A Primal Dual Network for Low-Level Vision Problems

Christoph Vogel[1] Thomas Pock[1,2]

[1] Graz University of Technology [2] AIT Austrian Institute of Technology

Abstract. In the past, classic energy optimization techniques were the driving force in many innovations and are a building block for almost any problem in computer vision. Efficient algorithms are mandatory to achieve real-time processing, needed in many applications like autonomous driving. However, energy models - even if designed by human experts - might never be able to fully capture the complexity of natural scenes and images. Similar to optimization techniques, Deep Learning has changed the landscape of computer vision in recent years and has helped to push the performance of many models to never experienced heights. Our idea of a *primal-dual network* is to combine the structure of regular energy optimization techniques, in particular of first order methods, with the flexibility of Deep Learning to adapt to the statistics of the input data.

1 Introduction

Classic energy optimization techniques, like dynamic programming, graph-cut, belief propagation, dual decomposition and first-order methods [26,16,25,8,17] paved the way for several milestones of computer vision. This led to efficient algorithms for stereo [23], optical flow [3] and image editing [38], to name only a few examples. However, there is reason to believe that energy based models like Potts model [48], Total Variation (TV) [39], general Markov-Random-Fields (MRF) [27] or even any expert designed energy formulation will never be able to fully capture the complexity of natural images.

More recently, convolutional neural networks (CNN), omnipresent for classical machine learning problems like object detection and image classification [40], also start to surpass energy based models for more and more low-level vision tasks *e.g.* [15,14].

Our idea is to unify both fields. Relying on the structure of regular energy optimization algorithms we seek to utilize the capability of Deep Learning (DL) to adapt to the statistics of data, to learn a generalized optimization algorithm. In other words, we do not predefine the energy but rather provide a specific algorithmic structure to generate an output for a certain task. This paper aims to take a step towards this ambitious goal.

To provide this algorithmic framework we believe that first order (FO)-methods are the most appropriate of the optimization algorithms mentioned. FO-methods possess a structure that can be generalized, are inherently parallel and simple enough to be implemented in neural network. As an example for this class of algorithms, we utilize the algorithm of Chambolle and Pock [8], which is designed for general non-smooth problems and can be applied for many tasks in computer vision.

Not only the high memory consumption of backpropagation limit the number of iterations the network can perform, such that we can only expect an approximate solution. Here, machine learning techniques like DL can be helpful to improve the approximation

quality. Our learned algorithm does not need to reserve equal capacity to all possible inputs, but can adapt its local filters to activate on the more likely and w.r.t. the approximation quality, more relevant substructures in the data. On the other hand, given the algorithmic framework to solve a specific problem, DL can potentially generalize the structure to also provide approximate solutions for related problem instances.

In a CNN, the receptive field of a neuron is limited by the size of its filters, such that the number of necessary steps to transport local information between unknowns that are not coupled directly will easily exceed the maximal possible number of iterations of a network. FO methods have essentially the same problems. Here, multigrid methods [2,4,7] can accelerate convergence. In this work we resort to a simpler strategy, which works on the same problem over multiple scales. In the encoding stage, operating from fine to coarse, a feature representation of the input is generated and used later, at the respective resolution in the inference stage. The inference phase proceeds from coarse to fine and executes several optimization steps given the upsampled output of the stage below and the features. To apply the multiscale scheme on arbitrarily sized input data, we introduce the idea of weight sharing across the different resolution levels.

The contributions of this work are threefold. At first, we train and analyze our network on known models from the TV family to obtain high-quality approximate solutions for all these models in only a fraction of time, compared to regular energy optimization. We further exploit that multi-label problems with a linearly ordered label set can be formulated as a ROF problem and solved globally optimal [24,5,35]. Our multiscale extension allows to overcome the limited receptive field of convolutions and due to our idea of sharing weights across resolution levels we can apply our model on inputs of arbitrary size. Finally, we train our proposed network structure without the requirement to minimize a certain energy model and achieve state-of-the-art results for image denoising and competitive results for stereo estimation.

2 Related Work

Provably optimal convergence rates for different problem classes, and a low memory footprint render first-order methods [13,1,8] attractive for large-scale computer vision problems. Despite optimality, accelerating these methods further is ongoing research. These efforts include pre-conditioning [34], inertia or heavy-ball like acceleration for strongly convex problems [1,8] and lately accelerating the computation of certain proximal operators [9]. In contrast to these methods we voluntarily abandon the idea of optimality and (partially) even that of minimizing an energy. Instead we search for the best solution that can be reached in a limited and very small number of steps. We only specify the algorithmic framework, namely we follow the primal-dual method proposed in [8] that has also been applied to non-convex optimization problems [44].

Multi-grid methods [2] are a general framework to accelerate optimization and have lately been extended to non-smooth optimization problems [4,7]. In this work we resort to a simpler, less sophisticated multiscale scheme. However, apart from a coarse solution, our method also has access to higher dimensional feature maps.

First-order methods have already found their way into neural networks. Possibly the first application was proposed in the context of sparse coding [21]. Here, an unrolled

and generalized implementation of the Ista algorithm [13] is shown to require less iterations to achieve a certain approximation quality than its accelerated version [1]. This framework was later extended by [46] and used for image super-resolution in [47]. [11] proposes a more general network structure in which each iteration can be interpreted as one gradient step of a certain (at each step different) energy function. Later a network that essentially implements the same idea was introduced in [50]. More recently, [37] suggest to unroll iterations of a primal-dual method for the same application and learn the step sizes and smoothness weights. In contrast, we propose a general structure that can be applied to multiple problems. We augment our network with features generated from the input that are used to generalize our projection operations and filters. Parameters of the algorithm, like the smoothness weight, are not learned and kept fixed, but treated as additional input, such that our network behaves just like a general algorithm. We provide a principled evaluation of different generalizations and finally, all of these works do not use a multiscale scheme, but always operate on the finest resolution level.

Our coarse-to-fine scheme is realized as a stacked autoencoder [45] with additional lateral connections [36]. Similar architectures have been successfully used for different applications, *e.g.* for semantic segmentation [29], stereo [31] and optical flow [15].

We focus our investigation on problems from the Total Variation family, and in particular on the model proposed by Rudin, Osher and Fatemi [39]. Its relation to graph-cut [26] was analyzed in [5] and exploited in [35] to solve a certain class of multi-label problems following the construction of [24]. This links our work also to approaches that combine deep neural networks and Markov Random Fields (MRF) [28,41,51].

Our design is further inspired by the residual network architecture (Resnet) [22], which introduces skip connections to alleviate the gradient flow of the network.

3 Method

In this work we start by putting our emphasis on the well known Rudin-Osher Fatemi (ROF) [39] model for image restoration. To write the ROF model in its discretized form we partition a subset of $\mathbb{R}^d$ into equally sized and piecewise constant elements $\mathbf{i} \in I$, with $I = \{1, \dots, N^1\} \times \dots \times \{1, \dots, N^d\}, N^i \in \mathbb{N},\ \forall i \leq d$. ROF can then be written as the following non-smooth convex optimization problem:

$$\min_{\mathbf{x}} \lambda \|\mathrm{Q}\mathbf{x}\|_{p,1} + \frac{1}{2}\|\mathbf{x} - \mathbf{f}\|_2^2, \quad \text{with } \|\mathrm{Q}\mathbf{x}\|_{p,1} = \sum_{\mathbf{i}} |(\mathrm{Q}\mathbf{x})_{\mathbf{i},\cdot}|_p. \tag{1}$$

where $\mathbf{x}, \mathbf{f} \in \mathbb{R}^{N^1 \times \dots \times N^d}$, with $\mathbf{f}$ being the input and $\mathbf{x}$ the regularized output. The term $\|\mathrm{Q}\mathbf{x}\|_{p,1}$ refers to the discrete Total Variation. For $p=1$ we obtain the anisotropic (ℓ_1) and for $p=2$ the isotropic (ℓ_2) Total Variation. In this work, we will mainly stick to the anisotropic case $p=1$. The free parameter $\lambda \geq 0$ controls the amount of smoothness of the solution. The linear operator $\mathrm{Q} : \mathbb{R}^{N^1 \times \dots \times N^d} \to \mathbb{R}^{N^1 \times \dots \times N^d \times d}$ approximates the spatial gradient via finite differences and is defined by $(\mathrm{Q}\mathbf{x})_{\mathbf{i},k} = \mathbf{x}_{\mathbf{i}+e_k} - \mathbf{x}_{\mathbf{i}}$, if $\mathbf{i}, \mathbf{i} + e_k \in I$ and 0 else. Here, e_k denotes the unit vector in direction k.

There are several reasons why we consider the ROF model in this paper: It is simple enough to render results interpretable for the human observer, but nevertheless has immediate applications, *e.g.* in image denoising [39] or segmentation [10].

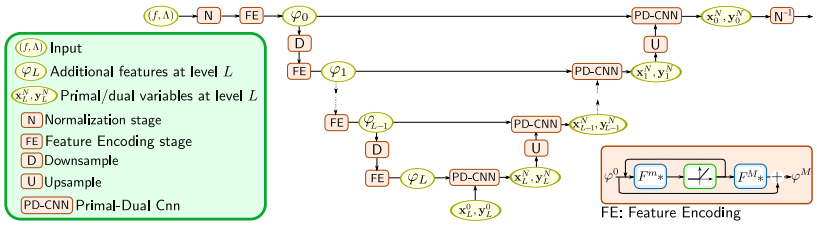

Fig. 1. Our network architecture is known as U-shaped network with lateral links. In our case to enable a multiscale framework. The network is divided in 2 stages: Feature encoding and optimization. Both are applied at each resolution, with variables initialized from the previous level.

Moreover, it defines the archtype model for more advanced TV based models that proved to be effective for other low-level vision problems such as stereo, depth-map–fusion [49] and optical flow [3]. Finally, its strong connections to max-flow/minimum-cut problems [26] make it further attractive. To solve (1), common options are first-order (FO) algorithms, *e.g.* [33] and parametric max-flow [19]. In this paper, we posit that FO-methods are a suitable and convenient representation, to be transformed into a neural network and generalized via learning. To that end, we first write (1) in its primal-dual form [8], *i.e.* we introduce (dual) variables y and replace the ℓ_1-norm by its conjugate:

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \langle \mathrm{Q}\mathbf{x}, \mathbf{y} \rangle + \frac{1}{2}\|\mathbf{x} - \mathbf{f}\|_2^2, \quad \text{s.t. } \|\mathbf{y}\|_\infty \leq \lambda. \tag{2}$$

This saddle-point problem can be solved by alternating gradient ascent and descent steps on the primal and dual variables. In particular we select the algorithm proposed in [8], which for problem (2) takes the following form:

$$\begin{aligned} \mathbf{y}^{n+1} &= \mathrm{proj}_\lambda(\mathbf{y}^n + \sigma \mathrm{Q}(2\mathbf{x}^n - \mathbf{x}^{n-1})), \\ \mathbf{x}^{n+1} &= (1+\tau)^{-1}(\mathbf{x}^n - \tau(\mathrm{Q}^*\mathbf{y}^{n+1} - \mathbf{f})). \end{aligned} \tag{3}$$

Here, $\mathrm{proj}_\lambda(\cdot) := \max(-\lambda, \min(\cdot, \lambda))$ denotes the projection into the interval $[-\lambda, \lambda]$, which is applied per component in the anisotropic case. The step sizes τ and σ are chosen such that $(\tau\sigma) \leq ||\mathrm{Q}||_2^{-2}$ (*cf.* [8] for a more detailed explanation). We picked this particular primal-dual algorithm since it provides a rather flexible and general optimization framework with many applications. Lately, this specific update scheme was extended to the non-convex case [44]. This finding further confirms our choice.

3.1 Network structure

Our network architecture is depicted in Fig. 1. Its input consists of a discretized version of $\mathbf{f}$ (*e.g.* an image) and in addition a scalar field Λ (8), encoding spatial smoothness weights between adjacent elements of $\mathbf{f}$. Thus, we always model the more general, weighted ROF problem (Sec. 3.2). After normalization (*cf.* supplementary), we identify two main stages: an encoding stage, to generate the feature representation φ and the optimization stage realizing our primal-dual network (PD-CNN). Both stages are embedded into a multiscale framework. The principal idea is to work on a low resolution

version of the problem first and reuse the output at the next higher level as initialization. Between levels we use simple fixed up- and downsample operators, mean-pooling (D in Fig. 1) and nearest-neighbor interpolation (U in Fig. 1), and let the encoding and PD-CNN modules compensate potential artifacts. We initialize our variables at the coarsest resolution ($\mathbf{x}_L^0, \mathbf{y}_L^0$ in Fig. 1) with a downsampled version of $\mathbf{f}$ and set $\mathbf{y}$ to zero.

Feature encoding. In the encoding stage, operating from fine to coarse, a feature representation, φ, of the input is created that is used later, at the respective resolution in the PD-CNN. At each resolution level we apply a small sub-network on the current feature representation to perform this task. In practice, the sub-networks are so called residual functions [22] of depth M; *i.e.* M pairs of conv-relu before a final convolution takes place. The result is then added back to the input feature vector (Fig. 1, lower right). These, so called, identity skip connections should mitigate effects of vanishing gradients and allow to train very deep architectures [22]. The resulting features φ, consist of multiple channels. Before they are used in our PD-CNN, a downsampled version of the input $\mathbf{f}$ is concatenated as an additional channel.

Optimization Network. The mentioned properties, render the primal-dual framework a good candidate to serve as template for the optimization stage of our network. Eqs. (4) – (7) compare our proposed PD-CNN and the original primal-dual method of (3).

Primal-Dual algorithm	Primal-Dual-CNN
$\mathbf{y}^{n+1} = \text{proj}_\lambda(\mathbf{y}^n + \sigma \mathrm{Q}(2\mathbf{x}^n - \mathbf{x}^{n-1}))$ (4)	$\mathbf{y}^{n+1} = \text{proj}_{\rho(T^n * (\mathbf{y}^n; \mathbf{x}^n; \varphi))}(\mathbf{y}^n + s_\sigma^n D^n * (2\mathbf{x}^n - \mathbf{x}^{n-1}))$ (5)
$\mathbf{x}^{n+1} = (1+\tau)^{-1}(\mathbf{x}^n - \tau(\mathrm{Q}^* \mathbf{y}^{n+1} - \mathbf{f}))$ (6)	$\mathbf{x}^{n+1} = (1 + s_\tau^n)^{-1}(\mathbf{x}^n - s_\tau^n((D^n)^* * \mathbf{y}^{n+1} - C^n * \varphi))$ (7)

First notice that all utilized operations posses an easy to compute subgradient, which directly enables backpropagation in a CNN. In our PD-CNN all linear operators of (4) and (6) are replaced by iteration dependent convolutions, expressed by the superscript n. For instance, the learned operator D is equivalent to a linear map $\hat{D} : \mathbb{R}^{N^1 \times \ldots \times N^d} \rightarrow \mathbb{R}^{N^1 \times \ldots \times N^d \times d}$, to replace the forward differences encoded by Q. T and C are defined in a similar manner. In the proximal step for the primal variable $\mathbf{x}$ (7), our generated feature maps replace our input $\mathbf{f}$, for which we learn a convolution operator 'C' per iteration. The feature maps also occur in our spatially adaptive projection operator. Here, the thresholds are not constant any more, but a function of the current primal and dual variables and feature encoded by 'T'. Further, we utilize a 'softplus' $\rho(\cdot) := \log(1 + \exp(\cdot))$ and employ the outcome as a direct replacement for λ in (5). Thus, the projection operations are based on local information of all variables and features. We also use different step sizes (s_σ^n, s_τ^n) per iteration, to potentially learn some form of acceleration [1,8]. The last form of generalization is achieved by allowing the network to utilize more channels during the computation than the optimization problem would suggest. In this work we donate multiple dimensions to $\mathbf{y}$ and the feature vector φ, but restrict our primal variable $\mathbf{x}$ to a single dimension. *I.e.* $\mathbf{y}, \varphi \in \mathbb{R}^c$ for some $c \geq d$. The linear operators are adjusted accordingly. Note that the algorithm described by (7) and (5) can still be interpreted as minimizing a (in every step) different energy. Because the convolutions 'D' are still adjoint linear operators, the saddle-point structure remains intact. Note that the residual structure is already inherent to FO-methods and, thus, also to our PD-CNN.

Parameter sharing. The primal dual algorithm (3) has no restriction on the size of the input data $\mathbf{f}$. We want our network to have similar characteristics. To apply the network on arbitrarily sized input data, we can rely on the fact that our network is 'fully convolutional' (*e.g.* [29]). Yet, while we have no spatial restriction on the input, the number of iterations in each PD-CNN stage and the depth of our multiscale framework is limited. RNNs [20] address this problem by sharing the parameters of the convolutions in each step. In contrast, the design of our coarse-to-fine network suggests to share parameters across the different levels, not across iterations within one resolution level. This allows to adjust the number of levels to the input size after training. Here, the basic assumption is that similar problems have to be solved at each resolution and not at each step. Compared to weight-sharing across iterations, this treatment provides more freedom to the network, *e.g.*, the step size can still be variable per iteration. It is not required to share the filter maps at the finest resolution to create this behavior. In fact, we experience better results if we exploit this freedom and only share weights across all lower levels.

Training. Recall that we treat the smoothness parameter as additional input to the network (Fig. 1) and specifically design the training data to contain a large range of values. To guide the training process further, we combine the same $\mathbf{f}$ with multiple values for λ in the training set. This enables the network to apply the correct regularization for any value in the learned range and even to extrapolate the amount of smoothing up to a certain degree (about 1.5 times the maximal trained value). To further guide the training, we optionally specify a loss at each resolution level, penalizing the difference to the downsampled ground-truth. This treatment can stabilize the training in its early stages. At least, this finding confirms our assumption that per resolution level similar problems are solved and with that our idea of sharing the parameters across resolution levels.

3.2 Extension to related problems (from the TV-family)

Our network was designed to be applicable to different vision problems. Here we provide some examples. We start with a more general version of the ROF model. An extension for TV-ℓ_1 can be found in the supplementary.

Weighted ROF. The problem in (1) can be extended, by allowing the smoothness weight λ to be spatially different. Analogous to (2) we directly describe the primal-dual form of the weighted ROF problem (WROF):

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \langle \mathrm{Q}\mathbf{x}, \mathbf{y} \rangle + \frac{1}{2}\|\mathbf{x}-\mathbf{f}\|_2^2, \text{ s.t. } |\mathbf{y}_{\mathbf{i},k}| \leq \Lambda(\mathbf{i},k). \tag{8}$$

Thus, we can encode the weights Λ with d dimensions. The problem is now exactly in a form to be used by our proposed network from Sec. 3.1.

MRF with TV regularization. Multi-label problems are in general NP-hard and cannot be minimized globally. An exception occurs in the case of a linearly ordered label set. If the pairwise interactions form a convex function on this set, an optimal solution can be obtained by finding a minimum cut in a higher-dimensional graph [24]. The problem can alternatively be formulated in the form of (8) [6,35]. Summarized in (9) the graph construction works by extending the domain of the problem by one dimension, the

Fig. 2. Training our stereo network. *Left*: Example scene of out training set. [18] *Right*: Truncated 3D distance function. The disparity is recovered as the 0-levelset of the distance function.

'label-direction'. Edge-costs along this dimension are associated with the data costs for a particular label $l \in L$ and spatial position $\mathbf{i}$. All spatial weights are copied along the label-axis. The data cost at the vertices is zero, except for vertices above (below) the first (last) set of edges in label direction. Here, one uses a cost of $-\gamma$ (γ).

$$\begin{aligned} \Lambda((\mathbf{i},l),k) &:= \theta_{\mathbf{i},k} \quad \text{for } k \leq d \quad \text{and} \quad \Lambda((\mathbf{i},l),d+1) := \vartheta_{\mathbf{i},l} \quad \text{else} \\ \mathbf{f}_{(\mathbf{i},0)} &:= -\gamma, \quad \mathbf{f}_{(\mathbf{i},|L|)} := \gamma \quad \text{and} \quad \mathbf{f}_{(\mathbf{i},l)} := 0 \quad \text{for } 0 < l < |L|. \end{aligned} \tag{9}$$

Here, we use θ and ϑ to denote edge and data costs of the original multi-label problem. The solution of the original problem is recovered as the 0-levelset of problem (9) [6].

From energy optimization to general models. We have motivated our network design to mimic and generalize a FO-method from convex analysis. Apart from allowing for a detailed analysis and fast approximate solutions, the construction admits to utilize the generalized algorithmic structures for related problems. We argue that a structure that is used to solve related (convex) problems could be a good starting point to generalize to more complex tasks via learning. As a demonstration we train networks for Gaussian denoising and stereo estimation. For the denoising task we use our 2D WROF network, but train it directly with easily available ground truth data. For the stereo problem we use the 3D WROF model from (9). To go beyond TV regularization, we train the model on a truncated 3D distance function, generated by applying a fast marching method [42] on the provided ground-truth (Fig. 2). Note that employing a 3D graph significantly increases the number of variables in the network. Here, sharing weights across levels proves useful. Because of the weight sharing, the network can be trained on data with a small label-space, but still be applied to much larger label-spaces; simply by extending the hierarchy with additional resolution levels, until the whole label-space is covered.

4 Evaluation

Our evaluation is split into two distinct parts. First, we analyze the capability of our network to approximate solutions for optimization problems from the TV-family and investigate different levels of generalization of the algorithm. Second, we test our network on the tasks of denoising and stereo estimation as explained in the previous section.

Data Set and Training Procedure. We train our networks with the Theano package [43] on a machine equipped with a NVIDIA Titan X GPU with 12 GB of RAM. Our standard training sets consist of 1000 images from the KITTI dataset [18] randomly cropped to the size of 256×256 pixels. For our test dataset we use 300 images from the Berkeley segmentation dataset (BSD68) [30]. All images are in the interval $[0, 1]$.

Table 1. Step-by-step generalization of the PD-CNN algorithm for the ROF model.

Step	Algorithm (6 level of 6 iterations)	channels	PSNR
(i)	$\mathbf{y}^{n+1} = \text{proj}_\lambda(\mathbf{y}^n + s_\sigma \mathrm{Q}(2\mathbf{x}^n - \mathbf{x}^{n-1}))$ $\mathbf{x}^{n+1} = (1 + s_\tau)^{-1}(\mathbf{x}^n - s_\tau(\mathrm{Q}^*\mathbf{y}^{n+1} - \mathbf{f}))$	2	18.57
(ii)	$\mathbf{y}^{n+1} = \text{proj}_\lambda(\mathbf{y}^n + s_\sigma^n \mathrm{Q}(2\mathbf{x}^n - \mathbf{x}^{n-1}))$ $\mathbf{x}^{n+1} = (1 + s_\tau^n)^{-1}(\mathbf{x}^n - s_\tau^n(\mathrm{Q}^*\mathbf{y}^{n+1} - \mathbf{f}))$	2	30.92
(iii)	$\mathbf{y}^{n+1} = \text{proj}_\lambda(\mathbf{y}^n + D^n * (2\mathbf{x}^n - \mathbf{x}^{n-1}))$ $\mathbf{x}^{n+1} = (1 + s_\tau^n)^{-1}(\mathbf{x}^n - s_\tau^n((D^n)^* * \mathbf{y}^{n+1} - \mathbf{f}))$	2	32.23
(iv)	$\mathbf{y}^{n+1} = \text{proj}_{\rho(T^n * (\mathbf{y}^n, \mathbf{x}^n; \varphi))}(\mathbf{y}^n + s_\sigma^n D^n * (2\mathbf{x}^n - \mathbf{x}^{n-1}))$ $\mathbf{x}^{n+1} = (1 + s_\tau^n)^{-1}(\mathbf{x}^n - s_\tau^n((D^n)^* * \mathbf{y}^{n+1} - \mathbf{C^n} * \varphi))$	2	35.34
(v)	$\mathbf{y}^{n+1} = \text{proj}_{\rho(T^n * (\mathbf{y}^n, \mathbf{x}^n; \varphi))}(\mathbf{y}^n + s_\sigma^n D^n * (2\mathbf{x}^n - \mathbf{x}^{n-1}))$ $\mathbf{x}^{n+1} = (1 + s_\tau^n)^{-1}(\mathbf{x}^n - s_\tau^n((D^n)^* * \mathbf{y}^{n+1} - \mathbf{C^n} * \varphi))$	8	45.38

Table 2. Evaluation of models from the TV family including sharing parameters across levels.

	Sharing: ✗			Sharing: ✓		
	ROF	WROF	TV-ℓ_1	ROF	WROF	TV-ℓ_1
PSNR	53.75	49.83	46.45	52.78	49.21	46.12
MADP	0.132	0.211	0.277	0.145	0.228	0.285

To train our models we use 6 different smoothness weights per image, ranging from 0.1 to 3, *cf*. Fig. 3. Thus, our full data sets contains 7800 images. The edges weights for the WROF model are extracted via filtering from random images. We randomly flip and rotate the images for data augmentation. We measure the peak signal-to-noise ratio (PSNR) and the mean absolute difference in percent (MADP) to analyze our network. Our networks are fully convolutional and use a filter size of 3×3 everywhere. As training objective we use an ℓ_1-loss. For more details please refer to the supplementary material.

Generalization of Primal-Dual. We start our evaluation by analyzing the effect of different forms of generalization of our PD-CNN and use the ROF model for that purpose. In this experiment we employ 6 resolution levels and 6 iterations of PD-CNN per level. At first, we restrict ourselves to the minimal 2 channels for $\mathbf{y}$ and φ. Tab. 1 summarizes our procedure, we mark relevant changes for each step in blue. In the first steps *(i,ii)* we use finite differences to implement gradient and divergence and only learn the step-sizes. Interestingly, compared to sharing all values *(i)*, *i.e*. learning just 2 scalars, the PSNR increases by 66%, if we allow the step-sizes to vary per iteration *(ii)*. Next, we replace the fixed linear operators with convolutions but keep the saddle-point structure per iteration *(iii)*. After we utilize our learned features for the primal updates and generalize our projection operator *(iv)* we again observe a significant improvement, which is even more visible after employing additional channels for φ and $\mathbf{y}$ *(v)*.

Performance Evaluation. The results for all TV problems ROF, WROF and TV-$\ell_1$are summarized in Tab. 2. We use 16 channels for $\mathbf{y}$ and φ, 6 levels, 6 iterations of PD-CNN and 2 in the encoding stage per level. Note that PSNR and the number of iter-

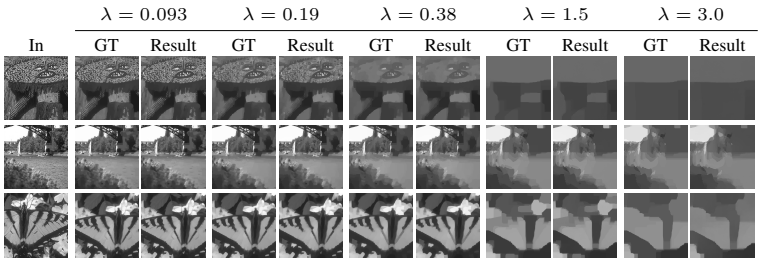

Fig. 3. Test set examples for varying smoothness costs (λ) for our ROF(*top*) / WROF(*middle*) / TV-$\ell_1$models(*bottom*). Displayed are input (*In*), ground truth (*GT*) and prediction (*Result*).

ations of our prototype algorithm are related via the primal dual gap [8]. *E.g.*, to achieve a PSNR of 50 requires about 530 iterations. TV-$\ell_1$appears to be the hardest problem of the three, followed by WROF. Sharing parameters across levels does barely harm our performance. Fig. 3 displays examples with the worst MADP score in the test set. An inspection reveals that some edges are not as crisp as in the ground truth, especially, if long range interactions have to be modeled. This finding is confirmed in Tab. 3. As expected, ROF becomes the harder the stronger we regularize. Note, that our multiscale framework proves effective in reducing the error for stronger regularization (*cf*. supplementary). For WROF we exemplarily trained a deeper model with 10 iterations per level (Tab. 4). Using a loss function at every resolution level led to consistently better results. With only a single loss the training might get stuck in a bad local minimum.

4.1 From energy optimization to algorithm learning

For our denoising example we train our ROF network on gray images from the BSD68 dataset. We deviate from our standard settings and use only 2 levels, but 48 channels and 10 iterations. Following the standard protocol, the dataset is split into 432 training image from which we randomly crop one 256×256 block and add random Gaussian noise ranging from 0.02 to 0.21. We use the other 68 images for evaluation. Two examples are shown in Fig. 4 for a noise level of 20%. Tab. 5 lists the PSNR and the structured similarity index (SSIM) for our and competing work. Our results are very close to the state-of-the-art [50], while we apply the same network to all noise levels.

To perform stereo estimation we employ the multi-label MRF from Sec. 3.2. We train our network on 64×64 snippets from the KITTI dataset [32] following the described lifting approach, *i.e.* we train the 3D WROF model (9). We use normalized cross correlation (NCC) as data cost (ϑ) and image gradients to define the spatial edge

Table 3. Test set performance, for various smoothness costs.

	ROF-Model					
λ	0.093	0.19	0.38	0.75	1.5	3.0
PSNR	53.10	53.21	52.71	52.00	51.22	49.89

Table 4. Training a deep network with or without an explicit level-wise loss.

Model	Multi-Loss	PSNR	MADP
WROF	✓	51.86	0.168
	✗	41.72	0.586

Table 5. Performance of selected Gaussian denoising algorithms on the BSD68 dataset.

Noise level	BM3D [12]		EPLL [52]		TNRD [11]		DnCNN-S [50]		Ours	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$\sigma = 15$	31.08	0.8722	31.21	-	31.42	0.8826	**31.73**	-	31.66	**0.8964**
$\sigma = 25$	28.57	0.8017	28.68	-	28.92	0.8157	**29.23**	-	29.21	**0.8359**
$\sigma = 50$	25.62	0.6869	25.67	-	25.97	0.7029	26.23	-	**26.28**	**0.7275**

Fig. 4. *Top:* Denoising at 20% noise level. From left to right: Input; result; original. *Bottom:* Stereo examples from [32]. From left to right [9]; ours, trained via [9]; ours, trained via the truncated 3D distance function. Disparity color code: White - near, blue - far.

Table 6. Results of our stereo network; trained on [18], evaluated on [32]. Given are the average endpoint error (AEP) and the percentage of pixel with > N pixel deviation from the ground truth.

All				Non-Occluded			
%> 3pix.	%> 4pix.	%> 5pix.	AEP	%> 3pix.	%> 4pix.	%> 5pix.	AEP
4.9	4.4	4.0	1.3	3.6	3.0	2.7	0.9

weights (θ). At first, we generate our training data with the model described in [9] (Fig. 4, *left*). Optimization in this 3D graph with [9] is several orders slower than in our network (Fig. 4 middle). To allow for larger disparities than what we trained on, we employ our weight sharing technique. Recall, that our trained network allows to use different data costs; the weights, and with that the data costs are input to the network. The model trained on the 3D distance function is shown in Fig. 4, *right*. This model (*cf*. Tab. 6) behaves differently, surfaces appear to be more planar and boundaries are tighter, *e.g.* around the cyclist. However, in these challenging conditions, training a data cost network jointly with our inference network might be an interesting path to investigate.

5 Conclusion

We have presented our idea of a primal-dual network as a deep CNN that leverages the algorithmic structure provided by energy optimization techniques into learning a generalized optimization algorithm. To achieve this goal, we integrated a modern and general FO-method into a multiscale framework, in order to overcome a limited receptive field and number of iterations. We further introduced weight sharing across resolution levels to apply our framework on arbitrarily sized input. We have already shown several possible applications that are worthwhile to be investigated further. We also plan to extend the scheme to more advanced Total Variation based models such as optical flow.

Acknowledgements: This work was supported from the ERC grant HOMOVIS No. 640156.

References

1. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Img. Sci. (2009)
2. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial (2nd Ed.). Society for Industrial and Applied Mathematics (2000)
3. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: ECCV (2004)
4. Bruhn, A., Weickert, J., Kohlberger, T., Schnörr, C.: A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. IJCV (2006)
5. Chambolle, A.: Total variation minimization and a class of binary mrf models. EMMCVPR (2005)
6. Chambolle, A., Darbon, J.: A parametric maximum flow approach for discrete total variation regularization. In: Image Processing and Analysis with Graphs, chap. 4 (2012)
7. Chambolle, A., Levine, S.E., Lucier, B.J.: An upwind finite-difference method for total variation-based image smoothing. SIAM J. Imaging Sciences (2011)
8. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. JMIV 40(1) (2011)
9. Chambolle, A., Pock, T.: A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions. SMAI-JCM (2015)
10. Chan, T.F., Esedoglu, S., Nikolova, M.: Algorithms for finding global minimizers of image segmentation and denoising models. SIAM Journal of Applied Mathematics (2006)
11. Chen, Y., Yu, W., Pock, T.: On learning optimized reaction diffusion processes for effective image restoration. In: CVPR (June 2015)
12. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.O.: Image restoration by sparse 3d transform-domain collaborative filtering. In: Transactions on Image Processing (2008)
13. Daubechies, I., Defrise, M., De Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on P. and Appl. Mathematics (2004)
14. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV (2014)
15. Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., v.d. Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: ICCV (2015)
16. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. IJCV (2006)
17. Felzenszwalb, P.F., Zabih, R.: Dynamic programming and graph algorithms in computer vision. PAMI (2011)
18. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? In: CVPR (2012)
19. Goldfarb, D., Yin, W.: Parametric maximum flow algorithms for fast total variation minimization. SIAM J. SCI-COMP (2009)
20. Goller, C., Küchler, A.: Learning task-dependent distributed representations by backpropagation through structure. IEEE Transactions on Neural Networks (1996)
21. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: ICML (2010)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CVPR (2016)
23. Hirschmüller, H.: Stereo processing by semiglobal matching and mutual information. PAMI 30(2) (2008)
24. Ishikawa, H.: Exact optimization for markov random fields with convex priors. PAMI (2003)
25. Kolmogorov, V., Rother, C.: Minimizing nonsubmodular functions with graph cuts-a review. PAMI (2007)

26. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? PAMI (2004)
27. Li, S.Z.: Markov Random Field Modeling in Image Analysis. Springer Publishing (2009)
28. Lin, G., Shen, C., Reid, I.D., van den Hengel, A.: Efficient piecewise training of deep structured models for semantic segmentation. CoRR (2015)
29. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
30. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms. In: ICCV (2001)
31. Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. CVPR (2016)
32. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: CVPR (2015)
33. Pock, T., Unger, M., Cremers, D., Bischof, H.: Fast and exact solution of Total Variation models on the GPU. In: CVPR - Workshop (2008)
34. Pock, T., Chambolle, A.: Diagonal preconditioning for first order primal-dual algorithms in convex optimization. ICCV (2011)
35. Pock, T., Schoenemann, T., Graber, G., Bischof, H., Cremers, D.: A convex formulation of continuous multi-label problems. ECCV (2008)
36. Rasmus, A., Valpola, H., Honkala, M., Berglund, M., Raiko, T.: Semi-supervised learning with ladder networks. In: NIPS (2015)
37. Riegler, G., Ferstl, D., Rüther, M., Bischof, H.: A deep primal-dual network for guided depth super-resolution. CoRR (2016)
38. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": Interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. (2004)
39. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D (1992)
40. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015)
41. Schwing, A.G., Urtasun, R.: Fully connected deep structured networks. CoRR (2015)
42. Sethian, J.A.: Level set methods and fast marching methods. Cambridge monographs on applied and computational mathematics, Cambridge University Press (1999)
43. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. CoRR (2016)
44. Valkonen, T.: A primal-dual hybrid gradient method for nonlinear operators with applications to mri. Inverse Problems (2014)
45. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR (2010)
46. Wang, Z., Ling, Q., Huang, T.: Learning deep ℓ_0 encoders. AAAI (2016)
47. Wang, Z., Liu, D., Yang, J., Han, W., Huang, T.: Deep networks for image super-resolution with sparse prior. In: ICCV. pp. 370–378 (2015)
48. Wu, F.Y.: The potts model. Rev. Mod. Phys. (1982)
49. Zach, C., Pock, T., Bischof, H.: A globally optimal algorithm for robust tv-l1 range image integration. In: ICCV (2007)
50. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. CoRR (2016)
51. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.: Conditional random fields as recurrent neural networks. In: ICCV (2015)
52. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: ICCV (2011)

Supplementary – A Primal Dual Network for Low-Level Vision Problems

Christoph Vogel[1] Thomas Pock[1,2]

[1] Graz University of Technology [2] AIT Austrian Institute of Technology

Our supplementary material starts with a deeper analysis of the behavior of our network. At first we compare the impact of the number of iterations and resolution levels. At second we investigate to which extent our models can be trained by working with the energy alone, which might lead to semi-supervised training procedures for adequate problems. Afterwards, we consider the normalization stage of our network. Finally we give more details to our denoising and stereo scenarios and show different examples. We also extend our model to the TV-$\ell_1$case and give more details on our stereo model. We start by providing more details about our training procedure.

1 Training Procedure

Our filters are initialized to be zero-mean and, if possible, orthogonal to each other. We also make sure that initially the sum of the standard deviation along the channels remains constant through filtering. Our networks are optimized via ADAM [8]. We refrain form using Batch Normalization (BN) [7] to allow sharing parameters of the convolutions. Otherwise, we could not extend our network with additional levels after training, unless the parameters of the BN are also shared. In our experience this leads to sharper edges and superior results compared to the usual squared ℓ_2-loss. Please refer to [11] for a deeper discussion of this phenomenon.

2 Iterations versus resolution levels

We continue with a more detailed analysis on how the number of resolution levels and iterations per level influence the performance of the network. Recall that we usually employ 6 resolution levels and 6 iterations of our PD-CNN per level. At first, Tab. 1 shows that operating on the highest resolution level alone does not lead to good results. Although the results can be significantly improved with a second stage, we still value

Table 1. Different number of resolution levels and PD-CNN iterations per level.

	ROF-Model						
	Resolution levels / iterations per level						
	1/6	2/6	4/6	6/1	6/2	6/4	6/6
PSNR	37.58	42.83	50.14	43.00	46.13	49.71	53.75
MADP	3.76	1.98	0.58	1.28	0.87	0.55	0.34

the reached quality as insufficient. Only after we employ four resolution levels, results start to look promising. In our experience, this is about the level at which the resulting receptive field is sufficient to handle the cases of stronger regularization we have in our dataset. On the other hand, utilizing a larger number of levels helps to reconstruct the global structure, but with the proper number of iterations, results can be significantly improved. Indeed, the more iterations we apply (*cf*. Tab. 4 of the paper), the better the numbers become. However, often additional means (*e.g*. our multiloss) are required to successfully train these deeper networks. To summarize, in case of stronger regularization, which is needed, for instance, for all our models that perform some form of labeling task, our multiscale framework appears mandatory to achieve good results.

To gain some intuition how the networks can utilize these additional resolution levels we provide Fig. 1. Using our multiloss for the ROF model[1], we plot the output of the network over six different levels and for six different smoothness costs. The figure confirms our prior findings: For stronger regularization, intermediate results already look similar to the final output, even at very coarse resolutions. In case of weaker regularization, details are re-inserted at the higher levels of the network, and the global structure is shared at coarser resolutions for different weaker smoothness costs.

3 Energy based training

Recall that we train our models directly from input/ground truth pairs, which can be easily computed for our convex models. Otherwise, one has to rely on manually labeled or artificially generated ground truth. Both procedures either require significant effort or may lead to unrealistic data. These observations motivated us to test an alternative training procedure for our models. We utilize the available energy formulation as loss for our network. The result is depicted in Tab. 2 for the ROF and WROF models. At first we notice that both losses perform best on their respective training objective. For the energy loss we observe further that the final energies are indeed quite close to the true minimum, however, the PSNR is significantly worse, compared to training from examples. Here, the resulting solutions appear visually less sharp, especially for larger smoothness weights. Splitting the energy into a data and smoothness part reveals that, on average, the cost of the smoothness part of our energy loss version is even lower than that of the ground truth energy. We conjecture that a hybrid approach might mitigate this problem. When examples are expensive to generate, a strategy similar to semi-supervised learning [6] might be of benefit for training such a model. One application of this procedure could be models, for which an energy formulation is known, but leads to an NP-hard optimization problem, *e.g*. truncated Total Generalized Variation [2].

4 Normalization.

Almost all training procedures for neural networks apply some form of normalization on the training (and testing) data. This is also true for our method. In our case, we design

[1] In our experience, the results with and without multiloss differ only very little for all our models if we apply less or equal to five resolution levels per PD-CNN.

Table 2. Different training objectives: ℓ_1-loss vs. energy as training objective for the ROF/WROF problem.

	PSNR		Energy %	
Loss	ROF	WROF	ROF	WROF
Energy	46.36	42.92	100.7	104.9
ℓ_1	53.75	49.83	102.1	111.6
GT	∞	∞	100.0	100.0

our normalization stage such that we can exploit invariance properties of the modeled problems.

Normalization is applied on the input data of the network $(\mathbf{f}, \Lambda)$. To that end, we subtract the mean μ from $\mathbf{f}$ to receive $\bar{\mathbf{f}}$ and divide $(\bar{\mathbf{f}}, \Lambda)$ by its standard deviation s (Fig. 1 of the paper). The outcome is passed to the encoding stage. Later, the output of the network is multiplied by s and the mean μ is added back. We also use $\hat{\mathbf{f}} := \bar{\mathbf{f}}/s$ to provide the initial value for $\mathbf{x}$. The reason for this normalization is twofold. Firstly, neural network training is known to be more stable if the input remains in a certain input range and scale. Secondly, this transformation models exactly the invariance of the ROF problem. A quick look at the optimality conditions of Eq. (1) of the paper indicates that

$$\arg\min \text{ROF}(\mathbf{f}/s + c, \lambda/s) = 1/s \cdot \arg\min \text{ROF}(\mathbf{f}, \lambda) + c. \tag{1}$$

The normalization reduces the dimension of the problem and simplifies the task for the network, which otherwise would need to learn this property. In addition to our usual normalization we initially scale only our input $\mathbf{f}$, such that its standard deviation is equal to one and undo the operation on the output.

5 Extension for TV-ℓ_1.

At first sight TV-$\ell_1$is very similar to the ROF model, the only difference is that the square on the data-term vanishes.

$$\min_{\mathbf{x}} \lambda \|\mathrm{Q}\mathbf{x}\|_{p,1} + \|\mathbf{x} - \mathbf{f}\|_2. \tag{2}$$

Analogously to the ROF problem, we can the write the primal dual form of (2) as:

$$\min_{\mathbf{x}} \max_{\mathbf{y}} \langle \mathrm{Q}\mathbf{x}, \mathbf{y} \rangle + \|\mathbf{x} - \mathbf{f}\|_2 \;\; \text{s.t.} \; \|\mathbf{y}\|_\infty \leq \lambda. \tag{3}$$

This little change, however, renders the model contrast invariant. In analogy to the ROF model that means:

$$\arg\min \text{TV-}\ell_1(\mathbf{f}/s + c, \lambda) = 1/s \cdot \arg\min \text{TV-}\ell_1(\mathbf{f}, \lambda) + c. \tag{4}$$

We can introduce this invariance property to the network with a subtle change in the normalization stage (*cf*. Sec. 4).

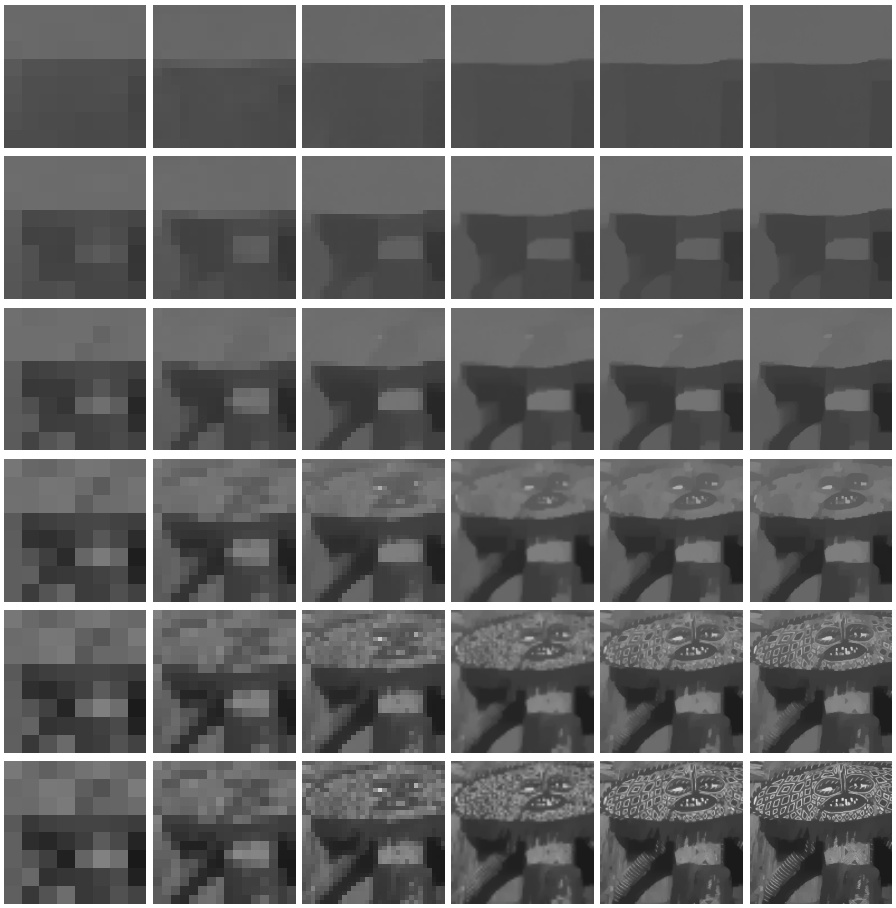

Fig. 1. Simple ROF model: Output of our network over 6 different resolution levels (*columns*) for 6 different smoothness costs (*rows*).

The algorithm for TV-$\ell_1$is summarized in Tab. 3. At first we define the shrinkage operation via Moreau's identity (*e.g.* [14]):

$$\text{shrink}(\mathbf{x}, \tau, \mathbf{f}) := \mathbf{x} - \text{proj}_{\pm\tau}(\mathbf{x} - \mathbf{f}). \tag{5}$$

Now, we can use (5) as a direct replacement for the solution of the proximal map of the square norm data term of the ROF problem – in the primal-dual algorithm as well as our PD-CNN. For more details on proximal operators please refer to *e.g.* [13].

6 More details on training without an energy model

At first notice that our selected applications are merely a subset of possible use cases for our network. Those were rather selected to give an idea of what our network is capable

Table 3. Generalized Primal-Dual algorithm (*PD-CNN*) for the TV-$\ell_1$model, implemented as a CNN. *Left:* Primal-Dual algorithm [4]. *Right:* Our *PD-CNN*.

Primal-Dual algorithm		Primal-Dual-CNN	
$\mathbf{y}^{n+1}=\text{proj}_{\pm\lambda}(\mathbf{y}^n+\sigma\text{Q}(2\mathbf{x}^n-\mathbf{x}^{n-1}))$	(6)	$\mathbf{y}^{n+1}=\text{proj}_{\pm\rho(T^n*(\mathbf{y}^n;\mathbf{x}^n;\varphi))}(\mathbf{y}^n+s_\sigma^n D^n*(2\mathbf{x}^n-\mathbf{x}^{n-1}))$	(7)
$\mathbf{x}^{n+1}=\text{shrink}(\mathbf{x}^n-\tau\text{Q}^*\mathbf{y}^{n+1},\tau,\mathbf{f})$	(8)	$\mathbf{x}^{n+1}=\text{shrink}(\mathbf{x}^n-s_\tau^n(D^n)^**\mathbf{y}^{n+1},s_\tau^n,C^n*\varphi)$	(9)

of. We believe that our work can also be useful for problems like semantic segmentation [10] or image inpainting [1] and many more.

Graph-Cut. The graph-cut problem [9] can be written as:

$$\min_{\mathbf{x}_\mathbf{i}\in\{0,1\}}\sum_{\mathbf{i},k}\theta_{\mathbf{i},k}|(\text{Q}\mathbf{x})_{\mathbf{i},k}|_1+\sum_{\mathbf{i}}\mathbf{x}_\mathbf{i}(\vartheta_{\mathbf{i},1}-\vartheta_{\mathbf{i},0}). \tag{10}$$

Here, $\vartheta_{\mathbf{i},j}$ denotes the data cost for assigning label j to position $\mathbf{i}$ and $\theta_{\mathbf{i},k}$ the weight of the edges at $\mathbf{i}$ in direction k. A solution to the binary graph-cut problem can be found by solving a problem of the form of Eq. (8) of the paper and thresholding the result (*cf*. [3]). Consequently, our network is also capable to solve (10). The transformation is fairly simple [3]: We define $\mathbf{f}:=(\vartheta_{\mathbf{i},1}-\vartheta_{\mathbf{i},0})$ and $\Lambda(\mathbf{i},k):=\theta_{\mathbf{i},k}$ and threshold the result of the network at 0. If desired, the loss function can be adjusted to specifically train the network for solving graph-cut, by focusing on the transition around 0.

Instead of adjusting the loss function we directly apply our WROF model and threshold the result. Fig. 2 shows an example segmentation. We use image gradients to define our edge-weights and simply employ the user input as 0/1 data cost, *cf*. Fig. 3. In the future, a better loss, *e.g*. counting mislabeled pixel and directly training the model for the task might lead to better models.

Gaussian Denoising. In Fig. 6 we display some additional examples.

Stereo. Recall that our network (Fig. 4) does not learn the edge weights and stereo data costs in this experiment. Instead both are treated as input to our network. This is not a principal restriction to our approach but rather a choice we make: We want to specifically learn the inference part of the MRF model, here used for stereo estimation. However, it should be rather simple to incorporate a sub-network that learns specialized data costs and edge weights, in order to jointly learn matching costs and inference in end-to-end fashion.

As mentioned in the paper we select normalized cross correlation (NCC) to provide the data cost for our stereo algorithm. In accordance with equation (9) from the paper we set

$$\vartheta_{\mathbf{i},d}:=\text{NCC}(\mathbf{i},d), \tag{11}$$

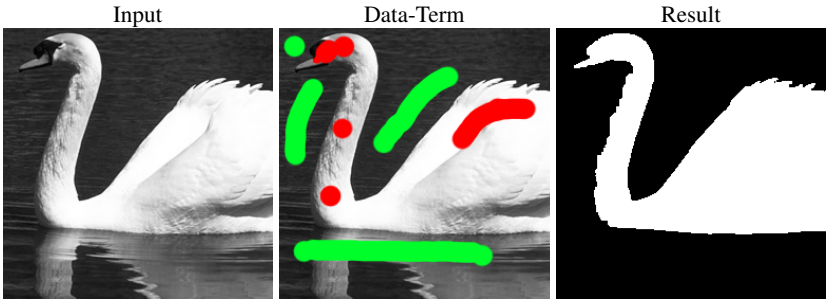

Fig. 2. Graph-cut computed with our WROF model.

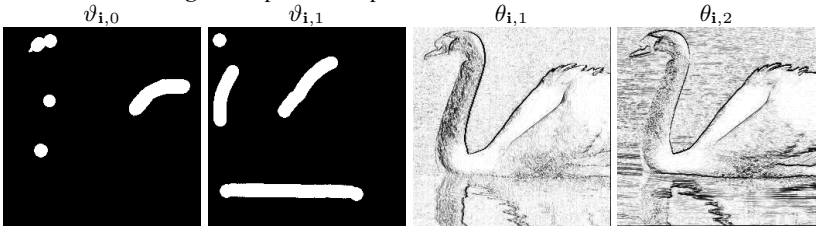

Fig. 3. Network input. *Left*: Data-costs, *Right*: Edge-weights.

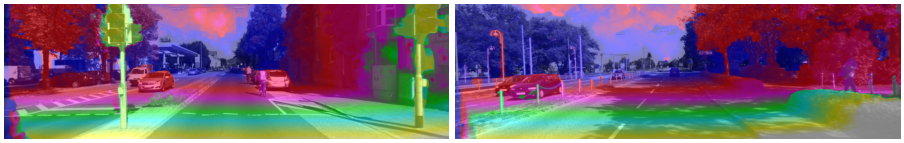

Fig. 4. Two additional results of our stereo model, trained via the truncated 3D distance function. Color code: White - near, blue - far.

where $\mathbf{i}$ denotes the position in the image and d the disparity. The normalized cross correlation (NCC) is defined as:

$$\mathrm{NCC}(\mathbf{i},d) = \sum_{\mathbf{q}\in\mathcal{N}(\mathbf{i})} \frac{(I_0(\mathbf{i}) - \mu_0(\mathbf{i}))(I_1(\mathbf{i},d) - \mu_1(\mathbf{i},d))}{\sigma_0(\mathbf{i})\sigma_1(\mathbf{i},d)}. \tag{12}$$

Here, the mean μ and variance σ are calculated over the neighborhood $\mathcal{N}(\mathbf{i})$ in the left image I_0 and the corresponding region in the right image I_1, displaced by the disparity d. The spatial edge-weights, $\theta_{\mathbf{i},k}$ from equation (9) in the paper, are computed similarly to the weights displayed in Fig. 3.

In the paper we train two variants, which only differ in the training data and not in the aforementioned input to the network. At first we use the model described in [5] to generate the data for our training. The respective inference network behaves like TV-regularization as expected. Note that the inf-convolution used in [5] is not easy to parallelize on the GPU – in contrast to our CNN, which is significantly faster. At second, we replace the output of [5] with a truncated 3D distance function, which is

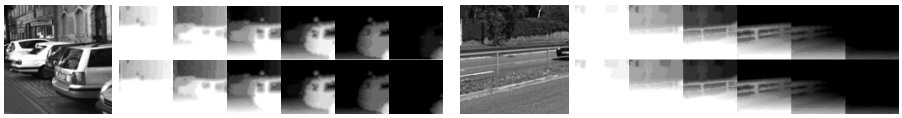

Fig. 5. Training our stereo network. *Left*: Example scene of out training set. *Right*: Truncated 3D distance function. The disparity is recovered as the 0-levelset of the distance function.

computed via fast marching [15], starting from the provided ground-truth data [12]. We receive an inference network that adheres to the properties of our training data. In particular, we do not observe staircasing artifacts and our reconstructions possess sharp object boundaries.

References

1. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH, ACM Press/Addison-Wesley Publishing Co. (2000)
2. Bredies, K., Kunisch, K., Pock, T.: Total generalized variation. SIAM J. Imaging Sciences (2010)
3. Chambolle, A.: Total variation minimization and a class of binary mrf models. EMMCVPR (2005)
4. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. JMIV 40(1) (2011)
5. Chambolle, A., Pock, T.: A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions. SMAI-JCM (2015)
6. Chapelle, O., Schlkopf, B., Zien, A.: Semi-Supervised Learning. The MIT Press (2010)
7. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F.R., Blei, D.M. (eds.) ICML. JMLR Workshop and Conference Proceedings (2015)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014)
9. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? PAMI (2004)
10. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
11. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. ICLR (2016)
12. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: CVPR (2015)
13. Parikh, N., Boyd, S., et al.: Proximal algorithms. Foundations and Trends® in Optimization 1(3), 127–239 (2014)
14. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D (1992)
15. Sethian, J.A.: Level set methods and fast marching methods. Cambridge monographs on applied and computational mathematics, Cambridge University Press (1999)

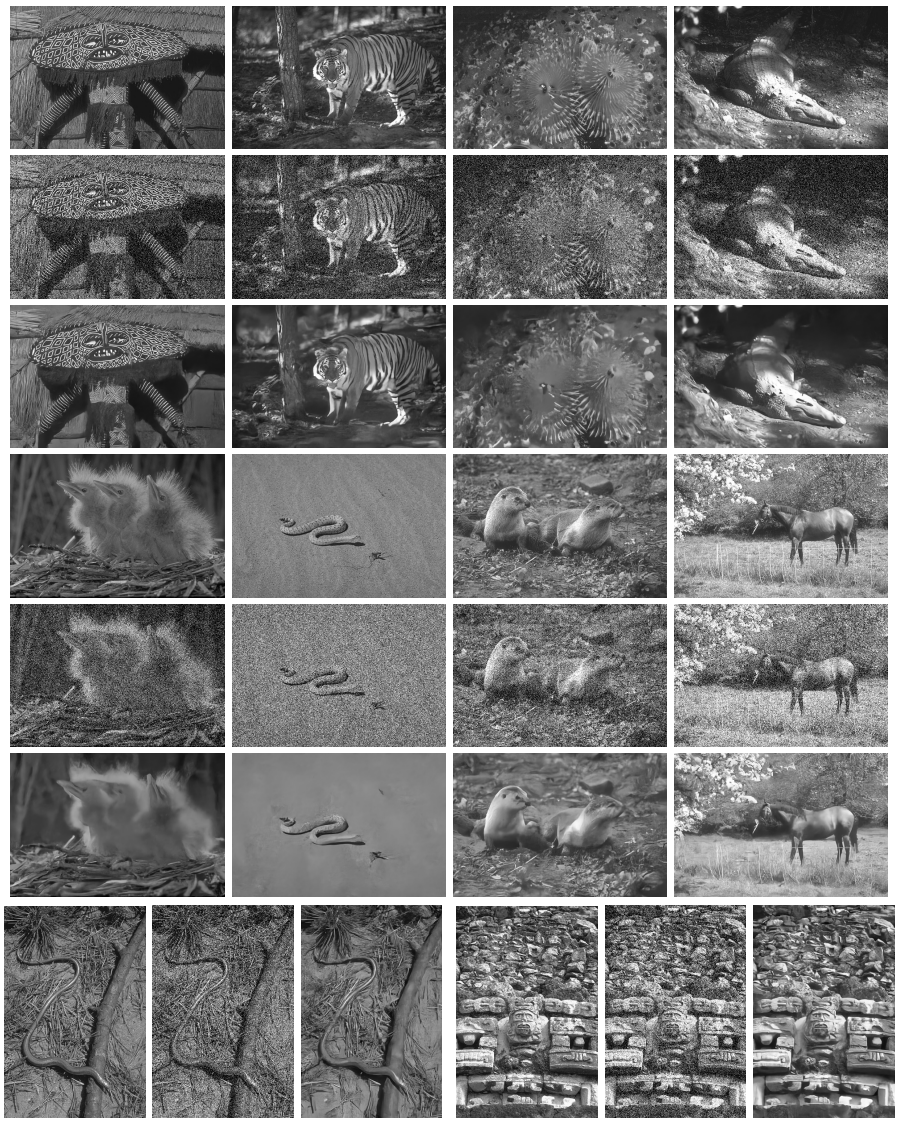

Fig. 6. Denoising examples (20% noise level). Landscape format, *top*: Original, *middle*: Noisy input, *bottom*: Reconstruction. Portrait format, *left*: Original, *middle*: Noisy input, *right*: Reconstruction.