Technische Universität Berlin

The Power of Patches for Training Normalizing Flows

Johannes Hertrich | TU Berlin

j.hertrich@math.tu-berlin.de
joint work with
Fabian Altekrüger, Alexander Denker, Paul Hagemann, Peter Maass, Gabriele Steidl
Code available at `https://github.com/FabianAltekrueger/patchNR`

Contents

Normalizing Flows

Patch Normalizing Flows - MAP Estimation

Wasserstein Patch Prior - MAP Estimation

Wasserstein Patch Prior - Posterior Sampling

Contents

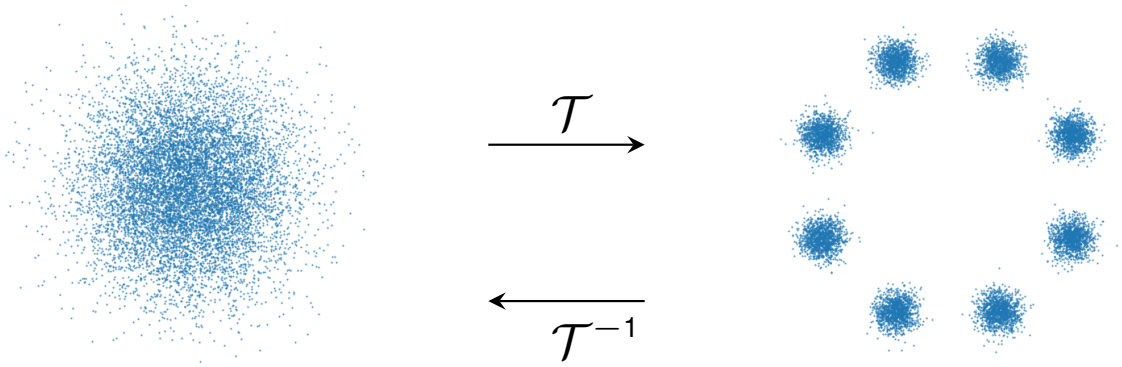
Normalizing Flows

Patch Normalizing Flows - MAP Estimation

Wasserstein Patch Prior - MAP Estimation

Wasserstein Patch Prior - Posterior Sampling

Normalizing Flows



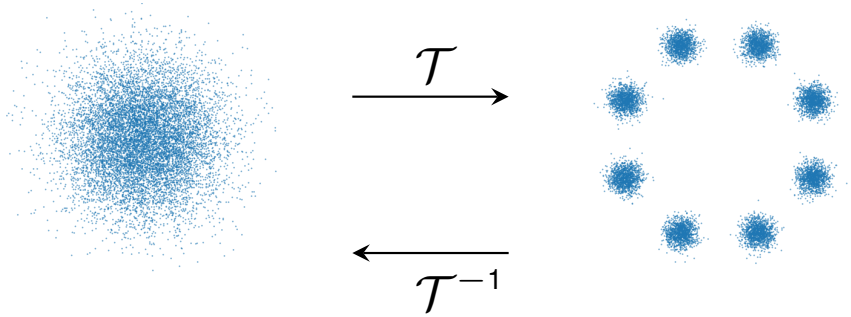
Latent distribution P_Z Data distribution P_X

- Goal: Approximate a complicated probability distribution P_X.
- Learn diffeomorphism $\mathcal{T}$ such that $P_X \approx \mathcal{T}_{\#} P_Z$.
- Sampling by drawing a sample z from P_Z and computing $\mathcal{T}(z)$.
- Density evaluation by the change-of-variables formula

$$p_{\mathcal{T}_{\#} P_Z}(x) = p_Z(\mathcal{T}^{-1}(x))|\det(\nabla \mathcal{T}^{-1}(x))|.$$

$\Rightarrow \mathcal{T}$ **will be an invertible neural network!**

Kullback-Leibler Divergence



Goal: $P_X \approx \mathcal{T}_{\#} P_Z$

- Loss Function: Kullback-Leibler divergence (KL) for densities

$$\mathrm{KL}(P_X, P_Y) := \mathbb{E}_{x\sim P_X}\left[\log\left(\frac{p_X(x)}{p_Y(x)}\right)\right] = \mathbb{E}_{x\sim P_X}[\log(p_X(x))] - \mathbb{E}_{x\sim P_X}[\log(p_Y(x))]$$

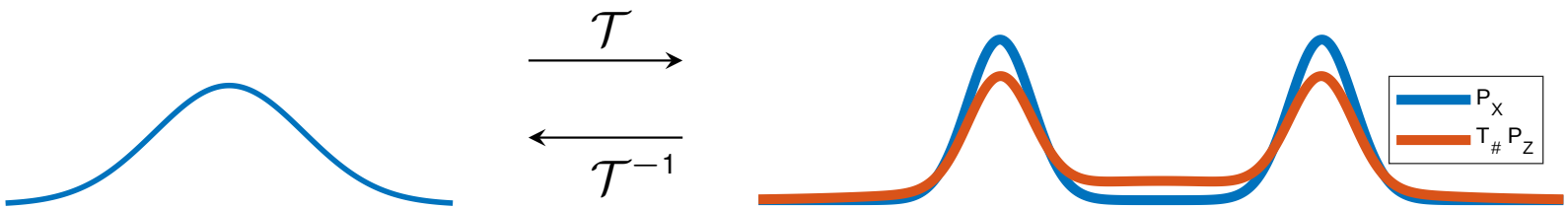
- $\mathrm{KL}(P_X, P_Y) \geq 0$ for all $P_X, P_Y \in \mathcal{P}(\mathbb{R}^d)$ with equality $\Leftrightarrow$ if $P_X = P_Y$.
- Not symmetric and no triangle inequality.

Forward KL

Minimize

$$\mathcal{J}_{\mathrm{NF}}(\theta) = \mathrm{KL}(P_X, \mathcal{T}_{\#} P_Z) = \mathbb{E}_{x\sim P_X}[\log(p_X(x))] - \mathbb{E}_{x\sim P_X}[\log(p_{\mathcal{T}_{\#} P_Z}(x))]$$
$$\propto -\mathbb{E}_{x\sim P_X}[\log(p_Z(\mathcal{T}^{-1}(x))) - \log(|\det(\nabla \mathcal{T}^{-1}(x))|)].$$

- Need samples from P_X for training, but not the density p_X.
- Penalizes samples from P_X which have a small probability under $\mathcal{T}_{\#} P_Z$.
- "Mode covering", i.e., all samples from the training set are included in $\mathcal{T}_{\#} P_Z$.
- Typical error: Generation of some unrealistic samples.

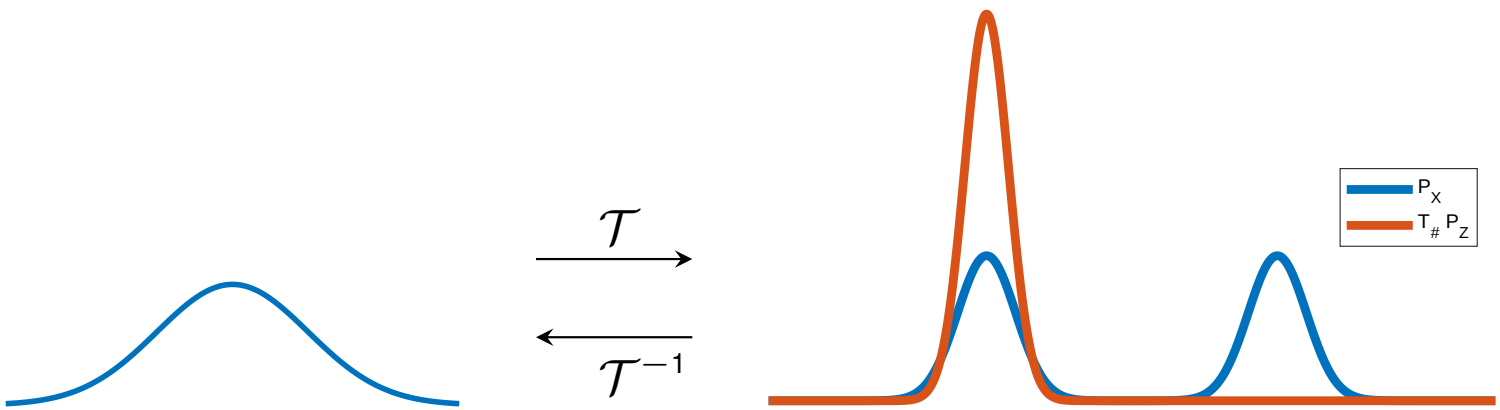


Backward KL

Minimize

$$\tilde{\mathcal{J}}_{\mathrm{NF}}(\theta) = \mathrm{KL}(\mathcal{T}_{\#}P_Z, P_X) = \mathrm{KL}(P_Z, \mathcal{T}_{\#}^{-1}P_X)$$
$$\propto -\mathbb{E}_{z\sim P_Z}[\log(p_X(\mathcal{T}(z))) - \log(|\det(\nabla\mathcal{T}(z))|)].$$

- Need the density p_X, but no samples.
- Penalizes samples from $\mathcal{T}_{\#}P_Z$ which have a small probability under P_X.
- "Mode seeking", i.e., samples generated from $\mathcal{T}_{\#}P_Z$ are realistic.
- Typical error: Mode collapse, i.e., some modes from P_X are missed.



Contents

Normalizing Flows

Patch Normalizing Flows - MAP Estimation

Wasserstein Patch Prior - MAP Estimation

Wasserstein Patch Prior - Posterior Sampling

(Bayesian) Inverse Problems

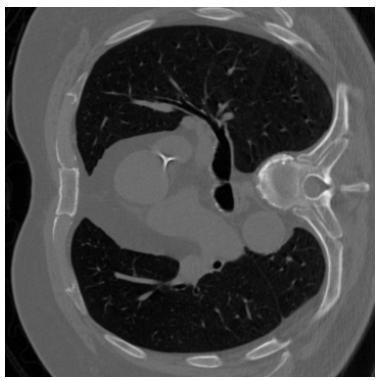
$$Y = \text{noisy}(F(X)), \qquad X \sim P_X$$

- Reconstruction by MAP estimation: Compute

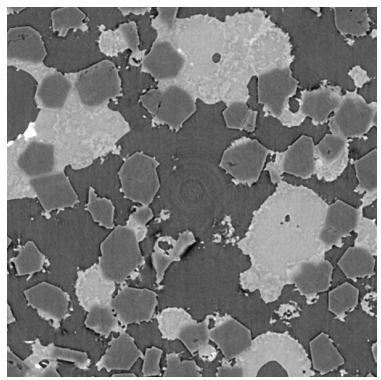
$$\hat{x} \in \operatorname*{argmax}_x \log(p_{X|Y=y}(x)) = \operatorname*{argmin}_x -\log(p_{Y|X=x}(y)) - \log(p_X(x))$$

$$= \operatorname*{argmin}_x \underbrace{\mathcal{D}(x, y)}_{\text{data-fidelity term}} + \lambda \underbrace{\mathcal{R}(x)}_{\text{regularizer}}$$

- Learn regularizer from data → can be used for different operators.
- Only few example images x_j, $j = 1, \ldots, n$ are available → use patches $P_i(x_j)$
- Adapt to a certain class of images:

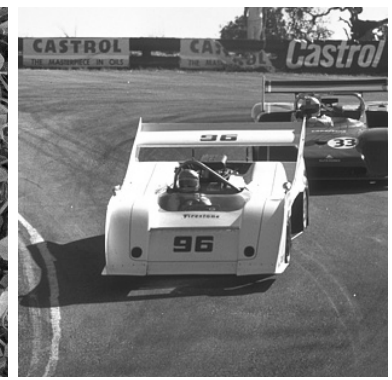


CT images (n=6)



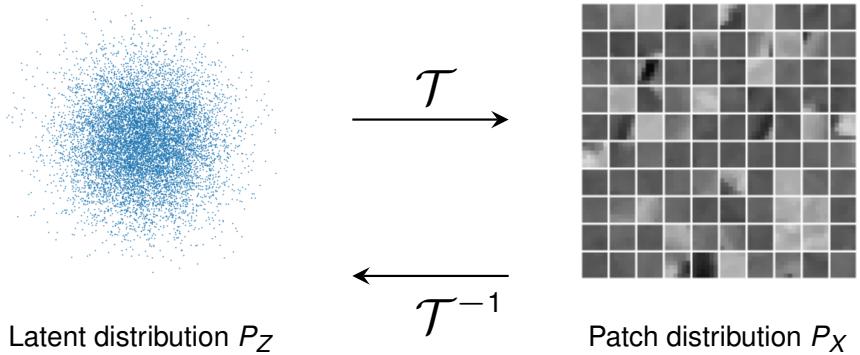
Material images (n=1)

Textures (n=1)



(Natural images - zero shot)

Patch Normalizing Flow Regularizer



1. Learn a normalizing flow $\mathcal{T} = \mathcal{T}_\theta(\cdot)$ using the forward KL

$$\mathcal{L}_{\mathrm{NF}}(\theta) = \sum_{j=1} \sum_{i \in \mathcal{I}} \frac{\|\mathcal{T}^{-1}\left(P_i(x_j)\right)\|^2}{2} - \log\left|\det \nabla \mathcal{T}^{-1}\left(P_i(x_j)\right)\right|$$

2. Patch Normalizing Flow Regularizer (patchNR):

$$\mathcal{R}(x) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \log(p_{\mathcal{T}_{\#} P_Z}(P_i(x)))$$

3. Reconstruction:

$$\hat{x} \in \operatorname*{argmin}_x \mathcal{D}(x, y) + \lambda \mathcal{R}(x)$$

Regularizers and Prior distributions

Recall that regularizer and prior distribution are related by

$$p_X(x) \propto \exp(-\lambda \mathcal{R}(x)).$$

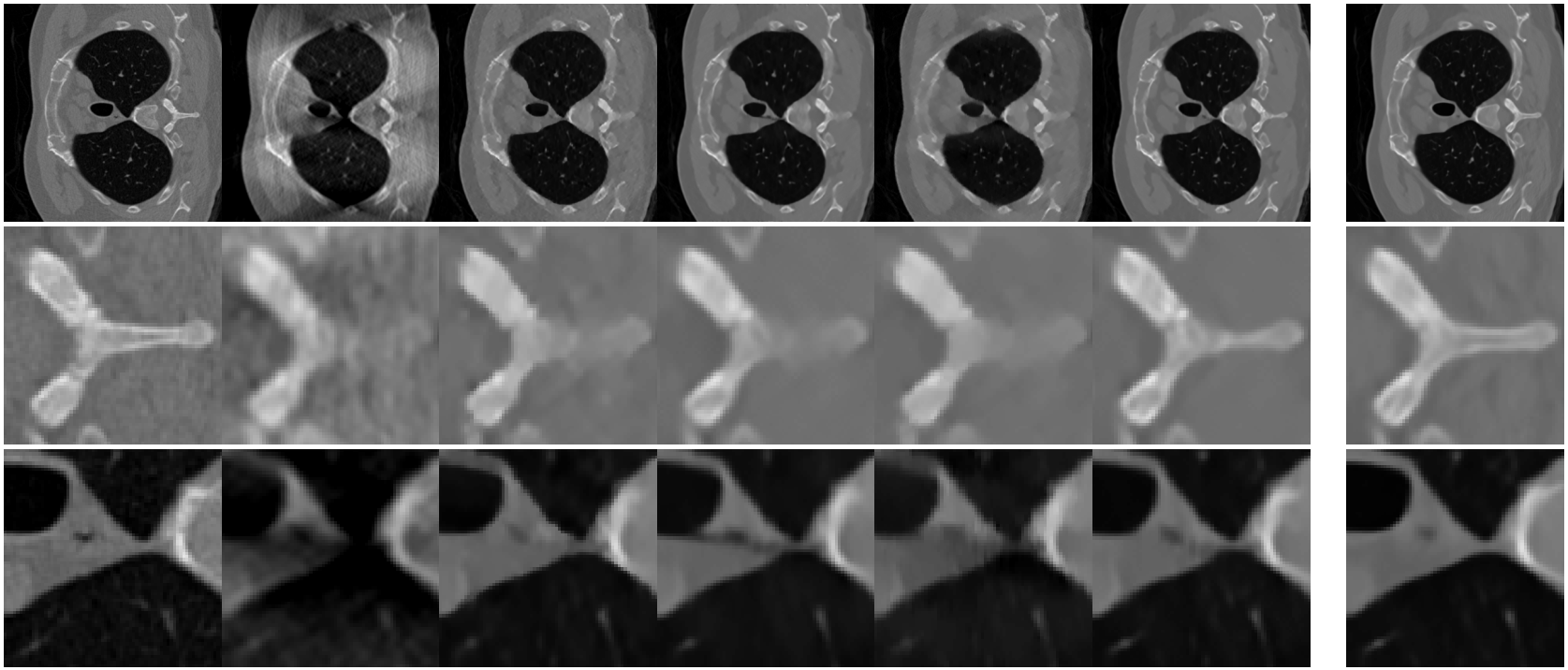
$\rightarrow$This is only a probability density if $\exp(-\lambda \mathcal{R}(x)) \in L^1(\mathbb{R}^d)$.

Theorem

Let $P_Z = \mathcal{N}(0, I)$ and let $\mathcal{T} : \mathbb{R}^s \to \mathbb{R}^s$ be a bi-Lipschitz diffeomorphism, i.e., $\mathrm{Lip}(\mathcal{T}) < \infty$ and $\mathrm{Lip}(\mathcal{T}^{-1}) < \infty$. Then, for any $\rho > 0$ the function $\varphi(x) = \exp(-\rho\, \mathcal{R}(x))$ belongs to $L^1(\mathbb{R}^d)$, where

$$\mathcal{R}(x) = \frac{1}{|\mathcal{I}|} \sum_{i=1}^{N_p} -\log\left(p_{\mathcal{T}_{\#}P_Z}(P_i(x))\right).$$

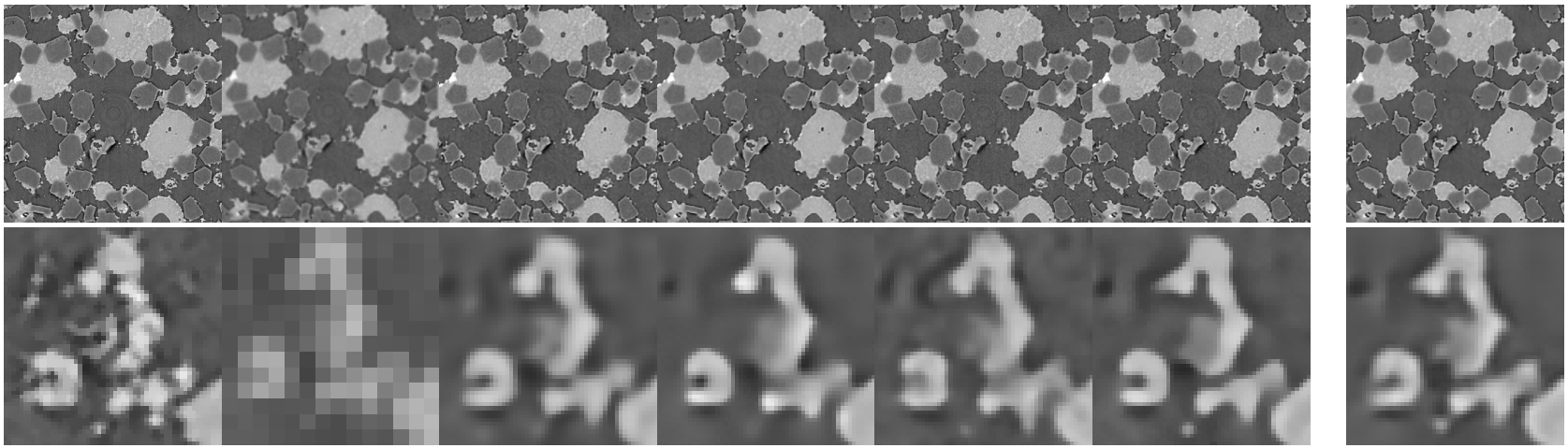
Numerical Results - Limited Angle CT (cutoff 36° of 180°)



	FBP	DIP + TV	EPLL	localAR	patchNR	FBP+UNet (s)
PSNR	21.96 ± 2.25	32.57 ± 3.25	32.78 ± 3.46	31.06 ± 2.95	**33.20** ± 3.55	33.75 ± 3.58
LPIPS	0.305 ± 0.117	**0.191** ± 0.165	0.216 ± 0.175	0.222 ± 0.166	0.201 ± 0.176	0.171 ± 0.134
SSIM	0.531 ± 0.097	0.803 ± 0.146	0.801 ± 0.151	0.779 ± 0.142	**0.811** ± 0.151	0.820 ± 0.140
Training images	0	0	6	6	6	35820

Refs: Ulyanov et al. 2018, Zoran/Weiss 2011, Prost et al. 2021, Jin et al. 2017

Numerical Results - Superresolution (magnification 4)



HR | LR | DIP + TV | EPLL | WPP | patchNR | ACNN (supervised)

	bicubic (not shown)	DPIR (not shown)	DIP+TV	EPLL	WPP	patchNR	ACNN (supervised)
PSNR	25.63 $\pm$ 0.56	27.78 $\pm$ 0.53	27.99 $\pm$ 0.54	28.11 $\pm$ 0.55	27.80 $\pm$ 0.37	**28.53** $\pm$ 0.49	28.89 $\pm$ 0.53
LPIPS	0.406 $\pm$ 0.013	0.322 $\pm$ 0.015	0.191 $\pm$ 0.009	0.244 $\pm$ 0.012	0.167 $\pm$ 0.014	**0.159** $\pm$ 0.008	0.203 $\pm$ 0.011
SSIM	0.699 $\pm$ 0.012	0.770 $\pm$ 0.011	0.764 $\pm$ 0.007	0.779 $\pm$ 0.010	0.749 $\pm$ 0.011	**0.780** $\pm$ 0.008	0.804 $\pm$ 0.010
Training images	0	0/500	0	1	1	1	1000

Refs: H. et al. 2022, Tian et al. 2021

Zero-shot Superresolution by Internal Learning[1]

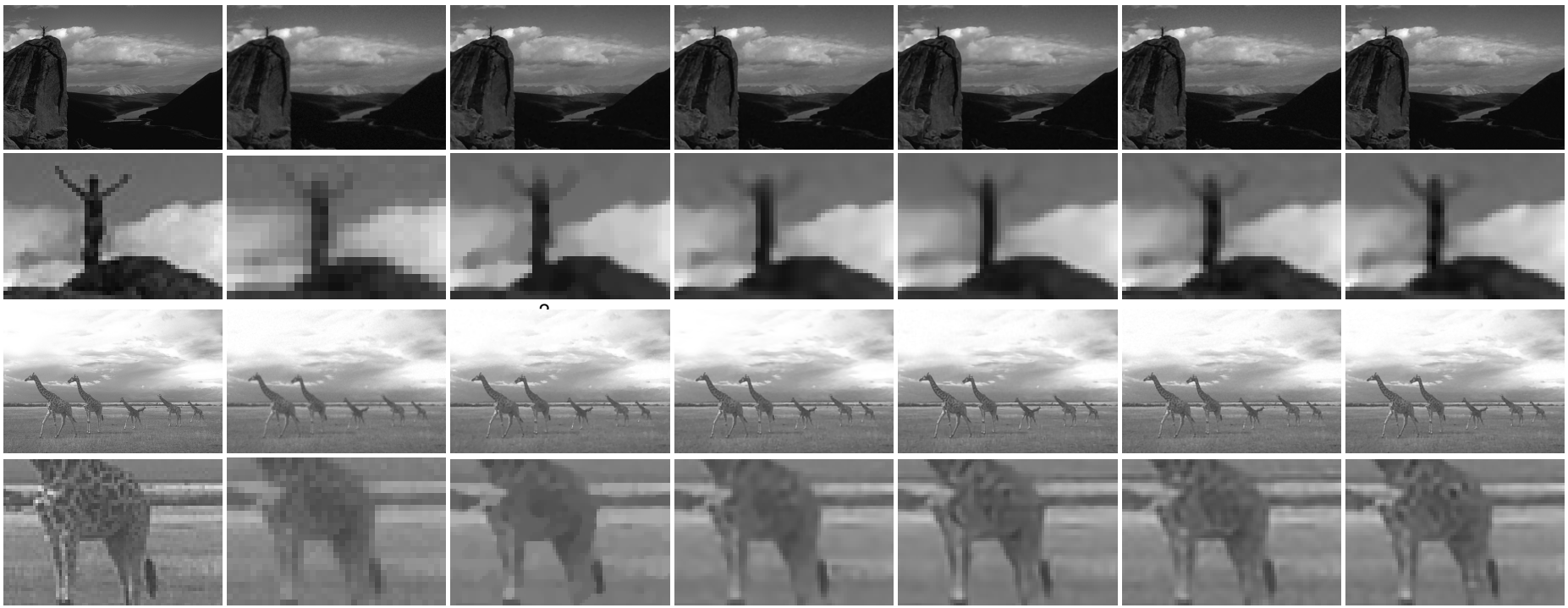
Use the concept of internal learning for superresolution

- Patches within natural images are self-similar across the scales.
- Train the patchNR on the low-resolution observation.
- Data augmentation by mirroring and rotations of the patches.

$\rightarrow$ Requires retraining for every observation.

[1] Refs: Glasner et al. 2009, Shocher et al. 2018

Numerical Results - Zero-shot Superresolution



HR | LR | L^2-TV | DIP+TV | ZSSR | DualSR | patchNR

	L^2-TV	DIP+TV	ZSSR	DualSR	patchNR
PSNR	28.35 ± 3.55	28.44 ± 3.69	28.83 ± 3.57	28.64 ± 3.47	**29.08** ± 3.58
LPIPS	**0.184** ± 0.073	0.215 ± 0.086	0.224 ± 0.085	0.216 ± 0.074	0.202 ± 0.076
SSIM	0.820 ± 0.072	0.821 ± 0.087	0.834 ± 0.066	0.829 ± 0.061	**0.846** ± 0.061

Refs: Shocher et al. 2018, Emad et al. 2021

Contents

Normalizing Flows

Patch Normalizing Flows - MAP Estimation

Wasserstein Patch Prior - MAP Estimation

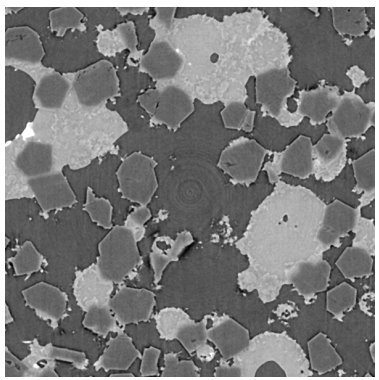
Wasserstein Patch Prior - Posterior Sampling

Superresolution of materials' microstructures

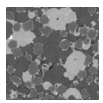
Consider the inverse problem

$$Y = F(X) + \Xi$$

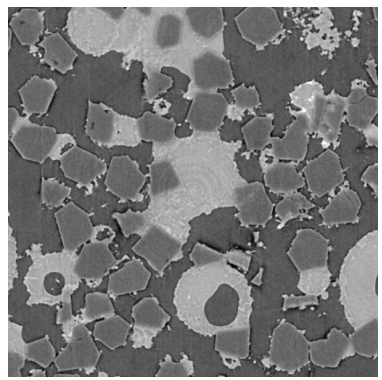
- Linear forward operator consisting of blur and downsampling.
- Additive Gaussian noise.
- One reference image from the same material is available.



Unknown HR image x



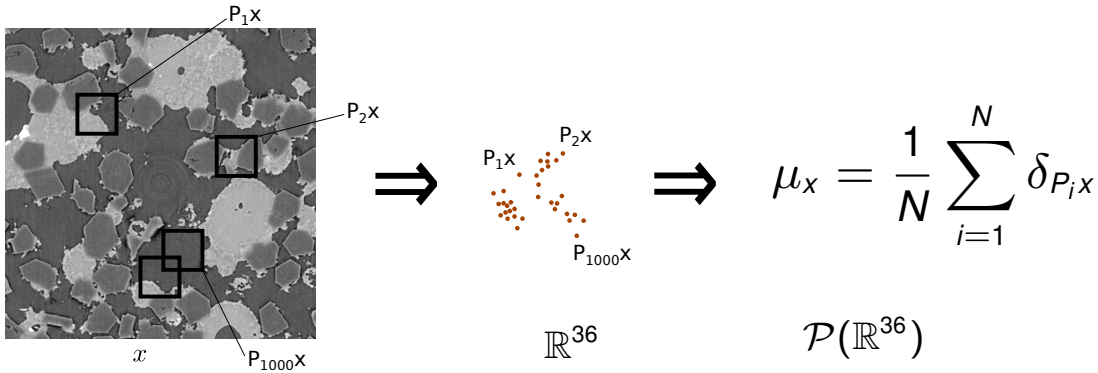
LR image y



Reference image $\tilde{x}$

Wasserstein Patch Prior

1. Compute the empircal patch distributions distributions μ_x and $\mu_{\tilde{x}}$:



2. Wasserstein Patch Prior (WPP):

$$\mathcal{R}(x) := W_2^2(\mu_x, \mu_{\tilde{x}}).$$

3. Use gradient descent to compute the MAP estimator

$$\hat{x} \in \operatorname{argmin} \tfrac{1}{2}\|F(x) - y\|^2 + \lambda W_2^2(\mu_x, \mu_{\tilde{x}}).$$

Theorem

The function $x \mapsto \exp(-\lambda W_2^2(\mu_x, \mu_{\tilde{x}}))$, $\lambda > 0$, is integrable.

Refs: H. et al. 2022, Altekrüger/H. 2022, Related: Houdard et al. 2021, Pinetz et al. 2021

Derivative of the Wasserstein Distance

- Dual formulation of the Wasserstein Distance

$$\mathcal{R}(x) := W_2^2(\mu_x, \mu_{\tilde{x}}) = \max_{\psi \in \mathbb{R}^{\tilde{N}}} \underbrace{\left(\frac{1}{N} \sum_{j=1}^{N} \psi^c\Big(P_j(x)\Big) + \frac{1}{\tilde{N}} \sum_{k=1}^{\tilde{N}} \psi_k \right)}_{H(\psi, x)}$$

  with $\psi^c(x) = \min_{j \in \{1,\dots,\tilde{N}\}} \{ \frac{1}{2} \|x - P_j \tilde{x}\|^2 - \psi_j \}$.

- Then one can show that

$$\nabla_x \max_{\psi} H(\psi, x) = \nabla_x H(\psi^*, x), \qquad \psi^* \in \operatorname*{argmax}_{\psi} H(\psi, x).$$

$\rightarrow$ One gradient descent step reads as

1. Compute $\psi^* \in \operatorname{argmax}_{\psi} H(\psi, x^{(r)})$ $\leftarrow$ Concave problem!
2. Set $x^{(r+1)} := \nabla F(x^{(r)})^{\mathrm{T}} (F(x^{(r)}) - y) + \nabla_x H(\psi^*, x)$.

Refs.: Houdard et al. 2021, H. et al. 2022.

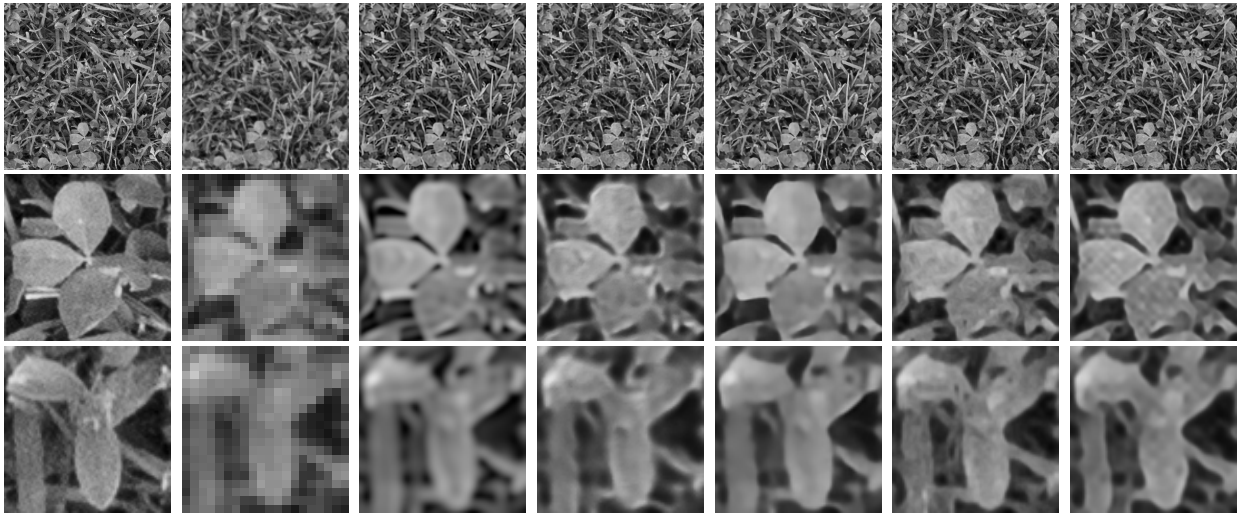
WPPNets

- Solving this min-max-problem is computationally costly.
- Several low-resolution observations $y_1, \ldots, y_m$ available.
- Train a neural network, which maps a low-resolution image onto the MAP estimator.

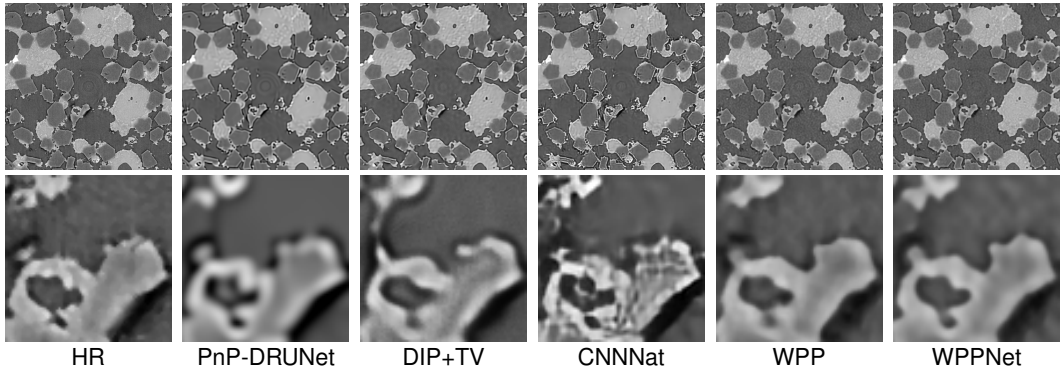
$\rightarrow$ Loss function:

$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \| F(G_\theta(y_i)) - y_i \|^2 + \lambda W_2^2(\mu_{G_\theta(y_i)}, \mu_{\tilde{x}}), \qquad x = G_\theta(y)$$

Superresolution with WPPNets



WPPNet with 2D texture (magnification 4)



Robustness against perturbations of the operator

Contents

Normalizing Flows

Patch Normalizing Flows - MAP Estimation

Wasserstein Patch Prior - MAP Estimation

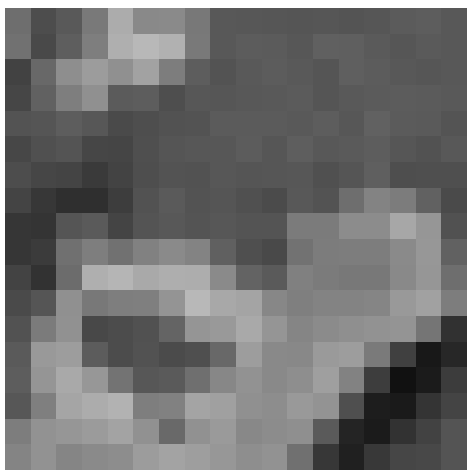
Wasserstein Patch Prior - Posterior Sampling

Posterior Sampling

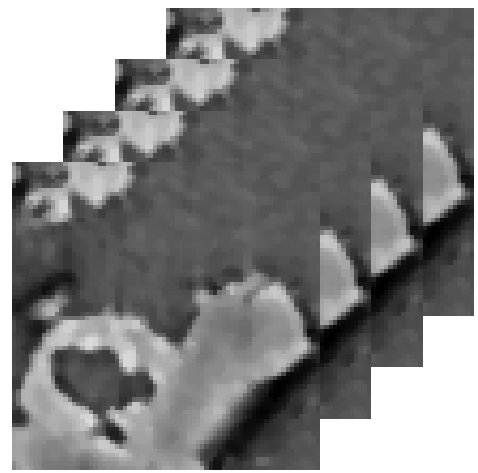
Given: Consider the Baysian inverse problem $Y = F(X) + \Xi$.

- F forward operator, Ξ Gaussian Noise
- Several LR observations $y_1, \ldots, y_m$.

Aim: Sample from the posterior distribution $P_{X \mid Y=y}$.



low resolution observation y



high resolution space $P_{X \mid Y=y}$

WPPFlows as Conditional Normalizing Flows

Goal: $P_{X|Y=y} \approx \mathcal{T}(y;\cdot)_{\#}P_Z$ for all y.

$\rightarrow$ $\mathcal{T}(y;\cdot)$ has to be invertible for any y. Loss functions:

- Forward KL: $\mathcal{J}_{\text{cNF}}(\theta) = \mathbb{E}_{y\sim P_Y}[\mathrm{KL}(P_{X|Y=y}, \mathcal{T}(y,\cdot)_{\#}P_Z)]$
  $\rightarrow$Supervised, forward operator F and prior density p_X might be unknown
- Backward KL: $\tilde{\mathcal{J}}_{\text{cNF}}(\theta) = \mathbb{E}_{y\sim P_Y}[\mathrm{KL}(\mathcal{T}(y,\cdot)_{\#}P_Z, P_{X|Y=y})]$
  $\rightarrow$Unsupervised, forward operator F and prior density p_X must be known

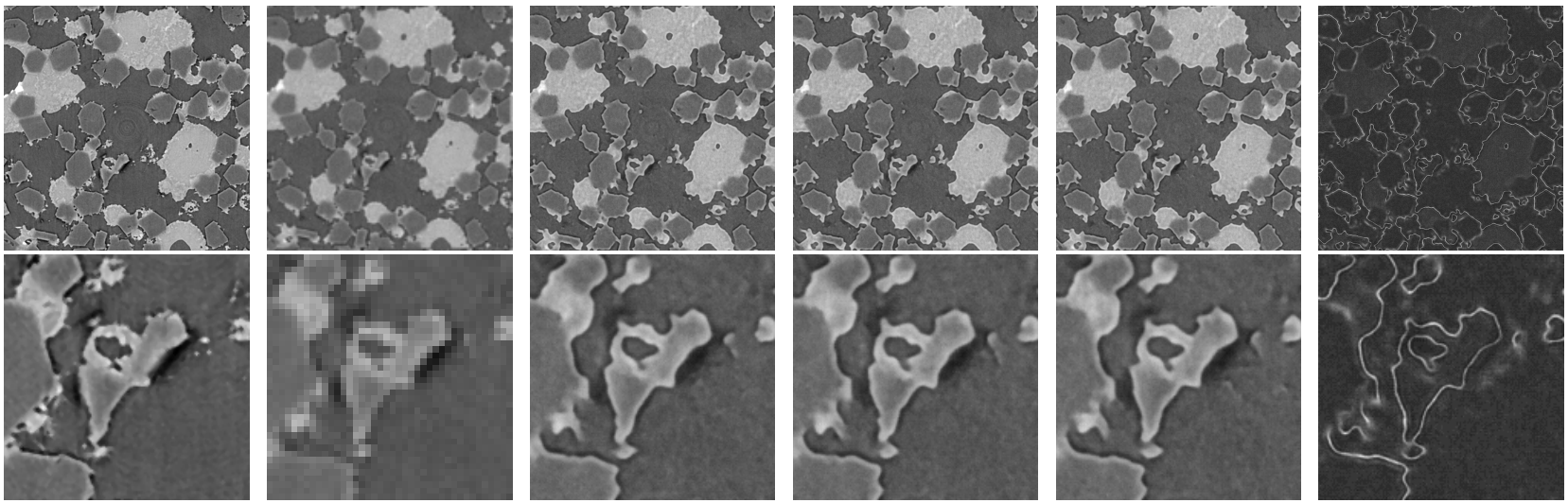
WPPFlows: Use the backward KL and insert $p_X(x) \propto \exp(-\rho W_2^2(\mu_x, \mu_{\tilde{x}}))$

$$\mathcal{L}(\theta) := \mathbb{E}_{y\sim P_Y}\left[\mathrm{KL}(\mathcal{T}(\cdot,y)_{\#}P_Z, P_{X|Y=y})\right]$$

$$\propto \mathbb{E}_{y\sim P_Y}\left[\mathbb{E}_{z\sim P_Z}\left[-\log p_{Y|X=x}(y) - \log p_X(\mathcal{T}(z)) - \log|\det \nabla\mathcal{T}(z;y)|\right]\right]$$

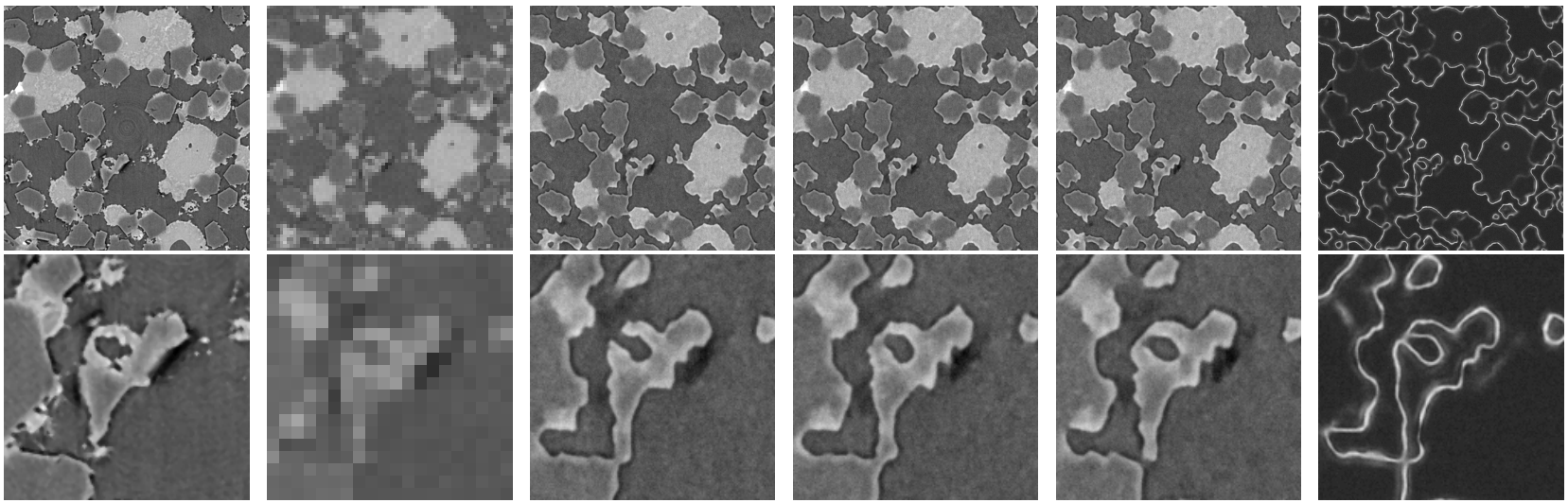
$$\propto \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{z\sim P_Z}\left[\frac{1}{2\sigma^2}\|f(\mathcal{T}_\theta(z;y_i)) - y_i\|^2 + \rho W_2^2\left(\mu_{\mathcal{T}(z;y_i)}, \mu_{\tilde{x}}\right) - \log|\det \nabla\mathcal{T}_\theta(z;y_i)|\right].$$

Uncertainty Quantification - magnification factor 4



HR image | LR image | Prediction 1 | Prediction 2 | Prediction 3 | Standard deviation

Uncertainty Quantification - magnification factor 8



HR image | LR image | Prediction 1 | Prediction 2 | Prediction 3 | Standard deviation

Conclusions

The power of patches:

- Patch priors can be used for learning with very few training images.
- MAP estimation by solving variational problems.
- Posterior sampling by training normalizing flows using the backward KL.
- Applicable for many classes of images including CT, materials' microstructures, textures and natural images.

Possible extensions:

- Combination with stochastic normalizing flows.
- Posterior sampling with patchNR?
- Zero-shot superresolution with Wasserstein patch priors?

Technische Universität Berlin

Thank you for your attention!

F. Altekrüger, A. Denker, P. Hagemann, J. Hertrich, P. Maass and G. Steidl (2022).
PatchNR: Learning from Small Data by Patch Normalizing Flow Regularization.
(arXiv Preprint#2205.12021)
Code available at `https://github.com/FabianAltekrueger/patchNR`.

F. Altekrüger and J. Hertrich (2022).
WPPNets and WPPFlows: The Power of Wasserstein Patch Priors for Superresolution.
(arXiv Preprint#2201.08157)
Code available at `https://github.com/FabianAltekrueger/WPPNets`.

J. Hertrich, A. Houdard and C. Redenbach (2022).
Wasserstein Patch Prior for Image Superresolution.
IEEE Transactions on Computational Imaging, vol. 8, pp. 693–704
Code available at `https://github.com/johertrich/Wasserstein_Patch_Prior`.

P. Hagemann, J. Hertrich and G. Steidl (2021).
Stochastic Normalizing Flows: a Markov Chains Viewpoint.
SIAM/ASA Journal on Uncertainty Quantification, vol. 10, pp. 1162–1190.
Code available at `https://github.com/PaulLyonel/conditionalSNF`.