.
  - It contracts exponentially fast towards **p(x|c)** as δ increases.
- The second line (implicit Euler) is equivalent to a so-called proximal step that can be solved exactly for many imaging problems.
- Key idea: replace the first line by a different Markov kernel that has similar properties.

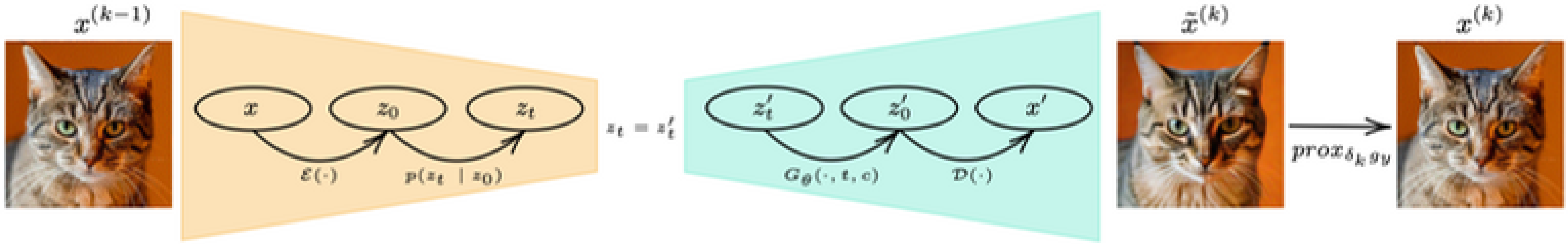
Auto-Encoding Stable Diffusion

$$\mathfrak{E}_t : \quad \boldsymbol{z}_t | \boldsymbol{x} \sim \mathcal{N}(\sqrt{\alpha_t}\mathcal{E}(\boldsymbol{x}), (1-\alpha_t)\mathrm{Id}_d)$$

$$\mathfrak{D}_{t,c} : \quad \boldsymbol{x}' = \mathcal{D}(G_\theta(\boldsymbol{z}'_t, t, c))$$



Proposed *Plug-and-Play* Langevin scheme



for $k = 1, \ldots, N$ **do**

$\quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{Id})$

$\quad \boldsymbol{z}_{t_k}^{(k)} \leftarrow \sqrt{\alpha_{t_k}}\mathcal{E}(\boldsymbol{x}^{(k-1)}) + \sqrt{1-\alpha_{t_k}}\boldsymbol{\epsilon}$ ▷ Encode

$\quad \boldsymbol{u}^{(k)} \leftarrow \mathcal{D}(G_\theta(\boldsymbol{z}_{t_k}^{(k)}, t_k, c))$ ▷ Decode

$\quad \boldsymbol{x}^{(k)} \leftarrow \mathrm{prox}_{\delta_k g_y}(\boldsymbol{u}^{(k)})$ ▷ $g_{\boldsymbol{y}} : \boldsymbol{x} \mapsto -\log p(\boldsymbol{y}|\boldsymbol{x})$

end for

LATINO (LAtent consisTency INverse sOlver)

Prompt Optimisation
Stochastic Approximation Projected Gradient

$$\hat{c}(\boldsymbol{y}) = \arg\max_{c \in \mathbb{R}^k} p(\boldsymbol{y} \mid c)$$

$$c_{m+1} = \Pi_C[c_m + \gamma_m \nabla_c \log p(\boldsymbol{y} \mid c_m)]$$

$$\begin{aligned} \nabla_c \log p(\boldsymbol{y} \mid c) &= \mathrm{E}_{\boldsymbol{x}\mid\boldsymbol{y},c}[\nabla_c \log p(\boldsymbol{y}, \boldsymbol{x} \mid c)], \\ &= \mathrm{E}_{\boldsymbol{x}\mid\boldsymbol{y},c}[\nabla_c \log p(\boldsymbol{x} \mid c)], \end{aligned}$$

$$\nabla_c \log p(\boldsymbol{y} \mid c_m) \approx \nabla_c \log p(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)} \mid c_m)$$

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

Prompt Optimisation
Stochastic Approximation Projected Gradient

for $m = 1, \dots, M$ **do**

  for $k = 1, \dots, N_m$ **do** $\triangleright$ LATINO

    $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{Id})$

    $\boldsymbol{z}_{t_k}^{(k)} \leftarrow \sqrt{\alpha_{t_k}}\mathcal{E}(\boldsymbol{x}^{(k-1)}) + \sqrt{1-\alpha_{t_k}}\boldsymbol{\epsilon}$

    $\boldsymbol{u}^{(k)} \leftarrow \mathcal{D}(G_\theta(\boldsymbol{z}_{t_k}^{(k)}, t_k, c_m))$

    $\boldsymbol{x}^{(k)} \leftarrow \mathrm{prox}_{\delta_k g_y}(\boldsymbol{u}^{(k)})$

  end for

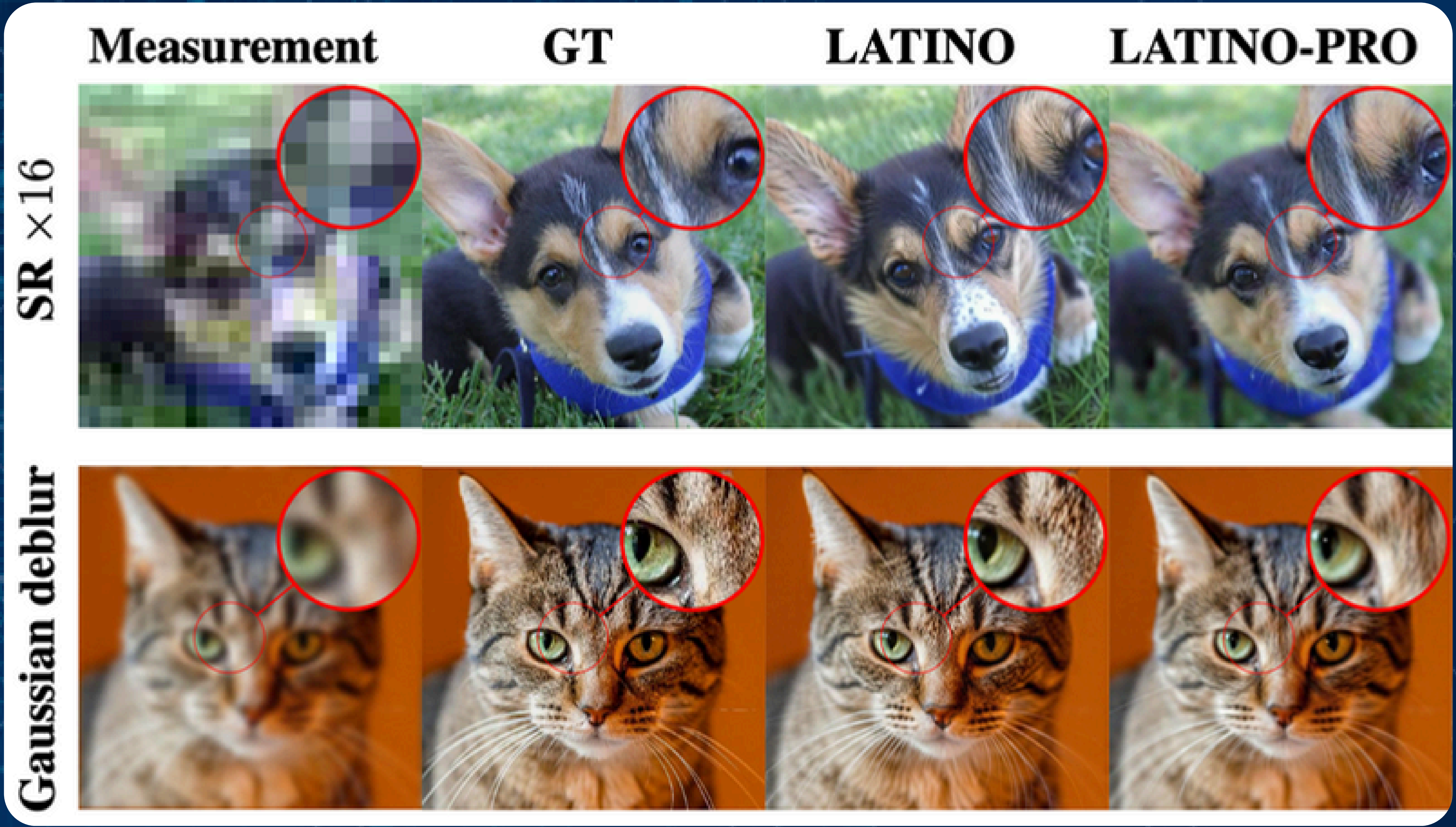
  $h(c_m) \leftarrow \nabla_c \log p(\boldsymbol{z}_{t_1}^{(1)}, \dots, \boldsymbol{z}_{t_{N_m}}^{(N_m)} | c_m)$

  $c_{m+1} = \Pi_C \left[c_m + \gamma_m h(c_m)\right]$ $\triangleright$ SAPG

  $\boldsymbol{x}^{(0)} \leftarrow \boldsymbol{x}^{(N_m)}$ $\triangleright$ Carry state forward

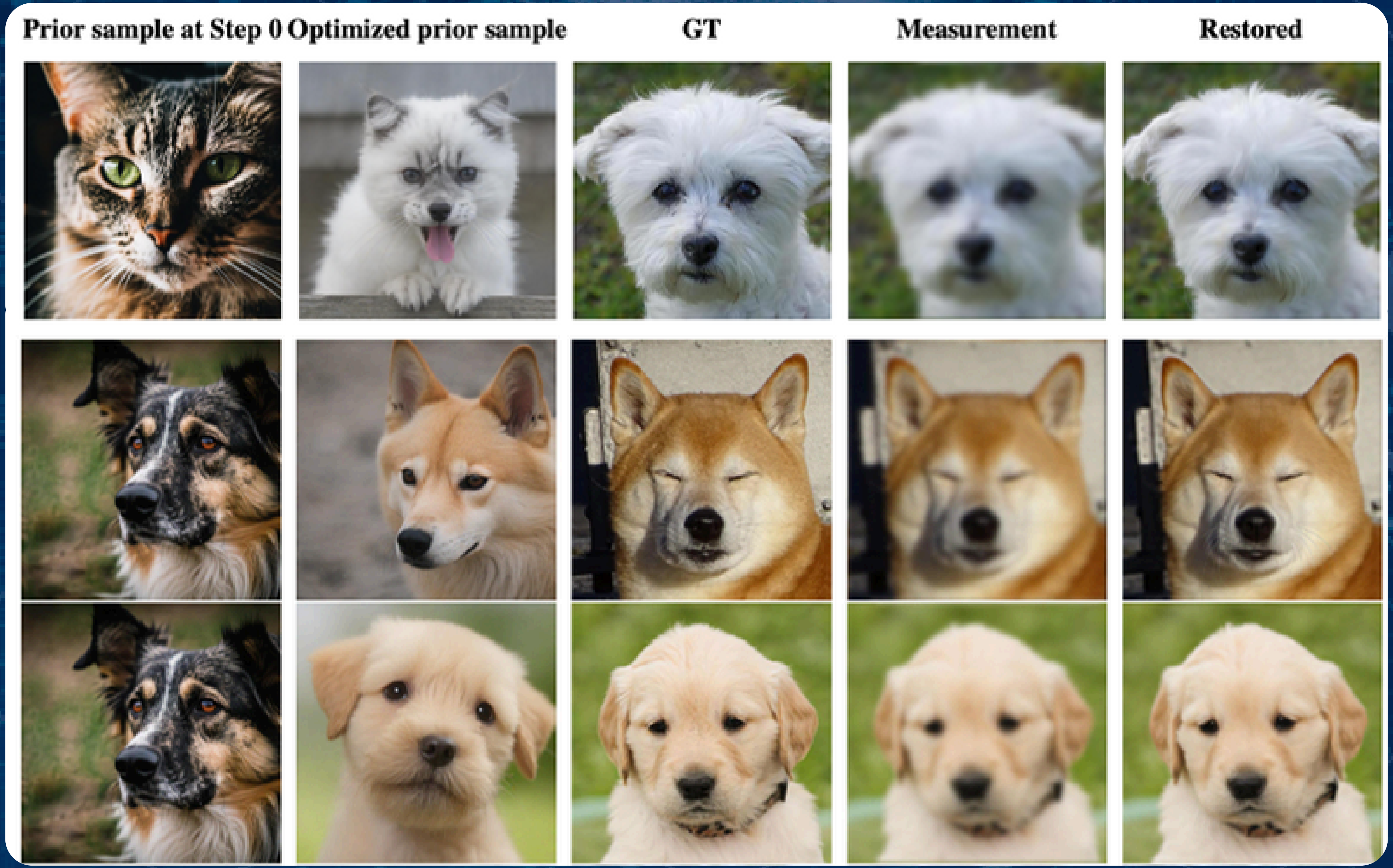
LATINO-PRO (LAtent consisTency INverse sOlver with PRompt Optimisation)

Some Results

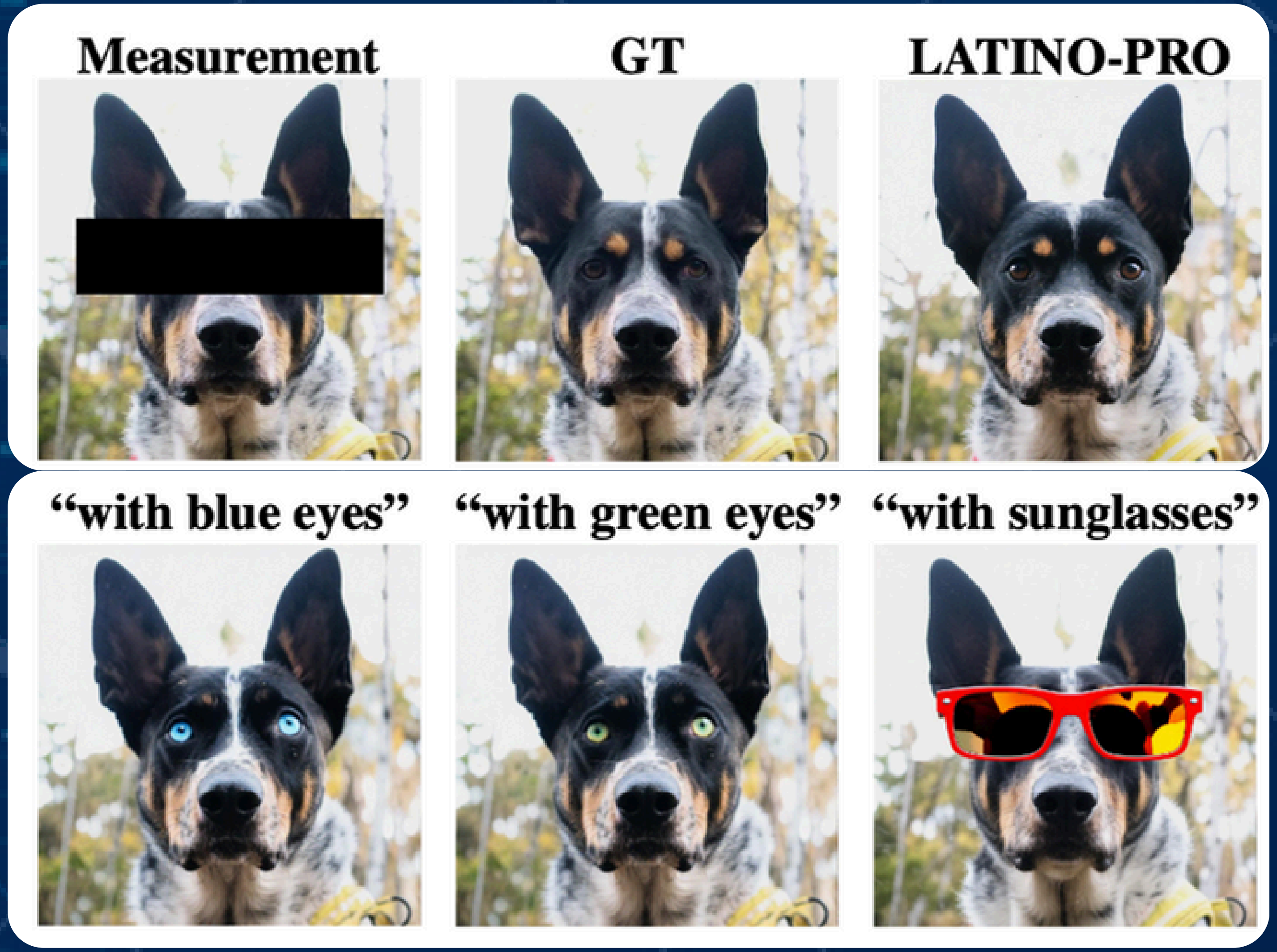


LATINO (8 NFEs) & LATINO-PRO (68 NFEs)

Visualisation of Prompt Optimisation

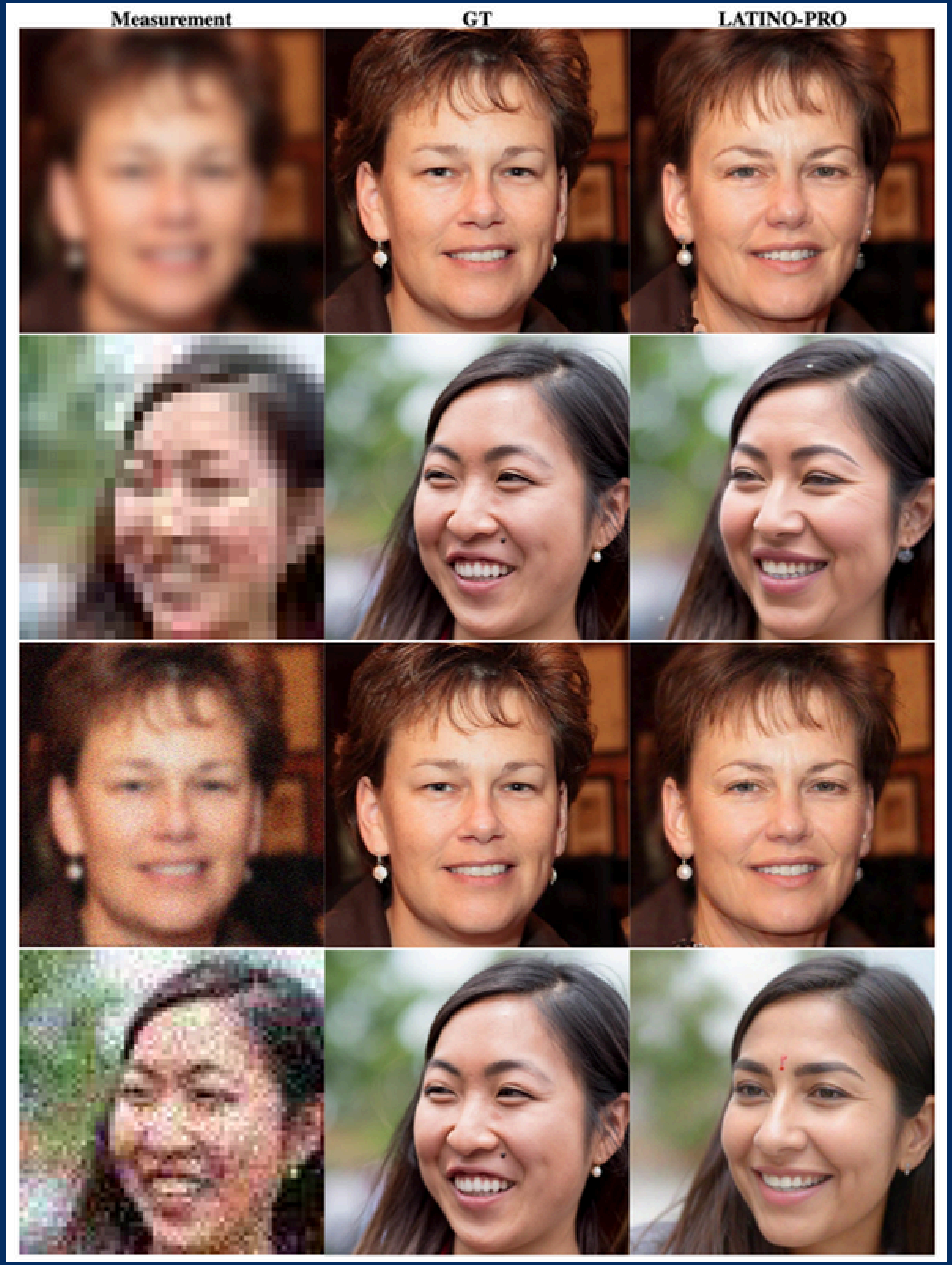


A sample from **p(x|c)** before and after 4 SAPG steps to adjust prompt (semantics)



Editing: sample from **p(x|c)** using constrained SAPG steps to enforce semantic constraints

Warning! The Internet is Biased



Open questions for adventurous NAs

- Asymptotic and non-asymptotic convergence analysis for large δ.
- What Markov kernels are "good" approximations of $\boldsymbol{u} = \boldsymbol{x}_k + \int_0^\delta \nabla \log p(\tilde{\boldsymbol{x}}_s|c)\mathrm{d}s + \sqrt{2}\mathrm{d}\boldsymbol{w}_s\,,\ \tilde{\boldsymbol{x}}_0 = \boldsymbol{x}_k\,,$?
- Constraining models to remain log-concave leads to worse models, but they also lead to slower algorithms. Why?
- No other known Langevin sampler (excluding trivial cases) converges in 4-8 steps in dimension 1M. We observe this behaviour with other DM priors, and on pixel space too. What's going on here?
- Good strategies for moving the forward model to the latent space (save encoder-decoder evals.)

Thank you!
https://arxiv.org/abs/2503.12615