Study of the paper : PNP-flow: Plug-and-play image restoration with Flow Matching

Mariia Baranova, Sara Meziane

March 21, 2025

1 Introduction

The core idea of the paper [1] is to leverage the Flow Matching method as a denoiser within the PnP framework. The authors propose integrating Flow Matching (FM) into the PnP approach by defining a time-dependent denoiser based on a pre-trained FM model. Their algorithm alternates between gradient descent steps on the data-fidelity term, reprojections onto the learned FM trajectory, and denoising.

2 Context

2.1 Inverse problems :

We place ourselves in the context of inverse problem resolution, where the goal is to recover an image x that has undergone degradation. This degradation process is typically modeled as:

$$y = H(x) + \xi, \tag{1}$$

where y is the observed degraded image, H represents the degradation operator (such as a blur or downsampling operator), and ξ is an additive Gaussian noise. The objective is to estimate x given y under appropriate prior assumptions. One of the most used approaches to solving this problem is MAP . We start by assuming that x comes from a r.v X and y from Y. By taking the MAP we end up with the following optimization problem:

$$x^* = \arg\min_x \left(-\log P(Y = y \mid X = x) - \log P(X = x)\right). \tag{2}$$

Here, the first term represents the data fidelity term derived from the likelihood model, while the second term acts as a regularization term incorporating prior knowledge about x. If we assume that the noise ξ follows a Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$, the problem becomes :

$$x^* = \arg\min_x \left(\frac{1}{2\sigma^2}\|y - H(x)\|^2 + \mathcal{R}(x)\right), \tag{3}$$

where $\mathcal{R}(x) = -\log P(X = x)$ is a regularization term that enforces prior constraints on x. Several regularization strategies exist, such as total variation (TV), sparse priors, or deep-learning-based denoising approaches like Plug-and-Play (PnP) methods.

Solving such an optimization problem generally relies on proximal splitting methods, which consist of performing a gradient descent step on the data fidelity term followed by a proximal step on the regularization term. In this paper, the authors propose an alternative Plug-and-Play (PnP) approach that replaces the proximal step with a learned denoiser, which directly captures the prior from data. More specifically, they leverage Flow Matching as a generative model to learn the underlying probability distribution.

2.2 Flow matching :

The main idea behind Flow Matching is to learn a velocity field v that transforms a **latent distribution** into the **target distribution**. Once v is learned, we can generate samples from the target distribution by solving the following Ordinary Differential Equation (ODE):

$$\frac{df(t,x)}{dt} = v(t, f(t,x)), \tag{4}$$

where $f(t,x)$ represents the trajectory of a sample evolving from the latent distribution (at $t = 0$) to the target distribution (at $t = 1$). The velocity field $v(t, f(t,x))$ is learned such that it guides the transformation smoothly over time.
The velocity field is learned in two steps: We design a probability path P_t, and then we train a velocity field $u_\theta(t)$ to generate P_t through a regression process.

First we define define the target probability path $t \mapsto P_t$, where $t \in [0,1]$, connecting the initial distribution P_0 and the target distribution P_1. Let $P_t := (e_t)_\# \pi$, where π is a coupling between P_0 and P_1 and the map $e_t(x_0, x_1) := (1-t)x_0 + tx_1$ interpolates between a latent sample x_0 and a data sample x_1. This path P_t is shown to be an absolutely continuous curve, meaning there exists a Borel vector field $v : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ such that the curve satisfies the continuity equation:

$$\frac{\partial P_t}{\partial t} + \nabla \cdot (P_t v_t) = 0. \tag{5}$$

We now proceed with the second step in the Flow Matching process: training our velocity field v_θ by performing regression towards a target velocity field v_t, which is known to generate the desired probability path P_t.

$$\mathcal{L}_{\mathrm{FM}}(\theta) := \mathbb{E}_{t\sim\mathcal{U}[0,1], x\sim P_t}\left[\|v_t^\theta(x) - v_t(x)\|^2\right] \tag{6}$$

Here $v_\theta(t)$ is approximated by a Neural Network. However, v_t is a complex object governing the joint transformation between two high-dimensional distributions, so we use a different objective instead. In particular we use the conditional flow matching :

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],(x_0,x_1)\sim\pi}\left[\left\|v_t^\theta\left(e_t(x_0,x_1)\right) - (x_1 - x_0)\right\|^2\right] \tag{7}$$

Minimizing the Conditional Flow Matching (CFM) loss is equivalent to minimizing the FM loss, and it only requires sampling pairs (x_0, x_1) from the coupling π which could be done thanks to an Optimal Transport Solver.

3 Method

The main novelty of the article is the introduction of a time-dependent denoiser, leveraging a learnt velocity field, as explained in the flow matching theory. The denoising operator is defined as:

$$D_t := \mathrm{Id} + (1-t)v_\theta \tag{8}$$

where v_θ is a pre-trained velocity field.
This definition is motivated by the fact that the best denoising performance is achieved with **straight-line flows**. Let $v : [0,1]\times\mathbb{R}^d \to \mathbb{R}$ and let f be a solution to the flow ODE associated with v. Given $(X_0, X_1) \sim \pi$, we call (f, v) a straight-line Flow Matching pair connecting X_0 and X_1 if $X_t = f(t, X_0)$ almost surely, where X_t is defined as

$$X_t := tX_1 + (1-t)X_0.$$

Straight or nearly straight paths are preferred since they represent the shortest distance between two points and can be simulated with fewer steps of an ODE solver.
It has been shown that the operator D_t can be understood as the best approximation of X_1 given the knowledge of X_t. The operator D_t is the minimizer of the L^2-problem

$$\min_g \mathbb{E}_{(X_0,X_1)\sim\pi}\left[\|X_1 - g(X_t)\|^2\right],$$

projecting any noisy point taken along the path onto the target distribution. Since it is a L2 norm, the output is expected to be "smooth".
A special case of straight-line flows is given by OT Flow Matching, for which we recover

$$D_t(X_t) = X_1.$$

The defined denoiser is plugged into a Forward Backward Splitting algorithm. However, it is different from classical PnP-FBS in 2 aspects: iterations of the algorithm depend on time and intermediate reprojection step between the gradient step on the data-fidelity term and the denoising step.
The algorithm is the following:

- **Gradient step on the data-fidelity term**: $z = x - \gamma\nabla F(x)$.
- **Interpolation step**: $\tilde{z} = (1-t)\varepsilon + tz$ (in case if the output z doesn't lie in the support of X_t which follows a staight path $X_t = (1-t)X_0 + tX_1$).
- **PnP Denoising step with denoising operator** D_t: $x = D_t(\tilde{z})$.

Algorithm 1 PnP Flow Matching

Require: Pre-trained network v^θ by Flow Matching, time sequence $(t_n)_n$ either finite with $t_n = n/N$, $N \in \mathbb{N}$ or infinite with $\lim_{n\to+\infty} t_n = 1$, adaptive stepsizes $(\gamma_n)_n$.

Initialize: $x_0 \in \mathbb{R}^d$.
for $n = 0, 1, \ldots$ **do**
  $z_n = x_n - \gamma_n \nabla F(x_n)$ ▷ Gradient step on the data-fidelity term
  $\tilde{z}_n = (1-t_n)\varepsilon + t_n z_n, \quad \varepsilon \sim P_0$ ▷ Linear interpolation
  $x_{n+1} = D_{t_n}(\tilde{z}_n)$ ▷ PnP step with denoiser (6)
end for
return x_{n+1}

We can notice in the algorithm that the learning rate is time dependant. The authors chose it as $\gamma_t = (1-t)^\alpha$ where α can take values between 0 and 1.

4 Experiments and results

Since the authors provided a GitHub repository (GitHub link), we did not reimplement the method. Instead, we began by testing various inverse problems on selected CelebA images, challenging the method with high noise variance, large occlusions, and other challenging conditions.

4.1 Denoising

Here we first take an image from CelebA and we apply a gaussian noise with $\sigma = 0.2$ and we try to see how the algorithm denoises it with different hyper-parameters values.
Now let's look how the images are generated using the PnP flow matching and different α values 0.01,0.2,0.5 where α is the exponent in the learning rate schedule

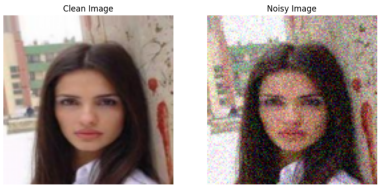

Figure 1: Cleaned and noised images with PSNR = 14

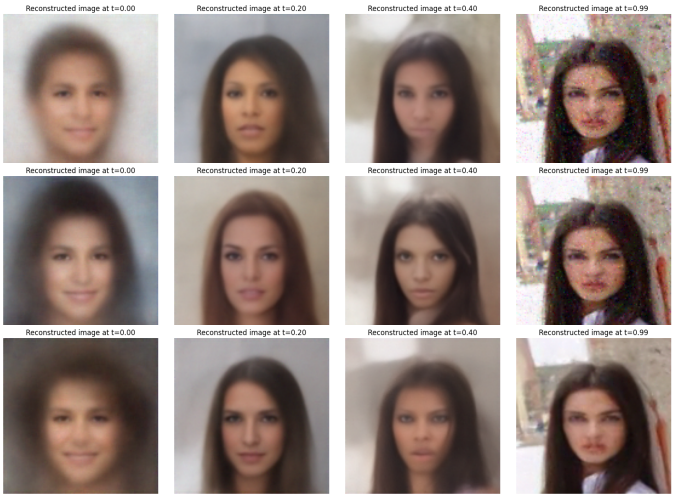

Figure 2: Reconstruction of the images with : First row (alpha = 0.01 , final PSNR = 17.8) , second row (alpha = 0.2 , final PSNR = 19.6) and final row (alpha = 0.5 , final PSNR = 21.3)

The visualizations above illustrate the impact of the parameter α on the reconstruction quality. When the learning rate exponent is not well adjusted, the algorithm struggles to remove noise effectively, leading to suboptimal results with residual artifacts. This sensitivity highlights the importance of careful parameter tuning. For this reason, the authors opted to fine-tune α and other hyperparameters separately for each inverse problem and dataset, ensuring optimal performance across different scenarios.

4.2 Pixels Inpaiting

To challenge the borderline cases of the proposed inverse problems scenarios, we tested how the algorithm reconstructs images with different quantity of masked pixels. The initial setting proposed by the authors was masking 70% of pixels of the original image, which indeed gave a good reconstruction (with a high PSNR), with α value fixed at 0.01. We progressively masked more pixels and noticed that the reconstruction stayed acceptably good (even though the PSNR score got lower), until 95% of masked pixels, which is a very good result.

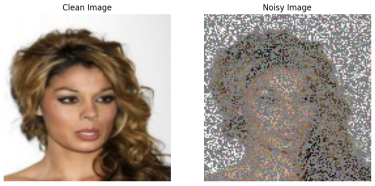

Figure 3: Original and image with 70% of masked pixels

(a) Masked 50%, PSNR = 28.1 (b) Masked 60%, PSNR = 25.9 (c) Masked 70%, PSNR = 24.1 (d) Masked 80%, PSNR = 21.4

(e) Masked 90%, PSNR = 16.9 (f) Masked 95%, PSNR = 12.9

Figure 4: Comparison of original and reconstructed images with different levels of masking.

4.3 Inpainting with large mask

One of the interesting tasks we investigated was the inpainting with a large mask around the image.

We observed that the reconstructed images with big masks often lack coherence with the surrounding regions, leading to noticeable inconsistencies such as abrupt changes in hair color, variations in skin tone, and an overall unnatural appearance. These artifacts make the results visually unconvincing and highlight the limitations of the current approach.

4.4 Deblurring

We challenged the proposed setting by increasing σ_b from 1 (suggested in the article) to 5 (and even more). The algorithm still performed well in these scenarios. We also noticed that it was not as sensitive to the tuning of alpha as other inverse problems, however the performance decreased with higher values as showed on Figure 7 .

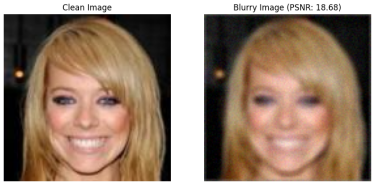

Figure 5: Blurred image with $\sigma_b = 5$ and $\sigma_{noise} = 0.05$

Figure 6: Restored with $\alpha = 0.01$, obtained PSNR=24.8

Figure 7: Restored with $\alpha = 0.5$, obtained PSNR=23.1

4.5 Replacing the denoiser

As suggested in the guidelines, we replaced the flow-based denoiser proposed by the authors with a deep neural network. Initially, we trained a DRUNet model [2] on the CelebA dataset, but the results were unsatisfactory, likely due to limited resources and time. Instead, we opted for the default weights from the DeepInv library, which demonstrated effective denoising on CelebA images.

Figure 8: Denoised with sigma=0.2

We then replaced the denoiser in the algorithm by our DruNet denoiser and got the following results :

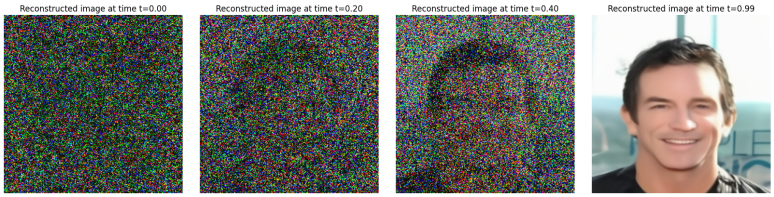

Figure 9: Reconstruction of the images with alpha = 0.01 and PNP-flow with Drunet as a denoiser (Final PSNR = 31)

Our experiments show promising results when replacing the flow denoiser with DRUNet, particularly for denoising, for which the only drawback is that the reconstructed images tend to appear overly smooth. However, our deblurring results were not satisfactory, which may be due to an issue in our implementation.

4.6 Remarks about Proposition 4

The proposition demonstrates the convergence of the sequence $(x_n)_{n\in\mathbb{N}}$ generated by the algorithm, given certain conditions. It assumes that the function $F : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable , the learned vector field $v : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is continuous and that the initial distribution P_0 has bounded support. If the time sequence $(t_n)_{n\in\mathbb{N}}$ satisfies $\sum_{n=0}^{\infty}(1-t_n) < +\infty$, and defining $\gamma_n := 1 - t_n$, then the sequence $(x_n)_{n\in\mathbb{N}}$ is guaranteed to converge, **provided it remains bounded**.

The demonstration is short and quite-straightforward, as they prove that $(x_n)_{n\in\mathbb{N}}$ is a cauchy sequence and therefore it converges (since we are in this case in a complete space). However , we find that the sentence "By assumption on F and v and since both (x_n) and the noise (ϵ_n) are bounded, the expression in the norm is bounded as well" is not precise enough. As we know that a continuous function applied on a bounded sequence is not necessarily bounded. A more rigorous justification would involve showing that the sequence remains within a compact set (i.e., a closed and bounded subset), which would ensure the boundedness of the expression under consideration.

Regarding the determinism of the algorithm, it is inherently non-deterministic since, at each iteration, we sample from the latent distribution. As suggested by the authors, we can make it deterministic by averaging over multiple noise realizations rather than drawing just one. Specifically, in the last step of the algorithm, we replace the update rule with the expectation over multiple samples $\varepsilon \sim P_0$:

$$x_{n+1} := \mathbb{E}_{\varepsilon\sim P_0}\left[D_{t_n}(\tilde{z}_n(\varepsilon))\right]$$

where

$$\tilde{z}_n(\varepsilon) := (1-t_n)\varepsilon + t_n z_n.$$

By adopting this approach, the algorithm's output becomes deterministic.

5 Conclusion

PnP-Flow Matching is a powerful approach that uses flow matching theory to solve inverse problems within the PnP framework. It achieves state-of-the-art performance, demonstrating strong results across various tasks. The algorithm is both intuitive and easy to implement, requiring minimal hyperparameter tuning. Additionally, it offers flexibility by supporting different latent distributions, not limited to Gaussian distributions.

However, the method encounters several challenges and limitations. One of the main issues is that most reconstructed images appear overly smooth, which can be attributed to the fact that the proposed denoiser acts as a mean squared estimator. Additionally, our experiments reveal that the method struggles in certain scenarios, particularly when dealing with high levels of noise or large masked regions. Furthermore, while the number of tunable parameters is relatively small, their impact on the quality of the results is significant, as demonstrated earlier.

References

[1] Ségolène Martin, Anne Gagneux, Paul Hagemann, and Gabriele Steidl. Pnp-flow: Plug-and-play image restoration with flow matching. *ICLR 2025*, 2024.

[2] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.