# Compressive learning of deep regularization for denoising

Hui Shi, Yann Traonmilin, Jean François Aujol

# Compressive learning of deep regularization for denoising

**Hui Shi** **Yann Traonmilin** **Jean-François Aujol**
Univ. Bordeaux, Bordeaux INP, CNRS, IMB, UMR 5251,F-33400 Talence, France.

## Abstract

Solving ill-posed inverse problems can be done accurately if a regularizer well adapted to the nature of the data is available. Such regularizer can be systematically linked with the distribution of the data itself through the maximum a posteriori Bayesian framework. Recently, regularizers designed with the help of deep neural networks received impressive success. Such regularizers are typically learned from voluminous training data. To reduce the computational burden of this task, we propose to adapt the compressive learning framework to the learning of regularizers parametrized by deep neural networks (DNN). Our work shows the feasibility of batchless learning of regularizers from a compressed dataset. In order to achieve this, we propose an approximation of the compression operator that can be calculated explicitly for the task of learning a regularizer by DNN. We show that the proposed regularizer is capable of modeling complex regularity prior and can be used to solve the denoising inverse problem.

## 1 Introduction

### 1.1 Modeling the problem

In this work, we are interested in solving the denoising inverse problem arising in signal processing. We aim at finding an accurate estimate $\hat{x}$ of the original signal $x \in \mathbb{R}^d$ from the observed noisy signal $y \in \mathbb{R}^d$. A general formulation of the forward model is

$$y = x + \epsilon, \tag{1}$$

where the measurement noise $\epsilon$ is independent of the signal of interest. It is assumed to be additive white Gaussian noise (AWGN) of standard deviation $\sigma$, i.e. $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_d)$. Recovering $x$ from its degraded version $y$ by inverting the observation model is usually an ill-posed problem as the solution of the problem is not unique or stable. One needs to use additional (prior) information about the unknown signal $x$ so that the estimation problem admits meaningful solutions. Hence, common strategies for solving inverse problems often define an estimator which minimizes an appropriate function of the form

$$\hat{x} \in \arg\min_x F(x) + \lambda R(x), \tag{2}$$

where $F$ is the data fidelity term making the solution consistent with the observation $y$ and $R$ is the regularization term weighted by $\lambda > 0$ that incorporates prior ensuring the stability of the solution (Demoment (1989)). The choice of regularization depends on the statistics of the signal of interest which is not always available in real-life applications.

The maximum a posteriori (MAP) Bayesian framework provides a useful tool to interpret such methods. From a probabilistic point of view, we consider the problem as a stochastic model:

$$y = x + \epsilon, \quad x \sim \mu, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_d), \tag{3}$$

where $\mu$ denotes a prior probability law (of density $\mu(\cdot)$) of the unknown data $x$. In this framework, the probability density of the likelihood of $y$ given $x$ is given by $p(y|x) \sim e^{-\frac{\|y-x\|_2^2}{2\sigma^2}}$. Using Bayes' rule, the posterior distribution of $x$ given by $y$ is derived by

$$p(x|y) \propto p(y|x)\mu(x). \tag{4}$$

The MAP estimator corresponds to:

$$\begin{aligned} \hat{x}_{\text{MAP}} &= \arg\max_x p(x|y) \\ &\propto \arg\min_x \|y - x\|_2^2 - \lambda log(\mu(x)) \end{aligned} \tag{5}$$

for an appropriate choice of $\lambda$. In this context, the regularizer is related to the prior distribution of the data, $i.e.$, $R(x) = -log(\mu(x))$.

It is not an easy task to accurately estimate the prior $\mu(x)$, especially in high-dimensional spaces. Classical Bayesian

approaches, e.g. in image processing, rely on explicit priors such as total variation ( Rudin et al. (1992); Chambolle (2004); Louchet and Moisan (2013)), Markov random field (MRF) ( Blake et al. (2011); Roth and Black (2005)) or Gaussian mixture models (GMM) ( Zoran and Weiss (2011); Yu et al. (2011)). Recently, researchers have proposed and studied the use of deep neural networks to design the regularizer. The methods such as the total deep variation (Kobler et al. (2020, 2021)), adversarial regularizers ( Lunz et al. (2018); Prost et al. (2021)), as well as the Plug & Play approach and its extensions (Venkatakrishnan et al. (2013); Zhang et al. (2021); Hurault et al. (2021)) deliver remarkably accurate results.

However, such models are typically learned from voluminous training data. Estimating their parameters from such a large-scale dataset is a serious computational challenge (in both time and memory requirements).

### 1.2 Compressive learning

One possibility to reduce the computational resources of the learning task consists in using the compressive learning (CL) framework (Keriven et al. (2018); Gribonval et al. (2020, 2021a,b,c)). The main idea of this framework, coined as sketching, is to compress the whole data collection into a fixed-size representation, a so-called *sketch* of data, such that enough information relevant to the considered learning task is captured. Then the learned parameters are estimated as a near-minimizer of a non-linear least-square problem built with the sketch. The size of sketch $m$ is chosen proportional to the intrinsic complexity of the learning task. Meanwhile, the cost of inferring the parameters of interest from the sketch does not depend on the number of data in the initial collection. Hence, it is possible to exploit arbitrarily large datasets in the sketching framework without demanding more computational resources.

More precisely, during the sketching phase, a huge collection of $n$ $d$-dimensional data vectors $X = \{x_i\}_{i=1}^n$ is summarized into a single $m$-dimensional ($m \ll n$) vector $\hat{z}$ with:

$$\hat{z} = \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i) = \mathcal{S}(\hat{\mu}_n), \tag{6}$$

where $\hat{\mu}_n := \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ the empirical probability distribution of the data, $\delta_{x_i}$ is the Dirac measure at $x_i$ and the function $\Phi : \mathbb{R}^d \to \mathbb{R}^m$ is called the feature map (typically random Fourier moments). The operator $\mathcal{S}$ is a linear operator such that the feature map $\Phi(\cdot)$ is integrable with respect to $\mu$, i.e.

$$\mathcal{S}\mu := \mathbb{E}_{X\sim\mu}\Phi(X). \tag{7}$$

Then an estimate of the distribution $\mu$ (or of distributional parameters $\theta$ of interest) is computed by solving the optimization problem:

$$\mu^* = \arg\min_{\mu} \|\hat{z} - \mathcal{S}\mu\|_2^2. \tag{8}$$

In practice, this "sketch matching" problem can be generally solved by greedy compressive learning Orthogonal Matching Pursuit (OMP) algorithm and its extension CL-OMP with replacement (Keriven et al. (2018)). When the distribution $\mu$ is a Gaussian mixture model (GMM) in high-dimension space with flat tail covariances, the problem can also be solved by the low-rank OMP algorithm (Shi et al. (2022)). These results show that the prior model learned from the compressed patch dataset with low-rank OMP can be used to perform image denoising.

These greedy algorithms are suitable for any sketching operator $\mathcal{S}$ and any distribution density $\mu$, as long as the sketch $\mathcal{S}\mu$ and its gradient with respect to the distributional parameters $\theta$ of interest have a closed-form expression. That is to say, the core of the OMP-based algorithms is computing the expression of $\mathcal{S}\mu$ and $\nabla_\theta\mathcal{S}\mu$. However, real-life data tends to be complex and needs to be modeled with complex distributions. In this case, the sketching feature map is not always integrable with respect to the prior density of the data or it may not have a closed-form. This limits the advantages of using the sketching framework in practice.

In this paper, we aim at recovering a good approximation of the probability distribution of any unknown data from its sketch (i.e. beyond Gaussian mixtures). As neural networks have great expressive power (Hornik et al. (1989); Pan and Srikumar (2016)), we propose to tackle the problems by adapting the sketching framework to neural networks. More precisely, we propose to define the regularizer $R_\theta$ parameterized by a DNN, precisely a ReLU network $f_\theta$, that is,

$$R_\theta(\cdot) = \|f_\theta(\cdot)\|_2^2. \tag{9}$$

Such a regularization corresponds to the parametric distribution density $\mu_\theta \propto e^{-\|f_\theta(\cdot)\|_2^2}$. Thus it can be viewed as a generalized Gaussian distribution, where the bilinear form induced by the covariance matrix is replaced by a network. Due to the fact that neural networks have good generalization properties, the proposed regularization should be capable of encoding complex probability distributions. We aim to apply the sketching framework to such parametric densities. Unfortunately, a direct practical application of existing tools is not possible as closed-form expressions of $\mathcal{S}\mu$ are not available for sketching operator $\mathcal{S}$ based on random Fourier features.

### 1.3 Contributions and outline

In this work, we show the feasibility of learning regularizers parametrized by a DNN from a compressed database. Once the network is trained, the regularizer can be used for inverse problems such as denoising.

To do so, we propose to approximate the sketching operator $\mathcal{S}\mu$ by a discrete version $\mathcal{S}_d\mu$ that can be calculated with closed-form expressions, and such that the approximation still permits to apply the sketch matching estimation method. The approximation is performed on a grid of the domain where the data is located.

To find an estimate of the distribution $\mu_\theta$ (of density $\propto e^{-\|f_\theta(\cdot)\|_2^2}$), we adapt the sketch matching problem with our approximate sketching operator in the following way:

$$\theta^* \in \arg\min_{\theta\in\Theta} \|\mathcal{S}_d\mu_\theta(p) - \hat{z}\|_2^2, \tag{10}$$

where $\Theta$ is a set were the DNN inducing the regularizer $R_\theta(\cdot) = \|f_\theta(\cdot)\|_2^2$ can be parametrized (i.e. weights and bias). This problem can be solved practically with gradient descent based methods.

There are various advantages of our proposed approach.

As we do not need the original dataset during the training process, then the training procedure does not need to build batches of data. As a consequence, *each gradient descent iteration in the training incorporates information from the whole original database*. Once the empirical sketch has been computed (in a single pass, possibly in parallel), the dataset can be removed from memory. This reduces the memory complexity of the learning task. Moreover, the Jacobian $\nabla\mathcal{S}\mu$ can be computed efficiently with back-propagation.

Our approach overcomes the limits of greedy learning algorithms of the original sketching framework: regardless of the complexity of the data distribution, the proposed sketching operator allows us to always have a closed form expression of $\mathcal{S}_d\mu_\theta$. Thus, the sketching is no longer limited to the distribution densities for which the Fourier transform is explicit.

As a result, the learned regularizer can be used to solve inverse problems. The effectiveness of the proposed scheme is tested on synthetic examples and real dataset. Due to the limitations of our approximation of the sketching operator, the feasibility is illustrated on 2-D and 3-D data with possibly complex distributions. Our work thus opens the broader open question of designing closed form sketching operators in high dimension.

The rest of this article is organized as follows. We start by introducing the sketching framework, ReLU networks and some related works in section 2. In section 3, we then describe the proposed framework: the adaptation of the compressive learning framework to the learning of regularizers parameterized by ReLU networks. We explain in Section 4 how is solved the denoising variational problem (5). Section 5 illustrates the performance of the proposed methods on both synthetic data and real-life data. Finally, conclusions are drawn in section 7.

# 2 Background, related works

## 2.1 Sketching

Sketching is a statistical compressive learning technique. Let $\mathcal{D}$ be the set of probability measures over $\mathbb{R}^d$. We suppose that the data samples $x_i$ are modeled as i.i.d. random vectors having an unknown probability distribution with density $\mu_\mathcal{X} \in \mathcal{D}$. We define the linear sketching operator $\mathcal{S}$ that maps the probability distribution $\mu_\mathcal{X}$ to the $m$-dimensional sketch vector $z$:

$$\begin{aligned} &\mathcal{S} : \mathcal{D} \to \mathbb{C}^m \\ &z = \mathcal{S}\mu_\mathcal{X} := \int_{\mathbb{R}^d} \mu_\mathcal{X}(x)\Phi(X)dx. \end{aligned} \tag{11}$$

When the transformation (sketching feature map) $\Phi(\cdot)$ is built with random frequencies of the Fourier transform, for $l = 1, \dots, m$, the $l$-th component of the sketch is

$$z_l = \int_{\mathbb{R}^d} e^{-j<\omega_l, x>}\mu_\mathcal{X}(x)dx, \tag{12}$$

where $\{\omega_l\}_{l=1}^m \in \mathbb{R}^d$ are $d$-dimensional frequencies drawn at random. Taking a statistical perspective, the components $z_l$ can be seen as samples of the characteristic function of $\mu_\mathcal{X}$. Accordingly, the empirical sketch $\hat{z}$ can be computed from the samples of the database as

$$\hat{z}_l = \frac{1}{n}\sum_{i=1}^{n} e^{-j<\omega_l, x_i>} \quad l = 1, \dots, m. \tag{13}$$

It was shown (Keriven et al. (2018); Gribonval et al. (2020, 2021b)) that when the probability distribution $\mu$ has a low dimension structure, e.g. a GMM, one can recover it (with high probability) from enough randomly chosen samples of its Fourier transform. The required size of the sketch is typically of the order of the number of parameters we need to estimate.

## 2.2 ReLU network

A ReLU network, that we denote by $f_\theta$, is defined as a fully connected, feed-forward network (multi-layer perceptrons) with rectified linear unit (ReLU) activations. The ReLU function is given by: $ReLU(x) = max(x, 0)$. This activation has grown in popularity in feed-forward networks due to the success of first-order gradient based heuristic algorithms and the improvement in convergence to the approximated function for training (Nair and Hinton (2010)). We consider that the ReLU network has $K$ hidden layers and each layer indexed by $k$ has $n_k$ neurons. Note that the output layer is a linear layer without ReLU activation. As the network is a weighted graph with bias, the network parameters are the weight matrices $W^k \in \mathbb{R}^{n_k \times n_{k-1}}$ and bias vectors $b^k \in \mathbb{R}^{n_k}$, i.e., $\theta = \{W^k, b^k\}_{k=1}^{K+1}$. Formally, given $n_0$ input data $\{x_i\}_{i=1}^{n_0}$, $n_{K+1}$ output data $\{y_i\}_{i=1}^{n_{K+1}}$, and a

loss function $L : \mathbb{R}^{n_0} \times \mathbb{R}^{n_{K+1}} \to \mathbb{R}_+$, the training task is to determine the network parameters $\theta^*$ such that

$$\theta^* = \arg\min_{\theta} L(f_\theta(x), y). \tag{14}$$

If $L$ is differentiable, this problem can be solved by gradient descent based methods. The gradients of the network can be computed efficiently with back-propagation.

### 2.3 Related works

The sketching frameworks has been successfully applied to parametric models including GMMs (Keriven et al. (2018); Gribonval et al. (2021b); Shi et al. (2022)) and $K$-means clustering (Keriven et al. (2018); Gribonval et al. (2021b)); classification (Schellekens and Jacques (2018)); as well as component analytic (Sheehan et al. (2019b)). A reformulation of the original sketching framework has been applied to semi-parametric models (Sheehan et al. (2019a)). The sketching has been also incorporated into neural networks once (Schellekens and Jacques (2020)). Note that in their work, the authors combine the sketching with the generative networks to generate data samples. While in our work, we aim to cast the sketching framework to the ReLU network to learn a regularizer for solving the inverse problem. In addition, the authors proposed to approximate the sketching map by Monte-Carlo sampling. In our approach, we propose to do the approximation with a discrete sketching operator. The sketching frameworks mentioned above focus on data-independent approximation, i.e. the sketches are obtained by averaging random features. In Chatalic et al. (2022), the authors propose to perform the sketching based on a Nyström approximation, the latter is data-dependent and shows empirically better performances for k-means clustering and Gaussian modeling.

The sketching mentioned in our work reduces the dimensionality by performing a linear "projection" of the probability distribution of the data set $\{x_i\}_{i=1}^n$. This differs from the approach where the dimensionality reduction is carried out on the data $x_i$ themselves like Daniely et al. (2016, 2017). The sketching framework also differs from the "random sketching strategies" scheme described in Williams and Seeger (2000); Gittens and Mahoney (2013). The latter random sketching is proposed for kernel methods. It refers to selecting representative elements from the kernel matrix to reduce the computational burden. Hence our method also differs from the method proposed in Wang et al. (2021) in which the authors designed random sketching strategies for ReLU networks.

## 3 Sketching densities parametrized by DNN

In this section, we explain how we adapt the sketching framework to estimate regularizations by DNN. Intuitively, since ReLU networks define piecewise affine functions, we can indeed express a ReLU network $f_\theta$ as:

$$f_\theta(x) = \sum_{\gamma=1}^{N_R} \mathbf{1}_{R_\gamma}(x)(W_\gamma x + b_\gamma), \tag{15}$$

where $\mathbf{1}_{R_\gamma}$ is the indicator function on each linear region $R_\gamma$. The domain is partitioned into $N_R$ linear regions within which $f$ corresponds to an affine function. We characterize each linear region by the set of units that are active in that domain. We have that $W_\gamma = W^{(K+1)} W_\gamma^{(K)} \cdots W_\gamma^{(1)}$ where $W_\gamma^k$ is obtained by setting the $i$-th coordinate of $W^k$ to 0 whenever the neuron $i$ of the $k$-th layer is not active.

Given a dataset $X$, we aim at learning, from only the sketch $z$, an approximation $\mu_\theta$ for the probability distribution $\mu$ generating $X$. As considering a regularizer of the form $R_\theta(\cdot) = \|f_\theta(\cdot)\|_2^2$, it corresponds to parametric densities of the form $\mu_\theta(\cdot) \propto e^{-R_\theta(\cdot)}$. Ideally, with the definition in (12), the sketch would have to be calculated as

$$\begin{aligned} z_l &= \int_{\mathbb{R}^d} e^{-j<\omega,x>} e^{-\|f_\theta(x)\|_2^2} dx \\ &= \int_{\mathbb{R}^d} e^{-j<\omega,x>} e^{-\| \sum_i^{N_R} \mathbf{1}_{R_i}(x)(W_i x + b_i)\|_2^2} dx \\ &= \int_{x_d} \cdots \int_{x_1} e^{-j \sum_{p=1}^d \omega_p x_p} \\ &\quad e^{-\sum_{p=1}^d (\sum_i^{N_R} \mathbf{1}_{R_i}(x)((W_i x)_p + b_{ip}))^2} dx_1 \cdots dx_d. \end{aligned} \tag{16}$$

However, to the best of our knowledge, there is no analytic expression of such Fourier transform (Fourier transform on polygons). To tackle this issue, we consider approximating the continuous Fourier transform on a set of discrete points. This can be done by approximating the integral in the Fourier transform as a Riemann sum. The Riemann sum approximation can be expressed in terms of the discrete Fourier transform.

To be specific, we define an approximation $\mathcal{S}_d : \mathbb{R}^d \to \mathbb{C}^m$ of the sketching operator $\mathcal{S}$ such that $\mathcal{S}_d \mu_\theta(\omega) \approx \mathcal{S}\mu(\omega)$ for a given frequency $\omega$. The approximated sketch $\tilde{z}$ then has components:

$$\begin{aligned} \tilde{z}_l &= |\Delta\Omega| \sum_{p_i \in \Omega} e^{-j<\omega_l, p_i>} \mu_\theta(p_i) \\ &= |\Delta\Omega| \sum_{p_i \in \Omega} e^{-j<\omega_l, p_i>} e^{-\|f_\theta(p_i)\|_2^2}, \end{aligned} \tag{17}$$

where $p_i$ is a point in the $d$-dimensional cell $\Omega$ with volume $|\Delta\Omega|$. Of course, the major pitfall of this approximation is the limitation for applications in high dimension as the number of points grows with respect to the dimension $d$. The required boundedness (or approximate boundedness such as in the Gaussian case) of the data is a valid assumption in many practical applications in signal and image processing.

As a consequence, given $N$ points $\{p_i\}_{i=1}^N$ on the grid where the dataset $X$ lives and the empirical sketch defined as (6), we consider a ReLU network sketch matching problem as finding the network parameters $\theta^*$ in the set $\Theta$ of possible parametrizations, such that

$$\begin{aligned}\theta^* &= \arg\min_{\theta\in\Theta} L\left(\tilde{z}_\theta, \hat{z}\right) \\ &= \arg\min_{\theta\in\Theta} \|\tilde{z}_\theta - \hat{z}\|_2^2 \\ &= \arg\min_{\theta\in\Theta} \|\mathcal{S}_d\mu_\theta - \hat{z}\|_2^2\end{aligned} \tag{18}$$

With the discretization, if $\mu_\theta$ is differentiable at point $p_i$, the gradient of $\mathcal{S}_d\mu_\theta$ with respect to the parameters $\theta$ can be computed easily by

$$\begin{aligned}&\nabla\mathcal{S}_d\mu_\theta(p_i) = \mathcal{S}_d\nabla\mu_\theta(p_i) \\ &= -2|\Delta\Omega| \sum_{p_i\in\Omega} e^{-j<\omega,p_i>} e^{-\|f_\theta(p_i)\|_2^2} f_\theta(p_i)\nabla f_\theta(p_i),\end{aligned} \tag{19}$$

where the gradient of the network $\nabla f_\theta(p_i)$ can be easily computed using the automatic differentiation. Note that the discretization is used only in the estimation of the regularizer from the sketch. It thus only impacts the calculation time and memory requirement of the estimation of the regularizer and not the size of the compressed dataset itself.

## 4 Denoising

With the learned regularization term, one can solve the variational problem (5) by minimizing the following function:

$$\begin{aligned}G(x) &= \|x - y\|_2^2 + \lambda R_\theta(x) \\ &= \|x - y\|_2^2 + \lambda\|f_\theta(x)\|_2^2.\end{aligned} \tag{20}$$

The optimization problem can be solved by gradient descent based methods. Let $x^t$ the data at iteration $t$, the step writes

$$x^t = x^{t-1} - \eta\left((y - x^{t-1}) + \lambda\nabla R_\theta(x^{t-1})\right) \tag{21}$$

where $\eta > 0$ is the learning rate. Similarly, we can compute the gradient by using automatic differentiation.

Also, note that this denoising method can easily be extended to other linear inverse problems, such as interpolation and deconvolution by including the corresponding forward measurement operator.

## 5 Experimental results with synthetic data

To validate the proposed framework, we first test it against 2-D and 3-D synthetic problems. The used training datasets are made of $n = 1000$ samples which are generated from: a spiral with parameters: the radius of circular curve R= 0.3 to 1, spiral length $L = 2\pi$, i.e. $\{(R_i, L_i)\}_{i=1}^n$, where $R_i \sim_{i.i.d.} \frac{L_i}{2\pi}$ and $L_i \sim_{i.i.d.} \mathcal{U}([0, 2\pi))$; and a zero-mean GMM of 2 Gaussians. The proposed approach is implemented with the PYTORCH framework. The source code to reproduce the experiments is available at Anonymous (2022). It contains parts of code taken from the Python Compressive Learning toolbox (Schellekens (2020)).

For the 2-D experiments, we compress the datasets into sketches of size $m = 100$. The network $f_\theta$ is designed as a ReLU network with 3 fully connected hidden layers with 64, 128, and 256 neurons in each layer respectively. To train the network, we use the Adam optimizer (Kingma and Ba (2014)) with a learning rate of value $10^{-6}$. The number of points on the grid is set to $N = 20^d$ where $d$ denotes the data dimension. For comparison, we propose to learn the regularizer on the non compressed dataset using the same network with the following learning objective function:

$$\theta' = \arg\min_\theta \sum_{i=1}^N \left\|\|f_\theta(p_i)\|_2^2 - dist_i\right\|_2^2, \tag{22}$$

where $dist_i = \min_j \|x_j - p_i\|_2^2$ is the distance between the data $x_j$ and its nearest grid point. This objective function imposes a regularizer that is close to the function "distance to the model". Note that for the non compressive learning, as we do not go through a (implicit) model of the density, we explicitly give the distance value, which is not necessary when using our proposed sketched method.

Figure 1 shows the experimental results of 2-D synthetic data. The first row shows the synthetic spiral and GMM samples for training. We compare distribution densities learned from a compressive dataset with the proposed framework ($2nd$ row) and distribution densities learned from a non compressive dataset ($3rd$ row). In the compressive approach, the dataset is compressed by a factor of 20, while producing comparable results, indicating that our approach accomplishes its objective: learning efficiently a probability density prior from a compressed database (which will be evaluated when used as a regularization on real audio data).

Figure 2 shows an experimental result of 3-D synthetic spiral data. The network $f_\theta$ is a ReLU network with 5 fully connected hidden layers of 64, 128, 192, 192, and 192 neurons respectively. The data is compressed to a sketch of size $m = 200$. This figure demonstrates again the capacity of our method to estimate accurate complex 3-D distribution densities from the compressed dataset.

### 5.1 Denoising results

We evaluate the regularizer learned with our method on white Gaussian noise denoising problem. Figure 3 illustrates the 2-D denoising results using regularizers learned

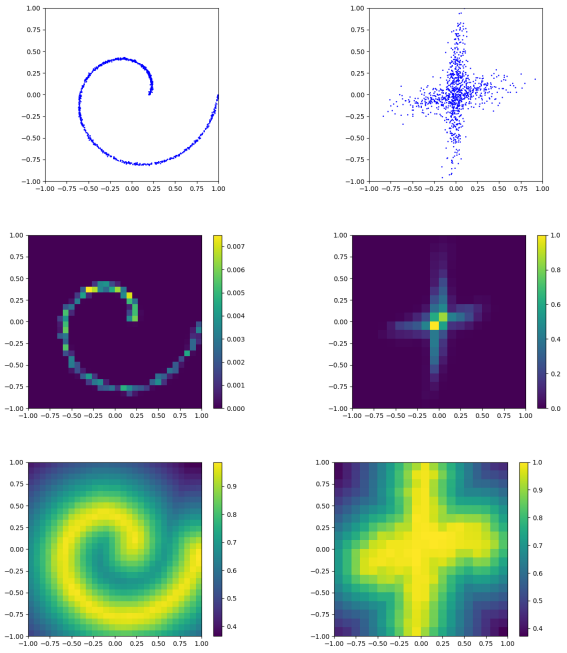

Figure 1: The distribution densities of sample data (top) learned from compressed dataset with sketching (middle) and original non compressed dataset (bottom).

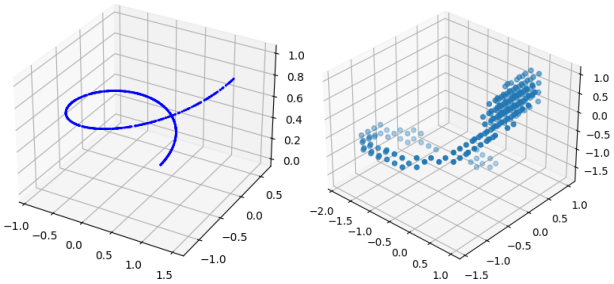

Figure 2: The 3-D spiral data samples (left) and the distribution density learned with the proposed approach (right, points of the learned distribution on a grid exceeding a given threshold).

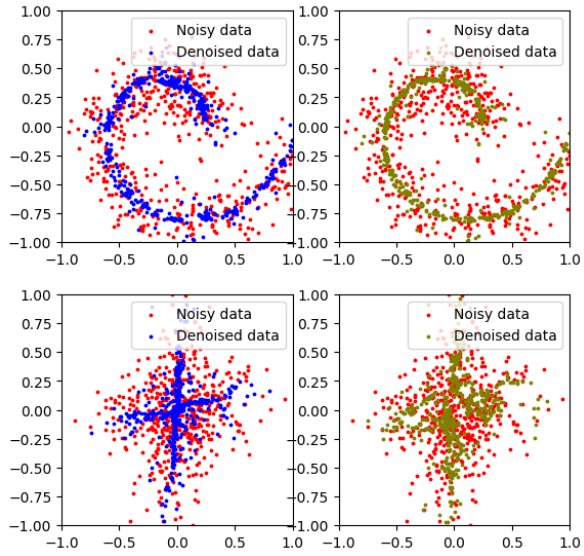


Figure 3: Denoising results with regularizers learned with the compressed (left) and the non compressed dataset (right). The noise level is set to $\sigma^2 = .15$.

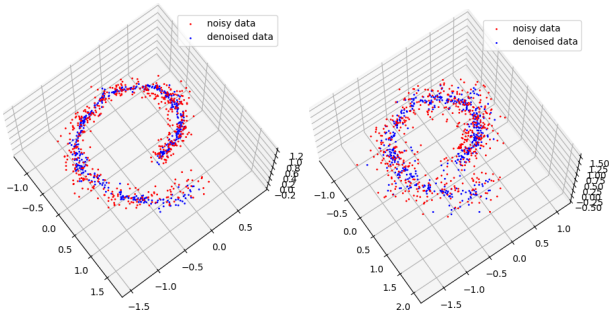


Figure 4: 3-D Denoising results with densities learned from the proposed method. The noise level is set to $\sigma^2 = 0.15$ (left) and $\sigma^2 = 0.2$ (right).

from the compressed dataset (left) and the original dataset (right). Figure 4 shows the 3-D denoising results with different noise level.

Results show that regularizers trained from the compressed datasets have comparable denoising performance with the regularizers trained on their original non compressed datasets.

### 5.2 Robustness to noise during training

To evaluate how robust our proposed method is to noise level during the training process, we train the regularizer on a compressed dataset of data samples generated with noise level $\sigma^2_{train} = 0.15$. We use the same network and training procedure as described above. Figure 5 shows the distribution density of the sampled data learned from the compressive noisy dataset and the denoising result. The result shows that the regularizer trained with a compressed noisy dataset has good denoising performance. This illustrates that our approach is robust to low noise level. This is easy to understand due to the fact that adding Gaussian noise corresponds to a convolution of the density with a Gaussian kernel, which does not change the shape of the distribution if small enough. It is even possible to add a deconvolution term to the distribution parameter estimation if the noise level in the training dataset is known.

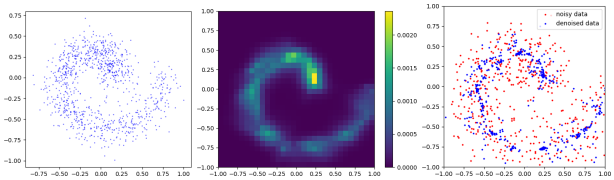

Figure 5: Denoising result (right) with densities (middle) learned from the compressived noisy data (left) with noise level $\sigma^2_{train} = 0.15$. The noise level of the data is set to $\sigma^2 = 0.2$.

### 5.3 Training parameters

We perform different experiments to evaluate potential factors that could affect the results. First we assess the compression rate, $i.e.$ the sketch size. Figure 6 shows the denoising results with regularization functions learned from sketches of different sizes. The original dataset is compressed 10 and 5 times respectively. It is shown that the proposed approach is capable to learn distributions from databases with high compression ratios. The number of

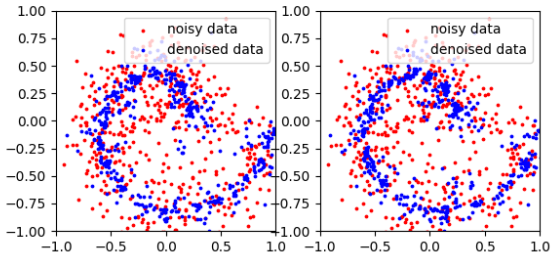


Figure 6: Denoising results with regularizers learned with different sketch size (left: $m = 100$, right: $m = 200$)

frequencies $\{\omega\}_{l=1}^{m}$ used to compute sketches affects memory storage, while the number of grid points used during training process affects the learning time. The number of grid points should be chosen well if we want to control the learning time well, since the number of grid points $N$ grows exponentially with respect to the dimension $d$.

Experiments from Figure 7 also show that overparameterized networks (with more layers or more neurons per layer) can achieve better results while using less learning time and fewer necessary grid points.

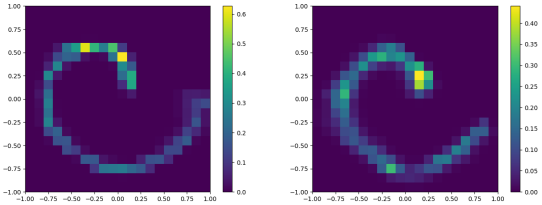

Figure 7: Regularizers learned via the proposed method with different learning parameters. Using the same number of grid points, we have better result when the network has more neurons. (Left) 3 hidden layers with 64, 128, 256 neurons in each layer. (Right) 3 hidden layers with 64, 128, 192 neurons in each layer.

## 6 Application for audio denoising

In order to evaluate the effectiveness of our approach, we test it on the audio denoising task. The experiment presented below is performed on recorded musical notes (monophonic 16kHz audio snippets) from the NSynth dataset (Engel et al. (2017)). The training data is an extracted 0.125s audio recorded from an acoustic guitar. After filtering the normalized audio data $s$ by two 4th-order Butterworth (Butterworth et al. (1930)) low-pass filters $h_1$ and $h_2$ with a cutoff frequency of 1.5kHz and 3.75kHz, three frequency responses are constructed with $s_1 = h_1 * s$, $s_2 = h2 * (s - s_1)$, and $s_3 = s - s2$. Then the frequency responses are concatenated, hence the training set is of dimension $2000 \times 3$; i.e. 2000 samples in dimension 3. Figure 8 shows a representation of the training dataset. The regularizer is learned from a sketch of size $m = 200$,

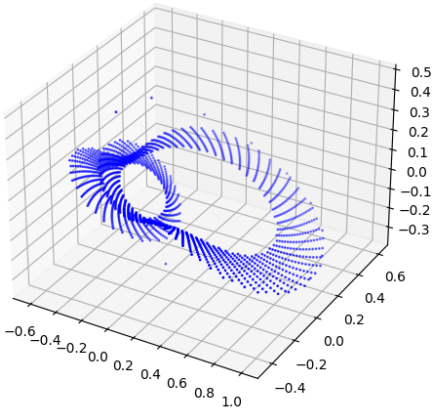

Figure 8: The training dataset for the experiment on audio file.

$i.e.$ the dataset is compressed by a factor of 30. Once the regularizer is learned, it is used to denoise the audio corrupted by Gaussian white noise of noise levels $\sigma^2 = 0.1$ and $\sigma^2 = 0.2$. When the regularizer is used for denois-

ing, we use the measurement signal-to-noise ratio (SNR) (Quackenbush et al. (1988)) to evaluate its effectiveness. The SNR is defined as

$$SNR = 10log_{10}\frac{P_{signal}}{P_{noise}} \tag{23}$$

where $P_{signal}$ and $P_{noise}$ are the total energy of the signal $x$ and noise respectively.

Figure 9 and 10 show the audio denoising result with different noise levels. In the two cases, we gain more than 1dB on SNR in the case of small noise and more than 3dB SNR in case of large noise. The results show that, in addition to the original low-pass filtering effect, denoising can be achieved in low dimensions even when temporal consistency between individual samples is not guaranteed. These first feasibility results for solving inverse problem using regularizers learned from sketch are promosising if we could use a sketching operator in higher dimensions.

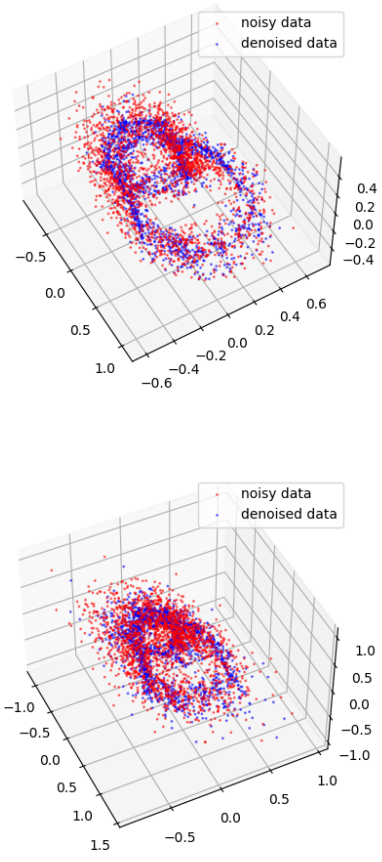


Figure 9: Audio denoising performances for different noise levels: (top) $\sigma^2 = 0.1$, SNR is 12.82 for noisy data and 13.85 for denoised data; (bottom) $\sigma^2 = 0.2$, SNR is 7.03 for noisy data and 10.26 for denoised data.

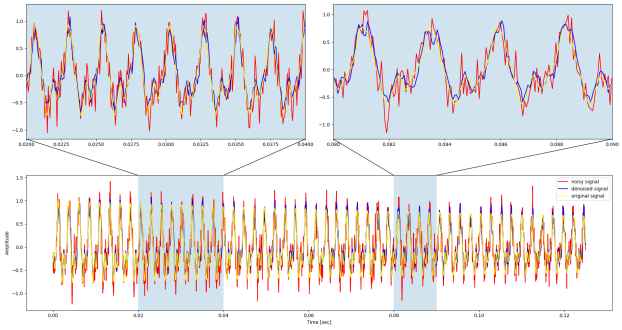

Figure 10: Audio denoising performance for noise level $\sigma^2 = 0.2$.

## 7 Conclusions

In this work, we illustrate the feasibility of adapting the compressive learning framework to the learning of a regularizer parameterized by a DNN. We achieve this by approximating the original sketching operator with a discrete one. With the proposed approximated sketching operator, the "sketch matching" problem can be solved with a gradient based algorithm. In addition, we define a new parametrization of the regularizer to solve the inverse problem. The regularizer is defined as the squared $\ell^2$-norm of a ReLU network and learned with a compressive dataset instead of the original dataset. It gathers the advantages of sketching which reduces the learning cost and of the neural networks which have great expressive power. Experiment results on 2-D/3-D synthetic data and audio data show that our method accomplish the objective of compressive learning in this context, illustrating the potential of sketched neural networks learning. However, our method relies on a discretization of the domain on which the data resides, which limits its use in high-dimensional domains ($e.g.$, for image denoising). Future works will be needed to overcome this limitation. We want to design a fast sketching operator that avoid such discretization. This leads to a major open question: can we find a sketching operator such that any distribution parametrized by a DNN can be estimated from the sketch? Does such a sketching operator exist? Also our experiments suggest that the overparameterization phenomenom that begins to be well understood in classical learning (Arora et al. (2018); Allen-Zhu et al. (2019)), could also be well behaved in the context of compressive learning. Hence understanding this regime in compressive learning is a also a key question in relation to this work.

### References

Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In 2019 International Conference on Machine Learning, pages 242–252. PMLR.

Anonymous, A. (2022). Code for this paper. To be released.

Arora, S., Cohen, N., and Hazan, E. (2018). On the optimization of deep networks: Implicit acceleration by overparameterization. In International Conference on Machine Learning, pages 244–253. PMLR.

Blake, A., Kohli, P., and Rother, C. (2011). Markov random fields for vision and image processing. MIT press.

Butterworth, S. et al. (1930). On the theory of filter amplifiers. Wireless Engineer, 7(6):536–541.

Chambolle, A. (2004). An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Vision, 20(1):89–97.

Chatalic, A., Carratino, L., De Vito, E., and Rosasco, L. (2022). Mean nyström embeddings for adaptive compressive learning. In 2022 International Conference on Artificial Intelligence and Statistics, pages 9869–9889. PMLR.

Daniely, A., Lazic, N., Singer, Y., and Talwar, K. (2016). Sketching and neural networks. arXiv preprint arXiv:1604.05753.

Daniely, A., Lazic, N., Singer, Y., and Talwar, K. (2017). Short and deep: Sketching and neural networks. 5th International Conference on Learning Representations.

Demoment, G. (1989). Image reconstruction and restoration: Overview of common estimation structures and problems. IEEE Transactions on Acoustics, Speech, and Signal Processing, 37(12):2024–2036.

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. In 2017 International Conference on Machine Learning, pages 1068–1077. PMLR.

Gittens, A. and Mahoney, M. (2013). Revisiting the nystrom method for improved large-scale machine learning. In 2013 International Conference on Machine Learning, pages 567–575. PMLR.

Gribonval, R., Blanchard, G., Keriven, N., and Traonmilin, Y. (2021a). Compressive statistical learning with random feature moments. Mathematical Statistics and Learning, 3(2):113–164.

Gribonval, R., Blanchard, G., Keriven, N., and Traonmilin, Y. (2021b). Statistical learning guarantees for compressive clustering and compressive mixture modeling. Mathematical Statistics and Learning, 3(2):165–257.

Gribonval, R., Chatalic, A., Keriven, N., Schellekens, V., Jacques, L., and Schniter, P. (2020). Sketching datasets for large-scale learning (long version). arXiv preprint arXiv:2008.01839.

Gribonval, R., Chatalic, A., Keriven, N., Schellekens, V., Jacques, L., and Schniter, P. (2021c). Sketching data sets for large-scale learning: Keeping only what you need. IEEE Signal Processing Magazine, 38(5):12–36.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366.

Hurault, S., Leclaire, A., and Papadakis, N. (2021). Gradient step denoiser for convergent plug-and-play. arXiv preprint arXiv:2110.03220.

Keriven, N., Bourrier, A., Gribonval, R., and Pérez, P. (2018). Sketching for large-scale learning of mixture models. Information and Inference: A Journal of the IMA, 7(3):447–508.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kobler, E., Effland, A., Kunisch, K., and Pock, T. (2020). Total deep variation for linear inverse problems. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7549–7558.

Kobler, E., Effland, A., Kunisch, K., and Pock, T. (2021). Total deep variation: A stable regularization method for inverse problems. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–1.

Louchet, C. and Moisan, L. (2013). Posterior expectation of the total variation model: properties and experiments. SIAM Journal on Imaging Sciences, 6(4):2640–2684.

Lunz, S., Öktem, O., and Schönlieb, C.-B. (2018). Adversarial regularizers in inverse problems. Advances in Neural Information Processing systems, 31.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Icml.

Pan, X. and Srikumar, V. (2016). Expressiveness of rectifier networks. In International Conference on Machine Learning, pages 2427–2435. PMLR.

Prost, J., Houdard, A., Almansa, A., and Papadakis, N. (2021). Learning local regularization for variational image restoration. In 2021 International Conference on Scale Space and Variational Methods in Computer Vision, pages 358–370. Springer.

Quackenbush, S. R., Barnwell, T. P., and Clements, M. A. (1988). Objective Measures of Speech Quality. Prentice-Hall, Englewood Cliffs, NJ.

Roth, S. and Black, M. J. (2005). Fields of experts: a framework for learning image priors. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2:860–867 vol. 2.

Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 60(1):259–268.

Schellekens, V. (2020). Pycle: a python compressive learning toolbox.

Schellekens, V. and Jacques, L. (2018). Compressive classification (machine learning without learning). arXiv preprint arXiv:1812.01410.

Schellekens, V. and Jacques, L. (2020). Compressive learning of generative networks. arXiv preprint arXiv:2002.05095.

Sheehan, M. P., Gonon, A., and Davies, M. E. (2019a). Compressive learning for semi-parametric models. arXiv preprint arXiv:1910.10024.

Sheehan, M. P., Kotzagiannidis, M. S., and Davies, M. E. (2019b). Compressive independent component analysis. In 2019 27th European Signal Processing Conference (EUSIPCO), pages 1–5. IEEE.

Shi, H., Traonmilin, Y., and Aujol, J.-F. (2022). Compressive learning for patch-based image denoising. SIAM Journal on Imaging Sciences, 15(3):1184–1212.

Venkatakrishnan, S. V., Bouman, C. A., and Wohlberg, B. (2013). Plug-and-play priors for model based reconstruction. In 2013 IEEE Global Conference on Signal and Information Processing, pages 945–948. IEEE.

Wang, D., Zeng, J., and Lin, S.-B. (2021). Random sketching for neural networks with relu. IEEE Transactions on Neural Networks and Learning Systems, 32(2):748–762.

Williams, C. K. I. and Seeger, M. (2000). Using the nyström method to speed up kernel machines. In 13th International Conference on Neural Information Processing Systems, NIPS'00, page 661–667, Cambridge, MA, USA. MIT Press.

Yu, G., Sapiro, G., and Mallat, S. (2011). Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. IEEE Transactions on Image Processing, 21(5):2481–2499.

Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., and Timofte, R. (2021). Plug-and-play image restoration with deep denoiser prior. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Zoran, D. and Weiss, Y. (2011). From learning models of natural image patches to whole image restoration. In 2011 International Conference on Computer Vision, pages 479–486. IEEE.