Solving Inverse Problems by Joint Posterior Maximization with Autoencoding Prior

Mario González, Andrés Almansa, Pauline Tan

Solving Inverse Problems by Joint Posterior Maximization with Autoencoding Prior*

Mario González†, Andrés Almansa ‡, and Pauline Tan §

Abstract. In this work we address the problem of solving ill-posed inverse problems in imaging where the prior is a variational autoencoder (VAE). Specifically we consider the decoupled case where the prior is trained once and can be reused for many different log-concave degradation models without retraining. Whereas previous MAP-based approaches to this problem lead to highly non-convex optimization algorithms, our approach computes the joint (space-latent) MAP that naturally leads to alternate optimization algorithms and to the use of a stochastic encoder to accelerate computations. The resulting technique (JPMAP) performs Joint Posterior Maximization using an Autoencoding Prior. We show theoretical and experimental evidence that the proposed objective function is quite close to bi-convex. Indeed it satisfies a weak bi-convexity property which is sufficient to guarantee that our optimization scheme converges to a stationary point. We also highlight the importance of correctly training the VAE using a denoising criterion, in order to ensure that the encoder generalizes well to out-of-distribution images, without affecting the quality of the generative model. This simple modification is key to providing robustness to the whole procedure. Finally we show how our joint MAP methodology relates to more common MAP approaches, and we propose a continuation scheme that makes use of our JPMAP algorithm to provide more robust MAP estimates. Experimental results also show the higher quality of the solutions obtained by our JPMAP approach with respect to other non-convex MAP approaches which more often get stuck in spurious local optima.

Key words. Image Restoration, Inverse Problems, Bi-convex Optimization, Bayesian Statistics, Generative Models, Variational Auto-encoders

AMS subject classifications. 68U10, 65K10, 65D18, 68T05, 90C26, 90C25, 90C30,

1. Introduction.

General inverse problems in imaging consist in estimating a clean image $\boldsymbol{x} \in \mathbb{R}^d$ from noisy, degraded measurements $\boldsymbol{y} \in \mathbb{R}^m$. In many cases the degradation model is known and its conditional density

$$p_{Y|X}\left(\boldsymbol{y} \,|\, \boldsymbol{x}\right) \propto e^{-F(\boldsymbol{x},\boldsymbol{y})}$$

is log-concave with respect to $\boldsymbol{x}$. To illustrate this, let us consider the case where the negative log-conditional is quadratic with respect to $\boldsymbol{x}$

$$F(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{1.1}$$

*The authors would like to sincerely thank Mauricio Delbracio, José Lezama and Pablo Musé for their help, their insightful comments, and their continuous support throughout this project.

Funding: This work was funded by ECOS Sud Project U17E04, by the French-Uruguayan Institute of Mathematics and Interactions (IFUMI), by CSIC I+D (Uruguay) and by ANII (Uruguay) under Grant 11 FCE_1_2017_1_135458. Computer experiments for this work ran on a Titan Xp GPU donated by NVIDIA, as well as on HPC resources from GENCI-IDRIS (Grant 2020-AD011011641).

†DMEL, CenUR RN, Universidad de la República, Salto, Uruguay (mgonzalez@unorte.edu.uy, http://dmel.interior.edu.uy/mario-gonzalez/).

‡MAP5, CNRS & Université de Paris, France (andres.almansa@parisdescartes.fr).

§LJLL, Sorbonne Université, Paris, France (pauline.tan@sorbonne-universite.fr)

This boils down to a linear degradation model that takes into account degradations such as, white Gaussian noise, blur, and missing pixels. When the degradation operator $\boldsymbol{A}$ is non-invertible or ill-conditioned, or when the noise level σ is high, obtaining a good estimate of $\boldsymbol{x}$ requires prior knowledge on the image, given by $p_X(\boldsymbol{x}) \propto e^{-G(\boldsymbol{x})}$. Variational and Bayesian methods in imaging are extensively used to derive MMSE or MAP estimators,

$$\hat{\boldsymbol{x}}_{\text{MAP}} = \arg\max_{\boldsymbol{x}} p_{X|Y}(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \{F(\boldsymbol{x}, \boldsymbol{y}) + G(\boldsymbol{x})\} \tag{1.2}$$

based on explicit priors like total variation [42, 8, 31, 35], or learning-based priors like patch-based Gaussian mixture models [59, 53, 48].

Since deep neural networks (NN) showed their superiority in image classification tasks [28] researchers started to look for ways to use this tool to solve inverse problems too. The most straightforward attempts employed neural networks as *regressors* to learn a risk minimizing mapping $\boldsymbol{y} \mapsto \boldsymbol{x}$ from many examples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ either agnostically [16, 54, 56, 18, 45, 17] or including the degradation model in the network architecture via unrolled optimization techniques [21, 10, 14, 19].

The main drawback of neural networks regression is that they require to retrain the neural network each time a single parameter of the degradation model changes. To avoid the need for retraining, another family of approaches seek to *decouple* the NN-based learned image prior from the degradation model. A popular approach within this methodology are *plug & play* methods. Instead of directly learning the log-prior $-\log p_X(\boldsymbol{x}) = G(\boldsymbol{x}) + C$, these methods seek to learn an approximation of its gradient ∇G [5, 4] or proximal operator prox_G [51, 32, 55, 9, 25, 43], by replacing it by a denoising NN. Then, these approximations are used in an iterative optimization algorithm to find the corresponding MAP estimator in Equation (1.2) or more generally some sort of consensus equilibrium among the data fitting term and the priors [7].

Plug & play approaches became very popular because of their convenience but obtaining convergence guarantees under realistic conditions is quite challenging. Indeed, the actual prior is unknown, and the existence of a density whose gradient or proximal operator is well approximated by a neural denoiser is most often not guaranteed [40], unless the denoiser is retrained with specific constraints [43, 22, 46]. In our experience these modifications may result in sub-optimal solutions, or in constraints like strongly convex data fit [43] to ensure convergence. A similar plug & play approach is proposed in [41] by using a denoiser D_σ to construct an explicit regularizer $G(\boldsymbol{x}) = \boldsymbol{x}^T(\boldsymbol{x} - D_\sigma(\boldsymbol{x}))$ (which is not necessarily a $-\log$ density) in such a way that ∇G can be conveniently computed, but only under very restrictive conditions [47] that exclude state of the art neural denoisers.

1.1. Maximum a Posteriori meets Generative Models.

It is tempting to use neural networks to learn an explicit prior for images. For instance one could use a generative adversarial network (GAN) to learn a generative model for $X = \mathsf{G}(Z)$ with $Z \sim N(0, I)$ a latent variable. The generative model induces a prior on X via the push-forward measure $p_X = \mathsf{G}\sharp p_Z$, which following [33, section 5] can be developed as

$$p_X(\boldsymbol{x}) = \frac{p_Z\left(\mathsf{G}^{-1}(\boldsymbol{x})\right)}{\sqrt{\det S(\mathsf{G}^{-1}(\boldsymbol{x}))}} \delta_{\mathcal{M}}(\boldsymbol{x})$$

where $S = \left(\frac{\partial \mathsf{G}}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial \mathsf{G}}{\partial \boldsymbol{z}}\right)$ is the squared Jacobian and the manifold $\mathcal{M} = \{\boldsymbol{x} : \exists \boldsymbol{z},\, \boldsymbol{x} = \mathsf{G}(\boldsymbol{z})\}$ represents the image of the generator G. With such a prior p_X, the $\boldsymbol{x}$-optimization (1.2) required to obtain $\hat{\boldsymbol{x}}_{\text{MAP}}$ becomes intractable (in general), for various reasons:

- the computation of $\det S$,
- the inversion of G, and
- the hard constraint $\boldsymbol{x} \in \mathcal{M}$.

These operations are are all memory and/or computationally intensive, except when they are partially addressed by the use of a normalizing flow like in [23, 52].

Current attempts to use such a generative model as a prior, like the one proposed by Bora et al. [6] for GANs, circumvent these difficulties by performing an optimization on $\boldsymbol{z}$ (in the latent domain) instead of $\boldsymbol{x}$. Instead of solving Equation (1.2), they solve

$$\begin{aligned} \hat{\boldsymbol{z}}_{\text{MAP}} &= \arg\max_{\boldsymbol{z}} \left\{ p_{Y|X}\left(\boldsymbol{y} \,|\, \mathsf{G}(\boldsymbol{z})\right) p_Z\left(\boldsymbol{z}\right) \right\} \\ &= \arg\min_{\boldsymbol{z}} \left\{ F(\mathsf{G}(\boldsymbol{z}), \boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2 \right\}, \end{aligned} \tag{1.3}$$

by assuming a standard Gaussian prior. This problem is much more tractable, and the corresponding $\boldsymbol{x}$-estimate is obtained as

$$\hat{\boldsymbol{x}}_{\text{MAP}-\boldsymbol{z}} = \mathsf{G}(\hat{\boldsymbol{z}}_{\text{MAP}}). \tag{1.4}$$

As we show in appendix B.2, this new estimator does not necessarily coincide with $\hat{\boldsymbol{x}}_{\text{MAP}}$ but it does correspond to the MAP-estimator of $\boldsymbol{x}$ after the change of variable $\boldsymbol{x} = \mathsf{G}(\boldsymbol{z})$, namely

$$\hat{\boldsymbol{x}}_{\text{MAP}-\boldsymbol{z}} = \mathsf{G}\left(\arg\max_{\boldsymbol{z}} \left\{ p_{Z|Y}\left(\boldsymbol{z} \,|\, \boldsymbol{y}\right) \right\} \right).$$

Despite the convenience of this $\boldsymbol{z}-$MAP approach with respect to the $\boldsymbol{x}-$MAP approach, convergence guarantees for this optimization problem are of course extremely difficult to establish, as confirmed by experimental results presented in Section 3.

A common technique to solve difficult optimization problems like the one in Equation (1.3) is to use (Half Quadratic) splitting methods

$$\hat{\boldsymbol{x}}_{\beta} = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{z}} \underbrace{\left\{ F(\boldsymbol{x}, \boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{x} - \mathsf{G}(\boldsymbol{z})\|^2 + \frac{1}{2}\|\boldsymbol{z}\|^2 \right\}}_{J_{1,\beta}(\boldsymbol{x},\boldsymbol{z})} \tag{1.5}$$

combined with a continuation scheme, namely:

$$\hat{\boldsymbol{x}}_{\text{MAP}-\boldsymbol{z}} = \lim_{\beta \to \infty} \hat{\boldsymbol{x}}_{\beta}. \tag{1.6}$$

The convergence of the continuation scheme in the last line is a standard result in Γ-convergence (see [13] and appendix C). The corresponding splitting algorithm is presented in Algorithm 1.1.

Algorithm 1.1 MAP-$\boldsymbol{z}$ splitting

Require: Measurements $\boldsymbol{y}$, Initial condition $\boldsymbol{x}_0$
Ensure: $\hat{\boldsymbol{x}} = \mathsf{G}\left(\arg\max_{\boldsymbol{z}} p_{Z|Y}\left(\boldsymbol{z} \,|\, \boldsymbol{y}\right)\right)$

1: **for** $k := 0$ **to** $k_{\max}$ **do**
2: $\quad \beta := \beta_k$
3: $\quad$ **for** $n := 0$ **to** maxiter **do**
4: $\quad\quad \boldsymbol{z}_{n+1} := \arg\min_{\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x}_n, \boldsymbol{z})$ // Nonconvex
5: $\quad\quad \boldsymbol{x}_{n+1} := \arg\min_{\boldsymbol{x}} J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z}_{n+1})$ // Quadratic
6: $\quad$ **end for**
7: $\quad \boldsymbol{x}_0 := \boldsymbol{x}_{n+1}$
8: **end for**
9: **return** $\boldsymbol{x}_{n+1}$

However, unlike most cases of HQS which include a linear constraint between the two variables, this splitting algorithm still contains (line 4) a difficult non-convex optimization problem. [1]

1.2. Proposed method: Joint MAP$_{x,z}$. In this work we propose to addresses this challenge by substituting the difficult non-convex sub-problem by a local quadratic approximation provided by the encoder of a variational autoencoder.

Indeed, as we show in Section 2, a variational autoencoder allows to interpret the splitting Equation (1.5) as the negative logarithm of the joint posterior density $p_{X,Z|Y}\left(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y}\right)$. Therefore, solving Equation (1.5) amounts to compute a joint MAP$_{x,z}$ estimator that we denote by $\hat{\boldsymbol{x}}^{\beta}_{\mathrm{MAP}_{\boldsymbol{x},\boldsymbol{z}}}$. Moreover if the same joint conditional density $p_{X,Z|Y}\left(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y}\right)$ is decomposed in a different manner, it leads to an approximate expression that makes use of the encoder, and is quadratic in $\boldsymbol{z}$. If this approximation is good enough then the maximization of the joint log-posterior becomes a bi-concave optimization problem or approximately so. And in that case, an extension of standard bi-convex optimization results [20] shows that the algorithm converges to a stationary point.

We also highlight the importance of correctly training the VAE in such a way that the encoder generalizes well to noisy values of $\boldsymbol{x}$ outside of the support of $p_X\left(\boldsymbol{x}\right)$. This can be achieved by training the VAE to reconstruct their clean inputs with noise injected at the input level, as proposed by Im et al. [24]. We observe that this modified training does not degrade the quality of the generative model, but makes our quasi-biconvex optimization procedure much more robust.

Finally we show that a continuation scheme allows to obtain the MAP$_{\boldsymbol{z}}$ estimator as the limit of a series of joint MAP$_{\boldsymbol{x},\boldsymbol{z}}$ optimizations. This continuation scheme, in addition to the quasi-bi-convex optimization, and the initialisation heuristic provided by the denoising encoder leads to a much more robust non-convex optimization scheme which more often converges to the right critical point than a straightforward gradient descent of the MAP$_{\boldsymbol{z}}$ model.

The remainder of this paper is organized as follows. In Section 2 we derive a model for

[1] In another context a primal-dual optimization algorithm was proposed to solve a similar optimization problem [2], but this approach was not explored in the context where G is a generative model.

the joint conditional posterior distribution of space and latent variables $\boldsymbol{x}$ and $\boldsymbol{z}$, given the observation $\boldsymbol{y}$. This model makes use of a generative model, more precisely a VAE with Gaussian decoder. We then propose an alternate optimization scheme to maximize for the joint posterior model, and state convergence guarantees. Section 3 presents first a set of experiments that illustrates the convergence properties of the optimization scheme. We then test our approach on classical image inverse problems, and compare its performance with state-of-the-art methods. Concluding remarks are presented in Section 4.

2. From Variational Autoencoders to Joint Posterior Maximization. Recently, some generative models based on neural networks have shown their capability to approximate the complex image distribution in a data-driven fashion. In particular, *Variational Autoencoders (VAE)* [27] combine variational inference to approximate unknown posterior distributions of latent variable models with the ability of neural networks to learn such approximations.

Consider a graphical model $\boldsymbol{z} \to \boldsymbol{x}$ in which we assume that a latent variable $\boldsymbol{z}$ is responsible of the observed image $\boldsymbol{x}$. For example, in an image of a handwritten digit we can imagine which digit is represented in the image, width, angle (and so on) as latent variables. We choose a generative model

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_Z(\boldsymbol{z})$$

where $p_Z(\boldsymbol{z})$ is some simple distribution (which we can easily sample from) and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the approximation of the probability distribution of $\boldsymbol{x}$ given $\boldsymbol{z}$ parameterized by a neural network (with weights θ) known as *stochastic decoder*.

The intractability of $p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_Z(\boldsymbol{z})\, d\boldsymbol{z}$ is related to the posterior distribution $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ by

$$p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_Z(\boldsymbol{z})}{p_\theta(\boldsymbol{x})}. \tag{2.1}$$

The *variational inference* approach consists in approximating this posterior with another model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ which, in our case, is another neural network with parameters ϕ, called a *stochastic encoder*.

Following [27], we consider the *Evidence Lower BOund (ELBO)* as

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) := \log p_\theta(\boldsymbol{x}) - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z}|\boldsymbol{x})) \leq \log p_\theta(\boldsymbol{x}) \tag{2.2}$$

where KL is the Kullback-Leibler divergence. Thus, given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of image samples, maximizing the averaged ELBO on $\mathcal{D}$ means maximizing $\log p_\theta(\mathcal{D})$ which is the maximum likelihood estimator of weights θ *and* minimizing $KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z}|\boldsymbol{x}))$ which enforces the approximated posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to be similar to the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$.

It can be shown [27] that the ELBO can be rewritten as

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_Z(\boldsymbol{z})). \tag{2.3}$$

The first term in (2.3) is a *reconstruction loss* similar to the one of plain autoencoders: it enforces that the code $\boldsymbol{z} \sim q_\phi(\cdot|\boldsymbol{x})$ generated by the encoder q_ϕ can be used by the decoder

p_θ to reconstruct the original input $\boldsymbol{x}$. The second term is a *regularization term* that enforces the distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ of the latent code $\boldsymbol{z}$ (given $\boldsymbol{x}$) to be close to the prior distribution $p_Z(\boldsymbol{z})$. It is common to choose an isotropic Gaussian as the prior distribution of the latent code:

$$p_Z(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z} \,|\, 0, I) \propto e^{-\|\boldsymbol{z}\|^2/2}$$

and a Gaussian encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z} \,|\, \mu_\phi(\boldsymbol{x}), \Sigma_\phi(\boldsymbol{x}))$, so that the KL divergence in (2.3) is straightforward to compute. For the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ a Gaussian decoder is the most common choice and as we will see we benefit from that.

2.1. Learning approximations vs. encoder approximations. In this work we construct an image prior using a Variational Autoencoder (VAE). Like any machine learning tool VAEs make different kinds of approximations. Let's distinguish two types of approximations that shall be important in the sequel:

Learning approximation: The ideal prior p_X^* can only be approximated by our VAE due to its architectural constraints, finite complexity, truncated optimization algorithms, finite amount of data and possible biases in the data. Due to all these approximations, after learning we have only access to an approximate prior $p_X \approx p_X^*$. VAEs give access to this approximate prior p_X via a generative model: taking samples of a latent variable Z with known distribution $\mathcal{N}(0, I)$ in $\mathbb{R}^l$ (with $l \ll d$), and feeding these samples through a learned decoder network, we obtain samples of $X \sim p_X$. The approximate prior itself

$$p_X(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z})\, p_Z(\boldsymbol{z})\, d\boldsymbol{z} \tag{2.4}$$

is intractable because it requires computing an integral over all possible latent codes $\boldsymbol{z}$. However the approximate joint distribution is readily accessible

$$p_{X,Z}(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})\, p_Z(\boldsymbol{z})$$

thanks to $p_{X|Z}(\boldsymbol{x} \,|\, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})$ which is provided by the decoder network.

Encoder approximation: In the previous item we considered the VAE as a generative model without making use of the encoder network. The encoder network

$$\tilde{p}_{Z|X}(\boldsymbol{z} \,|\, \boldsymbol{x}) := q_\phi(\boldsymbol{z}|\boldsymbol{x}) \approx p_{Z|X}(\boldsymbol{z} \,|\, \boldsymbol{x})$$

is introduced as an approximate way to solve the intractability of $p_{Z|X}(\boldsymbol{z} \,|\, \boldsymbol{x}) = p_\theta(\boldsymbol{z}|\boldsymbol{x})$ (which is related to the intractability of $p_\theta(\boldsymbol{x})$ as observed in equation (2.1)).
Using the encoder network we can provide an alternative approximation for the joint distribution

$$\tilde{p}_{X,Z}(\boldsymbol{x}, \boldsymbol{z}) := q_\phi(\boldsymbol{z}|\boldsymbol{x})\, p_X(\boldsymbol{x}) \approx p_{X,Z}(\boldsymbol{x}, \boldsymbol{z})$$

which shall be useful in the sequel.

Put another way, the ideal joint distribution $p_{X,Z}^*$ is inaccessible, but can be approximated in two different ways:

The first expression denoted $p_{X,Z}(\boldsymbol{x},\boldsymbol{z})$ only uses the decoder and is only affected by the *learning approximation*

$$p^*_{X,Z}(\boldsymbol{x},\boldsymbol{z}) \approx p_{X,Z}(\boldsymbol{x},\boldsymbol{z}) := p_\theta(\boldsymbol{x}|\boldsymbol{z})\, p_Z(\boldsymbol{z})\,.$$

The second expression denoted $\tilde{p}_{X,Z}(\boldsymbol{x},\boldsymbol{z})$ uses both encoder and decoder and is affected both by the *learning approximation* and by the *encoder approximation*

$$p_{X,Z}(\boldsymbol{x},\boldsymbol{z}) \approx \tilde{p}_{X,Z}(\boldsymbol{x},\boldsymbol{z}) := q_\phi(\boldsymbol{z}|\boldsymbol{x})\, p_X(\boldsymbol{x})$$

In the following subsection we shall forget about the ideal prior p^*_X and joint distribution $p^*_{X,Z}$ which are both inaccessible. Instead we accept p_X (with its learning approximations) as our prior model which shall guide all our estimations. The approximation symbol shall be reserved to expressions that are affected by the encoder approximation *in addition to* the learning approximation.

2.2. Variational Autoencoders as Image Priors. To obtain the Maximum a Posteriori estimator (MAP), we could plug in the approximate prior p_X in equation (1.2), but this leads to a numerically difficult problem to solve due to the intractability of p_X. Instead, we propose to maximize the joint posterior $p_{X,Z|Y}(\boldsymbol{x},\boldsymbol{z}\,|\,\boldsymbol{y})$ over $(\boldsymbol{x},\boldsymbol{z})$ which is equivalent to minimizing

$$\begin{aligned} J_1(\boldsymbol{x},\boldsymbol{z}) &:= -\log p_{X,Z|Y}(\boldsymbol{x},\boldsymbol{z}\,|\,\boldsymbol{y}) \\ &= -\log p_{Y|X,Z}(\boldsymbol{y}\,|\,\boldsymbol{x},\boldsymbol{z})\, p_\theta(\boldsymbol{x}\,|\,\boldsymbol{z}) p_Z(\boldsymbol{z}) \\ &= F(\boldsymbol{x},\boldsymbol{y}) + H_\theta(\boldsymbol{x},\boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2. \end{aligned} \tag{2.5}$$

Note that the first term is quadratic in $\boldsymbol{x}$ (assuming (1.1)), the third term is quadratic in $\boldsymbol{z}$ and all the difficulty lies in the coupling term $H_\theta(\boldsymbol{x},\boldsymbol{z}) = -\log p_\theta(\boldsymbol{x}\,|\,\boldsymbol{z})$. For Gaussian decoders [27], the latter can be written as

$$\begin{aligned} H_\theta(\boldsymbol{x},\boldsymbol{z}) = \frac{1}{2}\Big(& d\log(2\pi) + \log\det\Sigma_\theta(\boldsymbol{z}) \\ & + \|\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x}-\boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2\Big). \end{aligned} \tag{2.6}$$

which is also convex in $\boldsymbol{x}$. Hence, minimization with respect to $\boldsymbol{x}$ takes the convenient closed form:

$$\begin{aligned} \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x},\boldsymbol{z}) = & \left(\boldsymbol{A}^T\boldsymbol{A} + \sigma^2\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{z})\right)^{-1} \\ & \times \left(\boldsymbol{A}^T\boldsymbol{y} + \sigma^2\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{z})\boldsymbol{\mu}_\theta(\boldsymbol{z})\right). \end{aligned} \tag{2.7}$$

Unfortunately the coupling term H and hence J_1 is a priori non-convex in $\boldsymbol{z}$. As a consequence the $\boldsymbol{z}$-minimization problem

$$\arg\min_{\boldsymbol{z}} J_1(\boldsymbol{x},\boldsymbol{z}) \tag{2.8}$$

is a priori more difficult. However, for Gaussian encoders, VAEs provide an approximate expression for this coupling term which is quadratic in $\boldsymbol{z}$. Indeed, given the equivalence

$$\begin{aligned} p_\theta(\boldsymbol{x} \,|\, \boldsymbol{z})\, p_Z\,(\boldsymbol{z}) &= p_{X,Z}\,(\boldsymbol{x}, \boldsymbol{z}) \\ &= p_{Z|X}\,(\boldsymbol{z} \,|\, \boldsymbol{x})\; p_X\,(\boldsymbol{x}) \\ &\approx q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x})\, p_X\,(\boldsymbol{x}) \end{aligned}$$

we have that

$$H_\theta(\boldsymbol{x}, \boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2 \approx K_\phi(\boldsymbol{x}, \boldsymbol{z}) - \log p_X\,(\boldsymbol{x})\,. \tag{2.9}$$

where $K_\phi(\boldsymbol{x}, \boldsymbol{z}) = -\log q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x})$. Therefore, this new coupling term becomes

$$\begin{aligned} K_\phi(\boldsymbol{x}, \boldsymbol{z}) &= -\log \mathcal{N}\,(\boldsymbol{z}; \boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\Sigma}_\phi(\boldsymbol{x})) \\ &= \frac{1}{2}\big[l \log(2\pi) + \log\det \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) \\ &\quad x + \|\boldsymbol{\Sigma}_\phi^{-1/2}(\boldsymbol{x})(\boldsymbol{z} - \boldsymbol{\mu}_\phi(\boldsymbol{x}))\|^2\big], \end{aligned}$$

which is quadratic in $\boldsymbol{z}$. This provides an approximate expression for the energy (2.5) that we want to minimize, namely

$$J_2(\boldsymbol{x}, \boldsymbol{z}) := F(\boldsymbol{x}, \boldsymbol{y}) + K_\phi(\boldsymbol{x}, \boldsymbol{z}) - \log p_X\,(\boldsymbol{x}) \approx J_1(\boldsymbol{x}, \boldsymbol{z}). \tag{2.10}$$

This approximate functional is quadratic in $\boldsymbol{z}$, and minimization with respect to this variable yields

$$\arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}). \tag{2.11}$$

2.3. Alternate Joint Posterior Maximization. The previous observations suggest to adopt an alternate scheme to minimize $-\log p_{X,Z|Y}\,(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y})$ in order to solve the inverse problem. We begin our presentation by a simple version of the proposed algorithm, which aims at managing the case where the approximation of J_1 by J_2 is exact (at least in the sense given in Assumption 1 below); then we propose an adaptation for the more realistic non-exact case and we explore its convergence properties.

When $J_1 = J_2$ is a biconvex function, Algorithm 2.1 is known as *Alternate Convex Search*. Its behavior has been studied in [20, 1]. Here we shall consider the following (strong) assumption, which includes the strictly bi-convex case ($J_1 = J_2$):

Assumption 1. *For any $\boldsymbol{x}$, if $\boldsymbol{z}^*$ is a global minimizer of $J_2(\boldsymbol{x}, \cdot)$, then $\boldsymbol{z}^*$ is a global minimizer of $J_1(\boldsymbol{x}, \cdot)$.*

The proposed alternate minimization takes the simple and fast form depicted in Algorithm 2.1, which can be shown to converge to a stationary point of J_1, as stated in Proposition 2.1 below. Note that the minimization in step 2 of Algorithm 2.1 does not require the knowledge of the unknown term $-\log p_X\,(\boldsymbol{x})$ in Equation (2.10) since it does not depend on $\boldsymbol{z}$.

The convergence analysis of the proposed schemes requires some general assumptions on the functions J_1 and J_2 :

Algorithm 2.1 Joint posterior maximization - exact case

Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial condition $\boldsymbol{x}_0$
Ensure: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{x},\boldsymbol{z}} p_{X,Z|Y}(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y})$

for $n := 0$ **to** maxiter **do**
  $\boldsymbol{z}_{n+1} := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$ // Quadratic approx
  $\boldsymbol{x}_{n+1} := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}_{n+1})$ // Quadratic
end for
return $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

Assumption 2. *$J_1(\cdot, \boldsymbol{z})$ is convex and admits a minimizer for any $\boldsymbol{z}$. Moreover, J_1 is coercive and continuously differentiable.*

The convergence property of Algorithm 2.1 will be investigated in a wider framework below (Proposition 2.1). Note that all the properties required in Assumption 2 are satisfied if we use a differentiable activation function like the Exponential Linear Unit (ELU) [11] with $\alpha = 1$, instead of the more common ReLU activation function. More details can be found in Appendix A.

2.4. Approximate Alternate Joint Posterior Maximization. When the autoencoder approximation in (2.10) is not exact (Assumption 1), the energy we want to minimize in Algorithm 2.1, namely J_1 may not decrease. To ensure the decay, some additional steps can be added. Noting that the approximation provided by J_2 provides a fast and accurate heuristic to initialize the minimization of J_1, an alternative scheme is proposed in Algorithm 2.2.

Algorithm 2.2 Joint posterior maximization - approximate case

Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial conditions $\boldsymbol{x}_0, \boldsymbol{z}_0$
Ensure: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{x},\boldsymbol{z}} p_{X,Z|Y}(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y})$

for $n := 0$ **to** maxiter **do**
  $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$ // Equation (2.11)
  $\boldsymbol{z}^2 := \mathrm{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}^1$
  $\boldsymbol{z}^3 := \mathrm{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}_n$
  for $i := 1$ **to** 3 **do**
    $\boldsymbol{x}^i := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^i)$ // Equation (2.7)
  end for
  $i^* := \arg\min_{i \in \{1,2,3\}} J_1(\boldsymbol{x}^i, \boldsymbol{z}^i)$
  $(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) := (\boldsymbol{x}^{i^*}, \boldsymbol{z}^{i^*})$
end for
return $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

In Algorithm 2.2, GD is a gradient descent scheme such that for any starting point $\boldsymbol{z}_0$, the output $\boldsymbol{z}^+$ satisfies

$$\frac{\partial J_1}{\partial \boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{z}^+) = 0 \quad \text{and} \quad J_1(\boldsymbol{x}, \boldsymbol{z}^+) \leq J_1(\boldsymbol{x}, \boldsymbol{z}_0)$$

Hence, one can consider for instance a gradient descent scheme which finds a local minimizer

of $J_1(\boldsymbol{x}, \cdot)$ starting from $\boldsymbol{z}_0$.

Our experiments with Algorithm 2.2 (Section 3.3) show that during the first few iterations (where the approximation provided by J_2 is good enough) $\boldsymbol{z}^1$ and $\boldsymbol{z}^2$ reach convergence faster than $\boldsymbol{z}^3$. After a critical number of iterations the opposite is true (the initialization provided by the previous iteration is better than the J_2 approximation, and $\boldsymbol{z}^3$ converges faster).

These observations suggest that a faster execution, with the same convergence properties, can be achieved by the variant in Algorithm 2.3, which avoids the costly computation of $\boldsymbol{z}^2$ and $\boldsymbol{z}^3$ when unnecessary. Hence, in practice, we will use Algorithm 2.3 rather than Algorithm 2.2. However, Algorithm 2.2 provides a useful tool for diagnostics. Indeed, the comparison of the evaluation of $J_1(\boldsymbol{x}^i, \boldsymbol{z}^i)$ for $i = 1, 2, 3$ performed in step 8 permits to assess the evolution of the approximation of J_1 by J_2.

Algorithm 2.3 Joint posterior maximization - approximate case (faster version)

Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial condition $\boldsymbol{x}_0$, iterations $n_1 \leq n_2 \leq n_{\max}$

Ensure: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{x},\boldsymbol{z}} p_{X,Z|Y}(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{y})$

1: **for** $n := 0$ **to** $n_{\max}$ **do**
2: done := FALSE
3: **if** $n < n_1$ **then**
4: $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$ // Equation (2.11)
5: $\boldsymbol{x}^1 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^1)$ // Equation (2.7)
6: **if** $J_1(\boldsymbol{x}^1, \boldsymbol{z}^1) < J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$ **then**
7: $i^* := 1$ // J_2 is good enough
8: done := TRUE
9: **end if**
10: **end if**
11: **if not** done **and** $n < n_2$ **then**
12: $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$
13: $\boldsymbol{z}^2 := \mathrm{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}^1$
14: $\boldsymbol{x}^2 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^2)$ // Equation (2.7)
15: **if** $J_1(\boldsymbol{x}^2, \boldsymbol{z}^2) < J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$ **then**
16: $i^* := 2$ // J_2 init is good enough
17: done := TRUE
18: **end if**
19: **end if**
20: **if not** done **then**
21: $\boldsymbol{z}^3 := \mathrm{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}_n$
22: $\boldsymbol{x}^3 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^3)$ // Equation (2.7)
23: $i^* := 3$
24: **end if**
25: $(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) := (\boldsymbol{x}^{i^*}, \boldsymbol{z}^{i^*})$
26: **end for**
27: **return** $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

Algorithm 2.3 is still quite fast when J_2 provides a sufficiently good approximation, since in that case the algorithm chooses $i^* = 1$, and avoids any call to the iterative gradient descent algorithm. Even if we cannot give a precise definition of what *sufficiently good* means, the sample comparison of K_ϕ and H_θ as functions of $\boldsymbol{z}$, displayed in Figure 3(a), shows that the approximation is fair enough in the sense that it preserves the global structure of J_1. The same behavior was observed for a large number of random tests.

Note that Algorithm 2.1 is a particular instance of Algorithm 2.3 in the case where Assumption 1 holds, and $n_1 = n_2 = 0$ and if grad descent gives a global minimizer of the considered function (in this case, the computation of $\boldsymbol{z}^1$, $\boldsymbol{z}^2$, are skipped and only $\boldsymbol{z}^3$ is computed).

Proposition 2.1 (Convergence of Algorithm 2.3). *Let $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ be a sequence generated by Algorithm 2.3. Under Assumption 2 we have that:*

1. *The sequence $\{J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ converges monotonically when $n \to \infty$.*
2. *The sequence $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ has at least one accumulation point.*
3. *All accumulation points of $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ are stationary points of J_1 and they all have the same function value.*

Proof. Since we are interested in the behaviour for $n \to \infty$, we assume $n > n_2$ in Algorithm 2.3.

1. Since $n > n_2$ the algorithm chooses $i^* = 3$ and $\boldsymbol{z}_{n+1} = \boldsymbol{z}^3$. According to the definition of grad descent, one has

$$J_1(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

and by optimality one has

$$J_1(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}).$$

Hence, since J_1 is coercive (thus, lowerbounded), Statement 1 is straightforward.

2. Thanks to the coercivity of J_1, the sequences $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ and $\{(\boldsymbol{x}_n, \boldsymbol{z}_{n+1})\}$ are bounded, thus admit an accumulation point.

3. Using Fermat's rule and the definition of grad descent, one has

$$\frac{\partial J_1}{\partial \boldsymbol{z}}(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}) = 0 \quad \text{and} \quad \frac{\partial J_1}{\partial \boldsymbol{x}}(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) = 0.$$

Let $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ be an accumulation point of $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$. By double extraction, one can find two subsequences such that

$$(\boldsymbol{x}_{n_j+1}, \boldsymbol{z}_{n_j+1}) \to (\boldsymbol{x}^*, \boldsymbol{z}^*) \text{ and } (\boldsymbol{x}_{n_j}, \boldsymbol{z}_{n_j+1}) \to (\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*)$$

By continuity of ∇J_1, one gets that

$$\frac{\partial J_1}{\partial \boldsymbol{z}}(\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*) = 0 \quad \text{and} \quad \frac{\partial J_1}{\partial \boldsymbol{x}}(\boldsymbol{x}^*, \boldsymbol{z}^*) = 0$$

In particular, the convexity of $J_1(\cdot, \boldsymbol{z}^*)$ and Assumption 2 ensure that $\boldsymbol{x}^*$ is a global minimizer of $J_1(\cdot, \boldsymbol{z}^*)$. Besides, the inequalities proved in Point 1 above show that

$$J_1(\boldsymbol{x}^*, \boldsymbol{z}^*) = J_1(\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*) = \lim_{n \to \infty} J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

that is, $\hat{\boldsymbol{x}}^*$ is also a global minimizer of $J_1(\cdot, \boldsymbol{z}^*)$. Hence, we get

$$\frac{\partial J_1}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*) = 0$$

namely $(\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*)$ is a stationary point of J_1. ∎

Remark 2.2. Note that if $n_1 = n_2 = \infty$ we cannot assume that $i^* = 3$. In that case statements 1 and 2 are still valid but the third statement is not. The reason is that for $i^* \in \{1, 2\}$ we cannot guarantee the chain of inequalities

$$J_1(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

but only

$$J_1(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_n).$$

This is consistent with the design of the algorithm where iterations $n < n_2$ serve as an heuristic to guide the algorithm to a sensible critical point. However, convergence to a critical point is only guaranteed by the final iterations $n > n_2$.

2.5. MAP-z as the limit case for $\beta \to \infty$. If one wishes to compute the MAP-$\boldsymbol{z}$ estimator instead of the joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ from the previous section, one has two options:

1. Use your favorite gradient descent algorithm to solve equation (1.3).
2. Use Algorithm 2.3 to solve a series of joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ problems with increasing values of $\beta \to \infty$ as suggested in Algorithm 1.1.

In the experimental section we show that the second approach most often leads to a better optimum.

In practice, in order to provide a stopping criterion for Algorithm 1.1 and to make a sensible choice of β-values we reformulate Algorithm 1.1 as a constrained optimization problem

$$\underset{\boldsymbol{x},\boldsymbol{z}\,:\,\|\mathsf{G}(\boldsymbol{z})-\boldsymbol{x}\|^2 \leq \varepsilon}{\arg\min} F(\boldsymbol{x}, \boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2.$$

The corresponding Lagrangian form is

$$\max_{\beta} \min_{\boldsymbol{x},\boldsymbol{z}} F(\boldsymbol{x}, \boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2 + \beta \left(\|\mathsf{G}(\boldsymbol{z}) - \boldsymbol{x}\|^2 - \varepsilon\right)^+ \tag{2.12}$$

and we use the exponential multiplier method [49] to guide the search for the optimal value of β (see Algorithm 2.4)

3. Experimental results.

3.1. AutoEncoder and dataset. In order to test our joint prior maximization model we first train a Variational Autoencoder like in [27] on the training data of MNIST handwritten digits [29].

The *stochastic encoder* takes as input an image $\boldsymbol{x}$ of $28 \times 28 = 784$ pixels and produces as an output the mean and (diagonal) covariance matrix of the Gaussian distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, where the latent variable $\boldsymbol{z}$ has dimension 8. The architecture of the encoder is composed of

Algorithm 2.4 MAP-$\boldsymbol{z}$ as the limit of joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$.

Require: Measurements $\boldsymbol{y}$, Tolerance ε, Rate $\rho > 0$, Initial β_0, Initial $\boldsymbol{x}_0$, Iterations $0 \leq n_1 \leq n_2 \leq n_{\max}$

Ensure: $\arg\min_{\boldsymbol{z}\,:\,\|\mathsf{G}(\boldsymbol{z})-\boldsymbol{x}\|^2\leq\varepsilon} F(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2$.

$\beta := \beta_0$
$\boldsymbol{x}^0, \boldsymbol{z}^0 :=$ Algorithm 2.3 starting from $\boldsymbol{x} = \boldsymbol{x}_0$ with $\beta, n_1, n_2, n_{\max}$.
converged := FALSE
$k := 0$
while not converged **do**
  $\boldsymbol{x}^{k+1}, \boldsymbol{z}^{k+1} :=$ Algorithm 2.3 starting from $\boldsymbol{x} = \boldsymbol{x}^k$ with β and $n_1 = n_2 = 0$
  $C = \|\mathsf{G}(\boldsymbol{z}^{k+1}) - \boldsymbol{x}^{k+1}\|^2 - \varepsilon$
  $\beta := \beta \exp(\rho C)$
  converged := $(C \leq 0)$
  $k := k + 1$
end while
return $\boldsymbol{x}^k, \boldsymbol{z}^k$

3 fully connected layers with ELU activations (to preserve continuous differentiability). The sizes of the layers are as follows: $784 \to 500 \to 500 \to (8+8)$. Note that the output is of size $8+8$ in order to encode the mean and diagonal covariance matrix, both of size 8.

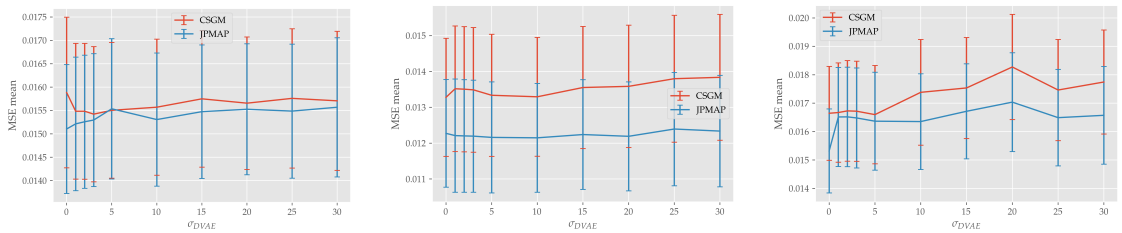
The *stochastic decoder* takes as an input the latent variable $\boldsymbol{z}$ and outputs the mean and covariance matrix of the Gaussian distribution $p_\theta(x|z)$. Following [12] we chose here an isotropic covariance $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}) = \gamma^2 I$ where $\gamma > 0$ is trained, but independent of $\boldsymbol{z}$. This choice simplifies the minimization problem (2.8), because the term $\det \Sigma_\theta(\boldsymbol{z})$ (being constant) has no effect on the $\boldsymbol{z}$-minimization. The architecture of the decoder is also composed of 3 fully connected layers with ELU activations (to preserve continuous differentiability). The sizes of the layers are as follows: $8 \to 500 \to 500 \to 784$. Note that the covariance matrix is constant, so it does not augment the size of the output layer which is still $784 = 28 \times 28$ pixels.

We also trained a VAE on CelebA [30] images cropped to $64 \times 64 \times 3$. The latent dimension in this case was set to 32. We choose a DCGAN-like [38] CNN architecture as encoder and a symmetrical one as decoder with ELU activations, batch normalization and isotropic covariance as before. For more details, see the code[2].

We train these architectures using PyTorch [34] with batch size 128 and Adam algorithm for 200 epochs with learning rate 0.0001 and rest of the parameters as default.

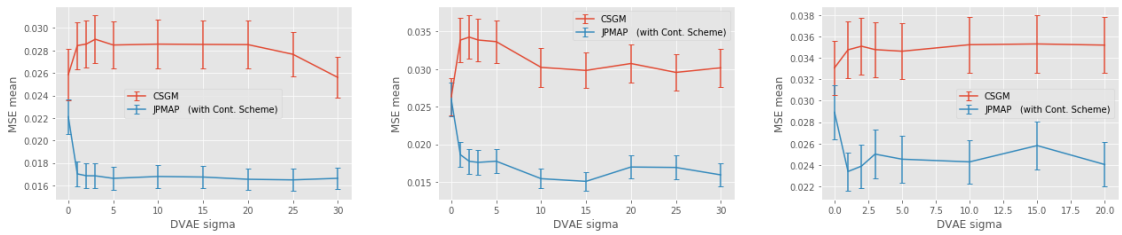
3.2. Need to train the VAE with a denoising criterion. It should be noted that when training our Variational Autoencoder we should be more careful that usual. Indeed in the most widespread applications of VAEs they are only used as a generative model or as a way to interpolate between images that are close to $\mathcal{M}$, *i.e.* the image of the generator $\boldsymbol{\mu}_\theta$. For such

[2]Code available at https://github.com/mago876/JPMAP.



(a) Denoising (b) Compressed Sensing (c) Inpainting

Figure 1. *Evaluating the quality of the generative model as a function of* σ_{DVAE}. *On (a) Denoising (Gaussian noise* $\sigma = 150$*), (b) Compressed Sensing (*$\sim 10.2\%$ *measurements, noise* $\sigma = 10$*) and (c) Inpainting (80% of missing pixels, noise* $\sigma = 10$*). Results of both algorithms are computed on a batch of 50 images and initialising on ground truth* $\boldsymbol{x}^*$ *(for CSGM we use* $\boldsymbol{z}_0 = \boldsymbol{\mu}_\phi(\boldsymbol{x}^*)$*).*



(a) Denoising (b) Compressed Sensing (c) Inpainting

Figure 2. *Evaluating the effectiveness of JPMAP vs CGSM as a function of* σ_{DVAE} *(same setup of Figure 1). Without a denoising criterion* $\sigma_{\text{DVAE}} = 0$ *the JPMAP algorithm may provide wrong guesses* $\boldsymbol{z}^1$ *when applying the encoder in step 2 of Algorithm 2.2. For* $\sigma_{\text{DVAE}} > 0$ *however, the alternating minimization algorithm can benefit from the robust initialization heuristics provided by the encoder, and it consistently converges to a better local optimum than the simple gradient descent in CSGM.*

applications it is sufficient to train the encoder $\boldsymbol{\mu}_\phi$, $\boldsymbol{\Sigma}_\phi$ on a training set that is restricted to $\mathcal{M}$.

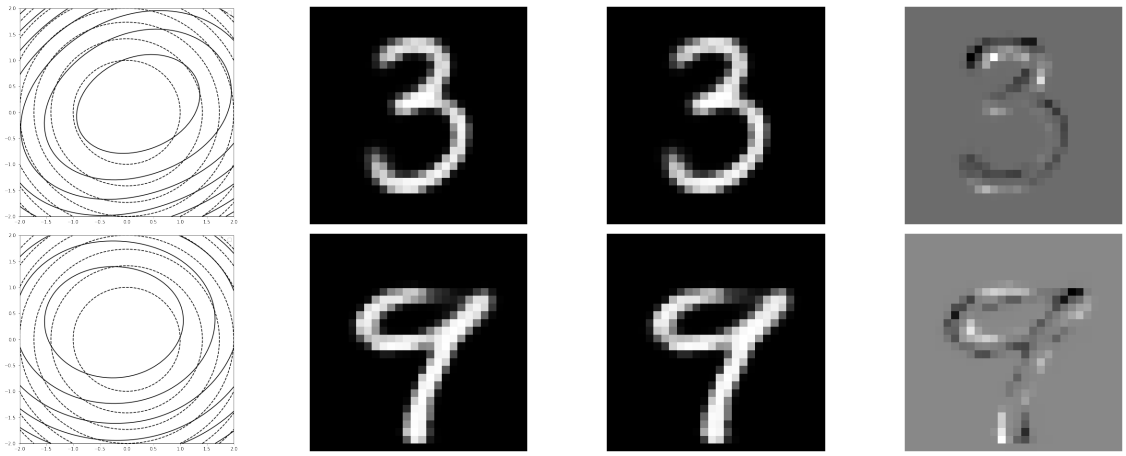
In our case however, we need the encoder to provide sensible values even when its input $\boldsymbol{x}$ is quite far away from $\mathcal{M}$: the encoder has to actually fulfil two functions at the same time:

1. (Approximately) project $\boldsymbol{x}$ to its closest point in $\mathcal{M}$, and
2. compute the encoding of this projected value (which should be the same as the encoding of the original $\boldsymbol{x}$.

Traditional VAE training procedures do not ensure that the encoder generalizes well to $\boldsymbol{x} \notin \mathcal{M}$. In order to ensure this generalization ability we adopt the training procedure of the DVAE (Denoising VAE) proposed by Im et al. [24], which consists in adding various realizations of zero-mean Gaussian noise of variance σ^2_{DVAE} to the samples $\boldsymbol{x}$ presented to the encoder, while still requiring the decoder to match the noiseless value, *i.e.* we optimize the parameters in such a way that

$$\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\tilde{\boldsymbol{x}})) \approx \boldsymbol{x} \tag{3.1}$$

where $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \sigma_{\text{DVAE}}\varepsilon$ and $\varepsilon \sim \mathcal{N}(0, I)$ for all $\boldsymbol{x}$ in the training set and for many realizations



(a) Encoder approximation (b) Decoded exact optimum (c) Decoded approx. optimum (d) Difference (b)-(c)

Figure 3. *Encoder approximation:* (a) *Contour plots of* $-\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2$ *and* $-\log q_\phi(\boldsymbol{z}|\boldsymbol{x})$ *for a fixed* $\boldsymbol{x}$ *and for a random 2D subspace in the* $\boldsymbol{z}$ *domain (the plot shows* $\pm 2\boldsymbol{\Sigma}_\phi^{1/2}$ *around* $\boldsymbol{\mu}_\phi$*). Observe the relatively small gap between the true posterior* $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ *and its variational approximation* $q_\phi(\boldsymbol{z}|\boldsymbol{x})$*. This figure shows some evidence of partial* $\boldsymbol{z}$*-convexity of* J_1 *around the minimum of* J_2*, but it does not show how far is* $\boldsymbol{z}^1$ *from* $\boldsymbol{z}^2$*.* (b) *Decoded exact optimum* $\boldsymbol{x}_1 = \boldsymbol{\mu}_\theta\left(\arg\max_{\boldsymbol{z}} p_\theta(\boldsymbol{x}|\boldsymbol{z}) e^{\frac{1}{2}\|\boldsymbol{z}\|^2}\right)$*.* (c) *Decoded approximate optimum* $\boldsymbol{x}_2 = \boldsymbol{\mu}_\theta\left(\arg\max_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x})\right)$*.* (d) *Difference betweeen (b) and (c).*

of ε.

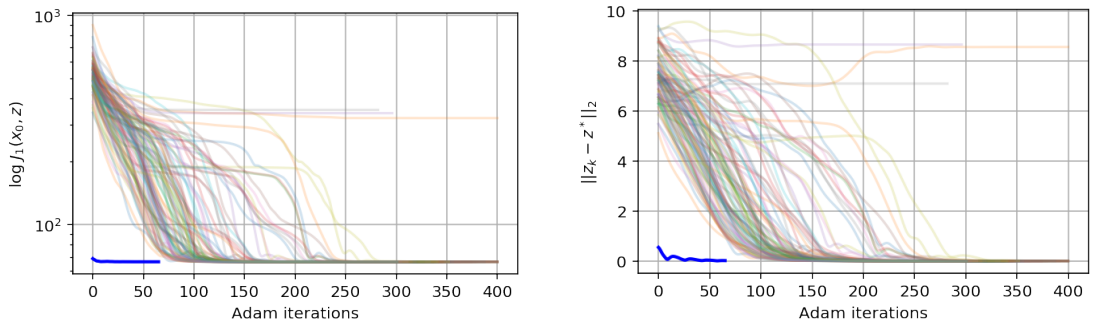
More specifically, if we take a corruption model $p(\tilde{\boldsymbol{x}}\,|\,\boldsymbol{x})$ like above, it can be shown [24] that

$$\tilde{\mathcal{L}}_{\theta,\phi}(\boldsymbol{x}) = \mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\tilde{\boldsymbol{x}})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - KL(q_\phi(\boldsymbol{z}|\tilde{\boldsymbol{x}}) \,||\, p_Z(\boldsymbol{z}))\right] \tag{3.2}$$

is an alternative ELBO of 2.3. In practice, using Monte Carlo for estimating the expectation $\mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}$ in 3.2, we only need to add noise to $\boldsymbol{x}$ before passing it to the encoder q_ϕ during training, as mentioned in 3.1.

Our experiments with this denoising criterion confirm the observation by Im et al. [24] that it does not degrade the quality of the generative model, as long as σ_{DVAE} is not too large (see Figure 1). As a side benefit, however, we obtain a more robust encoder that generalizes well for values of $\boldsymbol{x}$ that are not in $\mathcal{M}$ but within a neighbourhood of size $\approx \sigma_{\text{DVAE}}$ around $\mathcal{M}$. This side benefit, which was not the original intention of the DVAE training algorithm in [24] is nevertheless crucial for the success of our algorithm as demonstrated in Figure 2. The same figure shows that as long as $\sigma_{\text{DVAE}} \geq 5$ its value does not significantly affect the performance. In the sequel we use $\sigma_{\text{DVAE}} = 15$.

3.3. Effectiveness of the encoder as a fast minimizer. Proposition 2.1 shows that the proposed alternate minimization scheme in Algorithm 2.3 converges to a stationary point of J_1. And so does the gradient descent scheme in [6]. Since both algorithms have to deal with



(a) Energy evolution, initializing with $\mathcal{N}(0, I)$. (b) Distance to the optimum at each iteration of (a).

Figure 4. Effectiveness of the encoder approximation: *We take $\boldsymbol{x}_0$ from the* test *set of MNIST and minimize $J_1(\boldsymbol{x}_0, \boldsymbol{z})$ with respect to $\boldsymbol{z}$ using gradient descent from random Gaussian initializations $\boldsymbol{z}_0$. The blue thick curve represents the trajectory if we initialize at the encoder approximation $\boldsymbol{z}^1 = \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_0, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}_0)$.* (a)*: Plots of the energy iterates $J_1(\boldsymbol{x}_0, \boldsymbol{z}_k)$.* (b)*: ℓ^2 distances of each trajectory with respect to the global optimum $\boldsymbol{z}^*$.* Conclusion: *Observe that the encoder initialization allows much faster convergence both in energy and in $\boldsymbol{z}$, and it avoids the few random initializations that lead to a wrong stationary point different from the unique global minimizer.*

non-convex energies, they both risk to converge to spurious local minima. Also both algorithms solve essentially the same model when the variance γ of the coupling term tends to zero.

If our algorithm shows better performance (see next subsection), it is mainly because it relies on a previously trained VAE in two fundamental ways: *(i)* to avoid getting trapped in spurious local minima and *(ii)* to accelerate performance during the initial iterations ($n < n_{\min}$). These two features are only possible if the autoencoder approximation is good enough and if the encoder is able to provide good initializations for the non-convex $\boldsymbol{z}$- optimization subproblem in line 13 of Algorithm 2.3.

Figures 3 and 4 illustrate these two properties of our VAE. We do so by selecting a random $\boldsymbol{x}_0$ from MNIST test set and by computing $\boldsymbol{z}^*(\boldsymbol{z}_0) := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_0, \boldsymbol{z})$ with different initial values $\boldsymbol{z}_0$. These experiments were performed using the ADAM minimization algorithm with learning rate equal to 0.01. Figure 4(a) shows that $\boldsymbol{z}^*(\boldsymbol{z}_0)$ reaches the global optimum for most (but not all) initializations $\boldsymbol{z}_0$. Indeed from 200 random initializations $\boldsymbol{z}_0 \sim \mathcal{N}(0, I)$, 195 reach the same global minimum, whereas 5 get stuck at a higher energy value. However these 5 initial values have energy values $J_1(\boldsymbol{x}_0, \boldsymbol{z}_0) \gg J_1(\boldsymbol{x}_0, \boldsymbol{z}^1)$ far larger than those of the encoder initialization $\boldsymbol{z}^1 = \boldsymbol{\mu}_\phi(\boldsymbol{x}_0)$, and are thus never chosen by Algorithm 2.3. The encoder initialization $\boldsymbol{z}^1$ on the other hand provides much faster convergence to the global optimum.

In addition, this experiment shows that we cannot assume $\boldsymbol{z}$-convexity: The presence of plateaux in the trajectories of many random initializations as well as the fact that a few initializations do not lead to the global minimum indicates that J_1 may not be everywhere convex with respect to $\boldsymbol{z}$. However, in contrast to classical works on alternate convex search, our approach adopts weaker assumptions and does not require convexity on $\boldsymbol{z}$ to prove con-

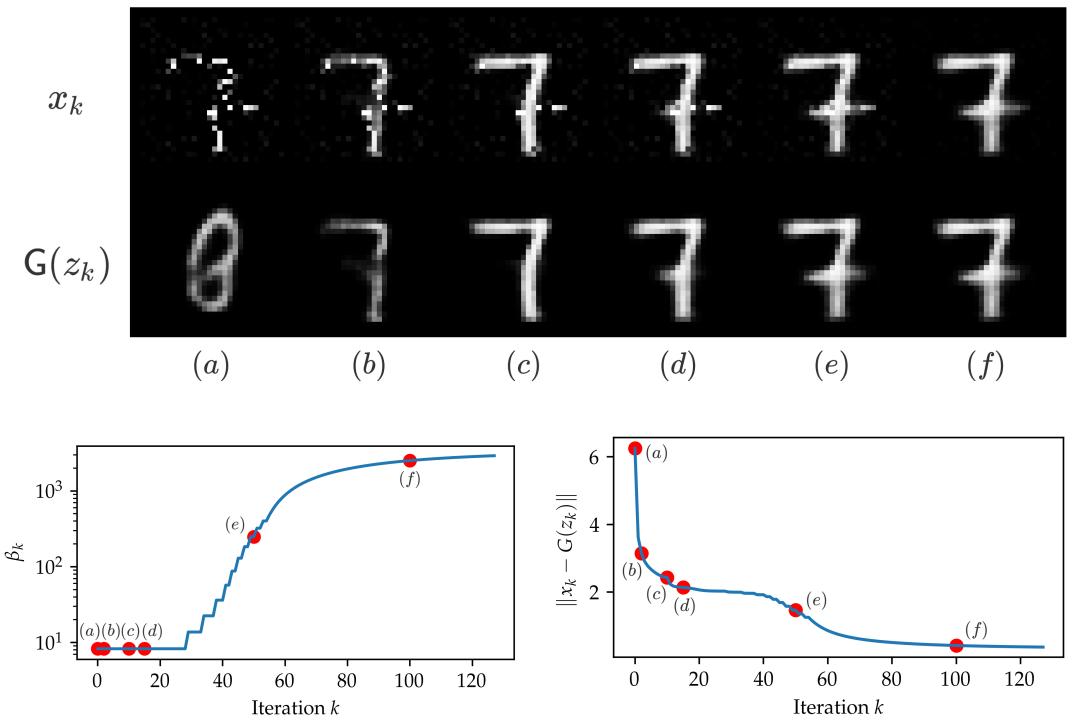


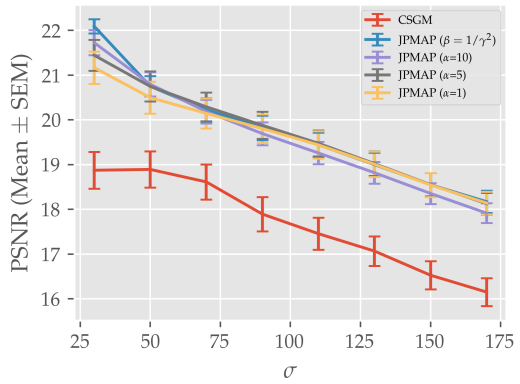
Figure 5. Evolution of Algorithm 2.4. *In this inpainting example, JPMAP starts with the initialization in* (a)*. During first iterations* $(b)-(d)$ *where* β_k *is small,* $\boldsymbol{x}_k$ *and* $\mathsf{G}(\boldsymbol{z}_k)$ *start loosely approaching each other at a coarse scale, and* $\boldsymbol{x}_k$ *only fills missing pixels with the ones of* $\mathsf{G}(\boldsymbol{z}_k)$ *(in particular the noise of* $\boldsymbol{y}$ *is still present). By increasing* β_k *in* $(e)-(f)$ *we enforce* $\|\mathsf{G}(\boldsymbol{z}_k)-\boldsymbol{x}_k\|^2 \leq \varepsilon$*. Here we set* $\epsilon = \left(\frac{3}{255}\right)^2 d$*, that is, MSE of 3 gray levels.*

vergence in Proposition 2.1.

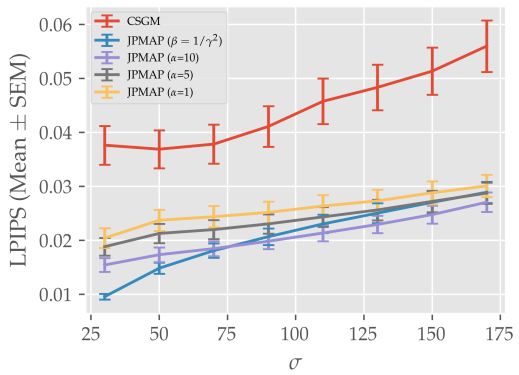
In Figure 4(b) we display the distances of each trajectory to the global optimum $\boldsymbol{z}^*$ (taken as the median over all initializations $\boldsymbol{z}_0$ of the final iterates $\boldsymbol{z}^*(\boldsymbol{z}_0)$); note that this optimum is always reached, which suggests that $\boldsymbol{z} \mapsto J_1(\boldsymbol{x}_0, \boldsymbol{z})$ has a unique global minimizer in this case. Finally, Figure 3 shows that the encoder approximation is quite good both in the latent space (Figure 3(a)) and in image space (Figures 3(b) and 3(c)). It also shows that the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is pretty close to log-concave near the maximum of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$.

3.4. Image restoration experiments. *Choice of* $\boldsymbol{x}_0$*:* In the previous section, our validation experiments used a random $\boldsymbol{x}_0$ from the data set as initialization. When dealing with an image restoration problem, Algorithms 2.2 and 2.3 require an initial value of $\boldsymbol{x}_0$ to be chosen. In all experiments we choose this initial value as the simplest possible inversion algorithm, namely the regularized pseudo-inverse of the degradation matrix:

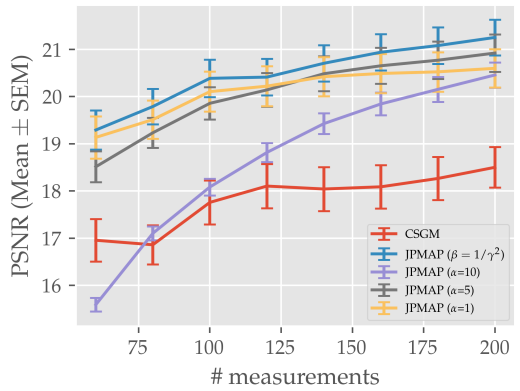
$$\boldsymbol{x}_0 = A^\dagger y = (A^T A + \varepsilon Id)^{-1} A^T y.$$



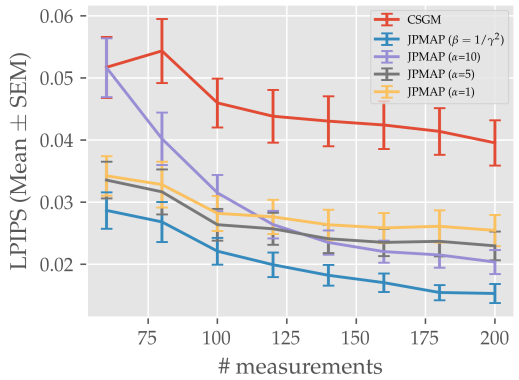
(a) Denoising (PSNR)



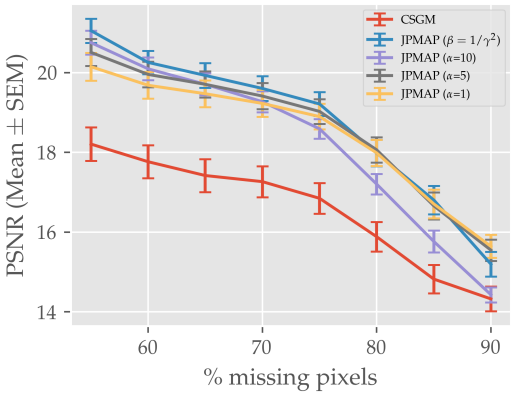
(b) Denoising (LPIPS)



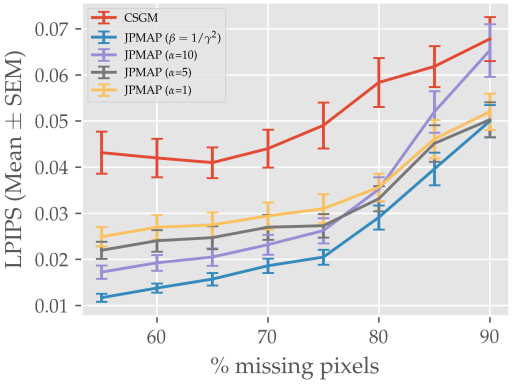
(c) Compressed Sensing (PSNR)



(d) Compressed Sensing (LPIPS)

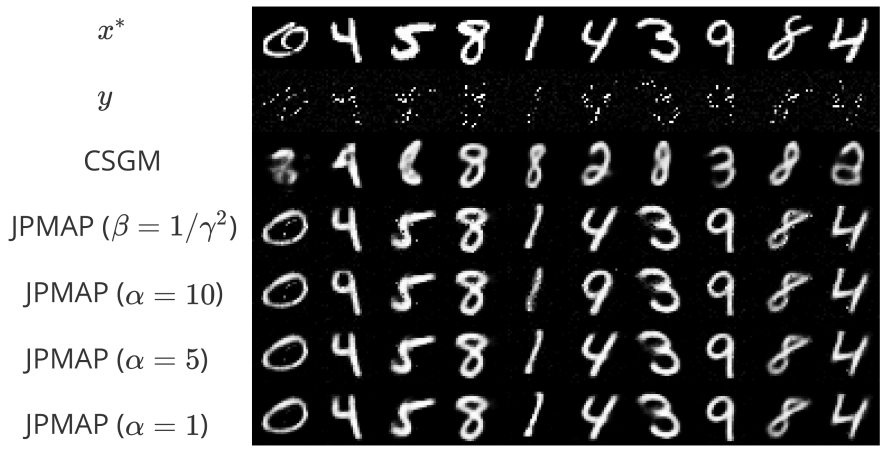


(e) Inpainting (PSNR)

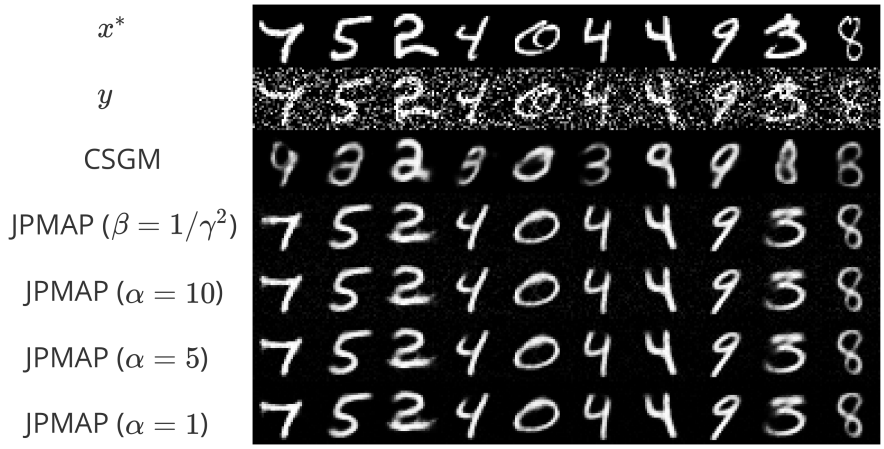


(f) Inpainting (LPIPS)

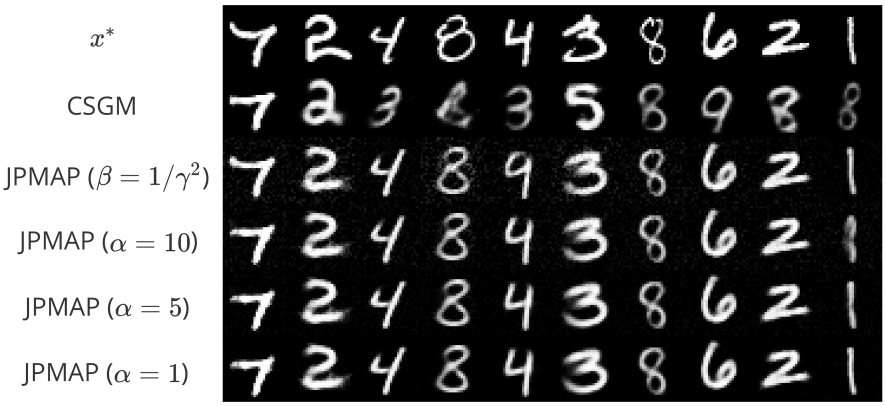
Figure 6. Evaluating the effectiveness of Algorithm 2.3 (fixed β) and Algorithm 2.4 *for different values of* $\epsilon = \left(\frac{\alpha}{255}\right)^2 n$, *with* $\sigma_{\text{DVAE}} = 15$ *(metrics were computed on a batch of 100 test images). For PSNR, higher is better and for LPIPS, lower is better.*



(a) Results on inpainting.



(b) Results on denoising



(c) Results on compressed sensing.

Figure 7. Experimental results on MNIST. *Comparison with CSGM algorithm [6].* (a) *Some selected results from the inpainting experiment with 80% of missing pixels and Gaussian noise with* $\sigma = 10$. *From top to bottom: original image* $\boldsymbol{x}^*$, *corrupted image* $\boldsymbol{y}$, *restored by [6], and restored images* $\hat{\boldsymbol{x}}$ *by our framework with Algorithm 2.3 (fixed* β*) and Algorithm 2.4 for different values of* $\epsilon = \left(\frac{\alpha}{255}\right)^2$ *d.*(b) *Same as (a) in a denoising experiment with Gaussian noise std* $\sigma = 110/255$ (c) *Same as (a) in a compressed sensing experiment with* $m = 140$ *random measurements and Gaussian noise with* $\sigma = 10/255$ *(without showing the second row).* Conclusion: *Our algorithm performs consistently better than CSGM. In addition our algorithm gets less often stuck in spurious local minima.*

(a) Results of inpainting on CelebA. (b) Reconstructions $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x}))$.

Figure 8. (a) *Some preliminary results on CelebA: 80% of missing pixels, noise std* $\sigma = 10/255$. *From top to bottom: original image* $\boldsymbol{x}^*$, *corrupted image* $\tilde{\boldsymbol{x}}$, *restored by CSGM [6], restored image* $\hat{\boldsymbol{x}}$ *by our framework.* (b) *Reconstructions* $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x}))$ *(even columns) for some test samples* $\boldsymbol{x}$ *(odd columns), showing the over-regularization of data manifold imposed by the trained vanilla VAE. As a consequence,* $-\log p_{Z|Y}(\boldsymbol{z} \,|\, \boldsymbol{y})$ *does not have as many local minima and then a simple gradient descent yields almost the same result as JPMAP (except on third column of (a)).*

Choice of n_1 *and* n_2*:* After a few runs of Algorithm 2.2 we find that in most cases, during the first 10 or 20 iterations $\boldsymbol{z}^1$ decreases the energy with respect to the previous iteration, and this value depends on the inverse problems (for example, for denoising is smaller than for compressed sensing). But after at most 150 iterations the autoencoder approximation is no longer good enough and we need to perform gradient descent on $\boldsymbol{z}_n$ in order to further decrease the energy. Based on these findings we set $n_1 = 25$ and $n_2 = 150$ in Algorithm 2.3 for all experiments. Note that we could also choose $n_1 = n_2 = n_{\max}$, since in all our experiments we observed that the algorithm auto-regulates itself, evolving from $i^* = 1$ in the first few dozen iterations to $i^* = 3$ when it is close to convergence. Choosing a finite value for n_1 and n_2 is only needed to ensure that $i^* = 3$ when $n \to \infty$, which is a necessary condition to prove statement 3 of Proposition 2.1.

Figure 5 shows the evolution of $\boldsymbol{x}_k$ and $\mathsf{G}(\boldsymbol{z}_k)$ from Algorithm 2.4 in an inpainting example. Here we can see how the exponential multiplier method in Equation 2.12 updates the values of β_k to ensure $\|\mathsf{G}(\boldsymbol{z}_k) - \boldsymbol{x}_k\|^2 \leq \varepsilon$.

Figure 6 shows the results of inpainting, denoising and compressed sensing experiments on MNIST using the proposed approach, for differents degradation levels. The metrics used are PSNR and LPIPS[3] [57]. Figure 7 displays some selected results. For comparison we provide also the result of CSGM [6] with $\lambda = \sigma^2$ which uses the same generative model to compute the MAP estimator as in Equation (1.3), but does not make use of the encoder.

Figure 6 shows the systematic superiority of the JPMAP algorithm with respect to the gradient descent in CGSM, in a wide variety of inverse problems on MNIST. The gap between

[3]MNIST images were zero-padded to 32×32 because LPIPS does not accept 28×28 images.

both grows larger for the most ill-posed inverse problems. So, the proposed method significantly outperforms CSGM in the vast majority of cases because our alternate minimization scheme does not get stuck in local optima, as CSGM does.

In the case of CelebA, we did not observe as much difference between JPMAP and CSGM as on MNIST. In Figure 8(a) the restorations on an inpainting problem (80% of missing pixels) are very similar to each other, but blurry. Also, although this problem is very ill-posed, both algorithms impressively find a solution $\boldsymbol{z}^*$ very close to the code $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ of the ground truth image $\boldsymbol{x}$, except for the third column where CSGM converges to a local minimum.

We hypothesize that, as CelebA is a substantially more complex dataset than MNIST, a simple model like vanilla VAE is over-regularizing the manifold of samples (underfitting problem). In particular, because of the spectral bias [39] the learned manifold perhaps only contains low-frequency approximations of the true images as we can see in the reconstructions $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x}))$ of test samples (see Figure 8(b)). This may cause that the posterior $p_{Z|Y}(\boldsymbol{z}\,|\,\boldsymbol{y})$ does not have as many local minima. With a more realistic generative model which better represents the true data manifold, we expect the objective function $-\log p_{Z|Y}(\boldsymbol{z}\,|\,\boldsymbol{y})$ to exhibit a much larger number of local minima, thus making it more difficult to optimize by a simple gradient descent scheme. In that situation the proposed JPMAP method would more clearly show its advantages.

4. Conclusions and Future work. In this work we presented a new framework to solve inverse problems with a convex data-fitting term and a non-convex regularizer learned in the latent space via variational autoencoders. Unlike similar approaches like CSGM [6] which learns the prior based on generative models, our approach is based on a generalization of alternate convex search to quasi-biconvex functionals. This quasi-biconvexity is the result of considering the joint posterior distribution of latent and image spaces. As a result, the proposed approach provides stronger convergence guarantees. Experiments on denoising, inpainting and compressed sensing confirm this, since our approach gets stuck much less often in spurious local minima than CSGM, which is simply based on gradient descent of a highly non-convex functional. This leads to restored images which are significantly better in terms of PSNR and LPIPS.

JPMAP vs related Plug & Play *approaches.* When compared to other decoupled *plug & play* approaches that solve inverse problems using NN-based priors, our approach is constrained in different ways:

(a) In a certain sense our approach is *less constrained* than existing decoupled approaches since we do not require to retrain the NN-based denoiser to enforce any particular property to ensure convergence: Ryu et al. [43] requires the denoiser's residual operator to be non-expansive, and Gupta et al. [22], Shah and Hegde [46] require the denoiser to act as a projector. The effect of these modifications to the denoiser on the quality of the underlying image prior has never been studied in detail and chances are that such constraints degrade it. Our method only requires a variational autoencoder without any further constraints, and the quality and expressiveness of this prior can be easily checked by sampling and reconstruction experiments. Checking the quality of the prior is a much more difficult task for Ryu et al. [43], Gupta et al.

[22], Shah and Hegde [46] which rely on an implicit prior, and do not provide a generative model.

(b) Unlike [43] which requires the data-fitting term $F(\boldsymbol{x})$ to be *strongly convex* to ensure convergence, our method admits weakly convex and ill-posed data-fitting terms like missing pixels, compressed sensing and non-invertible blurring for instance.

(c) On the other hand our method is *more constrained* in the sense that it relies on a generative model of a *fixed size*. Even if the generator and encoder are both convolutional neural networks, training and testing the same model on images of different sizes is a priori not possible because the latent space has a fixed dimension and a fixed distribution. As a future work we plan to explore different ways to address this limitation. The most straightforward way is to use our model to learn a prior of image patches of a fixed size and stitch this model via aggregation schemes like in EPLL [59] to obtain a global prior model for images of any size. Alternatively we can use hierarchical generative models like in [26, 50] or resizable ones like in [3, 52], and adapt our framework accordingly.

MAP-x or MAP-z or joint MAP-x-z. In this work we explored and clarified the tight relationships between joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ estimation, splitting and continuation schemes and the more common MAP-$\boldsymbol{z}$ estimator in the context of inverse problems with a generative prior. On the other hand MAP-$\boldsymbol{x}$ estimators (which are otherwise standard in bayesian imaging) remained largely unexplored in the context of generative priors, due to the optimization challenges they impose, until the recent work of Helminger et al. [23], Whang et al. [52] showed that a normalizing flow-based generative model allows to overcome those challenges and deems this problem tractable. Yet an extensive comparison of the advantages and weaknesses of these three families of estimators under the same prior model is still missing, and will be the subject of future work.

Extension to higher dimensional problems. The present paper provides a first proof of concept of our framework, on a very simple dataset (MNIST) with a very simple VAE. More experiments are needed to verify that the framework preserves its qualitative advantages on more high-dimensional datasets (like CelebA, CelebA-HQ, etc.), and a larger selection of inverse problems.

Generalizing our proposed method to much higher dimensional problems implies training much more complex generative models which can match the finer details and higher complexity of such data. We can still use over-simplified generative models in those cases, but our preliminary experiments suggest that in that situation, not only do we obtain relatively poor reconstructions, but the objective function associated to the MAP-$\boldsymbol{z}$ problem presents less spurious local-minima: as a consequence our proposed joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ is overkill in that configuration, and does not present such a great competitive advantage.

The big challenge of generalizing our proposed method to much higher dimensional problems is then to train sufficiently detailed and complex generative models. And in this area VAEs traditionally lagged behind GANs in terms of quality of the generated samples, the former producing in general more blurred samples. Nevertheless some studies [44] show that VAEs and Normalizing Flows produce more accurate representations of the probability distribution. In the short term our work can immediatly benefit from recent advances in VAE architectures [50, 12], and adversarial training for VAEs [36, 37, 58] that reach GAN-quality samples with the additional benefits of VAEs. With some additional effort GAN and flow-

based generative models can be augmented with an encoder network [15] and adapted by relaxation techniques to fit our quasi-bi-convex optimization framework.

Acknowledgments. We would like to sincerely thank Mauricio Delbracio, José Lezama and Pablo Musé for their help, their insightful comments, and their continuous support throughout this project.

References.

[1] Cecilia Aguerrebere, Andres Almansa, Julie Delon, Yann Gousseau, and Pablo Muse. A Bayesian Hyperprior Approach for Joint Image Denoising and Interpolation, With an Application to HDR Imaging. *IEEE Transactions on Computational Imaging*, 3(4):633–646, dec 2017. ISSN 2333-9403. doi: 10.1109/TCI.2017.2704439. URL https://nounsse.github.io/HBE_project/.

[2] Martin Benning, Florian Knoll, Carola Bibiane Schönlieb, and Tuomo Valkonen. Preconditioned ADMM with nonlinear operator constraint. *IFIP Advances in Information and Communication Technology*, 494:117–126, 2016. ISSN 18684238. doi: 10.1007/978-3-319-55795-3_10.

[3] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning Texture Manifolds with the Periodic Spatial GAN. *(ICML) International Conference on Machine Learning*, 1: 722–730, may 2017.

[4] Siavash Arjomand Bigdeli and Matthias Zwicker. Image Restoration using Autoencoding Priors. Technical report, 2017.

[5] Siavash Arjomand Bigdeli, Meiguang Jin, Paolo Favaro, and Matthias Zwicker. Deep Mean-Shift Priors for Image Restoration. In *(NIPS) Advances in Neural Information Processing Systems 30*, pages 763–772, sep 2017. URL http://papers.nips.cc/paper/6678-deep-mean-shift-priors-for-image-restoration.

[6] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *(ICML) International Conference on Machine Learning*, volume 2, pages 537–546. JMLR. org, 2017. ISBN 9781510855144.

[7] Gregery T Buzzard, Stanley H Chan, Suhas Sreehari, and Charles A Bouman. Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium. *SIAM Journal on Imaging Sciences*, 11(3):2001–2020, 2018.

[8] A Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20:89–97, 2004. doi: 10.1023/B:JMIV.0000011325.36760.1e.

[9] S. H. Chan, X. Wang, and O. A. Elgendy. Plug-and-play admm for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, March 2017. ISSN 2333-9403. doi: 10.1109/TCI.2016.2629286.

[10] Yunjin Chen and Thomas Pock. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2596743.

[11] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *(ICLR) International Conference on Learning Representations*, nov 2016.

[12] Bin Dai and David Wipf. Diagnosing and Enhancing VAE Models. *ICLR*, pages 1–42, 2019. URL https://openreview.net/forum?id=B1e0X3C9tQ.

[13] Gianni Dal Maso. *An Introduction to Γ-Convergence*. Birkhäuser Boston, Boston, MA, 1993. ISBN 978-1-4612-6709-6. doi: 10.1007/978-1-4612-0327-8. URL http://link.springer.com/10.1007/978-1-4612-0327-8.

[14] Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. Unrolled optimization with deep priors. 2017.

[15] Jeff Donahue and Karen Simonyan. Large Scale Adversarial Representation Learning. 2019. URL http://arxiv.org/abs/1907.02544.

[16] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[17] Hongyun Gao, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3848–3856, 2019.

[18] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 35(6):191, 2016.

[19] Davis Gilton, Greg Ongie, and Rebecca Willett. Neumann networks for inverse problems in imaging. 2019.

[20] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, nov 2007. ISSN 1432-2994. doi: 10.1007/s00186-007-0161-1.

[21] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 399–406. Omnipress, 2010.

[22] Harshit Gupta, Kyong Hwan Jin, Ha Q Nguyen, Michael T McCann, and Michael Unser. Cnn-based projected gradient descent for consistent ct image reconstruction. *IEEE transactions on medical imaging*, 37(6):1440–1453, 2018. doi: 10.1109/TMI.2018.2832656.

[23] Leonhard Helminger, Michael Bernasconi, Abdelaziz Djelouah, Markus Gross, and Christopher Schroers. Blind Image Restoration with Flow Based Priors. Technical report, sep 2020. URL http://arxiv.org/abs/2009.04583.

[24] Daniel Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 2059–2065. AAAI press, nov 2017.

[25] Ulugbek S Kamilov, Hassan Mansour, and Brendt Wohlberg. A plug-and-play priors approach for solving nonlinear imaging inverse problems. *IEEE Signal Processing Letters*, 24(12):1872–1876, 2017.

[26] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *(ICLR) International Conference on Learning Representations*, 10(2):327–331, oct 2017. URL https://openreview.net/forum?id=Hk99zCeAb.

[27] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *(ICLR) International Conference on Learning Representations*, number Ml, pages 1–14, dec 2013. ISBN 1312.6114v10. doi: 10.1051/0004-6361/201527329.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *(NIPS) Advances in neural information processing systems*, pages 1097–1105, 2012. ISSN 10495258.

[29] Yann Lecun, Leon Bottou, Yoshua Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 00189219. doi: 10.1109/5.726791.

[30] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[31] Cécile Louchet and Lionel Moisan. Posterior expectation of the total variation model: Properties and experiments. *SIAM Journal on Imaging Sciences*, 6(4):2640–2684, dec 2013. ISSN 19364954. doi: 10.1137/120902276.

[32] Tim Meinhardt, Michael Moller, Caner Hazirbas, and Daniel Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *(ICCV) International Conference on Computer Vision*, pages 1781–1790, 2017. doi: 10.1109/ICCV.2017.198. URL http://openaccess.thecvf.com/content_iccv_2017/html/Meinhardt_Learning_Proximal_Operators_ICCV_2017_paper.html.

[33] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. 2019.

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[35] Marcelo Pereyra. Proximal Markov chain Monte Carlo algorithms. *Statistics and Computing*, 26(4):745–760, jul 2016. ISSN 0960-3174. doi: 10.1007/s11222-015-9567-4. URL http://dx.doi.org/10.1007/s11222-015-9567-4.

[36] Yunchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *(NIPS) Advances in Neural Information Processing Systems*, pages 4331–4340, 2017.

[37] Yunchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *(NIPS) Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 4331–4340, 2017.

[38] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[39] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[40] Edward T Reehorst and Philip Schniter. Regularization by denoising: Clarifications and new interpretations. *IEEE Transactions on Computational Imaging*, 5(1):52–67, 2018. doi: 10.1109/TCI.2018.2880326.

[41] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[42] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992. ISSN 01672789. doi: 10.1016/0167-2789(92)90242-F.

[43] Ernest K. Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5546–5557, 2019. URL http://proceedings.mlr.press/v97/ryu19a.html.

[44] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing Generative Models via Precision and Recall. In *(NeurIPS) Neural Information Processing Systems*, may 2018.

[45] Eli Schwartz, Raja Giryes, and Alex M Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018.

[46] Viraj Shah and Chinmay Hegde. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4609–4613. IEEE, 2018.

[47] Suhas Sreehari, Singanallur V. Venkatakrishnan, Brendt Wohlberg, Gregery T. Buzzard, Lawrence F. Drummy, Jeffrey P. Simmons, and Charles A. Bouman. Plug-and-Play Priors for Bright Field Electron Tomography and Sparse Interpolation. *IEEE Transactions on Computational Imaging*, 2(4):1–1, 2016. ISSN 2333-9403. doi: 10.1109/TCI.2016.2599778.

[48] Afonso M. Teodoro, José M. Bioucas-Dias, and Mário A. T. Figueiredo. Scene-Adapted Plug-and-Play Algorithm with Guaranteed Convergence: Applications to Data Fusion in Imaging, jan 2018.

[49] Paul Tseng and Dimitri P. Bertsekas. On the convergence of the exponential multiplier method for convex programming. *Mathematical Programming*, 60(1-3):1–19, jun 1993. ISSN 00255610. doi: 10.1007/BF01580598. URL https://link.springer.com/article/10.1007/BF01580598.

[50] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33, 2020.

[51] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. Plug-and-Play priors for model based reconstruction. *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, pages 945–948, 2013. doi: 10.1109/GlobalSIP.2013.6737048.

[52] Jay Whang, Qi Lei, and Alexandros G. Dimakis. Compressed Sensing with Invertible Generative Models and Dependent Noise. In *NeurIPS deep-inverse workshop*, 2020.

[53] Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2011.

[54] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[55] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning Deep CNN Denoiser

Prior for Image Restoration. In *(CVPR) IEEE Conference on Computer Vision and Pattern Recognition*, pages 2808–2817. IEEE, apr 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.300. URL http://openaccess.thecvf.com/content_cvpr_2017/html/Zhang_Learning_Deep_CNN_CVPR_2017_paper.html.

[56] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[57] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[58] Zijun Zhang, Ruixiang Zhang, Zongpeng Li, Yoshua Bengio, and Liam Paull. Perceptual Generative Autoencoders. In *(ICLR) International Conference on Learning Representations*, pages 1–7, jun 2019. URL https://github.com/zj10/PGA.

[59] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486. IEEE, nov 2011. ISBN 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126278. URL http://people.csail.mit.edu/danielzoran/EPLLICCVCameraReady.pdf.

Appendix A. Properties of J_1. In this section, we establish that the objective function J_1 fulfills the assumptions required to prove the convergence of Algorithm 3, namely

- $J_1(\cdot, \boldsymbol{z})$ is convex for any $\boldsymbol{z}$;
- $J_1(\cdot, \boldsymbol{z})$ has a unique minimizer for any $\boldsymbol{z}$;
- J_1 is coercive;
- J_1 is continuously differentiable;

We recall that

$$\begin{aligned} J_1(\boldsymbol{x}, \boldsymbol{z}) = & \underbrace{\frac{1}{2\,\sigma^2} \|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}\|^2}_{F(\boldsymbol{x}, \boldsymbol{y})} \\ & + \underbrace{\frac{1}{2}\Big(Z_\theta(\boldsymbol{z}) + \|\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2\Big)}_{H_\theta(\boldsymbol{x}, \boldsymbol{z})} \\ & + \frac{1}{2}\|\boldsymbol{z}\|^2 \end{aligned}$$

where

$$Z_\theta(\boldsymbol{z}) = d\log(2\pi) + \log\det \boldsymbol{\Sigma}_\theta(\boldsymbol{z})$$

Thus, it is the sum of three non-negative terms.

A.1. Convexity and unicity of the minimizer of $J_1(\cdot, z)$. Let $\boldsymbol{z}$ be fixed. Then there exists a constant $C \in \mathbb{R}$ such that $\forall\, \boldsymbol{x}$

$$J_1(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{2\,\sigma^2} \|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}\|^2 + \|\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2 + C$$

Being the sum of two quadratic forms, $J_1(\cdot, \boldsymbol{z})$ is obviously twice differentiable. Its gradient is given by

$$\begin{aligned} \frac{\partial J_1}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{z}) = & \frac{1}{\sigma^2}\,\boldsymbol{A}^T(\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}) \\ & + 2\,(\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z}))^T\big(\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\big)\big) \end{aligned}$$

and its Hessian is

$$\mathrm{Hess}_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{\sigma^2}\,\boldsymbol{A}^T\boldsymbol{A} + 2\,(\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z}))^T\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})$$

Since $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}) = \gamma^2 I$ the Hessian is positive definite (without the need to assume that A is full rank), and we have that

Lemma A.1. *$J_1(\cdot, \boldsymbol{z})$ is strictly convex for any $\boldsymbol{z}$.*

An immediate consequence is the unicity of the minimizer of the partial function $J_1(\cdot, \boldsymbol{z})$.

A.2. Coercivity of J_1.

Lemma A.2. *J_1 is coercive.*

Proof. First, let us note that J_1 is the sum of three non-negative terms. If it was not coercive, then we could find a sequence $(\boldsymbol{x}_k, \boldsymbol{z}_k) \to \infty$ such that $J_1(\boldsymbol{x}_k, \boldsymbol{z}_k)$ is bounded. As a consequence all three terms are bounded. In particular the last term $\|\boldsymbol{z}_k\|$ is bounded, which means that $\boldsymbol{x}_k \to \infty$. From Property 1, $\{\boldsymbol{\mu}_\theta(\boldsymbol{z}_k)\}$ and $\{\boldsymbol{\Sigma}_\theta(\boldsymbol{z}_k)\}$ are bounded for bounded $\{\boldsymbol{z}_k\}$. Now, from the definition of the second term of J_1, we get that, $\{\boldsymbol{\mu}_\theta(\boldsymbol{z}_k)\}$ and $\{\boldsymbol{\Sigma}_\theta(\boldsymbol{z}_k)\}$ being bounded and $\boldsymbol{x}_k$ going to ∞ yield that $H_\theta(x_k, z_k)$ goes to infinity, while being bounded. This leads to a contradiction and thus proves that J_1 is coercive.

A.3. Regularity of J_1. In the sequel we adopt the common assumption that all neural networks used in this work are composed of a finite number d of layers, each layer being composed of: *(a)* a linear operator (*e.g.* convolutional or fully connected layer), followed by *(b)* a non-linear L-Lipschitz component-wise activation function with $0 < L < \infty$.

Therefore we have the following property:

Property 1. *For any neural network f_θ with parameters θ having the structure described above:*
There exists a constant C_θ such that $\forall \boldsymbol{u}$,

$$\|f_\theta(\boldsymbol{u})\|_2 \leq C_\theta \|\boldsymbol{u}\|_2.$$

Concerning activation functions we use two kinds:

- continously differentiable activations like ELU, or
- continuous but non-differentiable activations like ReLU

Hence, by composition, we have that

Lemma A.3. *For continuously differentiable activation functions, J_1 is continuously differentiable.*

Appendix B. MAP-x and MAP-z for deterministic generative models.

Assume that the stochastic γ-generative model is

$$p_{X_\gamma|Z_\gamma}(\boldsymbol{x} \,|\, \boldsymbol{z}) = \mathcal{N}(\mathsf{G}(\boldsymbol{z}), \gamma^2 I)$$

meaning that when $\gamma \to 0$

$$p_{X|Z}(\boldsymbol{x} \,|\, \boldsymbol{z}) = \delta(\boldsymbol{x} - \mathsf{G}(\boldsymbol{z}))$$

We now analyze the MAP-$\boldsymbol{z}$ and MAP-$\boldsymbol{x}$ estimators for the limit case when $\gamma = 0$. This is what we call a deterministic generative model, and it includes GANs for instance.

B.1. MAP-z. By definition the MAP-$\boldsymbol{z}$ estimator is obtained by maximising the posterior with respect to $\boldsymbol{z}$:

$$\begin{aligned} \hat{\boldsymbol{z}}_{\text{MAP}-\boldsymbol{z}} &= \arg\max_{\boldsymbol{z}} \left\{ p_{Z|Y}(\boldsymbol{z} \,|\, \boldsymbol{y}) \right\} \\ &= \arg\max_{\boldsymbol{z}} \left\{ p_{Y|Z}(\boldsymbol{y} \,|\, \boldsymbol{z})\, p_Z(\boldsymbol{z}) \right\}. \end{aligned} \tag{B.1}$$

In the last line we used Bayes rule to rewrite this posterior in more simple terms. However, this expression still involves the unknown conditional $p_{Y|Z}(\boldsymbol{y}\,|\,\boldsymbol{z})$.
Let us express this maximization in terms of $p_{Y|X}(\boldsymbol{y}\,|\,\boldsymbol{x})$.

To do so we recall the relation between the conditionals and the joint:

$$p_{Y|Z}(\boldsymbol{y}\,|\,\boldsymbol{z})\,p_Z(\boldsymbol{z}) = p_{Y,Z}(\boldsymbol{y},\boldsymbol{z}) = p_{Z|Y}(\boldsymbol{z}\,|\,\boldsymbol{y})\,p_Y(\boldsymbol{y}) \tag{B.2}$$

We can also compute the joint distribution $p_{Y,Z}(\boldsymbol{y},\boldsymbol{z})$ by marginalization on a third random variable X:

$$\begin{aligned} p_{Y,Z}(\boldsymbol{y},\boldsymbol{z}) &= \int p_{X,Y,Z}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\,d\boldsymbol{x} \\ &= \int p_{Y|X,Z}(\boldsymbol{y}\,|\,\boldsymbol{x},\boldsymbol{z})\,p_{X|Z}(\boldsymbol{x}\,|\,\boldsymbol{z})\,p_Z(\boldsymbol{z})\,d\boldsymbol{x} \\ &= \int p_{Y|X}(\boldsymbol{y}\,|\,\boldsymbol{x})\,\delta(\boldsymbol{x}-\mathsf{G}(\boldsymbol{z}))p_Z(\boldsymbol{z})\,d\boldsymbol{x} \\ &= p_{Y|X}(\boldsymbol{y}\,|\,\mathsf{G}(\boldsymbol{z}))\,p_Z(\boldsymbol{z}) \end{aligned} \tag{B.3}$$

The third line follows from our graphical model $Z \to X \to Y$ which implies that once we know $X = \boldsymbol{x}$, then Z provides no additional information, therefore

$$p_{Y|X,Z}(\boldsymbol{y}\,|\,\boldsymbol{x},\boldsymbol{z}) = p_{Y|X}(\boldsymbol{y}\,|\,\boldsymbol{x})\,.$$

The last line follows simply from the integration on $\boldsymbol{x}$ of a delta function.

From equations (B.2) and (B.3) we can derive an expression of $p_{Z|Y}(\boldsymbol{z}\,|\,\boldsymbol{y})$ in terms of $p_{Y|X}(\cdot\,|\,\cdot)$ and the generator G namely:

$$p_{Z|Y}(\boldsymbol{z}\,|\,\boldsymbol{y}) = \frac{1}{p_Y(\boldsymbol{y})} p_{Y|X}(\boldsymbol{y}\,|\,\mathsf{G}(\boldsymbol{z}))\,p_Z(\boldsymbol{z})$$

This proves the main result of this section:

Proposition B.1 (MAP-$\boldsymbol{z}$ estimator for deterministic generative models). *Assume we have*

- *a deterministic generative model where* $X = \mathsf{G}(Z)$ *and*
- *an inverse problem characterised by the log conditional distribution* $\log p_{Y|X}(\boldsymbol{y}\,|\,\boldsymbol{x}) = -F(\boldsymbol{x},\boldsymbol{y})$.

Then the MAP-$\boldsymbol{z}$ *estimator is computed as* $\hat{\boldsymbol{x}}_{\text{MAP}-\boldsymbol{z}} = \mathsf{G}(\hat{\boldsymbol{z}}_{\text{MAP}-\boldsymbol{z}})$ *where*

$$\begin{aligned} \hat{\boldsymbol{z}}_{\text{MAP}-\boldsymbol{z}} &= \arg\max_{\boldsymbol{z}} \left\{ p_{Y|X}(\boldsymbol{y}\,|\,\mathsf{G}(\boldsymbol{z}))\,p_Z(\boldsymbol{z}) \right\} \\ &= \arg\min_{\boldsymbol{z}} \left\{ F(\mathsf{G}(\boldsymbol{z}),\boldsymbol{y}) - \log p_Z(\boldsymbol{z}) \right\}. \end{aligned} \tag{B.4}$$

B.2. MAP-x. The MAP-$\boldsymbol{x}$ estimator is obtained by maximizing the posterior with respect to $\boldsymbol{x}$. The generative model induces a prior on X via the push-forward measure $p_X = \mathsf{G}\sharp p_Z$, which following [33, section 5] can be developed as

$$p_X(\boldsymbol{x}) = \frac{p_Z\left(\mathsf{G}^{-1}(\boldsymbol{x})\right)}{\sqrt{\det S(\mathsf{G}^{-1}(\boldsymbol{x}))}}\delta_{\mathcal{M}}(\boldsymbol{x})$$

where $S = \left(\frac{\partial \mathsf{G}}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial \mathsf{G}}{\partial \boldsymbol{z}}\right)$ is the squared Jacobian and the manifold $\mathcal{M} = \{\boldsymbol{x} : \exists \boldsymbol{z},\, \boldsymbol{x} = \mathsf{G}(\boldsymbol{z})\}$ represents the image of the generator G.

With such a prior p_X, the $\boldsymbol{x}$-optimization (1.2) required to obtain $\hat{\boldsymbol{x}}_{\text{MAP}}$ becomes intractable (in general), for various reasons:

- the computation of $\det S$,
- the inversion of G, and
- the hard constraint $\boldsymbol{x} \in \mathcal{M}$.

These operations are are all memory and/or computationally intensive, except when they are partially addressed by the use of a normalizing flow like in [23, 52].

Appendix C. Joint MAP-x-z, Continuation Scheme and convergence to MAP-z.

The functional $J_{1,\beta}$ introduced in Equation 1.5 can be seen from two different perspectives.

From a machine learning perspective it corresponds to the joint log-posterior J_1 in the case where $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}) = \frac{1}{\beta} I$ and $\boldsymbol{\mu}_\theta = \mathsf{G}$, namely:

$$\begin{aligned} J_{1,\beta}(\boldsymbol{x},\boldsymbol{z}) = & \underbrace{\frac{1}{2\,\sigma^2}\,\|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}\|^2}_{F(\boldsymbol{x},\boldsymbol{y})} \\ & + \underbrace{\frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2}_{H_\theta(\boldsymbol{x},\boldsymbol{z}) = \varphi_\beta(\boldsymbol{x},\boldsymbol{z})} \\ & + \frac{1}{2}\,\|\boldsymbol{z}\|^2 + C_\beta \end{aligned}$$

From an optimization standpoint it can be considered as an inexact penalisation procedure: We want to solve the constrained problem

$$\min_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{C}} \underbrace{F(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\,\|\boldsymbol{z}\|^2}_{=J_{1,0}(\boldsymbol{x},\boldsymbol{z})}$$

with $\mathcal{C} = \{(\boldsymbol{x},\boldsymbol{z}) \mid \boldsymbol{x} = \boldsymbol{\mu}_\theta(\boldsymbol{z})\}$ whose solution provides the MAP-$\boldsymbol{z}$ estimator

$$(\boldsymbol{x}^*,\boldsymbol{z}^*) = \arg\min_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{C}} J_{1,0}(\boldsymbol{x},\boldsymbol{z}). \tag{C.1}$$

To do so, we introduced the family of unconstrained problems

$$\min_{\boldsymbol{x},\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x},\boldsymbol{z})$$

and their corresponding minimizers

$$(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{z}}_\beta) = \arg\min_{\boldsymbol{x},\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z})$$

which for $\beta = \frac{1}{\gamma^2}$ provide the MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ estimator.

We can show that the MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ estimator converges to the MAP-$\boldsymbol{z}$ estimator when $\beta \to \infty$ (or equivalently $\gamma \to 0$).

Proposition C.1. *The unconstrained functional tends to the constrained functional plus the constraint:*

$$J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z}) \xrightarrow{\beta\to\infty} J_{1,\infty}(\boldsymbol{x}, \boldsymbol{z}) = F(\boldsymbol{x}, \boldsymbol{z}) + \iota_{\boldsymbol{x}=\boldsymbol{\mu}_\theta(\boldsymbol{z})}(\boldsymbol{x}, \boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2 \tag{C.2}$$

and the unconstrained minimizers tend to the constrained minimizer as $\beta \to \infty$:

$$\lim_{\beta\to\infty} \arg\min_{\boldsymbol{x},\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z}) = \arg\min_{\boldsymbol{x},\boldsymbol{z}} J_{1,\infty}(\boldsymbol{x}, \boldsymbol{z}) = \arg\min_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{C}} J_{1,0}(\boldsymbol{x}, \boldsymbol{z}) \tag{C.3}$$

or equivalently

$$\lim_{\beta\to\infty} (\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{z}}_\beta) = (\boldsymbol{x}^*, \boldsymbol{z}^*).$$

Proof. The pointwise convergence of φ_β to $\iota_{\boldsymbol{x}=\mathsf{G}(\boldsymbol{z})}$ as β goes to ∞ is straightforward. Moreover, we have uniform convergence on any compact set $\mathcal{K} \subset \mathcal{C}$ of φ_β to $\iota_{\boldsymbol{x}=\mathsf{G}(\boldsymbol{z})}$ as β goes to ∞. Indeed, $\varphi_\beta = \beta\varphi_1$, with φ_1 a continuous function. Hence φ_1 admits a (positive) minimizer on $\mathcal{K}$, thus for β large enough, φ_β's minimum is as large as wanted.

One has that if $(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{z}}_\beta)$ denotes a minimizer of $J_{1,\beta}$, then when β goes to ∞, any limit point of $(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{z}}_\beta)$ is a solution of the constrained problem. ∎

The previous result motivates Algorithm 1.1.

Consider Algorithm 1.1 in the ideal case (maxiter=∞) where the internal loop converges.

Proposition C.2 (Convergence of Algorithm 1.1). *Let $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)_k$ be a sequence generated by Algorithm 1.1. If $(\boldsymbol{z}_\infty^k)_k$ is bounded, then any limit point of $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)_k$ is a stationary point of problem (C.1).*

Proof. Let $(\beta_k)_k$ a sequence that converges to ∞. Let $k \in \mathbb{N}$. We consider the sequence $(\boldsymbol{x}_n^k, \boldsymbol{z}_n^k)_n$ generated by

$$\forall\, n \in \mathbb{N}, \qquad \boldsymbol{z}_{n+1}^k \in \arg\min_{\boldsymbol{z}} J_{1,\beta_k}(\boldsymbol{x}_n^k, \boldsymbol{z}) \text{ and } \boldsymbol{x}_{n+1}^k = \arg\min_{\boldsymbol{x}} J_{1,\beta_k}(\boldsymbol{x}, \boldsymbol{z}_{n+1}^k)$$

with $\boldsymbol{x}_0^k = \boldsymbol{x}_\infty^{k-1}$. If J_1 satisfies Assumption 2, then so does J_{1,β_k} . Hence, the sequence $(\boldsymbol{x}_n^k, \boldsymbol{z}_n^k)_n$ is bounded. Let $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ be a limit point of this sequence. According to Proposition 1, one has

$$\frac{\partial J_{1,\beta_k}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) = \frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) + \beta_k(\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)) = 0$$

and

$$\frac{\partial J_{1,\beta_k}}{\partial \boldsymbol{z}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) = \frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) + \beta_k (D\mathsf{G}(\boldsymbol{z}_\infty^k))^* (\mathsf{G}(\boldsymbol{z}_\infty^k) - \boldsymbol{x}_\infty^k) = 0$$

Assume that the sequence $(\boldsymbol{z}_\infty^k)_k$ is bounded. By convexity and optimality, one has

$$J_{1,0}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) \leq J_{1,\beta_k}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) \leq J_{1,\beta_k}(\mathsf{G}(\boldsymbol{z}_\infty^k), \boldsymbol{z}_\infty^k) = J_{1,0}(\mathsf{G}(\boldsymbol{z}_\infty^k), \boldsymbol{z}_\infty^k)$$

Since $(J_{1,0}(\mathsf{G}(\boldsymbol{z}_\infty^k), \boldsymbol{z}_\infty^k))_k$ is bounded, so is $(J_{1,0}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k))_k$. By coercivity, the sequence $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)_k$ is also bounded. Then it admits a limit point denoted $(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty)$. Let $(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j})_j$ be a convergent subsequence. Let us assume that $\hat{\boldsymbol{x}}_\infty \neq \mathsf{G}(\hat{\boldsymbol{z}}_\infty)$. Then, there exists $a > 0$ and $j_0 \in \mathbb{N}$ such that

$$\forall\, j \geq j_0, \qquad \|\boldsymbol{x}_\infty^{k_j} - \mathsf{G}(\boldsymbol{z}_\infty^{k_j})\|^2 > a$$

Hence, one has

$$J_{1,\beta_{k_j}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) \geq J_{1,0}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) + \beta_{k_j}\, a \geq \min J_{1,0} + \beta_{k_j}\, a \underset{j\to+\infty}{\longrightarrow} \infty$$

which leads to a contradiction. This proves that $\hat{\boldsymbol{x}}_\infty = \mathsf{G}(\hat{\boldsymbol{z}}_\infty)$. Otherwise said, $(\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k))_k$ goes to zero.

Since we have for any k

$$\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) + \beta_k(\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)) = 0$$

the continuity of $\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}$ ensures that $\left(\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j})\right)_j$ converges; thus, so is $(\beta_{k_j}(\boldsymbol{x}_\infty^{k_j} - \mathsf{G}(\boldsymbol{z}_\infty^{k_j})))_j$. Then there exists $\lambda^* \in \mathbb{R}^d$ such that

$$\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) = -\beta_{k_j}(\boldsymbol{x}_\infty^{k_j} - \mathsf{G}(\boldsymbol{z}_\infty^{k_j})) \underset{j\to+\infty}{\longrightarrow} \lambda^* = \frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty)$$

and

$$\frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) = -\beta_{k_j}(D\mathsf{G}(\boldsymbol{z}_\infty^{k_j}))^*(\mathsf{G}(\boldsymbol{z}_\infty^{k_j}) - \boldsymbol{x}_\infty^{k_j}) \underset{j\to+\infty}{\longrightarrow} -(D\mathsf{G}(\hat{\boldsymbol{z}}_\infty))^*(\lambda^*)$$

Let us introduce

$$f(\boldsymbol{z}) = F(\mathsf{G}(\boldsymbol{z}), \boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2 \tag{C.4}$$

One can check that f is differentiable and that

$$\nabla f(\boldsymbol{z}) = (D\mathsf{G}(\boldsymbol{z}))^* \left(\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\mathsf{G}(\boldsymbol{z}), \boldsymbol{z})\right) + \frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\mathsf{G}(\boldsymbol{z}), \boldsymbol{z})$$

Hence, we have proved that

$$\nabla f(\hat{\boldsymbol{z}}_\infty) = 0$$

Conclusion: If $(\boldsymbol{z}_\infty^k)_k$ is bounded, any limit point of $(\boldsymbol{z}_\infty^k)_k$ is a stationary point of (C.4). ■

Algorithm 2.4 is a particular (truncated) case of Algorithm 1.1 with an adaptive choice of β that does not need to go to ∞.

Proposition C.3 (Convergence of Algorithm 2.4).

Proof. Let us write the Lagrangian of the problem solved in Algorithm 2.4:

$$\forall\, \lambda \geq 0, \qquad \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \lambda) = J_{1,0}(\boldsymbol{x}, \boldsymbol{z}) + \lambda \left(\|\boldsymbol{x} - \mathsf{G}(\boldsymbol{z})\|^2 - \varepsilon \right)$$

KKT conditions ensure that any solution $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ of the constrained problem is associated to at least one Lagrange multiplier $\lambda^* \geq 0$ such that

$$\frac{\partial \mathcal{L}}{\partial(\boldsymbol{x}, \boldsymbol{z})}(\boldsymbol{x}^*, \boldsymbol{z}^*; \lambda^*) = 0 = \begin{pmatrix} \frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*, \boldsymbol{z}^*) + 2\lambda^* \left(\boldsymbol{x}^* - \mathsf{G}(\boldsymbol{z}^*)\right) \\ \frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\boldsymbol{x}^*, \boldsymbol{z}^*) + 2\lambda^* \left(D\mathsf{G}(\boldsymbol{z}^*)\right)^* (\boldsymbol{x}^* - \mathsf{G}(\boldsymbol{z}^*)) \end{pmatrix}$$

According to the calculus above, this proves that $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ is a stationary point of $J_{1,2\lambda^*}$. Note that, if $\lambda^* = 0$, then $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ is a minimizer of $J_{1,0}$. Otherwise, one has $\|\boldsymbol{x}^* - \mathsf{G}(\boldsymbol{z}^*)\|^2 = \varepsilon$.

Hence, if we consider Algorithm 1.1 with the update rule for β_k as in Algorithm 2.4 and a stopping rule saying that the iterations stop as soon as, for any given k,

$$\|\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)\|^2 \leq \varepsilon$$

there are two possible cases:

1. **case $\lambda^* = 0$:** then $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ is a solution of the constraint problem iff $\nabla J_{1,0}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) = 0$ (that is, $\boldsymbol{x}_\infty^k = \mathsf{G}(\boldsymbol{z}_\infty^k)$);
2. **case $\lambda^* > 0$:** unless $\|\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)\|^2$ exactly equals ε, $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ is **not** a solution of the constraint problem

However, in general, $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ is a solution of the following constraint problem

$$\min_{\|\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)\|^2 \leq \tilde{\varepsilon}} J_{1,0}(\boldsymbol{x}, \boldsymbol{z})$$

with $\tilde{\varepsilon} = \|\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)\|^2 \leq \varepsilon$. Hence, if we stop the iterations when $\|\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)\|^2 \leq \varepsilon$, we will get a solution of

$$\min_{\|\boldsymbol{x}_\infty^k - \mathsf{G}(\boldsymbol{z}_\infty^k)\|^2 \leq \tilde{\varepsilon}} J_{1,0}(\boldsymbol{x}, \boldsymbol{z}), \qquad \tilde{\varepsilon} \leq \varepsilon$$

which provides an error control as well. ∎