Solving Inverse Problems in Imaging
by Joint Posterior Maximization with Autoencoding Prior

Andrés Almansa

Université de Paris

cnrs

Joint work with: Mario González, Pauline Tan
Thanks to: Pablo Musé, Mauricio Delbracio, José Lezama

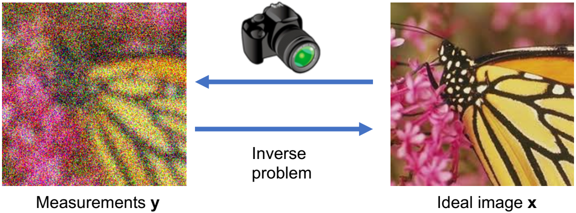
Preprint and code available here
http://up5.fr/jpmap

March 9th 2021 - Lecture 8 - M2 MVA course

https://delires.wp.imt.fr

Inverse Problems in Imaging

Estimate clean image $\boldsymbol{x} \in \mathbb{R}^d$
from noisy, degraded measurements $\boldsymbol{y} \in \mathbb{R}^m$.



Known degradation model (usually log-concave):

$$p_{Y|X}(\boldsymbol{y} \mid \boldsymbol{x}) \propto e^{-F(\boldsymbol{x},\boldsymbol{y})} \quad \text{where} \quad F(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2. \quad (1)$$

Inverse Problems in Imaging

Estimate clean image $\boldsymbol{x} \in \mathbb{R}^d$
from noisy, degraded measurements $\boldsymbol{y} \in \mathbb{R}^m$.
Known degradation model (usually log-concave):

$$p_{Y|X}(\boldsymbol{y} \mid \boldsymbol{x}) \propto e^{-F(\boldsymbol{x},\boldsymbol{y})} \quad \text{where} \quad F(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2. \quad (1)$$

Variational/Bayesian Approach

Use image prior $p_X(\boldsymbol{x}) \propto e^{-\lambda R(\boldsymbol{x})}$ to compute estimator

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}} p_{X|Y}(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \{F(\boldsymbol{x},\boldsymbol{y}) + \lambda R(\boldsymbol{x})\} \quad (2)$$

$$\hat{\boldsymbol{x}}_{\mathrm{MMSE}} = \arg\min_{\boldsymbol{x}} \mathbb{E}\left[\, \|X - \boldsymbol{x}\|^2 \,\middle|\, Y = \boldsymbol{y} \,\right] \quad (3)$$

Inverse Problems in Imaging

$$p_{Y|X}(\boldsymbol{y} \mid \boldsymbol{x}) \propto e^{-F(\boldsymbol{x},\boldsymbol{y})} \quad \text{where} \quad F(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2. \quad (1)$$

Variational/Bayesian Approach

Use image prior $p_X(\boldsymbol{x}) \propto e^{-\lambda R(\boldsymbol{x})}$ to compute estimator

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}} p_{X|Y}(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \{F(\boldsymbol{x},\boldsymbol{y}) + \lambda R(\boldsymbol{x})\} \quad (2)$$

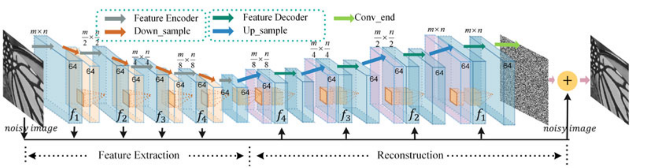
$$\hat{\boldsymbol{x}}_{\mathrm{MMSE}} = \arg\min_{\boldsymbol{x}} \mathbb{E}\left[\|X - \boldsymbol{x}\|^2 \,\middle|\, Y = \boldsymbol{y} \right] \quad (3)$$

Common explicit priors

- Total Variation (Chambolle, 2004; Louchet and Moisan, 2013; Pereyra, 2016; Rudin et al., 1992)
- Gaussian Mixtures (Teodoro et al., 2018; Yu et al., 2011; Zoran and Weiss, 2011)

Neural Networks for inverse problems:

Two paradigms



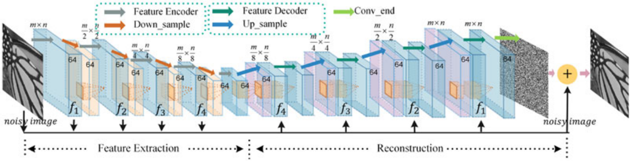
- Agnostic approach : find a sufficient number of image pairs $(\boldsymbol{x}^i, \boldsymbol{y}^i)$ and train a neural network f_θ to invert A by minimizing the empirical risk $\sum_i \|f_\theta(\boldsymbol{y}^i) - \boldsymbol{x}^i\|_2^2$

  ✓ no need to model A, $\mathbf{n}$ nor prior for $\boldsymbol{x}$

  ✗ needs retraining if A or $\mathbf{n}$ change

Neural Networks for inverse problems:

Two paradigms



- Agnostic approach : find a sufficient number of image pairs $(\boldsymbol{x}^i, \boldsymbol{y}^i)$ and train a neural network f_θ to invert A by minimizing the empirical risk $\sum_i \|f_\theta(\boldsymbol{y}^i) - \boldsymbol{x}^i\|_2^2$
  - ✓ no need to model A, $\mathbf{n}$ nor prior for $\boldsymbol{x}$
  - ✗ needs retraining if A or $\mathbf{n}$ change
- Decoupled (plug & play) approach : Model separately
  1. conditional density $p_{Y|X}(\boldsymbol{y} \mid \boldsymbol{x})$ (using physical model, calibration)
  2. prior model $p_X(\boldsymbol{x})$ (through NN learning)
  3. Use Bayes theorem to estimate $\boldsymbol{x}$ via MAP or MMSE

Neural Networks for inverse problems:

Two paradigms

- Agnostic approach : find a sufficient number of image pairs $(\boldsymbol{x}^i, \boldsymbol{y}^i)$ and train a neural network f_θ to invert A by minimizing the empirical risk $\sum_i \|f_\theta(\boldsymbol{y}^i) - \boldsymbol{x}^i\|_2^2$
  - ✓ no need to model A, $\mathbf{n}$ nor prior for $\boldsymbol{x}$
  - ✗ needs retraining if A or $\mathbf{n}$ change
- Decoupled (plug & play) approach : Model separately
  1. conditional density $p_{Y|X}(\boldsymbol{y} \mid \boldsymbol{x})$ (using physical model, calibration)
  2. prior model $p_X(\boldsymbol{x})$ (through NN learning)
  3. Use Bayes theorem to estimate $\boldsymbol{x}$ via MAP or MMSE
  - ✓ uses all available modeling information
  - ✓ train once, use for many inverse problems
  - ⚠ difficult to learn $p_X(\boldsymbol{x})$ directly
  - ⚠ Non-convex optimization

Neural Networks for inverse problems:

Implicitly decoupled approach

Solve the optimization problem

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}} p_{X|Y}(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \{F(\boldsymbol{x}, \boldsymbol{y}) + \lambda R(\boldsymbol{x})\}$$

via ADMM splitting (Ryu et al., 2019)

1. $\boldsymbol{v}_{k+1} = \arg\min_{\boldsymbol{v}} R(\boldsymbol{v}) + \frac{1}{2\delta^2}\|\boldsymbol{v} - (\boldsymbol{x}_k - \boldsymbol{u}_k)\|^2$
2. $\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}, \boldsymbol{y}) + \frac{\lambda}{2\delta^2}\|\boldsymbol{x} - (\boldsymbol{v}_{k+1} - \boldsymbol{u}_k)\|^2$
3. $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \boldsymbol{v}_{k+1} - \boldsymbol{x}_{k+1}$

R is unknown but we can use a train a neural network to approximate the δ-denoising problem in step 1:

$$D_\delta(\tilde{\boldsymbol{x}}) = \arg\min_{\boldsymbol{v}} R(\boldsymbol{v}) + \frac{1}{2\delta^2}\|\boldsymbol{v} - \tilde{\boldsymbol{x}}\|^2$$

Neural Networks for inverse problems:

Implicitly decoupled approach

Solve the optimization problem via ADMM splitting

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}} p_{X|Y}(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \{F(\boldsymbol{x}, \boldsymbol{y}) + \lambda R(\boldsymbol{x})\}$$

R is unknown but a NN approximates its proximal operator:

$$D_\delta(\tilde{\boldsymbol{x}}) = \arg\min_{\boldsymbol{v}} R(\boldsymbol{v}) + \frac{1}{2\delta^2}\|\boldsymbol{v} - \tilde{\boldsymbol{x}}\|^2$$

Challenges

- NN training produces an MMSE rather than a MAP estimator for D_δ
- Convergence guarantees

Neural Networks for inverse problems:

Implicitly decoupled approach

Solve the optimization problem via ADMM splitting (Ryu et al., 2019)

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}} p_{X|Y}(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \{F(\boldsymbol{x}, \boldsymbol{y}) + \lambda R(\boldsymbol{x})\}$$

Assumption (A)

1. ✓ $D_\delta - I$ *is L-Lipschitz with* $L \in (0,1)$
2. ✗ $F(\cdot, \boldsymbol{y})$ *is* μ*-strongly convex*
3. ✗ $\lambda < \frac{\sigma^2 \mu (1+L-2L^2)}{L} \underset{L \to 1^-}{\longrightarrow} 0$

Theorem (Ryu et al. (2019))

Under assumption A, the Plug & Play ADMM algorithm converges to a critical point.

Explicitly decoupled approach (MAP-x):

How to use neural networks to learn the prior $p_X(x)$?

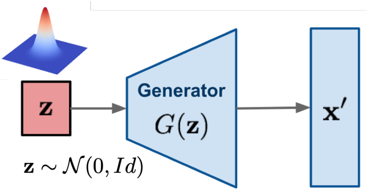
Generative Adversarial Networks (GANs) (Arjovsky and Bottou, 2017; Goodfellow et al., 2014)

Learn a generator function G that maps

$$z \sim \mathcal{N}(0, Id)$$

to

$$x = G(z) \sim p_X$$



Explicitly decoupled approach (MAP-x):

How to use neural networks to learn the prior $p_X(x)$?

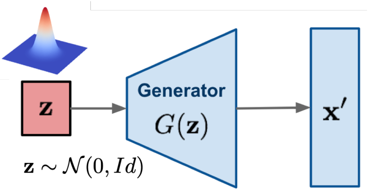
Generative Adversarial Networks (GANs) (Arjovsky and Bottou, 2017; Goodfellow et al., 2014)

Learn a generator function G that maps

$$z \sim \mathcal{N}(0, Id)$$

to

$$x = G(z) \sim p_X$$



MAP-x Following Papamakarios et al. (2019, section 5), the push-forward measure $p_X = G\sharp p_Z$ can be developed as

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

where

$$S = \left(\frac{\partial G}{\partial z}\right)^T \left(\frac{\partial G}{\partial z}\right)$$

$$\mathcal{M} = \{x : \exists z,\ x = G(z)\}$$

Explicitly decoupled approach (MAP-$\boldsymbol{x}$):

How to use neural networks to learn the prior $p_X(\boldsymbol{x})$?

Generative Adversarial Networks (GANs) (Arjovsky and Bottou, 2017; Goodfellow et al., 2014) Learn a generator function G that maps

$$\boldsymbol{z} \sim \mathcal{N}(0, Id) \quad \text{to} \quad \boldsymbol{x} = \mathrm{G}(\boldsymbol{z}) \sim p_X$$

MAP-$\boldsymbol{x}$ Following Papamakarios et al. (2019, section 5), the push-forward measure $p_X = \mathrm{G}\sharp p_Z$ can be developed as

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

where

$$S = \left(\frac{\partial \mathrm{G}}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial \mathrm{G}}{\partial \boldsymbol{z}}\right)$$

$$\mathcal{M} = \{\boldsymbol{x} : \exists \boldsymbol{z}, \boldsymbol{x} = \mathrm{G}(\boldsymbol{z})\}$$

$\boldsymbol{x}$-optimization required to obtain $\hat{\boldsymbol{x}}_{\mathrm{MAP}}$ becomes intractable due to:

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

Explicitly decoupled approach (MAP-z):

Instead of solving the x-optimisation problem:

$$\hat{x}_{\mathrm{MAP}} = \arg\max_{x} p_{Y|X}(y \mid x)\, p_X(x) = \arg\min_{x} \{F(x, y) + R(x)\}$$

BORA ET AL. (2017) propose to optimize over z

$$\hat{z} = \arg\max_{z} \{p_{Y|X}(y \mid \mathrm{G}(z))\, p_Z(z)\}$$
$$= \arg\min_{z} \left\{F(\mathrm{G}(z), y) + \frac{1}{2}\|z\|^2\right\}$$
$$\hat{x}_{z-\mathrm{MAP}} = \mathrm{G}(\hat{z})$$

$\hat{x}_{z-\mathrm{MAP}}$ $(\neq \hat{x}_{\mathrm{MAP}})$ but it maximizes the latent posterior:

$$\hat{x}_{z-\mathrm{MAP}} = \mathrm{G}\left(\arg\max_{z} \{p_{Z|Y}(z \mid y)\}\right)$$

Explicitly decoupled approach (MAP-z):

$\hat{\boldsymbol{x}}_{\boldsymbol{z}-\mathrm{MAP}}$ ($\neq \hat{\boldsymbol{x}}_{\mathrm{MAP}}$) maximizes the latent posterior:

$$\begin{aligned}\hat{\boldsymbol{x}}_{\boldsymbol{z}-\mathrm{MAP}} &= \mathrm{G}\left(\arg\max_{\boldsymbol{z}} \left\{p_{Z|Y}(\boldsymbol{z} \mid \boldsymbol{y})\right\}\right) \\ &= \mathrm{G}\left(\arg\min_{\boldsymbol{z}} \left\{F(\mathrm{G}(\boldsymbol{z}), \boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2\right\}\right)\end{aligned}$$

Challenges

- Nonconvex optimization using gradient descent
- may get stuck in spurious local minima

Common solution: Splitting + continuation scheme

MAP-z splitting and continuation scheme.

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

$$\hat{\mathbf{x}}_{\mathrm{MAP}-\mathbf{z}} = \lim_{\beta\to\infty} \hat{\mathbf{x}}_\beta .$$

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

Require: Measurements $\boldsymbol{y}$, Initial condition $\boldsymbol{x}_0$
Ensure: $\hat{\boldsymbol{x}} = \mathsf{G}\left(\arg\max_{\boldsymbol{z}} p_{Z|Y}(\boldsymbol{z}\,|\,\boldsymbol{y})\right)$
1: **for** $k := 0$ **to** $k_{\max}$ **do**
2: $\beta := \beta_k$
3: **for** $n := 0$ **to** maxiter **do**
4: $\boldsymbol{z}_{n+1} := \arg\min_{\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x}_n, \boldsymbol{z})$ // Nonconvex
5: $\boldsymbol{x}_{n+1} := \arg\min_{\boldsymbol{x}} J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z}_{n+1})$ // Quadratic
6: **end for**
7: $\boldsymbol{x}_0 := \boldsymbol{x}_{n+1}$
8: **end for**
9: **return** $\boldsymbol{x}_{n+1}$

Non-convex step 4: Use a local quadratic approximation (VAE encoder) ...

VAEs and Joint Posterior

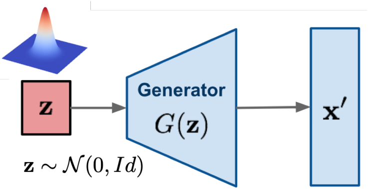
Generative Adversarial Networks (GANs) (Goodfellow et al., 2014)

Learn a generator function G that maps

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

to

$$\boldsymbol{x} = \mathrm{G}(\boldsymbol{z}) \sim p_X$$



VAEs and Joint Posterior

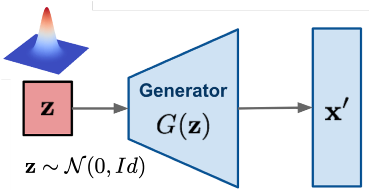
Generative Adversarial Networks (GANs) (Goodfellow et al., 2014)

Learn a generator function G that maps

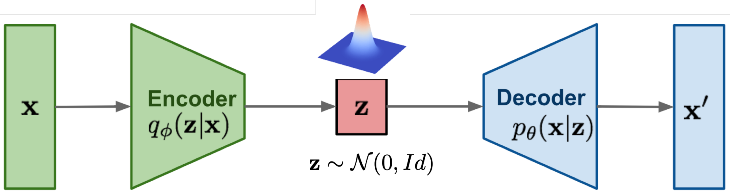
$$z \sim \mathcal{N}(0, Id)$$

to

$$x = \mathrm{G}(z) \sim p_X$$



Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)

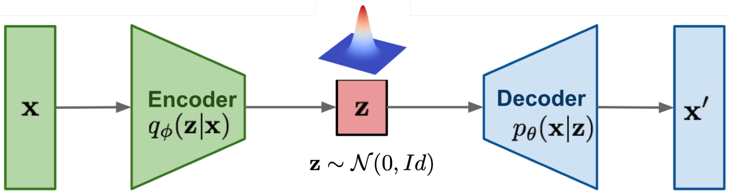


Generative model: $p_{X|Z}(x \mid z) = p_\theta(x|z) = \mathcal{N}(x;\ \mu_\theta(z),\ \gamma Id)$

Approximate inverse: $p_{Z|X}(z \mid x) \approx q_\phi(z|x) = \mathcal{N}(z;\ \mu_\phi(x),\ \Sigma_\phi(x))$

VAEs and Joint Posterior

Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)



Generative model: $p_{X|Z}(\mathbf{x} \mid \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x};\ \boldsymbol{\mu}_\theta(\mathbf{z}),\ \gamma Id)$

Approximate inverse: $p_{Z|X}(\mathbf{z} \mid \mathbf{x}) \approx q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\ \boldsymbol{\mu}_\phi(\mathbf{x}),\ \boldsymbol{\Sigma}_\phi(\mathbf{x}))$

Learning: Maximize the averaged *Evidence Lower BOund (ELBO)* for $\mathbf{x} \in \mathcal{D}$

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

VAEs and Joint Posterior

Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)

Generative model: $p_{X|Z}(\mathbf{x} \mid \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x};\, \boldsymbol{\mu}_\theta(\mathbf{z}),\, \gamma Id)$

Joint density: $p_{X,Z}(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})\, p_Z(\mathbf{z})$

Approximate inverse: $p_{Z|X}(\mathbf{z} \mid \mathbf{x}) \approx q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\, \boldsymbol{\mu}_\phi(\mathbf{x}),\, \boldsymbol{\Sigma}_\phi(\mathbf{x}))$

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

VAEs and Joint Posterior

Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)

Generative model: $p_{X|Z}(\mathbf{x} \mid \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x};\ \boldsymbol{\mu}_\theta(\mathbf{z}),\ \gamma Id)$

Joint density: $p_{X,Z}(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})\, p_Z(\mathbf{z})$

Approximate inverse: $p_{Z|X}(\mathbf{z} \mid \mathbf{x}) \approx q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\ \boldsymbol{\mu}_\phi(\mathbf{x}),\ \boldsymbol{\Sigma}_\phi(\mathbf{x}))$

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

Joint Posterior: (log-quadratic in $\mathbf{x}$)

$$\begin{aligned} J_1(\mathbf{x}, \mathbf{z}) &:= -\log p_{X,Z|Y}(\mathbf{x}, \mathbf{z} \mid \mathbf{y}) \\ &= -\log p_{Y|X,Z}(\mathbf{y} \mid \mathbf{x}, \mathbf{z})\, p_\theta(\mathbf{x} \mid \mathbf{z}) p_Z(\mathbf{z}) \\ &= F(\mathbf{x}, \mathbf{y}) + \underbrace{\frac{1}{2\gamma}\|\mathbf{x} - \boldsymbol{\mu}_\theta(\mathbf{z})\|^2}_{H_\theta(\mathbf{x},\mathbf{z})} + \frac{1}{2}\|\mathbf{z}\|^2. \end{aligned} \tag{4}$$

VAEs and Joint Posterior

Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)

Generative model: $p_{X|Z}(\boldsymbol{x} \mid \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x};\, \boldsymbol{\mu}_\theta(\boldsymbol{z}),\, \gamma Id)$

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

Approximate inverse: $p_{Z|X}(\boldsymbol{z} \mid \boldsymbol{x}) \approx q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z};\, \boldsymbol{\mu}_\phi(\boldsymbol{x}),\, \boldsymbol{\Sigma}_\phi(\boldsymbol{x}))$

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

Joint Posterior: (log-quadratic in $\boldsymbol{x}$)

$$\begin{aligned} J_1(\boldsymbol{x}, \boldsymbol{z}) &:= -\log p_{X,Z|Y}(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{y}) \\ &= -\log p_{Y|X,Z}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})\, p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) p_Z(\boldsymbol{z}) \\ &= F(\boldsymbol{x}, \boldsymbol{y}) + \underbrace{\frac{1}{2\gamma}\|\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z})\|^2}_{H_\theta(\boldsymbol{x},\boldsymbol{z})} + \frac{1}{2}\|\boldsymbol{z}\|^2. \end{aligned} \tag{4}$$

Approximate Joint Posterior: (log-quadratic in $\boldsymbol{z}$)

$$\begin{aligned} J_2(\boldsymbol{x}, \boldsymbol{z}) &:= -\log p_{Y|X,Z}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})\, q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\, p_X(\boldsymbol{x}) \\ &= F(\boldsymbol{x}, \boldsymbol{y}) + \underbrace{\frac{1}{2}\|\boldsymbol{\Sigma}_\phi^{-1/2}(\boldsymbol{x})(\boldsymbol{z} - \boldsymbol{\mu}_\phi(\boldsymbol{x}))\|^2 + C(\boldsymbol{x})}_{K_\phi(\boldsymbol{x},\boldsymbol{z})} - \log p_X(\boldsymbol{x}). \end{aligned} \tag{5}$$

Joint Posterior Maximization - Alternate Convex Search

Algorithm 2.1 Joint posterior maximization - exact case

Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial condition $\boldsymbol{x}_0$

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

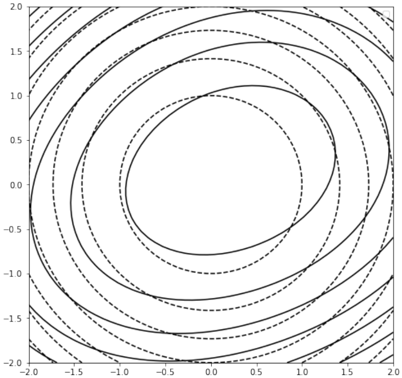
1: **for** $n := 0$ **to** maxiter **do**
2: $\quad \boldsymbol{z}_{n+1} := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}_n)$ // Quadratic approx
3: $\quad \boldsymbol{x}_{n+1} := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}_{n+1})$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

Proposition

If the encoder approximation is exact ($J_2 = J_1$) then

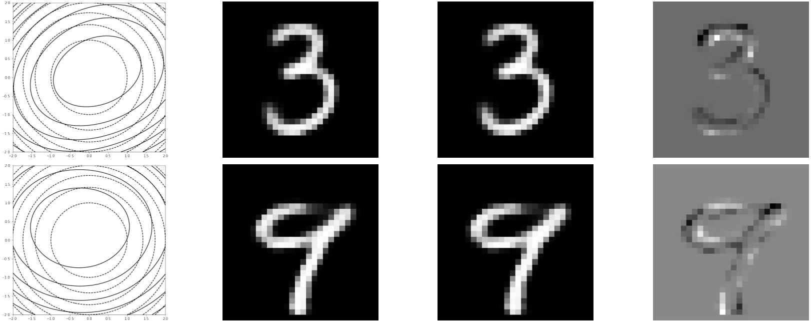
- J_1 is biconvex, and following Gorski et al. (2007):
- Algorithm 2.1 is an Alternate Convex Search
- Algorithm 2.1 converges to a critical point

JPMAP - Accuracy of encoder approximation



Contour plots of $-\log p_{Z|X}(\boldsymbol{z} \mid \boldsymbol{x})$ and $-\log q_\phi(\boldsymbol{z}|\boldsymbol{x})$ for a fixed $\boldsymbol{x}$ and for a random 2D subspace in the $\boldsymbol{z}$ domain.

JPMAP - Accuracy of encoder approximation



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

Figure 1. 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)

Joint Posterior Maximization - approximate case

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

for $n := 0$ **to** maxiter **do**
  $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}_n)$ // Quadratic approx
  $\boldsymbol{z}^2 := \mathrm{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, 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**
    $\boldsymbol{x}^i := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^i)$ // Quadratic
  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 $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

Joint Posterior Maximization - approximate case

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

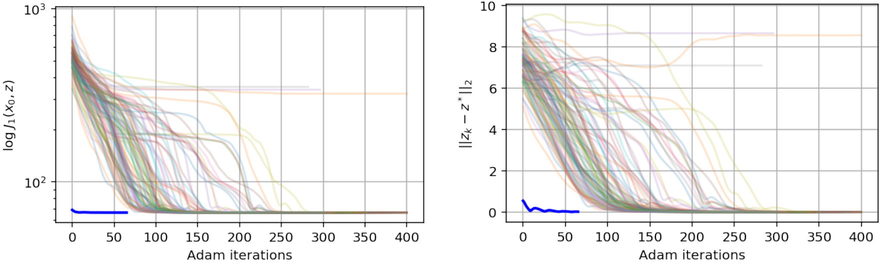
Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial condition $\boldsymbol{x}_0$, iterations $n_1 \leq n_2 \leq n_{\max}$

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

for $n := 0$ **to** $n_{\max}$ **do**
  done := FALSE
  if $n < n_1$ **then**
    $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}_n)$ // Quadratic approx
    $\boldsymbol{x}^1 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^1)$ // Quadratic
    if $J_1(\boldsymbol{x}^1, \boldsymbol{z}^1) < J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$ **then**
      $i^* := 1$ // Faster alternative while J_2 is good enough
      done := TRUE
    end if
  end if
  if not done **and** $n < n_2$ **then**
    $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}_n)$ // Quadratic approx
    $\boldsymbol{z}^2 := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}^1$
    $\boldsymbol{x}^2 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^2)$ // Quadratic
    if $J_1(\boldsymbol{x}^2, \boldsymbol{z}^2) < J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$ **then**
      $i^* := 2$ // J_2 init is good enough
      done := TRUE
    end if
  end if
  if not done **then**
    $\boldsymbol{z}^3 := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}_n$
    $\boldsymbol{x}^3 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^3)$ // Quadratic
    $i^* := 3$
  end if
  $(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) := (\boldsymbol{x}^{i^*}, \boldsymbol{z}^{i^*})$
end for
return $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

JPMAP - Effectivenes of the encoder initialization

Trajectories of $\mathrm{GD}_{\boldsymbol{z}} J_1(\boldsymbol{x}_0, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}_0$
Thick blue curve: $\boldsymbol{z}_0 = \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_0, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}_0)$
Thin curves: random initializations $\boldsymbol{z}_0 \sim \mathcal{N}(0, Id)$



JPMAP - Convergence

If we use ELU activations then the following assumption is verified:

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.

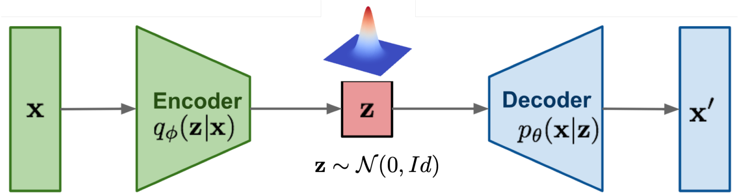
Proposition (Convergence of Algorithm 2.3)

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.*

Denoising Criterion to train VAEs (Im et al., 2017)

Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)

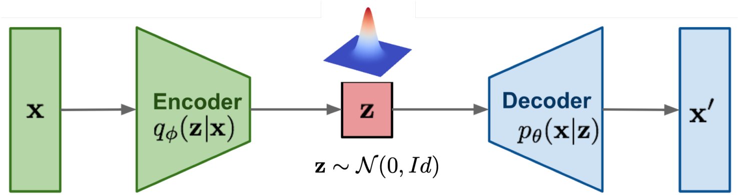


Learning: Maximize the averaged *Evidence Lower BOund (ELBO)* for $\boldsymbol{x} \in \mathcal{D}$

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

Denoising Criterion to train VAEs (Im et al., 2017)

Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)



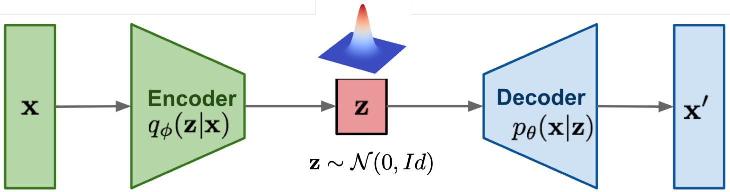
Learning: Maximize the averaged *Evidence Lower BOund (ELBO)* for $\boldsymbol{x} \in \mathcal{D}$

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

Problem: $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ only trained for $\boldsymbol{x} \in \mathcal{D}$ or $\boldsymbol{x} \in \mathcal{M} = \boldsymbol{\mu}_\theta(\mathbb{R}^m)$.
But: Step 2 in the algorithm evaluates $\boldsymbol{\mu}_\phi(\boldsymbol{x}_n)$ for degraded $\boldsymbol{x}_n \notin \mathcal{M}$

Denoising Criterion to train VAEs (Im et al., 2017)

Variational AutoEncoders (VAEs) (Kingma and Welling, 2013)



Learning: Maximize the averaged *Evidence Lower BOund (ELBO)* for $x \in \mathcal{D}$

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

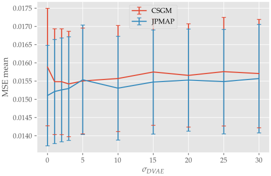
Denoising criterion: Train on $\tilde{\mathcal{D}}$ but still require $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\tilde{x})) \approx x$.

$$\tilde{\mathcal{D}} = \{\tilde{x} = x + \sigma_{\text{DVAE}}\varepsilon \;:\; x \in \mathcal{D} \text{ and } \varepsilon \sim \mathcal{N}(0, I)\}$$

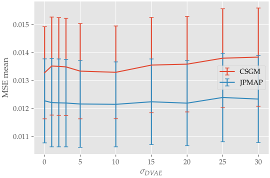
Maximize the denoising ELBO

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

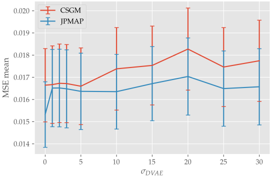
Denoising criterion does not degrade generative model



(a) Denoising



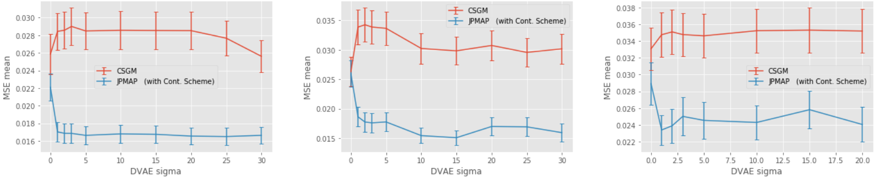
(b) Compressed Sensing



(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* $\boldsymbol{z}_0 = \boldsymbol{\mu}_\phi(\boldsymbol{x}^*)$*).*

Optimal value of σ_{DVAE}



(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.*

MAP-z as the limit case for $\beta \to \infty$

Two options for MAP-$\boldsymbol{z}$ estimator instead of the joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$

1. CSGM - gradient descent, may be stuck in local minima
2. Use Algorithm 2.3 to solve a series of joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ problems with increasing values of $\beta = \frac{1}{\gamma} \to \infty$ as suggested in Algorithm 1.1.

Stopping criterion: Inequality constrained problem

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

The corresponding Lagrangian form is

$$\max_{\beta}\min_{\boldsymbol{x},\boldsymbol{z}} F(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2 + \beta\left(\|\mathrm{G}(\boldsymbol{z}) - \boldsymbol{x}\|^2 - \varepsilon\right)^+ \qquad (6)$$

We use the exponential multiplier method (Tseng and Bertsekas, 1993) to guide the search for the optimal value of β (see Algorithm 2.4)

MAP-z as the limit case for $\beta \to \infty$

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 $\boldsymbol{x}^k, \boldsymbol{z}^k$

MAP-z as the limit case for $\beta \to \infty$

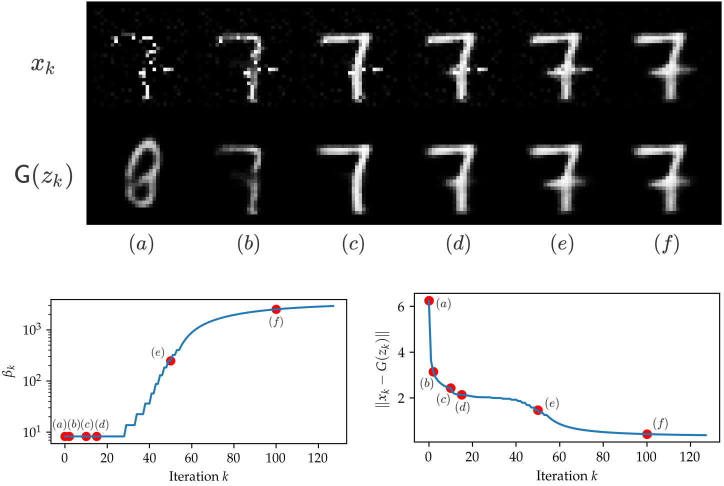
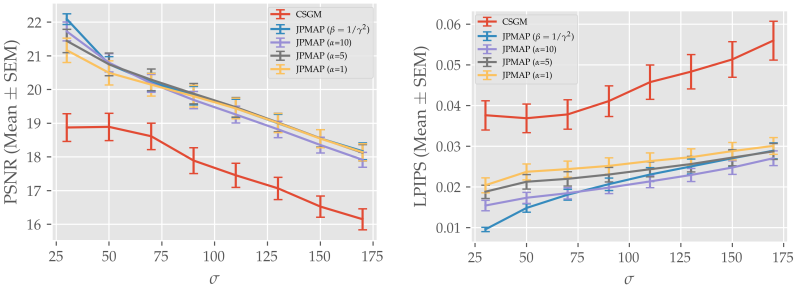
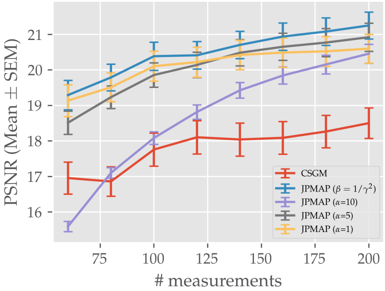


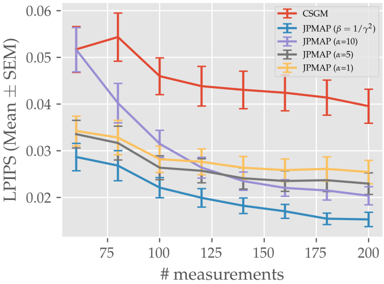
Figure 5. Evolution of Algorithm 2.4. *In this inpainting example, JPMAP starts with the initialization in (a). During first iterations (b) – (d) where β_k is small, $\boldsymbol{x}_k$ and $\mathsf{G}(\boldsymbol{z}_k)$ start loosely approaching each other*

Denoising experiments (MNIST)

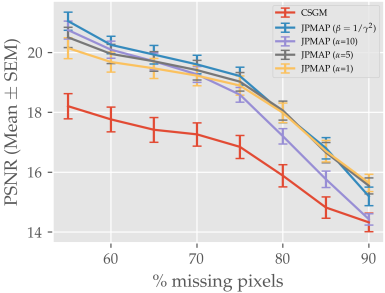


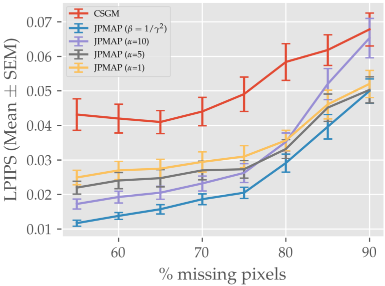
Compressed sensing experiments (MNIST)





Inpainting experiments (MNIST)





Denoising experiment: $\sigma = 110/255$

x^*

y

CSGM

JPMAP ($\beta = 1/\gamma^2$)

JPMAP ($\alpha = 10$)

JPMAP ($\alpha = 5$)

JPMAP ($\alpha = 1$)

Compressed sensing experiment: $m = 140$ random measurements

x^*

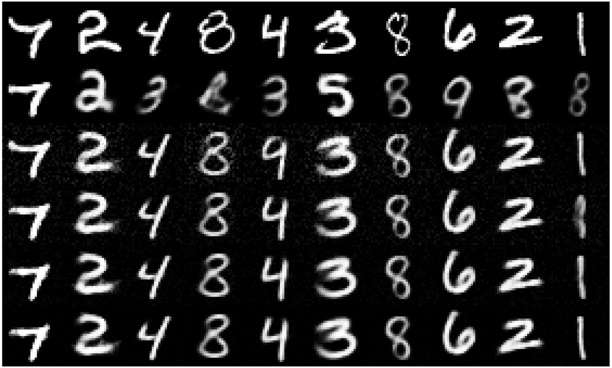
CSGM

JPMAP ($\beta = 1/\gamma^2$)

JPMAP ($\alpha = 10$)

JPMAP ($\alpha = 5$)

JPMAP ($\alpha = 1$)



Inpainting experiment: 80% missing pixels

x^*

y

CSGM

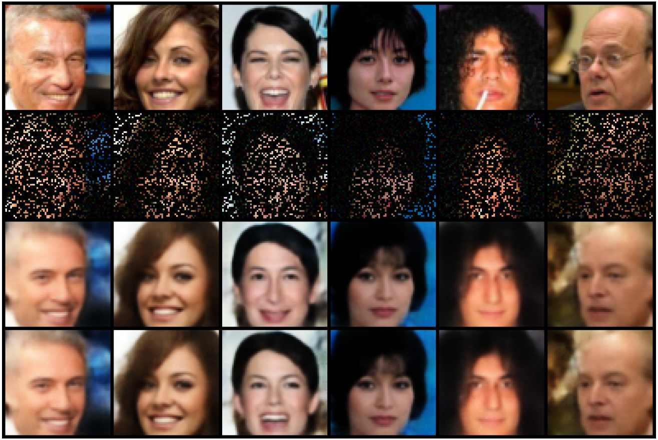
JPMAP ($\beta = 1/\gamma^2$)

JPMAP ($\alpha = 10$)

JPMAP ($\alpha = 5$)

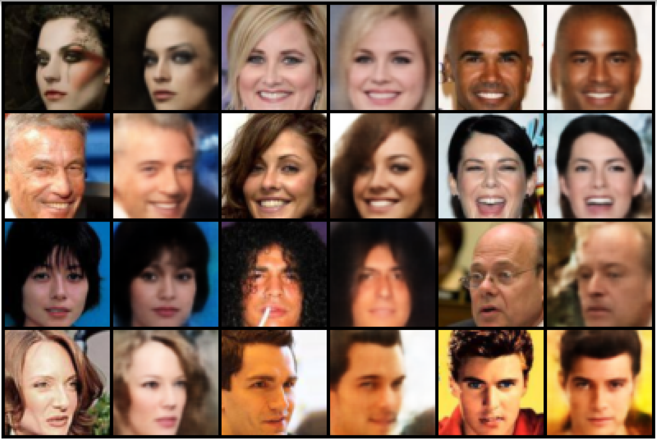
JPMAP ($\alpha = 1$)

Inpainting experiment: 80% missing pixels $\sigma = 10/255$ (CelebA)



From top to bottom: original image $\boldsymbol{x}^*$, corrupted image $\tilde{\boldsymbol{x}}$, restored by CSGM, restored image $\hat{\boldsymbol{x}}$ by our framework.

CelebA reconstructions $\mu_\theta(\mu_\phi(x))$



Reconstructions $\mu_\theta(\mu_\phi(x))$ (even columns) for some test samples x (odd columns), showing the over-regularization of data manifold imposed by the trained VAE. As a consequence, $-\log p_{Z|Y}(z \mid y)$ does not have as many local minima and then a simple gradient

Conclusion

- JPMAP avoids spurious local minima thanks to
  - Quasi bi-convex optimization
  - Encoder initialization
  - Denoising VAE
  - Splitting and continuation scheme
- JPMAP converges for all quadratic problems and regularisation parameters (unlike denoiser-based PnP approaches (Ryu et al., 2019) that are more restrictive)
- Constraints
  - Fixed size
  - VAEs lag behind GANs

Future work

- Use a more powerful VAE like NVAE (Vahdat and Kautz, 2020) or TwoStageVAE (Dai and Wipf, 2019)
- Patch-based JPMAP (EPLL-like)
- Use ADMM with non-linear constraints instead of continuation scheme for $\mathrm{MAP}-\boldsymbol{z}$
- Generalize the scheme to perform posterior sampling

Preprint and code available here
http://up5.fr/jpmap

Thank you for your attention!

Questions? Comments

Arjovsky, Martin and Léon Bottou (2017). "Towards Principled Methods for Training Generative Adversarial Networks". In: *(ICLR) International Conference on Learning Representations*, pp. 1–17 (cit. on pp. 12–14).

Bora, Ashish, Ajil Jalal, Eric Price, and Alexandros G Dimakis (2017). "Compressed sensing using generative models". In: *(ICML) International Conference on Machine Learning*. Vol. 2. JMLR. org, pp. 537–546. ISBN: 9781510855144. arXiv: arXiv:1703.03208v1 (cit. on p. 15).

Chambolle, A (2004). "An algorithm for total variation minimization and applications". In: *Journal of Mathematical Imaging and Vision* 20, pp. 89–97. DOI: 10.1023/B:JMIV.0000011325.36760.1e (cit. on p. 5).

Dai, Bin and David Wipf (2019). "Diagnosing and Enhancing VAE Models". In: *ICLR*, pp. 1–42. arXiv: 1903.05789 (cit. on p. 48).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversaria l Networks". In: *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. ISSN: 10495258. arXiv: 1406.2661 (cit. on pp. 12–14, 18, 19).

Gorski, Jochen, Frank Pfeuffer, and Kathrin Klamroth (2007). "Biconvex sets and optimization with biconvex functions: a survey and extensions". In: *Mathematical Methods of Operations Research* 66.3, pp. 373–407. ISSN: 1432-2994. DOI: 10.1007/s00186-007-0161-1 (cit. on p. 24).

Im, Daniel Jiwoong, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio (2017). "Denoising criterion for variational auto-encoding framework". In: *31st AAAI Conference on Artificial Intelligence, AAAI 2017*. AAAI press, pp. 2059–2065. arXiv: 1511.06406 (cit. on pp. 31–33).

Kingma, Diederik P and Max Welling (2013). "Auto-Encoding Variational Bayes". In: *(ICLR) International Conference on Learning Representations*. Ml, pp. 1–14. ISBN: 1312.6114v10. DOI: 10.1051/0004-6361/201527329. arXiv: 1312.6114 (cit. on pp. 19–23, 31–33).

Louchet, Cécile and Lionel Moisan (2013). "Posterior expectation of the total variation model: Properties and experiments". In: *SIAM Journal on Imaging Sciences* 6.4, pp. 2640–2684. ISSN: 19364954. DOI: 10.1137/120902276 (cit. on p. 5).

Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2019). "Normalizing Flows for Probabilistic Modeling and Inference". In: arXiv: 1912.02762 (cit. on pp. 13, 14).

Pereyra, Marcelo (2016). "Proximal Markov chain Monte Carlo algorithms". In: *Statistics and Computing* 26.4, pp. 745–760. ISSN: 0960-3174. DOI: 10.1007/s11222-015-9567-4. arXiv: 1306.0187 (cit. on p. 5).

Rudin, Leonid I., Stanley Osher, and Emad Fatemi (1992). "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1-4, pp. 259–268. ISSN: 01672789. DOI: 10.1016/0167-2789(92)90242-F (cit. on p. 5).

Ryu, Ernest K., Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin (2019). "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*, pp. 5546–5557. arXiv: 1905.05406 (cit. on pp. 9, 11, 47).

Teodoro, Afonso M., José M. Bioucas-Dias, and Mário A. T. Figueiredo (2018). *Scene-Adapted Plug-and-Play Algorithm with Guaranteed Convergence: Applications to Data Fusion in Imaging*. arXiv: 1801.00605 (cit. on p. 5).

Tseng, Paul and Dimitri P. Bertsekas (1993). "On the convergence of the exponential multiplier method for convex programming". In: *Mathematical Programming* 60.1-3, pp. 1–19. ISSN: 00255610. DOI: 10.1007/BF01580598 (cit. on p. 36).

Vahdat, Arash and Jan Kautz (2020). "Nvae: A deep hierarchical variational autoencoder". In: *Advances in Neural Information Processing Systems* 33 (cit. on p. 48).

Yu, Guoshen, Guillermo Sapiro, and Stéphane Mallat (2011). "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity". In: *IEEE Transactions on Image Processing* 21.5, pp. 2481–2499 (cit. on p. 5).

Zoran, Daniel and Yair Weiss (2011). "From learning models of natural image patches to whole image restoration". In: *2011 International Conference on*

Computer Vision. IEEE, pp. 479–486. ISBN: 978-1-4577-1102-2. DOI: 10.1109/ICCV.2011.6126278 (cit. on p. 5).