Variational Networks: Connecting Variational Methods and Deep Learning

Erich Kobler[1], Teresa Klatzer[1], Kerstin Hammernik[1] and Thomas Pock[1,2]

[1] Institute of Computer Graphics and Vision, Graz University of Technology, Austria
[2] Center for Vision, Automation and Control, Austrian Institute of Technology

Abstract. In this paper, we introduce variational networks (VNs) for image reconstruction. VNs are fully learned models based on the framework of incremental proximal gradient methods. They provide a natural transition between classical variational methods and state-of-the-art residual neural networks. Due to their incremental nature, VNs are very efficient, but only approximately minimize the underlying variational model. Surprisingly, in our numerical experiments on image reconstruction problems it turns out that giving up exact minimization leads to a consistent performance increase, in particular in the case of convex models.

1 Introduction

There has been a long tradition of using variational methods to tackle computer vision problems including denoising [38], deblurring [28, 45], segmentation [12, 34], tracking [3, 17] and optical flow [22] due to their simplicity, performance and profound theoretical foundations. In recent years, these approaches have been outperformed by deep learning methods. Despite the success of deep learning in computer vision [20, 31], it is unclear whether there exists a theoretical connection between variational methods and deep learning. In this paper, we try to answer this question by establishing relations between both worlds.

Variational methods are based on minimizing an energy functional. An archetype convex variational model (VM) for image restoration is the Rudin-Osher-Fatemi (ROF) model [38]. In the discrete setting it is defined as

$$\boldsymbol{x}^*(\boldsymbol{x}_0) = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) := \|\nabla \boldsymbol{x}\|_1 + \frac{\alpha}{2} \|\boldsymbol{x} - \boldsymbol{x_0}\|_2^2 \ , \tag{1}$$

where $\boldsymbol{x} \in \Re^n$ represents an image with n pixels, $\boldsymbol{x_0} \in \Re^n$ the noisy observation and $\nabla \in \Re^{2n \times n}$ is a linear operator that computes the discrete horizontal and vertical derivatives. As a motivational example, we analyze the 2×2 patch statistics of a set of natural images $\mathcal{G}$ and the set of minimizers $\mathcal{S} = \{\boldsymbol{x}^*(\boldsymbol{x}_0) : \partial F(\boldsymbol{x}^*) \ni 0, \boldsymbol{x}_0 = \boldsymbol{g} + \boldsymbol{n}, \boldsymbol{g} \in \mathcal{G}, \boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 I)\}$. Figure 1 visualizes these statistics along with those of noisy images. The solution set $\mathcal{S}$ shows a significant difference to the true image statistics especially in the polar regions, which suggests that the solution set $\mathcal{S}$ cannot capture the complexity of

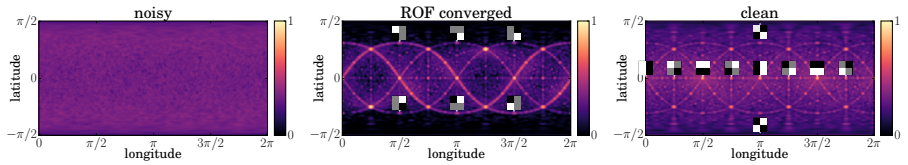


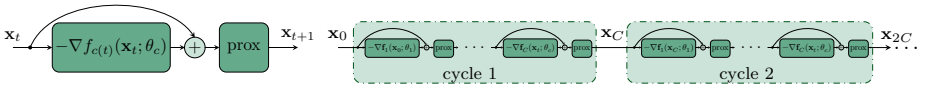
Fig. 1. Estimated log–probability density of 2×2 image patches from the BSDS500 data set [32] on the unit sphere in the zero-mean and contrast-normalized patch space. The projection onto this sphere is performed in analogy to [27] and its surface is parametrized by the longitudinal and the lateral angle.

natural images. This originates either from a too simple model or the optimality condition $\partial F(\boldsymbol{x}^*) \ni 0$ is too restrictive.

A natural idea for improving the ROF model is to increase its flexibility by introducing additional terms. Chambolle and Lions [11] increased the model complexity by formulating image reconstruction as a convex infimal convolution problem. Another convex VM is the total generalized variation [10], which extends the ROF model by *modeling* higher order statistics. However, Black and Anandan [7] demonstrated that incorporating non-convex functions improves results because the applied non-convex functions suppress outliers as known from robust statistics. They optimize the non-convex VMs using the graduated non-convexity method [8], which solves a sequence of VMs starting with a convex model that gradually becomes non-convex.

The idea of *learning* higher order statistics to enhance the results of variational methods for image reconstruction was introduced by Roth and Black [37]. They proposed to learn a prior (regularization) consisting of an ensemble of filters together with corresponding non-convex potential functions called Fields of Experts (FoE) using contrastive divergence. Later [25] formulated the learning of regularization parameters of a VM as a bi-level optimization problem, which was extended in [13] to learn analysis operators of (non-)convex VMs including the FoE model. Their results on image denoising indicate that non-convex models perform best, confirming the findings of Zhu and Mumford [49]. Also Domke [16] enhanced the performance of the FoE model by discriminatively learning incomplete energy minimization schemes that consist just of a few iterations inspired by [18]. The combination of 1) unrolling a gradient descent scheme for the FoE model and 2) abandoning energy minimization by parameterizing each step individually led to the optimized reaction-diffusion processes of Chen et al. [14], which improved the state-of-the-art on several reconstruction tasks [19, 23, 46].

The neural network community pursues a completely different approach for increasing the model complexity. Since the early convolutional neural networks [26, 39], advances in network training and the use of more complex, deeper networks have led to remarkable results in many areas of computer vision, including classification [20, 24] and restoration [31, 47]. Increasing the model complexity by stacking more and more layers works just to some extent due to a degradation problem reported by He et al. [20]. To avoid this problem, they introduced



(a) Variational Unit (VU) (b) Variational Network (VN)

Fig. 2. This figure shows an illustration of (a) our proposed variational units (3) and their combination to a variational network (b) that uses a cyclic scheme.

residual networks that have a simple computational structure which eases the training of very deep models.

In this work, we introduce variational networks that are developed by minimizing a parametrized energy using proximal incremental methods [5]. The VNs have the same computational structure as residual networks and thus are easy to train. Moreover, the concept of VNs enables us to explore theoretical properties such as the role of convexity in the field of natural image reconstruction. Therefore, we extend the FoE regularization structure by fully parametrized potential functions that can be trained either convex or non-convex.

2 Variational Networks

We propose *variational networks* (VNs) that are motivated by proximal gradient and proximal incremental methods and yield the same computation structure as residual networks. The basic structure of VNs evolves naturally by performing incremental proximal gradient steps [5] to solve problems of the form

$$\min_{\boldsymbol{x}} F(\boldsymbol{x}) := \sum_{c=1}^{C} f_c(\boldsymbol{x};\boldsymbol{\theta}_c) + h(\boldsymbol{x}) \;, \tag{2}$$

where C defines the number of components, $\boldsymbol{x} \in \mathfrak{R}^n$ represents some data, i. e., an image, $f_c : \mathfrak{R}^n \mapsto \mathfrak{R}$ are smooth component functions parametrized by $\boldsymbol{\theta}_c$ and $h : \mathfrak{R}^n \mapsto \mathfrak{R}$ is a convex, lower semi-continuous (l.s.c.) function. An incremental proximal gradient step is defined as

$$\boldsymbol{x}_{t+1} = \operatorname{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f_{c(t)}(\boldsymbol{x}_t;\boldsymbol{\theta}_{c(t)})\right), \tag{3}$$

where η_t is the step size of the t-th step. We fix the component selection function $c(t) = \operatorname{mod}(t, C)+1$ to obtain a cyclic procedure as depicted in Figure 2. We call the scheme (3) variational unit (VU) in analogy to residual units. The VU is the basic building block of a VN. The output of the C-th unit $\boldsymbol{x}_{t=C}$ ends the first cycle. It is also the output of a corresponding residual network [20]. Moreover, VNs generalize the optimized reaction-diffusion processes [14] as they can be interpreted as a single cycle of a parametrized incremental scheme.

2.1 Relation to Incremental Gradient Methods

The formulation of VNs is based on incremental proximal methods, which were proposed by Nedić and Bertsekas [5, 36]. These methods were designed to solve

large-scale energy minimization problems consisting of smooth and non-smooth components. Such problems can be cast into the form

$$\min_{\boldsymbol{x}\in\mathcal{X}} F(\boldsymbol{x}) := f(\boldsymbol{x}) + h(\boldsymbol{x}) = \sum_{c=1}^{C} f_c(\boldsymbol{x}) + h(\boldsymbol{x}) \ , \tag{4}$$

where f aggregates the smooth components $f_c : \mathfrak{R}^n \mapsto \mathfrak{R}$ and $h : \mathfrak{R}^n \mapsto \mathfrak{R}$ holds the convex, l.s.c. and non-smooth parts. Problem (19) can be turned into an unconstrained form by including the indicator function of $\mathcal{X}$ in $h(x)$. In analogy to [5] an incremental proximal gradient step is given by

$$\boldsymbol{x}_{t+1} = \operatorname{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f_{c(t)}(\boldsymbol{x}_t)\right) \ , \tag{5}$$

where $\nabla f_{c(t)}(\boldsymbol{x}_t)$ is the gradient of a single component selected by $c(t)$ and the proximal map is defined by

$$\operatorname{prox}_h^{\eta}(\boldsymbol{z}) := \arg\min_{\boldsymbol{x}} \left(h(\boldsymbol{x}) + \frac{1}{2\eta} \left\| \boldsymbol{x} - \boldsymbol{z} \right\|_2^2 \right) \ . \tag{6}$$

If f consists only of a single component, i. e., $f(\boldsymbol{x}) = f_1(\boldsymbol{x})$, the scheme (22) simplifies to the proximal gradient method defined as

$$\boldsymbol{x}_{t+1} = \operatorname{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t)\right) \ . \tag{7}$$

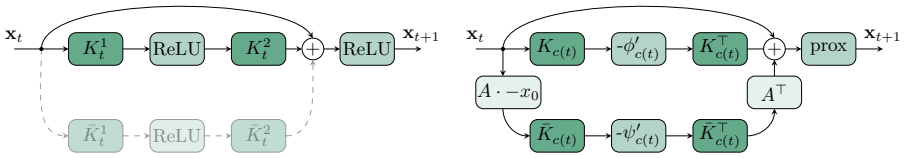
First assume that all components f_c are *convex*. In this case, Bertsekas [5] showed that the incremental proximal method (22) converges to a stationary point in the limit for a diminishing step size, satisfying $\sum_{t=0}^{\infty} \eta_t = \infty$, $\sum_{t=0}^{\infty} \eta_t^2 < \infty$, for both cyclic and random component selection $c(t)$. Moreover, he proved approximate convergence for a constant step size ($\eta_t = \eta > 0$). The assumptions of the proofs are fulfilled if all components f_c are Lipschitz continuous on $\mathcal{X}$.

If the components f_c are *non-convex*, one can still show approximate convergence of (22) in the limit using the inexact non-convex proximal splitting algorithm of Sra [43]. In addition to the requirements of Sra, i. e., all f_c have a Lipschitz continuous gradient on $\mathcal{X}$, we assume that the components f_c are Lipschitz on $\mathcal{X}$, just as in the convex case. Then (22) approximately converges to a stationary point for a constant step size $\eta_t = \eta > 0$. The proof can be found in the supplemental material.

2.2 Relation to Residual Networks

Deep residual networks were proposed by [20] to alleviate a degradation problem arising in deep neural network training, indicated by increasing training *and* test error despite growing model complexity. Residual networks circumvent this problem by stacking many simple residual units, which are characterized by

$$\boldsymbol{x}_{t+1} = p(\boldsymbol{x}_t + \boldsymbol{g}_t(\boldsymbol{x}_t)) \ , \tag{8}$$



(a) (multi-) Residual Unit (b) Variational Unit

Fig. 3. Visualization of the structural correspondence between (a) multi-residual units [33] and (b) variational units for image reconstruction (13). Note the data term gradient in (b) can be interpreted as a second residual mapping in the data domain. The multi-residual unit is turned into a residual unit [20] by omitting the dashed path.

where $\boldsymbol{x}_t, \boldsymbol{x}_{t+1} \in \mathfrak{R}^n$ are the input and output of the t-th layer, $p : \mathfrak{R}^n \mapsto \mathfrak{R}^n$ is a point-wise scalar function (e. g., ReLU) and $\boldsymbol{g}_t : \mathfrak{R}^n \mapsto \mathfrak{R}^n$ are residual functions. Typically, these residual functions are defined as

$$\boldsymbol{g}_t(\boldsymbol{x}_t) = \sum_{i=1}^{N_r} K_{t,i}^2 a(K_{t,i}^1 \boldsymbol{x}_t) \;, \tag{9}$$

where the matrices $K_{t,i}^1, K_{t,i}^2 \in \mathfrak{R}^{n \times n}$ model convolutions and N_r defines the number of convolution kernels. The function $a : \mathfrak{R}^n \mapsto \mathfrak{R}^n$ is often set to the ReLU activation. The resulting networks can be efficiently trained for more than 1000 layers. The combination of the individual residual units forms a powerful ensemble of networks [44], yielding state-of-the-art results on challenging competitions, e. g., ImageNet [24] and MS COCO [29].

By comparing the structure of variational units (3) and residual units (8), we see that the proximal map in (3) corresponds to $p(x) = \mathrm{ReLU}(x)$ in (8) if h is the indicator function of the positive orthant. If we assume $\eta_t = 1$, then $\boldsymbol{g}_t$ corresponds to $-\nabla f_{c(t)}(\boldsymbol{x}_t)$. This is either true for $t \leq C$ or if a residual net shares parameters in a periodic fashion [1]. To emphasize this structural resemblance, Fig. 3 visualizes a residual and a variational unit. The residual function (9) corresponds to a gradient if $K_{t,i}^2 = K_{t,i}^{1\top}$. If this relation is approximate ($K_{t,i}^2 \approxeq K_{t,i}^{1\top}$), $\boldsymbol{g}_t$ can still be interpreted as a gradient with error. Consequently, this type of networks fits into the VN formulation and both networks have the same computational structure. Hence, VNs combine the practical benefits of residual networks, i. e., avoid the degradation problem, and the rich theory of incremental methods, including convergence and convex optimization theory.

3 Variational Networks for Image Reconstruction

We formulate image reconstruction as a variational energy minimization problem with a fully trainable regularization as well as data term and cast this problem into the VN formulation.

3.1 Problem Formulation and Parametrization

A variational model for image reconstruction in the form of (2) is given by

$$\min_{\boldsymbol{x}\in\mathcal{X}^n} F(\boldsymbol{x}) := \sum_{c=1}^{C} f_c(\boldsymbol{x};\boldsymbol{\theta}_c) = R_c(\boldsymbol{x};\boldsymbol{\theta}_c) + D_c(\boldsymbol{x};\boldsymbol{\theta}_c) \ , \tag{10}$$

where $\boldsymbol{x} \in \mathcal{X}^n$ represents an image, constrained on $\mathcal{X} = \{x \in \mathfrak{R} : 0 \leq x \leq m\}$ with $m > 0$. The vector $\boldsymbol{\theta}_c$ holds the parameters for each component. The regularization term $R_c(\boldsymbol{x};\boldsymbol{\theta}_c)$ models prior knowledge, whereas, the data term $D_c(\boldsymbol{x};\boldsymbol{\theta}_c)$ models the data fidelity. The specific form of the FoE regularization term variant is given by

$$R_c(\boldsymbol{x};\boldsymbol{\theta}_c) = \sum_{i=1}^{N_r}\sum_{j=1}^{n} \phi_i^c\left((K_i^c\boldsymbol{x})_j\right) \ , \tag{11}$$

where $\phi_i^c(x) : \mathcal{Y} \mapsto \mathfrak{R}$ are potential functions defined on $\mathcal{Y} = \{y \in \mathfrak{R} : |y| \leq m\}$, their associated matrices $K_i^c \in \mathfrak{R}^{n\times n}$ model convolutions of the image x with kernels k_i^c and N_r defines the number of regularization functions. Some learned kernel-function pairs are depicted in Fig. 4. The convolution of a $s_k \times s_k$ kernel k_i^c can also be expressed as matrix-vector multiplication $X\boldsymbol{k}_i^c$ with the matrix $X \in \mathfrak{R}^{n\times s_k^2}$ and the vector $\boldsymbol{k}_i^c \in \mathfrak{R}^{s_k^2}$.

We parametrize the data term also with kernel-function pairs to incorporate higher-order statistics in the data domain, motivated by [42]. It is defined as

$$D_c(\boldsymbol{x};\boldsymbol{\theta}_c) = \sum_{i=1}^{N_d}\sum_{j=1}^{n} \psi_i^c\left(\left(\bar{K}_i^c(A\boldsymbol{x}-\boldsymbol{x}_0)\right)_j\right) \ , \tag{12}$$

where $\boldsymbol{x}_0 \in \mathcal{X}^n$ describes the degraded observation and $A \in \mathfrak{R}^{n\times n}$ models a linear operator. As before, the matrices $\bar{K}_i^c \in \mathfrak{R}^{n\times n}$ model convolutions with kernels $\bar{k}_i^c$, $\psi_i^c(y) : \mathcal{Y} \mapsto \mathfrak{R}$ are the corresponding potential functions and N_d specifies the number of kernel-function pairs.

We define the VUs for image reconstruction akin to (3) as

$$\boldsymbol{x}_{t+1} = \mathrm{proj}_{\mathcal{X}^n}(\boldsymbol{x}_t - \eta_t \nabla f_{c(t)}(\boldsymbol{x}_t;\boldsymbol{\theta}_{c(t)})) \tag{13}$$

where the proximal operator of (3) simplifies to the projection onto $\mathcal{X}^n$. The gradient for a selected component $f_c(\boldsymbol{x};\boldsymbol{\theta_c})$ is given by

$$\nabla f_c(\boldsymbol{x}_t;\boldsymbol{\theta}_c) = \sum_{i=1}^{N_r} K_i^{c\top}\phi_i'^c\left(K_i^c\boldsymbol{x}_t\right) + A^\top\sum_{i=1}^{N_d}\bar{K}_i^{c\top}\psi_i'^c\left(\bar{K}_i^c(A\boldsymbol{x}_t-\boldsymbol{x}_0)\right) \ . \tag{14}$$

Since we learn the influence functions $\phi_i'^c(y)$ and $\psi_i'^c(y)$, we can fix the step size $\eta_t = 1$ as it is reflected in the scale of both influence functions. Due to the above parametrization, all the component functions f_c of the according VN are smooth, Lipschitz continuous functions with bounded and Lipschitz continuous gradient as long as the functions $\phi_i'^c(y)$ and $\psi_i'^c(y)$ fulfill these constraints. The proofs are in the supplemental material. Note that the runtime and memory requirements of the VNs resemble those of [14], since the basic operations are identical.

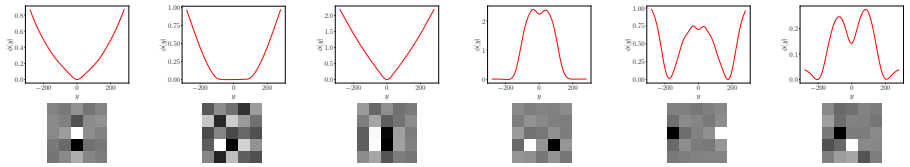


Fig. 4. Sample kernel-function pairs $(k_i^c, \phi_i^c(y))$ of the trained VNs. The left three pairs are convex samples, whereas the right three were extracted from non-convex VNs.

3.2 Training

To train the VNs for image reconstruction we parametrize the influence functions $\phi_i'^c(y)$ and $\psi_i'^c(y)$ in analogy to [14, 41] with radial basis functions

$$\phi_i'^c(y) = \sum_{j=1}^{N_w} \exp\left(-\frac{(y-\mu_j)^2}{2\sigma^2}\right) w_{ij}^c \ , \tag{15}$$

where w_{ij}^c are the individual basis weights that correspond to a single radial basis (μ_j, σ) and N_w defines the number of basis functions. To shorten notation we group the coefficients into $\boldsymbol{w}_i^c = (w_{i1}^c, \dots, w_{iN_w}^c)^\top$. The functions $\psi_i'^c(x)$ are parametized in the same way by $\bar{\boldsymbol{w}}_i^c$. We group the parameters of a single component c into the vector $\boldsymbol{\theta}_c = (\boldsymbol{k}_1^c, \boldsymbol{w}_1^c, \dots, \boldsymbol{k}_{N_r}^c, \boldsymbol{w}_{N_r}^c, \bar{\boldsymbol{k}}_1^c, \bar{\boldsymbol{w}}_1^c, \dots, \bar{\boldsymbol{k}}_{N_d}^c, \bar{\boldsymbol{w}}_{N_d}^c)$. The parameters of all components are gathered into $\boldsymbol{\theta} = (\boldsymbol{\theta}_i, i = 1 \dots C)$. We define the training cost for N_s input-target pairs $(\boldsymbol{x}_0^s, \boldsymbol{x}_{gt}^s)$ as

$$\min_{\boldsymbol{\theta} \in \mathcal{T}} L(\boldsymbol{\theta}) := \frac{1}{N_s} \sum_{s=1}^{N_s} \|\boldsymbol{x}_T^s(\boldsymbol{\theta}) - \boldsymbol{x}_{gt}^s\|_1 \ , \tag{16}$$

where $\boldsymbol{x}_T^s$ is the output after T steps (13). We use the ℓ_1-norm because of its robustness [48]. In addition, we constrain the parameters $\boldsymbol{\theta}$ to be in an admissible set $\mathcal{T}$. This set ensures that the kernels $\boldsymbol{k}_i^c$ and $\bar{\boldsymbol{k}}_i^c$ have zero-mean and ℓ_2-norm one, to avoid a scaling problem as outlined in [14]. $\mathcal{T}$ also allows us to incorporate constraints on the functions $\phi_i^c(y)$ and $\psi_i^c(x)$ such as convexity by defining suitable conditions for $\boldsymbol{w}_i^c$ and $\bar{\boldsymbol{w}}_i^c$ as shown in the supplemental material. Note if all $\phi_i^c(y)$ and $\psi_i^c(x)$ are convex, the entire energy (10) becomes convex [9].

We optimize the non-convex training problem (16) with the inertial incremental proximal method (IIPG) defined in Algorithm 1 in the supplemental material. It is an incremental proximal method that uses preconditioning for acceleration and is capable of handling the constraints incorporated in the admissible set $\mathcal{T}$.

4 Experiments

We conduct three groups of experiments to show the versatility of VNs and to explore the role of convexity. Table 1 defines all used VN types and outlines their relation to the previously discussed methods. We conduct all experiments

Table 1. Overview of the VN types. The subscript N defines the number of used kernel-function pairs $N_r = N$. The superscript specifies the number of components C and the step t for which the VN was optimized.

Type	Corresponding scheme
$\text{VN}_N^{1,t}$	proximal gradient method (20) (energy minimization)
$\text{VN}_N^{C,t}$	proximal incremental method (22) (approximate energy minimization)
$\text{VN}_N^{t,t}$	single cycle proximal incremental method (22) (reaction diffusion)

for denoising and non-blind delurring. In the case of denoising, the degraded input $\boldsymbol{x}_0$ is a noisy observation and the linear operator A in (12) simplifies to an identity operation. For non-blind deblurring, the input is a blurry and noisy observation and the linear operator A models a convolution with a known blur kernel. The denoising VNs (N-VN) use just a single data term $N_d = 1$ and an identity kernel $\bar{k}_1^1$, while the deblurring VNs (B-VN) apply $N_d = N_r$ kernel-function pairs. To train VNs for both problems, we use 400 training patches of size 180×180 extracted from the BSDS500 train and test sets [32]. We generate the noisy training inputs by adding white Gaussian noise with $\sigma = 25$ to the clean images. To generate the blurry training data, we extract 11×11 motion blur kernels from [40], convolve them with the clean training patches and add 1% white Gaussian noise. The test sets are generated in the same way for denoising and non-blind deblurring. We use 68 images from the BSDS500 [32] validation set and the motion blur kernels from [28] to ensure that neither the images nor the blur kernels are used during training. Finally, it is important to point out that all found schemes are local optima of the non-convex training problem (16).

4.1 Energy Minimization with VNs

In the first experiment, we set up VNs to perform energy minimization following the proximal gradient method (20) by fixing the number of components to $C = 1$, i. e., $F(\boldsymbol{x}) = f_1(\boldsymbol{x})$. For both denoising and non-blind deblurring, we train convex and non-convex VNs up to $t = 100$ steps. The resulting PSNR scores and the ℓ_2-norm of the gradients $\|\nabla F(\boldsymbol{x}_t)\|_2$ are depicted in green color in Fig. 5 and 6. As expected, the decreasing gradient-norm with increasing steps t indicates that the methods actually minimize the underlying energy (10).

The PSNR curves for denoising (Fig. 5) differ for convex and non-convex $\text{N-VN}_{24}^{1,t}$. The performance of the non-convex VNs increases initially and slowly declines with increasing t, while the convex $\text{N-VN}_{24}^{1,t}$ yield the best results after a single step. This indicates that a convex regularization of the form (11) is not a good prior for natural images because by approaching a minimizer (increasing t) the results become worse. Surprisingly, the highly parametrized convex $\text{N-VN}_{24}^{1,t}$ performs marginally better than the ROF model for $t > 10$, consistent with [25]. In the case of non-blind deblurring the PSNR curves (Fig. 6) are similar for convex and non-convex $\text{B-VN}_{24}^{1,t}$. Both VNs require more steps to yield satisfactory results since deblurring is a harder problem than denoising.

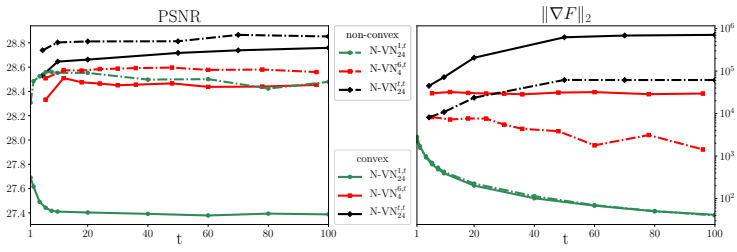


Fig. 5. Average PSNR curves on the test set of the trained VN types for Gaussian image denoising along with the gradient norm of the corresponding energy $F(\boldsymbol{x}_t)$.

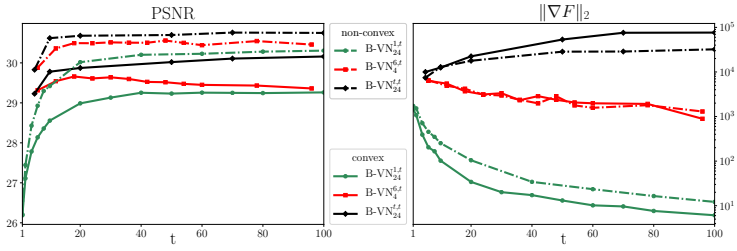


Fig. 6. Average PSNR scores and corresponding gradient norm on the test set of the different VN types for non-blind deblurring.

Nevertheless, the non-convex B-VN$_{24}^{1,t}$ outperform the convex ones by a large margin (1dB).

4.2 Approximate Incremental Minimization with VNs

In a second experiment, we evaluate the performance of VNs that follow an incremental approximate energy minimization scheme (22). We use $C = 6$ components and $N_r = 4$ kernel-function pairs. Thus, the number of parameters is approximately the same as in the previous experiment. The resulting PSNR scores as well as the gradient norm for the trained convex and non-convex VN$_4^{6,t}$ are depicted in red color in Fig. 5 for denoising and Fig. 6 for non-blind deblurring.

In contrast to the previous experiment, the PSNR curves for denoising and deblurring are rather flat for both convex and non-convex VN$_4^{6,t}$. So, they manage to generate good results after just a few steps and maintain the quality with increasing t. However, the results after 100 steps are far from approaching a stationary point, as indicated by the rather slowly decreasing gradient-norm $\|\nabla F\|_2$. This effect is very strong for the convex N-VN$_4^{6,t}$ because these VNs learn a sequence of components that alternate between strong blurring and detail recovery from the data term, leading to large gradients. In terms of PSNR scores this behavior yields superior results compared to the first experiment. The de-

Table 2. Average PSNR scores on the test set for the VN types. The reported PSNR scores are computed using the best performing depth t of each VN type.

	ROF[38]	convex			non-convex			BM3D[15]	$\text{TRD}^5_{5\times5}$[14]
		$\text{VN}^{1,t}_{24}$	$\text{VN}^{6,t}_{4}$	$\text{VN}^{t,t}_{24}$	$\text{VN}^{1,t}_{24}$	$\text{VN}^{6,t}_{4}$	$\text{VN}^{t,t}_{24}$		
denoising	27.39	27.69	28.51	*28.76*	28.56	28.60	**28.87**	28.56	*28.78*
non-blind deblurring	28.35	29.26	29.66	30.16	30.31	*30.56*	**30.76**	-	-

creasing PSNR of the convex B-$\text{VN}^{6,t}_{4}$ with increasing depth may originate from local optima of the learning problem.

4.3 VNs in a Reaction Diffusion Setup

In the final experiment, we investigate the performance of VNs in a residual network or trainable reaction-diffusion setting [14], i. e., each step (13) has its own parameter set $\boldsymbol{\theta}_t$ ($C = t$). Hence, the number of parameters increases linearly with the depth of the $\text{VN}^{t,t}_{24}$. These VN types can still be interpreted as an incremental proximal methods that apply each component just once.

The increasing model complexity with increasing t leads to a steady increase of the performance for the $\text{VN}^{t,t}_{24}$ on both reconstruction tasks, depicted in Fig. 5 and 6. The gradient-norm increases also along with the depth t due to the additional components. Consequently, these VNs do not minimize a corresponding energy. However, they yield the best performance on the image reconstruction tasks as shown in Table 2. In contrast to Chen et al. [14], our findings on image denoising suggest that the shape of the learned potential functions (Fig. 4) is of little importance since the convex and non-convex N-$\text{VN}^{t,t}_{24}$ perform almost equally well, as shown in Table 2. The convex N-VNs rather require the flexibility of incremental schemes in order to yield satisfactory results. Still, convexity seems to be a limiting factor for non-blind deblurring since all convex VNs perform worse than the non-convex ones.

5 Conclusion

In this work, we explored links between variational energy minimization methods and deep learning approaches by introducing variational networks (VNs). The VNs consist of stacked parametrized incremental proximal steps that have the same favorable computational structure as residual units. We demonstrated that the versatile VN formulation can be used to learn proximal gradient schemes, incremental proximal schemes as well as residual networks and optimized reaction-diffusion processes. Moreover, our parametrization of the VNs for image reconstruction allows us to learn corresponding *convex* energies.

We used this novel possibility to evaluate the limitations of convexity in the context of natural image reconstruction. Our findings on denoising and non-blind deblurring show that our convex formulations yield inferior results than

non-convex formulations. Additionally, the incremental VN types require just a few steps to yield reasonable results even for the challenging task of non-blind deblurring. In the future we would like to further investigate the role of convexity by learning different classes of convex models and analyze the stability of VNs.

Acknowledgements

We acknowledge grant support from the Austrian Science Fund (FWF) under the START project BIVISION, No. Y729 and the European Research Council under the Horizon 2020 program, ERC starting grant HOMOVIS, No. 640156.

References

1. Alexandre, B.: Sharesnet: reducing residual network parameter number by sharing weights. arXiv e-prints 1702.08782 (2017)
2. Beck, A., Teboulle, M.: A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. SIIMS 2(1), 183–202 (2009)
3. Bertalmio, M., Sapiro, G., Randall, G.: Morphing Active Contours. TPAMI 22(7), 733–737 (2000)
4. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1999)
5. Bertsekas, D.P.: Incremental proximal methods for large scale convex optimization. Mathematical Programming 129(2), 163 (Jun 2011), https://doi.org/10.1007/s10107-011-0472-0
6. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming, vol. 5. Athena Scientific (1996)
7. Black, M.J., Anandan, P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. Computer vision and image understanding 63(1), 75–104 (1996)
8. Blake, A., Zisserman, A.: Visual reconstruction. MIT press (1987)
9. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
10. Bredies, K., Kunisch, K., Pock, T.: Total generalized variation. SIIMS 3(3), 492–526 (2010)
11. Chambolle, A., Lions, P.L.: Image recovery via total variation minimization and related problems. Numerische Mathematik 76(2), 167–188 (1997)
12. Chan, T.F., Vese, L.A.: Active Contours Without Edges. IEEE Transactions on Image Processing 10(2), 266–277 (2001)
13. Chen, Y., Ranftl, R., Pock, T.: Insights Into Analysis Operator Learning: From Patch-based Sparse Models to Higher Order MRFs. IEEE Transactions on Image Processing 23(3), 1060–1072 (2014)
14. Chen, Y., Yu, W., Pock, T.: On Learning Optimized Reaction Diffusion Processes for Effective Image Restoration. In: CVPR (2015)
15. Dabov, K., Foi, A., Katkovnik, V.: Image Denoising by Sparse 3D Transformation-domain Collaborative Filtering. IEEE Transactions on Image Processing 16(8), 1–16 (2007)
16. Domke, J.: Generic Methods for Optimization-Based Modeling. AISTATS pp. 318–326 (2012)
17. Freedman, D., Zhang, T.: Active Contours for Tracking Distributions. IEEE Transactions on Image Processing 13(4), 518–526 (2004)

18. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: ICML (2010)
19. Hammernik, K., Knoll, F., Sodickson, D., Pock, T.: Learning a Variational Model for Compressed Sensing MRI Reconstruction. In: ISMRM (2016)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (2016)
21. Hinton, G.E.: Learning distributed representations of concepts. In: Proceedings of the eighth annual conference of the cognitive science society (1986)
22. Horn, B., Schunck, B.: Determining Optical Flow. Artificial Intelligence 17, 185–203 (1981)
23. Klatzer, T., Hammernik, K., Knöbelreiter, P., Pock, T.: Learning Joint Demosaicing and Denoising Based on Sequential Energy Minimization. In: ICCP (2016)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: NIPS (2012)
25. Kunisch, K., Pock, T.: A Bilevel Optimization Approach for Parameter Learning in Variational Models. SIIMS 6, 938–983 (2013)
26. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation 1(4), 541–551 (1989)
27. Lee, A.B., Pedersen, K.S., Mumford, D.: The Nonlinear Statistics of High-Contrast Patches in Natural Images. IJCV 54(5413), 83–103 (2003)
28. Levin, A., Weiss, Y., Durand, F., Freeman, W.T.: Understanding and Evaluating Blind Deconvolution Algorithms. In: CVPR (2009)
29. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common Objects in Context. In: ECCV (2014)
30. Mangasarian, L., Solodov, M.: Backpropagation Convergence Via Deterministic Nonmonotone Perturbed Minimization. NIPS 6, 383–390 (1994)
31. Mao, X.J., Shen, C., Yang, Y.B.: Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections. arXiv e-prints 1606.08921 (2016)
32. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In: ICCV (2001)
33. Masoud, A., Saeid, N.: Multi-residual networks. arXiv e-prints 1609.05672 (2016)
34. Mumford, D., Shah, J.: Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems. Communications on Pure and Applied Mathematics 42(5), 577–685 (1989)
35. Nedić, A., Bertsekas, D.P., Borkar, V.S.: Distributed Asynchronous Incremental Subgradient Methods. Studies in Computational Mathematics 8(C), 381–407 (2001)
36. Nedić, A., Bertsekas, D.: Convergence rate of incremental subgradient algorithms. In: Stochastic optimization: algorithms and applications, pp. 223–264. Springer (2001)
37. Roth, S., Black, M.J.: Fields of experts. IJCV 82, 205–229 (2009)
38. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear Total Variation Based Noise Removal Algorithms. Physica D: Nonlinear Phenomena 60(1-4), 259–268 (1992)
39. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Representations by Back-propagating Errors. Nature 323(6088), 533–536 (1986)
40. Schelten, K., Nowozin, S., Jancsary, J., Rother, C., Roth, S.: Interleaved regression tree field cascades for blind image deconvolution. In: IEEE Winter Conference on Applications of Computer Vision (2015)

41. Schmidt, U., Roth, S.: Shrinkage fields for effective image restoration. In: CVPR (2014)
42. Shan, Q., Jia, J., Agarwala, A.: High-quality motion deblurring from a single image. In: SIGGRAPH (2008)
43. Sra, S.: Scalable Nonconvex Inexact Proximal Splitting. In: NIPS (2012)
44. Veit, A., Wilber, M., Belongie, S.: Residual Networks are Exponential Ensembles of Relatively Shallow Networks. arXiv e-prints 1605.06431 (2016)
45. Xu, L., Zheng, S., Jia, J.: Unnatural L0 Sparse Representation for Natural Image Deblurring. In: CVPR (2013)
46. Yu, W., Heber, S., Pock, T.: Learning Reaction-Diffusion Models for Image Inpainting. In: GCPR. vol. 9358 (2015)
47. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. arXiv e-prints 1608.03981 (2016)
48. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss Functions for Neural Networks for Image Processing. arXiv e-prints 1511.08861 (2015)
49. Zhu, S.C., Mumford, D.: Prior learning and gibbs reaction-diffusion. TPAMI 19(11), 1236–1250 (1997)

Supplemental Material

Incremental Methods

In this part we briefly survey incremental gradient methods [4, 6] and incremental proximal methods [5, 43] to ease understanding of the main paper and the proofs.

Incremental Gradient Methods

Incremental gradient methods [4, 6] were developed to minimize problems of the form

$$\min_{\boldsymbol{x}\in\mathcal{X}} F(\boldsymbol{x}) := \sum_{c=1}^{C} f_c(\boldsymbol{x}) \ , \tag{17}$$

where the individual component functions $f_c : \Re^n \mapsto \Re$ are real-valued functions and $\mathcal{X} \subseteq \Re^n$ is a closed convex set. In empirical risk minimization or neural network training, the number of components C is typically very large. The basic idea is to operate on a single component function $f_c(x)$ at each minimization step in order to speed up the optimization procedure. This has implications for the type of optimization algorithms that can be used, and enables a parallel implementation which is very important for today's large-scale learning problems.

The most widespread incremental gradient method [21, 30] has the form

$$\boldsymbol{x}_{t+1} = \mathrm{proj}_{\mathcal{X}}\left(\boldsymbol{x}_t - \eta_t \nabla f_{c(t)}(\boldsymbol{x}_t)\right) \ , \tag{18}$$

where $\mathrm{proj}_{\mathcal{X}}(\cdot)$ is the projection onto a set $\mathcal{X}$, η_t defines the step size at iteration t and $c(t)$ selects the component for the t-th iteration. The basic differences

between variants of (18) are the selection of the step size η_t and how the components are distributed to each iteration $c(t)$, which can be either random or deterministic (e. g., repeated cycle). The convergence of all these variants has been proven under various conditions, e. g., [30] showed convergence for cyclic order and a diminishing step size.

Incremental Proximal Methods

We are especially interested in the incremental subgradient and proximal methods, which were proposed by Nedić and Bertsekas [5, 36, 35]. The intuition behind *incremental proximal methods* (IPM) is that the components $f_c(x)$ of Problem (17) can be partitioned in smooth and non-smooth functions to obtain

$$\min_{\boldsymbol{x}\in\mathcal{X}} F(\boldsymbol{x}) = f(\boldsymbol{x}) + h(\boldsymbol{x}) = \sum_{c=1}^{C} f_c(\boldsymbol{x}) + h(\boldsymbol{x}) \ , \tag{19}$$

where f aggregates the smooth components f_c and $h : \mathfrak{R}^n \mapsto \mathfrak{R}$ is lower semi-continuous (possibly non-smooth) and convex. Problem (19) can be turned into its unconstrained form by setting $h(x)$ to the indicator function of $\mathcal{X}$. A simple approach to minimize (19) is to use a proximal gradient scheme such as

$$\boldsymbol{x}_{t+1} = \operatorname{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t)\right) \ , \tag{20}$$

where $\nabla f(\boldsymbol{x}_t)$ is the gradient of the smooth components, η_t defines the step size and the proximal map is defined by

$$\operatorname{prox}_h^{\eta}(\boldsymbol{z}) = \arg\min_{\boldsymbol{x}} h(\boldsymbol{x}) + \frac{1}{2\eta}\left\|\boldsymbol{x} - \boldsymbol{z}\right\|_2^2 \ . \tag{21}$$

Analogous to [5] the incremental proximal gradient step is given by

$$\boldsymbol{x}_{t+1} = \operatorname{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f_{c(t)}(\boldsymbol{x}_t)\right) \ , \tag{22}$$

where $\nabla f_{c(t)}(\boldsymbol{x}_t)$ is the gradient of a single component selected by $c(t)$.

Convex Incremental Problems We first study the convergence properties of the convex version of Problem (19). For this, all its functions f_c and h must be convex. In this case, convergence follows in analogy to [5, 43]. We require that the functions f_c are Lipschitz continuous on $\mathcal{X}$. That is, for $c = 1 \dots C$ and all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ there exists a constant L such that

$$\left|f_c(\boldsymbol{x}) - f_c(\boldsymbol{y})\right| \leq L \left\|\boldsymbol{x} - \boldsymbol{y}\right\| \tag{23}$$

holds. Then, the incremental proximal method (22) converges to an approximate stationary point in the case of constant step sizes $\eta_t \geq \eta > 0$. Note that the formulation of [5] allows multiple functions h_c in the partitioning (19). Without loss of generality, we can subsume all non-smooth and convex parts into a single function h and set $h_c = h/C$, to end up with the same algorithm as in their original formulation.

Table 3. Summary of convergence properties for the proximal gradient method (20) and IPM (22) for solving Problem (19). A (✓) indicates whether a method converges in the limit to a global minimum in the case of convex functions or to a stationary point in the case of non-convex functions. However, (ϵ–✓) denotes approximate convergence. IPM converges exactly for diminishing step sizes [5, 43], i. e., $\sum_{t=0}^{\infty} \eta_t = \infty$, $\sum_{t=0}^{\infty} \eta_t^2 < \infty$. The right column indicates the type of the corresponding VN.

Method	Step Size	Convex	Non-Convex	Type
$\boldsymbol{x}_{t+1} = \text{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla F(\boldsymbol{x}_t)\right)$	$0 < \eta_t \leq \eta$	✓	✓	$\text{VN}_N^{1,t}$
$\boldsymbol{x}_{t+1} = \text{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f_{c(t)}(\boldsymbol{x}_t)\right)$	$0 < \eta_t \leq \eta$	ϵ–✓	ϵ–✓	$\text{VN}_N^{C,t}$
	diminishing	✓	✓	-

Non-convex Incremental Problems If we allow the component functions to be non-convex, Problem (17) becomes non-convex and possibly non-smooth, since h can still be non-smooth. We show convergence of (22) to an approximate stationary point in analogy to the NIPS framework of Sra [43], which considers problems of the form

$$\min_{\boldsymbol{x} \in \mathcal{X}^n} f(\boldsymbol{x}) + h(\boldsymbol{x}) \ , \tag{24}$$

where $f : \mathcal{X}^n \mapsto \mathfrak{R}$ is continuously differentiable and $h : \mathcal{X}^n \mapsto \mathfrak{R}$ is lower semi-continuous and convex (possibly non-smooth). NIPS requires that f has a Lipschitz continuous gradient, i.e. $\exists L_{\nabla f} > 0$:

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L_{\nabla f} \|\boldsymbol{x} - \boldsymbol{y}\| \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}^n \ . \tag{25}$$

The iterative scheme of NIPS is defined as

$$\boldsymbol{x}_{t+1} = \text{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t) + \eta_t e(\boldsymbol{x}_t)\right) \ , \tag{26}$$

where $e(\boldsymbol{x}_t)$ models an error in the gradient estimate $\nabla f(\boldsymbol{x}_t)$. The iterative scheme assumes that for $\eta_t \geq \eta > 0$ the computational error is uniformly bounded, that is

$$\eta \|e(\boldsymbol{x})\| \leq \epsilon \quad \forall \boldsymbol{x} \in \mathcal{X}^n. \tag{27}$$

Based on this assumption, [43] showed that (26) converges to an approximate stationary point.

If we apply the NIPS framework (26) to minimize (19) and rearrange the summands, we get

$$\boldsymbol{x}_{t+1} = \text{prox}_h^{\eta_t}\left(\boldsymbol{x}_t - \eta_t \nabla f_{c(t)}(\boldsymbol{x}_t) - \eta_t \sum_{\substack{j=1 \\ j \neq c(t)}}^{C} \nabla f_j(\boldsymbol{x}_t)\right) \ . \tag{28}$$

Thus, the gradient error of (22) is given by the gradients of the components that are *not* selected. If we assume that all components are Lipschitz continuous with

parameter L, as in the convex case, its upper bound is given by

$$\|e_t(\boldsymbol{x}_t)\| = \left\| \sum_{\substack{j=1 \\ j\neq c(t)}}^{C} \nabla f_j(\boldsymbol{x}_t) \right\| \leq \sum_{\substack{j=1 \\ j\neq c(t)}}^{C} L = (C-1)L \ . \tag{29}$$

In the non-convex case, NIPS ensures approximate convergence to a stationary point.

In both the convex and the non-convex case, (22) converges to an approximate stationary point for $0 < \eta_t \leq \eta$ if all components are Lipschitz continuous (and also their gradients in the non-convex case). Table 3 summarizes the convergence analysis and outlines their relation to VNs.

Lipschitz Continuity of the VNs for Image Reconstruction

In order to apply the theoretical properties of incremental methods to the VNs for image reconstruction, we need to show that the components as well as the gradients of

$$f_c(\boldsymbol{x}; \boldsymbol{\theta}_c) = R_c(\boldsymbol{x}; \boldsymbol{\theta}_c) + D_c(\boldsymbol{x}; \boldsymbol{\theta}_c) \tag{30}$$

are Lipschitz continuous in $\boldsymbol{X}$. The regularization term

$$R_c(\boldsymbol{x}; \boldsymbol{\theta}_c) = \sum_{i=1}^{N_k} \sum_{j=1}^{n} \phi_i^c((K_i^c \boldsymbol{x})_j) \tag{31}$$

is continuously differentiable iff the potential functions $\phi_i^c(y)$ are differentiable. Since we parametrize the gradient of the potential functions during learning by

$$\phi_i^{c\prime}(y) = \sum_{j=1}^{N_w} \exp\left(-\frac{(y-\mu_j)^2}{2\sigma^2}\right) w_{ij}^c \ , \tag{32}$$

its maximal value is bounded if the weights w_{ij}^c are bounded, which is ensured during training. Consequently, its Lipschitz constant is given by this bound. The same analysis can be applied to show that the gradient $\phi_i^{c\prime}(y)$ is Lipschitz continuous. Additionally, the Lipschitz continuity of the learned data term and its gradient can be shown in the same fashion.

Training and Projecting onto the Admissible Set $\mathcal{T}$

As described in the paper, we constrain the parameters $\boldsymbol{\theta}$ to lie in an admissible set $\mathcal{T}$. To solve the training problem we propose the inertial incremental proximal method (IIPG) defined in Algorithm 1, where $\delta_{\mathcal{T}}(\boldsymbol{\theta})$ is the indicator function of $\mathcal{T}$.

Algorithm 1: Inertial incremental proximal gradient (IIPG) algorithm.

Input: Training set $\mathcal{S}$, step size α and number of epochs N_E and minibatches N_B

Partition $\mathcal{S}$ into N_B minibatches $\mathcal{S} = \bigcup_{b=1}^{N_B} \mathcal{B}_b$
Choose initial parameters $\boldsymbol{\theta}^0$
$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0$;
$l \leftarrow 1$;
for $e \leftarrow 1$ **to** N_E **do**
  for $b \leftarrow 1$ **to** N_B **do**
    Perform over-relaxation
    $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}^l + \frac{e-1}{e+2}(\boldsymbol{\theta}^l - \boldsymbol{\theta}^{l-1})$;
    Compute gradient on $\mathcal{B}_b$
    $\boldsymbol{g}^l \leftarrow \partial L(\tilde{\boldsymbol{\theta}})/\partial \boldsymbol{\theta}$;
    Compute preconditioning P^l *by* (34) *and* (35)
    Perform proximal gradient descent step
    $\boldsymbol{\theta}^{l+1} \leftarrow \operatorname{prox}^{\alpha P^l}_{\delta(\mathcal{T})}(\tilde{\boldsymbol{\theta}} - \alpha P^l \boldsymbol{g}^l)$;
    $l \leftarrow l + 1$;

For image reconstruction, we introduce the following constraints on the parameters. We enforce that the convolution kernels $\boldsymbol{k}_i^c$ and $\bar{\boldsymbol{k}}_i^c$ have zero-mean and are normalized, i. e.,

$$\boldsymbol{k}_i^c, \bar{\boldsymbol{k}}_i^c \in \mathcal{K} = \left\{ \boldsymbol{k} \in \Re^{h^2} : \mathbf{1}^\top \boldsymbol{k} = 0, \|\boldsymbol{k}\|_2 = 1 \right\} , \tag{33}$$

in order to ensure that the domain $\mathcal{Y}^n$ of the convolution result $(K_i^c \boldsymbol{x})$ is bounded and symmetric around zero. The proximal map for the kernels in Algorithm 1 simplifies to the projection on $\mathcal{K}$ which can be simply computed by subtracting the mean and re-normalization.

To speed up Algorithm 1, we use a diagonal block-wise preconditioning matrix P^l given by

$$P^l = \operatorname{diag}\left(P^l_{\boldsymbol{k}_1^1}, P^l_{\boldsymbol{w}_1^1}, \ldots, P^l_{\boldsymbol{k}^C_{N_k}}, P^l_{\boldsymbol{w}^C_{N_k}}, P^l_{\lambda^C} \right) , \tag{34}$$

where the diagonal matrices $P^l_{\boldsymbol{p}}$ for the individual parameters are defined by

$$P^l_{\boldsymbol{p}} = \left\| \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{p}} \right\|_2^{-1} I , \tag{35}$$

where $\boldsymbol{p} \in \{\boldsymbol{k}_i^c, \boldsymbol{w}_i^c, \lambda^c\}$ and I is the corresponding identity matrix.

Enforcing convexity of the potential functions Our goal is to investigate the limitations of convexity due to its property that each local minimum is a global minimum. Therefore, we need to learn convex potential functions $\rho_i^c(y)$. Their domain is a closed bounded subset of $\mathcal{Y} \subset \Re$ because the input images are bounded ($\boldsymbol{x} \in \mathcal{X} := \{x \in \Re : 0 \leq x \leq m\}$) and the kernel have norm one. Thus,

$\mathcal{Y} = \{y \in \mathfrak{R} : |y| \leq m\}$ is a convex set. Since the potential functions are scalar, a sufficient condition for convexity is

$$\phi_i^{c\prime\prime}(y) \geq 0 \; \forall y \in \mathcal{Y} \; . \tag{36}$$

Hence, we need to ensure that $\phi_i^{c\prime\prime}$ is positive over $\mathcal{Y}$. Its is given by

$$\phi_i^{c\prime\prime}(y) = -\sum_{j=1}^{N_w} \frac{(y-\mu_j)}{\sigma^2} \exp\left(-\frac{(y-\mu_j)^2}{2\sigma^2}\right) w_{ij}^c \; , \tag{37}$$

which can be shortened in matrix vector notation to

$$\phi_i^{c\prime\prime}(\boldsymbol{y}) = \boldsymbol{\Phi}_i^{c\prime\prime}(\boldsymbol{y}) \boldsymbol{w}_i^c \; , \tag{38}$$

where the matrix $\boldsymbol{\Phi}_i^{c\prime\prime}(\boldsymbol{y}) \in \mathfrak{R}^{n \times N_w}$ holds coefficients for each radial base. Since we cannot test the convexity condition (36) for all elements in $\mathcal{Y}$, we define control points $\boldsymbol{y}_p \in \mathcal{Y}^{N_p}$. In practice it turned out that $N_p = 2N_w + 1$ yields enough control points to ensure convexity of $\phi_i^c(y)$ on $\mathcal{Y}$ due to the overlap of the individual radial basis functions. Consequently, the weights $\boldsymbol{w}_i^c$ of a influence function $\phi_i^c(y)$ have to lie in the set

$$\boldsymbol{w}_i^c \in \mathcal{W} = \left\{\boldsymbol{w} \in \mathfrak{R}^{N_w} : A\boldsymbol{w} \leq 0\right\} \tag{39}$$

with $A = -\boldsymbol{\Phi}^{c\prime\prime}(\boldsymbol{y}_p)$. We can easily incorporate this constraint in the proximal map of Algorithm 1 for $\boldsymbol{w}_i^c$

$$\boldsymbol{w}_i^{c,l} = \text{prox}_{\delta(\mathcal{W})}^{\eta P^l}(\boldsymbol{z}) = \underset{A\boldsymbol{w} \leq 0}{\arg\min} \frac{1}{2} \left\|\boldsymbol{w} - \boldsymbol{z}\right\|_2^2 \tag{40}$$

with $\boldsymbol{z} = \boldsymbol{w}_i^{c,l-1} - \alpha P_{\boldsymbol{w}_1^1}^l \partial L / \partial \boldsymbol{w}_i^c$. We add Lagrange multipliers $\boldsymbol{\tau} \in \mathfrak{R}^{N_c}$ to transform (40) into the saddle point problem

$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\tau} \geq 0} \frac{1}{2} \left\|\boldsymbol{w} - \boldsymbol{z}\right\|_2^2 + \boldsymbol{\tau}^\top A \boldsymbol{w} \; . \tag{41}$$

Its closed form solution is

$$\boldsymbol{w} = \boldsymbol{z} - A^\top \boldsymbol{\tau} \; . \tag{42}$$

By plugging this into (41) and rearranging terms, we get the quadratic problem

$$\min_{\boldsymbol{\tau}} \frac{1}{2} \left\|A^\top \boldsymbol{\tau} - \boldsymbol{z}\right\|_2^2 \quad \text{s.t.} \quad \boldsymbol{\tau} \geq 0 \; , \tag{43}$$

which can be efficiently solved by FISTA [2]. The proximal gradient step of $\boldsymbol{w}_i$ (42) can be performed with the minimizer of (43). Note that the quadratic problem (43) must be solved in every iteration of Algorithm 1. However, the problem can be easily parallelized for all potential functions, which helps to keep the overhead for convex functions minimal.

Table 4. Average PSNR scores and gradient norm $\|\nabla F\|_2$ on the test set for $\mathrm{VN}_{24}^{1,t}$. The reported PSNR scores are computed by performing t iterations on the energy learned for the $\mathrm{VN}_{24}^{1,100}$.

	convex				non-convex			
	$\mathrm{VN}_{24}^{1,1000}$	$\mathrm{VN}_{24}^{1,8000}$	$\mathrm{VN}_{4}^{6,996}$	$\mathrm{VN}_{4}^{1,7998}$	$\mathrm{VN}_{24}^{1,1000}$	$\mathrm{VN}_{24}^{1,8000}$	$\mathrm{VN}_{4}^{6,996}$	$\mathrm{VN}_{4}^{1,7998}$
denoising	21.84 / 3.40	18.57 / 0.45	28.45 / 29,593	28.45 / 29,591	25.32 / 1.155	25.13 / 0.02	28.37 / 1,499	28.36 / 1,501
non-blind deblurring	28.43 / 0.98	25.24 / 0.45	28.52 / 889.11	25.20 / 888.46	29.19 / 1.52	25.41 / 1.22	29.27 / 1,290	25.05 / 1,285

Training details

For all experiments, we partitioned the 400 training patches into mini-batches of size 20 and performed 150 epochs of the IIPGD algorithm 1 with step size 0.05. After every 50-th epoch, we reduced the step size by a factor of 0.5.

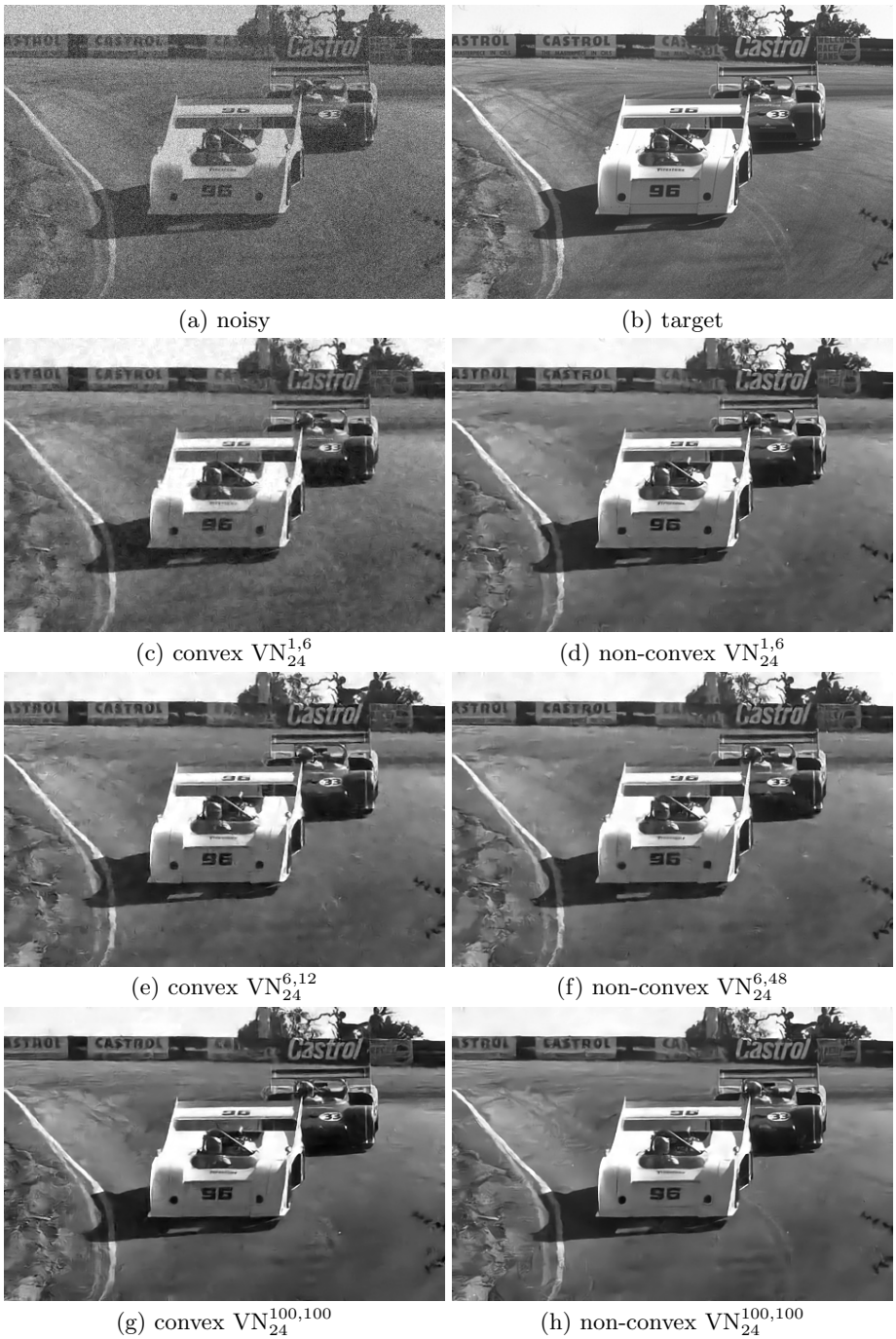
All the influence functions are parametrized by $N_w = 31$ radial basis functions in contrast to the 63 used by [14].

Minimizing the VN energy till approximate convergence

Since the VNs in an incremental setting learn a corresponding energy, an interesting experiment is to continue the minimization scheme. Therefore, we continued the minimization scheme of trained VNs for up to 8000 steps and evaluated the performance. The corresponding PSNR sores and gradient norms are depicted in Tab 4. In the case of direct energy minimization $C = 1$ the PSNR values decrease continuously for the convex and non-convex VNs on both tasks along with the gradient norm. This effect was expected since the networks were trained for just 100 steps. However, in the incremental setting $C = 6$, the denoising VNs maintain the PSNR performance. Also the the gradient norm remains stable, which indicates that the incremental denoising VNs tend towards a fixed point. The PSNR score of the non-blind deblurring VNs decreases with increasing t, while the gradient norm remains on a constant level. The PSNR decrease is mainly due to border handling artifacts.

Qualitative results

The Fig. 7 and 8 depict qualitative results of the different learned VN types. In general the non-convex models yield better results than the convex ones.



(a) noisy (b) target

(c) convex $\mathrm{VN}_{24}^{1,6}$ (d) non-convex $\mathrm{VN}_{24}^{1,6}$

(e) convex $\mathrm{VN}_{24}^{6,12}$ (f) non-convex $\mathrm{VN}_{24}^{6,48}$

(g) convex $\mathrm{VN}_{24}^{100,100}$ (h) non-convex $\mathrm{VN}_{24}^{100,100}$

Fig. 7. Qualitative results of the various VN types for image denoising. Note the convex VNs generate artifacts in smooth regions, whereas, the non-convex avoid those.

(a) noisy and blurry (b) target

(c) convex $\mathrm{VN}_{24}^{1,100}$ (d) non-convex $\mathrm{VN}_{24}^{1,100}$

(e) convex $\mathrm{VN}_{24}^{6,18}$ (f) non-convex $\mathrm{VN}_{24}^{6,18}$

(g) convex $\mathrm{VN}_{24}^{100,100}$ (h) non-convex $\mathrm{VN}_{24}^{100,100}$

Fig. 8. Qualitative results of the various VN types for non-blind image deblurring. Note the convex VNs results seem to be a bit more noisy than the non-convex results.