Regularized estimation of image statistics by Score Matching

Diederik P. Kingma
Department of Information and Computing Sciences
Universiteit Utrecht
d.p.kingma@students.uu.nl

Yann LeCun
Courant Institute of Mathematical Sciences
New York University
yann@cs.nyu.edu

Abstract

Score Matching is a recently-proposed criterion for training high-dimensional density models for which maximum likelihood training is intractable. It has been applied to learning natural image statistics but has so-far been limited to simple models due to the difficulty of differentiating the loss with respect to the model parameters. We show how this differentiation can be automated with an extended version of the double-backpropagation algorithm. In addition, we introduce a regularization term for the Score Matching loss that enables its use for a broader range of problem by suppressing instabilities that occur with finite training sample sizes and quantized input values. Results are reported for image denoising and super-resolution.

1 Introduction

Consider the subject of density estimation for high-dimensional continuous random variables, like images. Approaches for normalized density estimation, like mixture models, often suffer from the curse of dimensionality. An alternative approach is Product-of-Experts (PoE) [7], where we model the density as a product, rather than a sum, of component (expert) densities. The multiplicative nature of PoE models make them able to form complex densities: in contrast to mixture models, each expert has the ability to have a strongly negative influence on the density at any point by assigning it a very low component density. However, Maximum Likelihood Estimation (MLE) of the model requires differentiation of a normalizing term, which is infeasible even for low data dimensionality.

A recently introduced estimation method is Score Matching [10], which involves minimizing the square distance between the model log-density slope (score) and data log-density slope, which is independent of the normalizing term. Unfortunately, applications of SM estimation have thus far been limited. Besides ICA models, SM has been applied to Markov Random Fields [14] and a multi-layer model [13], but reported results on real-world data have been of qualitative, rather than quantitative nature. Differentiating the SM loss with respect to the parameters can be very challenging, which somewhat complicates the use of SM in many situations. Furthermore, the proof of the SM estimator [10] requires certain conditions that are often violated, like a smooth underlying density or an infinite number of samples.

Other estimation methods are Constrastive Divergence [8] (CD), Basis Rotation [23] and Noise-Contrastive Estimation [6] (NCE). CD is an MCMC method that has been sucessfully applied to Restricted Boltzmann Machines (RBM's) [8], overcomplete Independent Component Analysis
derivatives of the model density: is equivalent to minimizing the following expression, which involves only first and second partial density (the normalizing constant). Hyvärinen [10] shows that under some conditions, this objective underlying density at the data points, which is obviously independent of the vertical offset of the log-Score Matching works by fitting the slope (score) of the model density to the slope of the true, underlying density at the data points, which is obviously independent of the vertical offset of the log-density (the normalizing constant). Hyvärinen [10] shows that under some conditions, this objective is equivalent to minimizing the following expression, which involves only first and second partial derivatives of the model density:

\[
J(w) = \int_{x \in \mathbb{R}^N} p_x(x) \sum_{i=1}^{N} \left( \frac{1}{2} \left( \frac{\partial E(x; w)}{\partial x_i} \right)^2 - \frac{\partial^2 E(x; w)}{(\partial x_i)^2} \right) dx + \text{const} \tag{1}
\]

with \(N\)-dimensional data vector \(x\), weight vector \(w\) and true, underlying pdf \(p_x(x)\). Among the conditions 1 is that \(p_x(x)\) is differentiable, and (2) that the log-density is finite everywhere. In practice, the true pdf is unknown, and we have a finite sample of \(T\) discrete data points. The sample version of the SM loss function is:

\[
J^S(w) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \left( \frac{1}{2} \left( \frac{\partial E(x(t); w)}{\partial x_i} \right)^2 - \frac{\partial^2 E(x(t); w)}{(\partial x_i)^2} \right) \tag{2}
\]

which is asymptotically equivalent to the equation (1) as \(T\) approaches infinity, due to the law of large numbers. This loss function was used in previous publications on SM [10, 12, 13, 15].

2 Regularized Score Matching

Consider an energy-based [17] model \(E(x; w)\), where “energy” is the unnormalized negative log-density such that the pdf is: \(p(x; w) = e^{-E(x; w)} / Z(w)\), where \(Z(w)\) is the normalizing constant. In other words, low energies correspond to high probability density, and high energies correspond to low probability density.

Score Matching works by fitting the slope (score) of the model density to the data points, which is obviously independent of the vertical offset of the log-density (the normalizing constant). Hyvärinen [10] shows that under some conditions, this objective is equivalent to minimizing the following expression, which involves only first and second partial derivatives of the model density:

\[
J(w) = \int_{x \in \mathbb{R}^N} p_x(x) \sum_{i=1}^{N} \left( \frac{1}{2} \left( \frac{\partial E(x; w)}{\partial x_i} \right)^2 - \frac{\partial^2 E(x; w)}{(\partial x_i)^2} \right) dx + \text{const} \tag{1}
\]

with \(N\)-dimensional data vector \(x\), weight vector \(w\) and true, underlying pdf \(p_x(x)\). Among the conditions 1 is that \(p_x(x)\) is differentiable, and (2) that the log-density is finite everywhere. In practice, the true pdf is unknown, and we have a finite sample of \(T\) discrete data points. The sample version of the SM loss function is:

\[
J^S(w) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \left( \frac{1}{2} \left( \frac{\partial E(x(t); w)}{\partial x_i} \right)^2 - \frac{\partial^2 E(x(t); w)}{(\partial x_i)^2} \right) \tag{2}
\]

which is asymptotically equivalent to the equation (1) as \(T\) approaches infinity, due to the law of large numbers. This loss function was used in previous publications on SM [10, 12, 13, 15].

2.1 Issues

Should these conditions be violated, then (theoretically) the pdf cannot be estimated using equation (1). Only some specific special-case solutions exist, e.g. for non-negative data [11]. Unfortunately, situations where the mentioned conditions are violated are not rare. The distribution for quantized data (like images) is discontinuous, hence not differentiable, since the data points are concentrated at a finite number of discrete positions. Moreover, the fact that equation (2) is only equivalent to equation (1) as \(T\) approaches infinity may cause problems: the distribution of any finite training set of discrete data points is discrete, hence not differentiable. For proper estimation with SM, data can be smoothed by whitening; however, common whitening methods (such as PCA or SVD) are computational infeasible for large data dimensionality, and generally destroy the local structure of spatial and temporal data such as image and audio. Some previous publications on Score Matching apply zero-phase whitening (ZCA) [13] which computes a weighed sum over an input patch which removes some of the original quantization, and can potentially be applied convolutionally. However,

1. The conditions are: the true (underlying) pdf \(p_x(x)\) is differentiable, the expectations \(\mathbb{E} [\|\partial \log p_x(x) / \partial x\|^2]\) and \(\mathbb{E} [\|\partial E(x; w) / \partial x\|^2]\) w.r.t. \(x\) are finite for any \(w\), and \(p_x(x) \partial E(x; w) / \partial x\) goes to zero for any \(w\) when \(\|x\| \to \infty\).
the amount of information removed from the input by such whitening is not parameterized and potentially large.

2.2 Proposed solution

Our proposed solution is the addition of a regularization term to the loss, approximately equivalent to replacing each data point \( x \) with a Gaussian cloud of virtual datapoints \((x + \epsilon)\) with i.i.d. Gaussian noise \( \epsilon \sim \mathcal{N}(0, \sigma^2 I) \). By this replacement, the sample pdf becomes smooth and the conditions for proper SM estimation become satisfied. The expected value of the sample loss is:

\[
E \left[ J^S(x + \epsilon; w) \right] = \frac{1}{2} \sum_{i=1}^{N} \left( E \left[ \left( \frac{\partial E(x + \epsilon; w)}{\partial x_i + \epsilon_i} \right)^2 \right] - \sum_{i=1}^{N} \left( E \left[ \frac{\partial^2 E(x + \epsilon; w)}{(\partial(x_i + \epsilon_i))^2} \right] \right) \right)
\]

(3)

We approximate the first and second term with a simple first-order Taylor expansion. Recall that since the noise is i.i.d. Gaussian, \( E[\epsilon_i] = 0 \), \( E[\epsilon_i \epsilon_j] = E[\epsilon_i] E[\epsilon_j] = 0 \) if \( i \neq j \), and \( E[\epsilon_i^2] = \sigma^2 \).

The expected value of the first term is:

\[
\frac{1}{2} \sum_{i=1}^{N} E \left[ \left( \frac{\partial E(x + \epsilon; w)}{\partial x_i + \epsilon_i} \right)^2 \right] = \frac{1}{2} \sum_{i=1}^{N} E \left[ \left( \frac{\partial E(x; w)}{\partial x_i} + \sum_{j=1}^{N} \left( \frac{\partial^2 E(x; w)}{\partial x_i \partial x_j} \epsilon_j \right) + O(\epsilon_i^2) \right)^2 \right]
\]

(4)

The expected value of the second term is:

\[
\sum_{i=1}^{N} \left( E \left[ \frac{\partial^2 E(x + \epsilon; w)}{(\partial(x_i + \epsilon_i))^2} \right] \right) = \sum_{i=1}^{N} \left( E \left[ \frac{\partial^2 E(x; w)}{(\partial x_i)^2} + \sum_{j=1}^{N} \left( \frac{\partial^3 E(x; w)}{\partial x_i \partial x_j \partial x_j} \epsilon_j \right) + O(\epsilon_i^2) \right] \right)
\]

(5)

Putting the terms back together, we have:

\[
E \left[ J^S(x + \epsilon; w) \right] = \frac{1}{2} \sum_{i=1}^{N} \left( \frac{\partial E}{\partial x_i} \right)^2 - \sum_{i=1}^{N} \left( \frac{\partial^2 E}{(\partial x_i)^2} \right) + \frac{1}{2} \sigma^2 \sum_{i=1}^{N} \sum_{j=1}^{N} \left( \frac{\partial^2 E}{\partial x_i \partial x_j} \right)^2 + O(\epsilon^2)
\]

(6)

where \( E = E(x; w) \). This is the full regularized Score Matching loss. While minimization of above loss may be feasible in some situations, in general it requires differentiation of the full Hessian w.r.t. \( x \) which scales like \( O(W^2) \). However, the off-diagonal elements of the Hessian are often dominated by the diagonal. Therefore, we will use the diagonal approximation:

\[
J_{reg}(x; w; \lambda) = J^S(x; w) + \lambda \sum_{i=1}^{N} \left( \frac{\partial^2 E}{(\partial x_i)^2} \right)^2
\]

(7)

where \( \lambda \) sets regularization strength and is related to (but not exactly equal to) \( \frac{1}{2} \sigma^2 \) in equation (6). This regularized loss is computationally convenient: the added complexity is almost negligible since differentiation of the second derivative terms \( \frac{\partial^2 E}{(\partial x_i)^2} \) w.r.t. the weights is already required for unregularized Score Matching. The regularizer is related to Tikhonov regularization [22] and curvature-driven smoothing [2] where the square of the curvature of the energy surface at the data points are also penalized. However, its application has been limited since (contrary to our case) in the general case it adds considerable computational cost.

3
Figure 1: Illustration of local computational flow around some node $j$. Black lines: computation of quantities $\delta_j = \partial E / \partial g_j$, $\delta_j' = \partial^2 E / (\partial g_j)^2$ and the SM loss $J(x; w)$. Red lines indicate computational flow for differentiation of the Score Matching loss: computation of $\partial J / \partial g_j$ and $\partial J / \partial g_j'$. The influence of weights are not shown, for which the derivatives are computed in the last step.

3 Automatic differentiation of $J(x; w)$

In most optimization methods for energy-based models [17], the sample loss is defined in readily obtainable quantities obtained by forward inference in the model. In such situations, the required derivatives w.r.t. the weights can be obtained in a straightforward and efficient fashion by standard application of the backpropagation algorithm.

For Score Matching, the situation is more complex since the (regularized) loss (equations 2,7) is defined in terms of $\{\partial E / \partial x_i\}$ and $\{\partial^2 E / (\partial x_i)^2\}$, each term being some function of $x$ and $w$. In earlier publications on Score Matching for continuous variables [10, 12, 13, 15], the authors rewrote $\{\partial E / \partial x_i\}$ and $\{\partial^2 E / (\partial x_i)^2\}$ to their explicit forms in terms of $x$ and $w$ by manually differentiating the energy. Subsequently, derivatives of the loss w.r.t. the weights can be found. This manual differentiation was repeated for different models, and is arguably a rather inflexible approach. A procedure that could automatically (1) compute and (2) differentiate the loss would make SM estimation more accessible and flexible in practice.

A large class of models (e.g. ICA, Product-of-Experts and Fields-of-Experts), can be interpreted as a form of feed-forward neural network. Consequently, the terms $\{\partial E / \partial x_i\}$ and $\{\partial^2 E / (\partial x_i)^2\}$ can be efficiently computed using a forward and backward pass: the first pass performs forward inference (computation of $E(x; w)$) and the second pass applies the backpropagation algorithm [3] to obtain the derivatives of the energy w.r.t. the data point $\{\partial E / \partial x_i\}$ and $\{\partial^2 E / (\partial x_i)^2\}$. However, only the loss $J(x; w)$ is obtained by these two steps. For differentiation of this loss, one must perform an additional forward and backward pass.

3.1 Obtaining the loss

Consider a feed-forward neural network with input vector $x$ and weights $w$ and an ordered set of nodes indexed $1 \ldots N$, each node $j$ with child nodes $i \in \text{children}(j)$ with $j < i$ and parent nodes $k \in \text{parents}(j)$ with $k < j$. The first $D < N$ nodes are input nodes, for which the activation value is $g_j = x_j$. For the other nodes (hidden units and output unit), the activation value is determined by a differentiable scalar function $g_j(\{g_i\}_{i \in \text{parents}(j)}, w)$. The network’s “output” (energy) is determined as the activation of the last node: $E(x; w) = g_N(\cdot)$. The values $\delta_j = \partial E / \partial g_j$ are efficiently computed by backpropagation. However, backpropagation of the full Hessian scales like $O(W^2)$, where $W$ is the number of model weights. Here, we limit backpropagation to the diagonal approximation which scales like $O(W)$ [11]. This will still result in the correct gradients $\partial^2 E / (\partial x_j)^2$ for one-layer models and the models considered in this paper. Rewriting the equations for the full Hessian is a straightforward exercise. For brevity, we write $\delta_j' = \partial^2 E / (\partial g_j)^2$. The SM loss is split in two terms: $J(x; w) = K + L$ with $K = \frac{1}{T} \sum_{j=1}^{D} \delta_j' \delta_j$ and $L = \sum_{j=1}^{D} -\delta_j' + \lambda(\delta_j')^2$. The equations for inference and backpropagation are given as the first two $for$-loops in Algorithm 1.

Most previous publications do not express unnormalized neg. log-density as “energy”.

4
Input: \( x, w \) (data and weight vectors)

for \( j \leftarrow D + 1 \) to \( N \) do  

    // Forward propagation
    \( \text{compute } g_j(\cdot) \)
    for \( i \in \text{parents}(j) \) do
    \( \text{compute } \frac{\partial g_j}{\partial g_i}, \frac{\partial^2 g_j}{\partial (g_i)^2}, \frac{\partial^3 g_j}{\partial (g_i)^3} \)

\( \delta_N \leftarrow 1, \delta_N' \leftarrow 0 \)

for \( j \leftarrow N - 1 \) to \( 1 \) do  

    // Backpropagation
    \( \delta_j = \sum_{k \in \text{children}(j)} \delta_k \frac{\partial g_j}{\partial g_k} \)
    \( \delta_j' = \sum_{k \in \text{children}(j)} \frac{\partial^2 g_j}{\partial (g_k)^2} + \delta_k' \left( \frac{\partial g_j}{\partial g_k} \right)^2 \)

for \( j \leftarrow 1 \) to \( D \) do  

    \( \frac{\partial K}{\partial \delta_j} = \delta_j; \quad \frac{\partial L}{\partial \delta_j} \leftarrow -1 + 2\lambda \delta_j' \)

for \( j \leftarrow D + 1 \) to \( N \) do  

    // SM Forward propagation
    \( \frac{\partial K}{\partial \delta_j} \leftarrow \sum_{i \in \text{parents}(j)} \frac{\partial K}{\partial g_i} \frac{\partial g_i}{\partial \delta_j} \)
    \( \frac{\partial L}{\partial \delta_j} \leftarrow \sum_{i \in \text{parents}(j)} \frac{\partial L}{\partial g_i} \frac{\partial g_i}{\partial \delta_j} + \frac{\partial L}{\partial \delta_j} \left( \frac{\partial g_i}{\partial g_i} \right)^2 \)

for \( j \leftarrow N \) to \( D + 1 \) do  

    // SM Backward propagation
    \( \frac{\partial K}{\partial \delta_j} \leftarrow \sum_{k \in \text{children}(j)} \frac{\partial K}{\partial g_k} \frac{\partial g_k}{\partial \delta_j} + \frac{\partial K}{\partial g_k} \delta_k' \left( \frac{\partial g_k}{\partial \delta_j} \right)^2 + \frac{\partial L}{\partial \delta_j} \frac{\partial g_k}{\partial g_k} + \frac{\partial \delta_j}{\partial g_k} \left( \frac{\partial \delta_j}{\partial g_k} \right)^2 \)

for \( w \in w \) do  

    // Derivatives wrt weights
    \( \frac{\partial J}{\partial w} \leftarrow \sum_{j = D + 1}^{N} \frac{\partial K}{\partial \delta_j} \frac{\partial \delta_j}{\partial g_k} + \frac{\partial L}{\partial \delta_j} \frac{\partial \delta_j}{\partial g_k} + \frac{\partial K}{\partial \delta_j} \delta_j' + \frac{\partial L}{\partial \delta_j} \delta_j' + \frac{\partial L}{\partial \delta_j} \delta_j' \)

Algorithm 1: Compute \( \nabla_w J \). See sections 3.1 and 3.2 for context.

3.2 Differentiating the loss

Since the computation of the loss \( J(x; w) \) is performed by a deterministic forward-backward mechanism, this two-step computation can be interpreted as a combination of two networks: the original network for computing \( \{g_j\} \) and \( E(x; w) \), and an appended network for computing \( \{\delta_j\}, \{\delta_j'\} \) and eventually \( J(x; w) \). See figure 1. The combined network can be differentiated by an extended version of the double-backpropagation procedure [5], with the main difference that the appended network not only computes \( \{\delta_j\} \), but also \( \{\delta_j'\} \). Automatic differentiation of the combined network consists of two phases, corresponding to reverse traversal of the appended and original network respectively: (1) obtaining \( \frac{\partial K}{\partial \delta_j}, \frac{\partial L}{\partial \delta_j}, \frac{\partial L}{\partial \delta_j} \) for each node \( j \) in order 1 to \( N \); (2) obtaining \( \frac{\partial J}{\partial \delta_j} \) for each node \( j \) in order \( N \) to \( D + 1 \). These procedures are given as the last two for-loops in Algorithm 1. The complete algorithm scales like \( O(W) \).

4 Experiments

Consider the following Product-of-Experts (PoE) model:

\[
E(x; W, \alpha) = \sum_{i=1}^{M} \alpha_i g(w_i^T x)
\]

(8)

where \( M \) is the number of experts, \( w_i \) is an image filter and the \( i \)-th row of \( W \) and \( \alpha_i \) are scaling parameters. Like in [10], the filters are L2 normalized to prevent a large portion from vanishing. We use a slightly modified Student’s t-distribution \( g(z) = \log((c \varepsilon^2 + 1) / c) \) for latent space, so this is also a Product of Student’s t-distribution model [24]. The parameter \( c \) is a non-learnable horizontal scaling parameter, set to \( e^1.5 \). The vertical scaling parameters \( \alpha_i \) are restricted to positive, by setting \( \alpha_i = \exp \beta_i \) where \( \beta_i \) is the actual weight.
4.1 MNIST

The first task is to estimate a density model of the MNIST handwritten digits [16]. Since a large number of labels need to be learned, a 2× downsampling version of MNIST was used. The MNIST dataset is highly non-smooth: for each pixel, the extreme values (0 and 1) are highly frequent leading to sharp discontinuities in the data density at these points. It is well known that for models with square weight matrix $W$, normalized $g(.)$ (meaning $\int_{-\infty}^{\infty} \exp(-g(x))dx = 1$) and $\alpha = 1$, the normalizing constant can be computed [10]: $Z(w) = |\det W|$. For this special case, models can be compared by computing the log-likelihood for the training- and test set. Unregularized, and regularized models for different choices of $\lambda$ were estimated and log-likelihood values were computed. Subsequently, these models were compared on a classification task. For each MNIST digit class, a small sample of 100 data points was converted to internal features by different models. These features, combined with the original class label, were subsequently used to train a logistic regression classifier for each model. For the PoE model, the “activations” $g(w_i^T x)$ were used as features. Classification error on the test set was compared against reported results for optimal RBM and SESM models [20].

**Results.** As expected, unregularized estimation did not result in an accurate model. Figure 2 shows how the log-likelihood of the train- and test set is optimal at $\lambda^* \approx 0.01$, and decreases for smaller $\lambda$. Coincidentally, the classification performance is optimal for the same choice of $\lambda$.

4.2 Denoising

Consider grayscale natural image data from the Berkeley dataset [18]. The data quantized and therefore non-smooth, so regularization is potentially beneficial. In order to estimate the correct regularization magnitude, we again estimated a PoE model as in equation (8) with square $W$, such that $Z(w) = |\det W|$ and computed the log-likelihood of 10.000 random patches under different regularization levels. We found that $\lambda^* \approx 10^{-5}$ for maximum likelihood (see figure 2d). This value is lower than for MNIST data since natural image data is “less unsmooth”. Subsequently, a convolutional PoE model known as Fields-of-Experts [21] (FoE) was estimated using regularized SM:

$$E(x; W, \alpha) = \sum_{p} \sum_{i=1}^{M} \alpha_i g(w_i^T x_{(p)})$$

(9)

where $p$ runs over image positions, and $x_{(p)}$ is a square image patch at $p$. The first model has the same architecture as the CD-1 trained model in [21]: $5 \times 5$ receptive fields, 24 experts ($M = 24$), and $\alpha_i$ and $g(.)$ as in our PoE model. Note that qualitative results of a similar model estimated with SM have been reported earlier [15]. We found that for best performance, the model is learned on images “whitened” with a $5 \times 5$ Laplacian kernel. This is approximately equivalent to ZCA whitening used in [15].

Models are evaluated by means of Bayesian denoising using maximum a posteriori (MAP) estimation. As in a general Bayesian image restoration framework, the goal is to estimate the original input $x$ given a noisy image $y$ using the Bayesian proportionality $p(x|y) \propto p(y|x)p(x)$. The assumption is white Gaussian noise such that the likelihood is $p(y|x) \sim \mathcal{N}(0, \sigma^2 I)$. The model $E(x; w) = -\log p(x; w) - Z(w)$ is our prior. The gradient of the log-posterior is:

$$\nabla x \log p(x|y) = -\nabla_x E(x; w) + \frac{1}{2\sigma^2} \nabla_x \sum_{i=1}^{N} (y_i - x_i)^2$$

(10)

Denoising is performed by initializing $x$ to a noise image, and 300 subsequent steps of steepest descent according to $x' \leftarrow x + \alpha \nabla_x \log p(x|y)$, with $\alpha$ annealed from $2 \cdot 10^{-2}$ to $5 \cdot 10^{-4}$. For comparison, we ran the same denoising procedure with models estimated by CD-1 and Basis Rotation, from [21] and [23] respectively. Note that the CD-1 model is trained using PCA whitening. The CD-1 model has been extensively applied to denoising before [21] and shown to compare favourably to specialized denoising methods.

**Results.** Training of the convolutional model took about 1 hour on a 2Ghz machine. Regularization turns out to be important for optimal denoising (see figure 2[e-g]). See table 1 for denoising performance of the optimal model for specific standard images. Our model performed significantly better
Figure 2: (a) Top: selection of downsampled MNIST datapoints. Middle and bottom: random sample of filters from unregularized and regularized ($\lambda = 0.01$) models, respectively. (b) Average log-likelihood of MNIST digits in training- and test sets for choices of $\lambda$. Note that $\lambda^* \approx 0.01$, both for maximum likelihood and optimal classification. (c) Test set error of a logistic regression classifier learned on top of features, with only 100 samples per class, for different choices of $\lambda$. Optimal error rates of SESM and RBM (figure 1a in [20]) are shown for comparison. (d) Log-likelihood of 10,000 random natural image patches for complete model, for different choices of $\lambda$. (e-g) PSNR of 500 denoised images, for different levels of noise and choices of $\lambda$. Note that $\lambda^* \approx 10^{-5}$, both for maximum likelihood and best denoising performance. (h) Some natural images from the Berkeley dataset. (i) Filters of model with $5 \times 5 \times 24$ weights learned with CD-1 [21], (j) filters of our model with $5 \times 5 \times 24$ weights, (k) random selection of filters from the Basis Rotation [23] model with $15 \times 15 \times 25$ weights, (l) random selection of filters from our model with $8 \times 8 \times 64$ weights. (m) Detail of original Lena image. (n) Detail with noise added ($\sigma_{\text{noise}} = 1/256$). (o) Denoised with model learned with CD-1 [21], (p) Basis Rotation [23], (q) and Score Matching with (near) optimal regularization.
than the Basis Rotation model and slightly better than the CD-1 model. As reported earlier in [15],
we can verify that the filters are completely intuitive (Gabor filters with different phase, orientation
and scale) unlike the filters of CD-1 and Basis Rotation models (see figure 2[i-l]).

Table 1: Peak signal-to-noise ratio (PSNR) of denoised images with $\sigma_{\text{noise}} = 5/256$. Shown errors
are aggregated over different noisy images.

<table>
<thead>
<tr>
<th>Image</th>
<th>CD-1</th>
<th>Basis Rotation</th>
<th>Our model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights</td>
<td>$(5 \times 5) \times 24$</td>
<td>$(15 \times 15) \times 25$</td>
<td>$(5 \times 5) \times 24$</td>
</tr>
<tr>
<td>Barbara</td>
<td>37.30±0.01</td>
<td>37.08±0.02</td>
<td>37.31±0.01</td>
</tr>
<tr>
<td>Peppers</td>
<td>37.63±0.01</td>
<td>37.09±0.02</td>
<td>37.41±0.03</td>
</tr>
<tr>
<td>House</td>
<td>37.85±0.02</td>
<td>37.73±0.03</td>
<td>38.03±0.04</td>
</tr>
<tr>
<td>Lena</td>
<td>38.16±0.02</td>
<td>37.97±0.01</td>
<td>38.19±0.01</td>
</tr>
<tr>
<td>Boat</td>
<td>36.33±0.01</td>
<td>36.21±0.01</td>
<td>36.53±0.01</td>
</tr>
</tbody>
</table>

4.3 Super-resolution

In addition, models are compared with respect to their performance on a simple version of super-
resolution as follows. An original image $x_{\text{orig}}$ is sampled down to image $x_{\text{small}}$ by averaging blocks
of $2 \times 2$ pixels into a single pixel. A first approximation $x$ is computed by linearly scaling up $x_{\text{small}}$
and subsequent application of a low-pass filter to remove false high frequency information. The
image is then fine-tuned by 200 repetitions of two subsequent steps: (1) refining the image slightly
using $x' \leftarrow x + \alpha \nabla_x E(x; w)$ with $\alpha$ annealed from $2 \cdot 10^{-2}$ to $5 \cdot 10^{-4}$; (2) updating each $k \times k$
block of pixels such that their average corresponds to the down-sampled value. Note: the simple
block-downsampling results in serious aliasing artifacts in the Barbara image, so the Castle image
is used instead.

Results. PSNR values for standard images are shown in table 2. The considered models made give
slight improvements in terms of PSNR over the initial solution with low pass filter. Still, our model
did slightly better than the CD-1 and Basis Rotation models.

Table 2: Peak signal-to-noise ratio (PSNR) of super-resolved images for different models.

<table>
<thead>
<tr>
<th>Image</th>
<th>Low pass filter</th>
<th>CD-1</th>
<th>Basis Rotation</th>
<th>Our model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights</td>
<td>-</td>
<td>$(5 \times 5) \times 24$</td>
<td>$(15 \times 15) \times 25$</td>
<td>$(5 \times 5) \times 24$</td>
</tr>
<tr>
<td>Peppers</td>
<td>27.54</td>
<td>29.11</td>
<td>27.69</td>
<td>29.76</td>
</tr>
<tr>
<td>House</td>
<td>33.15</td>
<td>33.83</td>
<td>33.41</td>
<td>33.48</td>
</tr>
<tr>
<td>Lena</td>
<td>32.39</td>
<td>33.31</td>
<td>33.07</td>
<td>33.46</td>
</tr>
<tr>
<td>Boat</td>
<td>29.20</td>
<td>30.81</td>
<td>30.77</td>
<td>30.82</td>
</tr>
<tr>
<td>Castle</td>
<td>24.19</td>
<td>24.15</td>
<td>24.26</td>
<td>24.81</td>
</tr>
</tbody>
</table>

5 Conclusion

We have shown how the addition of a principled regularization term to the expression of the Score
Matching loss lifts continuity assumptions on the data density, such that the estimation method
becomes more generally applicable. The effectiveness of the regularizer was verified with the dis-
continuous MNIST and Berkeley datasets, with respect to likelihood of test data in the model. For
both datasets, the optimal regularization parameter is approximately equal for both likelihood and
subsequent classification and denoising tasks. In addition, we showed how computation and differ-
entiation of the Score Matching loss can be automated using an efficient algorithm.
References


