Fast Diffusion EM: a diffusion model for blind inverse problems with application to deconvolution

Charles Laroche
GoPro & MAP5
charles.laroche@u-paris.fr

Andrés Almansa
CNRS & Université Paris Cité
andres.almansa@parisdescartes.fr

Eva Coupette
GoPro
ecoupette@gopro.com

Abstract

Using diffusion models to solve inverse problems is a growing field of research. Current methods assume the degradation to be known and provide impressive results in terms of restoration quality and diversity. In this work, we leverage the efficiency of those models to jointly estimate the restored image and unknown parameters of the degradation model such as blur kernel. In particular, we designed an algorithm based on the well-known Expectation-Minimization (EM) estimation method and diffusion models. Our method alternates between approximating the expected log-likelihood of the inverse problem using samples drawn from a diffusion model and a maximization step to estimate unknown model parameters. For the maximization step, we also introduce a novel blur kernel regularization based on a Plug & Play denoiser. Diffusion models are long to run, thus we provide a fast version of our algorithm. Extensive experiments on blind image deblurring demonstrate the effectiveness of our method when compared to other state-of-the-art approaches. Our code is available at https://github.com/claroche-r/FastDiffusionEM.

1. Introduction

Image restoration aims to recover information that has been obscured by various degradations such as blur, noise, or compression artifacts. Deep-learning-based methods have revolutionized the field of image restoration by achieving impressive results in various tasks. They leverage the power of deep neural network architectures to learn a mapping between training data [11, 54, 56]. This data-driven approach allows deep-learning models to capture intricate patterns and relationships within the image data, enabling them to restore images with superior quality and perceptual fidelity [27, 50]. On the other hand, model-based approaches express the image restoration problem as an inverse problem and exploit the degradation process structure to design regularizations and optimization algorithms to find the optimal reconstruction [36]. They usually offer more control, flexibility, and interpretability. However, model-based approaches highly rely on the knowledge of the degradation forward process limiting their usefulness in practical applications. Some strategies try to bring the best of both worlds such as Plug-and-Play methods or deep unfolding networks [21, 22, 25, 40, 52]. One of the challenges behind inverse problems comes from their ill-posedness. In fact, for a single degraded image, there generally exist multiple plausible solutions. A common approach is to generate a single restored image that minimizes the mean squared error, but it does not allow the models to generate or hallucinate high-quality details [41, 49]. There is a growing interest in the field of image restoration to design models that can generate all the space of plausible solutions. Those models include Generative Adversarial Networks [15, 32] , conditional or PvP Diffusion Models [24, 41, 42] or Langevin dynamics [26]. This growing interest in diverse restoration is motivated by the impressive perceptual quality obtained by such methods. In particular, diffusion models that were first introduced for image synthesis tasks [19, 20, 43] are now used for a large diversity of tasks such as inverse problem solving [7, 24, 44]. In the field of blind deconvolution, it

Figure 1. Performance comparison of the different models using the PSNR metric depending on the runtime, “Ours” corresponds to Fast EM IIGDM method.
is common to use Bayesian methods to jointly estimate the blur kernel and the restored image [3, 30, 36, 37]. The kernel estimation highly relies on the restoration method that is used and it generally requires the restoration method to produce a sharp image. To do so, image regularizations such as TV, $\ell_1$ on the gradient can be used but they tend to over-sharpen the restored image leading to unpleasant results. Even with the sharp and blurry pairs, it is not easy to estimate any type of blur kernels without efficient regularization. Common regularization on the kernels are the $\ell_1$ norm [6], positivity, the sum to one constraint, and in some cases Gaussian constraints [4]. Some recent works also use deep neural networks such as normalizing flows to parameterize the kernels [28]. Motivated by the impressive quality of diffusion models for both estimated conditional distribution and returning high-quality images, it is natural to believe that they could be used in the context of kernel estimation. Also, a pioneer work [6] that combines parallel diffusion models for the kernel and image exhibits impressive results. Estimating the kernel and image is jointly done in the diffusion process using gradient descent on the forward model. Similarly, methods based on Monte Carlo sampling proposed parameters estimation derived from the Expectation-Maximization (EM) algorithm [13, 17], or the SAPG algorithm [12, 46]. Those methods are very efficient but Monte Carlo sampling is time-consuming. Also, the problem of kernel estimation is a complex problem so those methods highly depend on the regularization imposed in the M-step of the EM algorithm.

Motivated by the efficiency of diffusion models, we propose a diffusion model that solves the maximum a-posteriori estimator for blind deconvolution. Derived from the classical Expectation-Maximization algorithm, our model alternately estimates the expected value of the log-likelihood using samples drawn from a diffusion model and maximizes this quantity using half-quadratic splitting. In addition, we also propose a novel kernel regularization in a Plug & Play fashion. Finally, we proposed a fast version of our algorithm to facilitate the use of our method in real-world scenarios. Our experiments show that our proposed solution improves both in terms of fidelity and computational efficiency pushing the Pareto optimal curve further to the origin (Figure 1).

2. Background

Let us suppose that our deblurring problem fits the classical inverse problem formulation:

$$y = Hx + n \quad \text{with} \quad n \sim \mathcal{N}(0, \sigma^2)$$  \hspace{1cm} (1)

where $x$ is the clean image we want to estimate, $y$ is the blurry and noisy image and $H$ is the degradation operator, a convolution operator in the case of deconvolution. We suppose that we are in the real-world case where we only have access to the blurry image $y$ and the noise level $\sigma$ to reconstruct both the clean image and the blur kernel $H$. In such a setting, a common approach to estimate the blur kernel is to compute the marginalized maximum a-posteriori (MAP) estimator of the inverse problem described in Equation (1):

$$H_{MAP} = \arg \max_H p(H|y) = \arg \max_H p(y|H)p(H)$$  \hspace{1cm} (2)

$$= \arg \max_H \left[ \log \left( \int p(y|H, x)p(x)dx \right) + \log(p(H)) \right],$$

with $p(x)$ a natural image prior, $p(y|H, x)$ the likelihood of the blurry image and $p(H)$ the kernel’s prior distribution. This MAP estimator cannot be solved easily since the marginalization in the clean image $x$ is not tractable. Expectation-Maximization (EM) [10, 31] is an iterative algorithm that computes the MAP estimator for the parameters of a statistical model ($H$ in our case). It is very convenient when the model contains unobserved or missing data. The EM algorithm consists of two main steps. An $E$-step that computes the expected log-likelihood given the current model parameter estimates and an $M$-step, that maximizes this expected log-likelihood to update the estimated parameters. The whole algorithm alternates between the $E$-step and $M$-step until convergence. In the case of deblurring, the parameter we want to estimate is the blur kernel $H$ and our unobserved data are the clean images associated with the blurry image $y$ and the estimated blur kernel $H$. The EM algorithm alternates between the E-step and the M-step estimation. The E-step is performed by marginalizing the likelihood over the latent variable $x$ and maximizing the expected log-likelihood with respect to $H$. The M-step is performed by maximizing the expected log-likelihood with respect to $H$.

Figure 2. Overview of the method and evolution of the current estimates. We start with random noise and apply the diffusion process. The blurry image intervenes both for the guidance and for the M-step which estimates the blur kernel.
algorithm can be summarized as follows in such setting:

**E-Step:**

\[
Q(H, H_t) = E_{x \sim p(x|y, H_t)} [\log(p(y|x, H)) + \log(p(x))]
\]

(3)

**M-Step:**

\[
H_{t+1} = \arg \max_H [Q(H, H_t) + \log(p(H))]
\]

(4)

This formulation is very convenient but in many applications (including blind deblurring), the expected log-likelihood in Equation (3) cannot be computed explicitly, and even taking posterior samples \( x \sim p(x|y, H_t) \) is challenging. Our method proposes to approximate the expectation in the E-step by an empirical mean in Monte-Carlo EM fashion [48] and to use a diffusion model to obtain posterior samples.

**Diffusion models for posterior sampling:** To learn \( p(x_0) \) the distribution of the data, diffusion models define a family of distributions \( p(x_t) \) by gradually adding Gaussian noise of variance \( \beta(t) \) to samples of \( p(x_0) \) until the distribution \( p(x_T) \) reduces to a standard Gaussian with zero mean. For discrete timesteps \( t \in [0, T] \), we can define a Markov transition kernel \( p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta(t)}x_{t-1}, \beta(t)I) \) between two consecutive discrete timestamps. In the general continuous case, [45] described the forward noising process with the following stochastic differentiable equation (SDE):

\[
dx_t = -\frac{\beta(t)}{2} x_t dt + \sqrt{\beta(t)} dw
\]

(5)

where \( w(t) \) is the d-dimensional Wiener process. The reverse SDE of this process [2] can be written as:

\[
dx_t = \left[ -\frac{\beta(t)}{2} x_t - \beta(t) \nabla x_t \log p(x_t) \right] dt + \sqrt{\beta(t)} dw
\]

(6)

with \( dt \) corresponding to time running backwards and \( dw \) to the standard Wiener process running backwards. In the case of inverse problems, we want to use diffusion models to generate the posterior distribution \( \pi(x_t) = p(x_t|y, H) \). Using Bayes’ rule Equation (6) becomes:

\[
dx_t = \left[ -\frac{\beta(t)}{2} x_t - \beta(t) \nabla x_t \log p(x_t) \right] dt + \sqrt{\beta(t)} dw
\]

(7)

The main problem behind this equation is that in inverse problems, we have a relation between \( y \) and \( x_0 \) but not between \( x_t \) and \( y \). Marginalizing in \( x_0 \), we obtain:

\[
p(y|x_t) = \int p(y|x_0)p(x_0|x_t)dx_0
\]

(8)

that is intractable. The main challenge of non-blind diffusion for posterior sampling is to compute or approximate this integral. In our work, we conduct experiments with DPS [7] and IIGDM [44] that use different approximations for this integral. Both approximations are based on the mean of \( p(x_0|x_t) \), namely:

\[
\hat{x}_0(t) := E[x_0|x_t].
\]

DPS approximates \( p(x_0|x_t) \) by a delta function

\[
p(x_0|x_t) \approx \delta_{\hat{x}_0(t)}(x_0)
\]

(9)

whereas IIGDM approximates \( p(x_0|x_t) \) by a Gaussian distribution

\[
p(x_0|x_t) \approx \mathcal{N}(x_0|\hat{x}_0(t), \tau_r^2)
\]

(10)

with \( \tau_r \) a hyper-parameter. Both approximations allow us to solve the marginal in Equation (8) analytically and obtain explicit expressions for \( \nabla x_t \log p(y|x_t) \) as detailed below.

As a recall, one property of diffusion models is that we can express the noisy measurement \( x_t \) in the forward model using the original sample \( x_0 \):

\[
x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t} \epsilon
\]

(11)

with \( \alpha_t = 1 - \beta_t \) and \( \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_t \).

Using a noise predictor \( \epsilon(x_t, t) \), we can thus estimate \( \hat{x}_0(t) = E[x_0|x_t] \) at each step \( t \) using:

\[
\hat{x}_0(t) = \frac{1}{\sqrt{\bar{\alpha}_t}} (x_t - \sqrt{1-\bar{\alpha}_t} \epsilon(x_t, t)).
\]

(12)

Equivalently, we can use a score network \( s(x_t, t) \) using Tweedie’s identity:

\[
s(x_t, t) = \nabla x_t \log p(x_t) = -\frac{1}{\sqrt{1-\alpha_t}} \epsilon(x_t, t).
\]

(13)

Using DDPM [19] to discretize the unconditional reverse diffusion process (6) we obtain the update rule

\[
x_{t-1} = \frac{1}{\sqrt{\alpha_t}} (x_t + \beta_t s(x_t, t)) + \bar{\sigma}_t \mathcal{N}(0, I)
\]

(14)

where \( \bar{\sigma}_t = \sqrt{\beta_t} or \sqrt{\frac{1-\alpha_{t-1}}{1-\alpha_t}} \beta_t \). To simulate the conditional reverse diffusion process (7), we just have to add the likelihood term to the score

\[
x_{t-1} = \frac{1}{\sqrt{\alpha_t}} (x_t + \beta_t [s(x_t, t) + \nabla x_t \log p(y|x)] + \bar{\sigma}_t \mathcal{N}(0, I)
\]

(15)
Using Equation (12), the DPS [7] approximation for
\( p(x_0|x_i) \) leads to the following formula for the gradient of the
log-likelihood:
\[
\nabla_{x_t} \log p(y|x_t) = -\frac{1}{\sigma^2} \nabla_{x_t} \|y - H\hat{x}_0(t)\|^2
\] (16)

Similarly, the IIGDM [44] approximation leads to the fol-
lowing gradient for the log-likelihood:
\[
\nabla_{x_t} \log p(y|x_t) = \left( (y - H\hat{x}_0(t))^T (r_t^2 H H^T + \sigma^2 I)^{-1} H \left( \frac{\partial \hat{x}_0(t)}{\partial x_t} \right) \right)^T 
\] (17)

DPS and IIGDM derive different guidance terms for the in-
verse problem. While the DPS approximation leads to a
gradient that is easily implemented for any degradation op-
erator \( H \) using automatic differentiation, the IIGDM ap-
proximated gradient of Equation (17) is much more com-
plex to estimate for a general operator \( H \) because it re-
quires the computation of its pseudo-inverse. On the other
hand, the IIGDM approximation is more precise and thus
leads to stronger guidance which is very important for
kernel estimation. We summarize in Algorithm 1 the diffusion
process for inverse problems when the degradation operator
\( H \) is known. This case covers both DPS and IIGDM. The
pseudo-code is written using DDPM but is not limited to
this particular diffusion scheme. To compensate for the fact
that the first estimations of \( x_t \) are uncertain, it is common
to set \( \zeta_t = \sqrt{\alpha_t} \), instead of the theoretical \( \zeta_t = 1 \).

3. Method

Our method proposes to solve the MAP of the blur kernel
from a blurry and potentially noisy image. We estimate the
MAP estimator in an EM fashion. Iteratively, we first draw
samples from the posterior distribution knowing the current
kernel estimate using a diffusion model. It corresponds to
the E-step of the EM algorithm. Then, we update our esti-
mated kernel with the M-step by maximizing the expected
log-likelihood on the previously computed samples. To ef-
ficiently model the kernels’ distribution, we use a Plug &
Play kernel denoiser to regularize our MAP estimator.

3.1. E-step: Non-blind diffusion

The E-step of the EM algorithm consists in evaluating the
expectation from Equation (3). Instead of computing
its exact value, we propose to approximate it using random
samples in a Monte-Carlo EM fashion. To draw the random
samples, we use a non-blind diffusion model. Since the dif-
fusion model targets \( p(x|y,H_t) \), sampling several images
leads to a good approximation of the expectation. The num-
ber \( n \) of samples used to approximate the expectation is a
hyperparameter of the method. Having many samples leads
to a slow but accurate estimation while having only one
sample is equivalent to the Stochastic EM algorithm [35].
In practice, the E-step reduces to:

**Drawing samples**
\[
x = (x^1, \ldots, x^n) \sim p(x_0|y, H_t)
\] (18)

and updating
\[
\hat{Q}(H,H_t) = \frac{1}{n} \sum_{i=1}^{n} \log(p(y|x^i, H)).
\] (19)

The samples can be drawn by \( n \) parallel runs of Algo-
rum 1, and the empirical mean \( \hat{Q}(H,H_t) \approx Q(H,H_t) \)
approaches the expected value in Equation (3) as \( n \to \infty \).
Unlike in Equation (3), we remove the term in \( p(x) \) from
\( \hat{Q}(H,H_t) \) here since it does not affect the maximization in
the blur kernel \( H \).

3.2. M-step: Kernel estimation

The M-step computes the MAP estimator of the blur ker-
nel using the estimated samples from the E-step as mea-
surements. From equations (1), (4) and (19) this step can be
summarized as:
\[
H_{t+1} = \arg \max_H \hat{Q}(H,H_t) + \log(p(H))
\] (20)
\[
H_{t+1} = \arg \min_H \frac{1}{2n\sigma^2} \sum_{i=1}^{n} \|y - H x^i\|^2 + \lambda \Phi(H)
\] (21)

where (21) is obtained using Equation (1) and (19). Com-
mon choices for \( \Phi(\cdot) \) are \( \ell_2 \) or \( \ell_1 \) regularizations on top of
the simplex constraints on the blur kernel (non-negative val-
ues that add up to one). Despite being quite efficient when
the blurry image does not have noise, they generally fail to
provide good quality results when the noise increases.
On the other side, Plug & Play regularizations have become
more and more popular for many image restoration tasks.
By training a deep denoiser on Gaussian denoising, one can
obtain a powerful regularization in the domain on which the
denoiser was trained. Generally, we train the denoiser on
a dataset of natural images leading to a regularization on
natural images. Here, we propose to train a denoiser on a
dataset of blur kernels to build a Plug & Play regularization
for the blur kernels. We observed that this approach leads
to a kernel estimation algorithm that is more efficient and
robust to noise, see Figure 4. To solve Equation (21), we use
the Half-Quadratic Splitting (HQS) optimization scheme:
\[
Z_{j+1} = \arg \min_{Z} \frac{1}{2\sigma^2 n} \sum_{i=1}^{n} \|Z x^i - y\|^2 + \frac{\beta}{2} \|Z - K_j\|^2
\] (22)
\[
K_{j+1} = \arg \min_{K} \lambda \Phi(K) + \frac{\beta}{2} \|K - Z_{j+1}\|^2
\] (23)
For the deconvolution problem, Equation (22) can easily be solved in the Fourier domain (more details on the computations can be found in Appendix B). Equation (23) corresponds to the regularization step. It corresponds to the MAP estimator of a Gaussian denoising problem on the variable $Z_{j+1}$. The main idea behind Plug & Play regularization is to replace this regularization step with a pre-trained denoiser $D$ Measured Squared Error (MSE) loss. This substitution can be done thanks to the close relationship that exists between the MAP and the MMESE estimator of a Gaussian denoising problem [16]. Eventually, the $M$-step consists of the following iterations:

$$Z_{j+1} = \mathcal{F}^{-1} \left( \frac{\mathcal{F}(y) \sum_{i=1}^{n} \mathcal{F}(x_i) + n\beta^2 \mathcal{F}(K_j)}{\sum_{i=1}^{n} \mathcal{F}(x_i) \mathcal{F}(x_i)} + n\beta^2 \right),$$

$$K_{j+1} = D \sqrt{1/\beta}(Z_{j+1}).$$

While complex decreasing schemes for $\beta$ are often used to help HQS converge [53], we observed that using a constant $\beta$ was sufficient in our case. For the denoiser architecture, we use a simple DnCNN [54] with 5 blocks and 32 channels. In addition to the noisy channel, we also give the noise level as an extra channel to the network to control the denoising intensity. Eventually, the complete Diffusion EM algorithm alternates between sampling from the non-blind diffusion model and the HQS algorithm for the kernel estimation. In all our experiments, we use $L = 10$ EM iterations. See Algorithm A.1 in the supplementary.

### 3.3. Fast EM diffusion

The diffusion EM algorithm requires running a diffusion model at each step of the EM algorithm to produce a set of $n$ particles. Executing diffusion models is time-consuming, particularly in cases where inverse problems are addressed using score guidance, as the guidance must be applied to the full-size image, precluding the utilization of acceleration techniques like latent diffusion [38]. Consequently, the diffusion EM algorithm’s execution time becomes excessively long, significantly restricting its practical applicability.

To bypass this problem, we propose a fast version of diffusion EM that incorporates the $M$-step directly into the diffusion process, thereby reducing the number of required diffusion model runs to just one. To do so, we use the $n$ current samples $x_i \sim p(x_i | y, H)$ to build an approximation of $Q(H, H_t)$ at each timestep $t$, as follows. First, we use the current distribution estimates $p(x_i | z_t)$ (Equations (9) and (10)) for DPS, resp. IIGDM approximations) for each timestep $t$ to approximate the posterior $p(x_0 | y, H)$ by (discretized) marginalization on $x_t$:

$$p(x_0 | H, y) = \int p(x_0 | x_t) p(x_t | y, H) dx_t$$

$$\approx \sum_{i=1}^{n} p(x_0 | x_i) p(x_i | y, H)$$

$$= \frac{1}{n} \sum_{i=1}^{n} p(x_0 | x_i) =: q_t(x_0 | y, H).$$

Then, using this approximation, the $E$-step at timestep $t$ of the diffusion process is reformulated as follows:

$$Q(H, H_t) = E_{x \sim p(x | y, H_t)} [\log p(y | x, H)]$$

$$\approx E_{x \sim q_t(x_0 | y, H_t)} [\log p(y | x, H)].$$

Since the distribution $q_t(x_0 | y, H)$ progressively converges to the distribution $p(x_0 | y, H)$ as $t \to 0$, we have a finer and finer estimation of the expected log-likelihood and thus, the blur kernel, through the iterations.

Finally, the $E$-step reduces in the case of the DPS approxi-
Table 1. Model comparison on FFHQ synthetic dataset. Models with a “*” correspond to non-blind models used as baselines. Best blind models are in **bold** while second best are **underlined**. Note that baselines do not count for best model rankings.

<table>
<thead>
<tr>
<th>Method</th>
<th>Metric type</th>
<th>Time (sec/img)</th>
<th>Reference metrics</th>
<th>No-reference metrics</th>
<th>Kernel error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>PSNR ↑</td>
<td>SSIM ↑</td>
<td>LPIPS ↓</td>
</tr>
<tr>
<td>Blind DPS</td>
<td></td>
<td>5sec</td>
<td>25.81</td>
<td>0.76</td>
<td>0.34</td>
</tr>
<tr>
<td>Anger λi</td>
<td></td>
<td>0.73sec</td>
<td>12.46</td>
<td>0.13</td>
<td>0.8</td>
</tr>
<tr>
<td>Self-Deblur</td>
<td></td>
<td>1min53sec</td>
<td>14.53</td>
<td>0.15</td>
<td>0.69</td>
</tr>
<tr>
<td>MPRNet</td>
<td></td>
<td>3.7sec</td>
<td>19.52</td>
<td>0.42</td>
<td>0.54</td>
</tr>
<tr>
<td>Blind DPS</td>
<td></td>
<td>1min23</td>
<td>24.05</td>
<td>0.73</td>
<td>0.34</td>
</tr>
<tr>
<td>EM IIGDM (n=1)</td>
<td></td>
<td>1min30sec</td>
<td>23.4</td>
<td>0.71</td>
<td>0.43</td>
</tr>
<tr>
<td>EM IIGDM (n=4)</td>
<td></td>
<td>2min30sec</td>
<td>23.21</td>
<td>0.71</td>
<td>0.4</td>
</tr>
<tr>
<td>EM IIGDM (n=16)</td>
<td></td>
<td>9min10sec</td>
<td>23.09</td>
<td>0.71</td>
<td>0.39</td>
</tr>
<tr>
<td>Fast EM DPS (n=1)</td>
<td></td>
<td>1min41</td>
<td>24.68</td>
<td>0.75</td>
<td>0.34</td>
</tr>
<tr>
<td>Fast EM IIGDM (n=1)</td>
<td></td>
<td>9sec</td>
<td>25.66</td>
<td>0.79</td>
<td>0.34</td>
</tr>
<tr>
<td>Fast EM IIGDM (n=4)</td>
<td></td>
<td>15sec</td>
<td>25.74</td>
<td>0.8</td>
<td>0.34</td>
</tr>
<tr>
<td>Fast EM IIGDM (n=16)</td>
<td></td>
<td>55sec</td>
<td>25.75</td>
<td>0.8</td>
<td>0.34</td>
</tr>
</tbody>
</table>

4. Experiments

4.1. Experimental settings

We test our algorithm on the first 1000 validation images of the widely used FFHQ [23] 256x256 dataset that we degrade with random motion blur kernels computed using [14] and random Gaussian noise with noise level σ ∈ \{5, 10, 20\}. We also provide some results on DIV2K [1] dataset. To achieve a fair comparison, we use the code and

pre-trained weights provided by the authors of Blind DPS. For IIGDM, there is no public code so we re-implemented the model using the Blind DPS code backbone. In our experiments, we obtained that DPS needs more iterations to properly converge in comparison to IIGDM. Indeed, the DPS run needs 1000 iterations while we only use 100 iterations for IIGDM. For the kernel estimation, we use a bias-free FFDNet [55] denoiser trained on a dataset of motion blur kernels for the Plug & Play regularization. At test time, the M-step consists of 10 HQS iterations with hyper-parameters \( \lambda = 1 \) and \( \beta = 1e5 \). We provide experiments with different numbers of particles for both the Diffusion EM algorithm and the Fast diffusion EM algorithm. We use \( n \in \{1, 4, 16\} \). All the models are evaluated on a single A100 GPU.
4.3. Quantitative results

Table 1 shows the results of the different models on FFHQ synthetic dataset. We compute both classical metrics with full or reduced reference such as PSNR, SSIM [47], LPIPS [57] and FID [18], no-reference metrics to measure perceptual quality such as NIQE [34] and BRISQUE [33] and kernel metrics such as the Mean-Squared Error (MSE) on the reconstructed kernel. We also measure the consistency of the estimated image $\hat{x}$ and kernel $\hat{H}$ with the forward model by means of:

$$L_{\text{reblur}}(y, \hat{x}, \hat{H}) = \|\hat{H}\hat{x} - y\|^2_2 - \sigma^2 M$$

where $M = 3hw$ is the number of elements in vector $x$. We observe that classical optimization-based approaches such as Anger $\ell_0$ [3] and Self-Deblur [37] fail to estimate the blur and reconstruct the image efficiently. The main problem with those approaches is that they fail to produce pleasant results in the presence of noise. While Anger $\ell_0$ [3] produces results with over-sharpened noise, Self-Deblur [37] completely fails to both estimate the kernel and deblur the image. MPRNet produces better results but with artifacts due to the noise, it also fails to recover high-frequency details which is a common problem when using deep-learning models trained on mean-squared error. Diffusion-based models seem to be the most efficient. Blind DPS ranks best among the no-reference perceptual metrics and FID while ranking below our model both for reference metrics and kernel estimation. Figure 3, shows some example images where we can notice the sharpness and high quality of Blind DPS results. In our experiments, we observed that Blind DPS sometimes fails to efficiently estimate the blur kernel, especially in the presence of noise. We also noticed that on some images Blind DPS was producing sharper results than our model, even with a worst kernel prediction which is surprising since we use the same diffusion model. Yet, the fact that our model has better full-reference metrics and better measurement consistency points out the fact that Blind DPS hallucinates more details. We also conducted experiments on deblurring images from DIV2K dataset while keeping the same FFHQ-trained score model for testing. In that particular case, the prior of the score model does not match the distribution of the test images so the model won’t be able to hallucinate accurate details. Some visual results of those experiments can be found in Figure 5. Those experiments showed that our model and especially the one based on IIGDM diffusion produces sharper results. It highlights the fact that Blind DPS and DPS, in general, have weaker guidance than IIGDM, so it requires a more accurate score model which can be a limitation in practice since training a score model on the space of natural images is not an easy task. During our experiments, we realized that Fast Diffusion EM was both faster and better in terms of quality than
In this article, we present a novel approach for blind deconvolution based on diffusion models. In particular, we designed Diffusion EM, an algorithm based on the Expectation-Maximization algorithm. This algorithm consists of an E-step, which approximates the expected value of the log-likelihood using a diffusion model, and an M-step, which maximizes this expected log-likelihood with respect to the unknown parameters (the blur kernel). For the M-step, we introduced a novel kernel regularization based on a Plug & Play denoiser. The diffusion EM algorithm is slow since it requires running a diffusion model several times. We propose an acceleration of the algorithm that directly injects the EM iterations into the diffusion process (leveraging the intermediate diffusion steps as approximate posterior samples). We observed that this Fast EM diffusion model reaches better performance than the original diffusion EM algorithm while being significantly faster. Finally, we demonstrate the efficiency of our approach both quantitatively and visually. We compare our approach to state-of-the-art methods for blind deconvolution and provide several ablation studies that highlight the performance of our regularization and model and give insights into the behavior of the model. In its current form, our algorithm is limited to deconvolution. Future research will address more general blind deblurring problems. [5, 9]. Faster diffusion models such as latent diffusion [8, 39] or diffusion bridges [29] could also benefit our method.

## 5. Conclusion

Diffusion EM. Indeed, Diffusion EM is sometimes stuck in the no blur solution while we never observed this problem for Fast Diffusion EM. In terms of metrics, both Fast EM DPS and Fast EM IIGDM have better reference metrics than all the other methods, and for any number of particles. We observed better performance and faster runtime with the IIGDM model, probably because it has stronger guidance, thus, it is easier for the M-step to estimate the blur kernel. Fast EM IIGDM performance in no-reference metrics NIQE and BRISQUE is worse than the other diffusion-based methods: BlindDPS and Fast EM DPS have indeed slightly sharper results, but they are less accurate and less consistent (see the hallucinations of BlindDPS in the second line in Figure 3). In terms of runtime, our IIGDM-based model ranks best among diffusion models but it is significantly slower than MPRNet and Anger $\ell_0$.

### 4.4. Ablation studies

In this section, we discuss the efficiency of the different blocks of our algorithm. We first provide some additional results that show the efficiency of the proposed Plug & Play-based kernel regularization. Next, we study the influence of the number of samples used to estimate the E-step on the quality of the final results. To compare the efficiency of our regularization, we compared it against the $\ell_1$ and $\ell_2$ regularizations. To do so, we use our FFHQ synthetic dataset and estimate the blur kernel in the non-blind setting where the sharp and blurry images are both known. We compute the MSE of the reconstructed kernel for several noise levels. For all the regularizations, we used the same optimization scheme, HQS, and fine-tuned the hyper-parameters of the regularizations separately. Figure 4 shows the obtained results. We observed that our regularization is significantly better in the presence of noise and the loss of quality between $\sigma = 5$ and $\sigma = 20$ is very small. Finally, we also investigated the influence of the number of samples in our algorithms. We observed in Table 1 and Table 2 that increasing the number of samples increases the image reconstruction and kernel estimation accuracy. Using all the samples, we can also compute the PSNR on the average of the samples produced by the model. We refer to this metric as the “PSNR SA” in Table 2. Usually, the PSNR SA gives a higher PSNR than the PSNR on a single image, even if the average image is less sharp. We also observed that in the case of Diffusion EM, increasing the number of samples lowers the PSNR but improves all the other metrics. Averaging several samples is also possible with methods such as Blind-DPS, the main difference is that in our approach, all the samples have the same guidance at each diffusion step since we estimate a single kernel for all the samples. In Blind-DPS, all the samples have their respective kernels.

### Table 2. Influence of the number of samples used to estimate the E-step in Fast EM IIGDM. The image PSNR is computed on the first image of the batch.

<table>
<thead>
<tr>
<th>n-samples</th>
<th>Runtime</th>
<th>PSNR</th>
<th>PSNR SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=1</td>
<td>1min.50sec</td>
<td>23.4</td>
<td>23.4</td>
</tr>
<tr>
<td>n=4</td>
<td>2min.50sec</td>
<td>23.21</td>
<td>23.43</td>
</tr>
<tr>
<td>n=16</td>
<td>9min.10sec</td>
<td>23.09</td>
<td>23.37</td>
</tr>
<tr>
<td>Fast Diffusion EM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=1</td>
<td>9sec</td>
<td>25.66</td>
<td>25.66</td>
</tr>
<tr>
<td>n=4</td>
<td>15sec</td>
<td>25.74</td>
<td>26.14</td>
</tr>
<tr>
<td>n=16</td>
<td>55sec</td>
<td>25.75</td>
<td>26.16</td>
</tr>
</tbody>
</table>

Figure 5. Visual comparison on out-of-distribution images. The score network is trained on FFHQ dataset while we test on DIV2K.

(a) LR (b) HR (c) Blind DPS (d) Fast EM DPS (e) Fast EM IIGDM
References


