Sparse Convolutional Networks using the Permutohedral Lattice

Martin Kiefel
Varun Jampani
Peter V. Gehler
Max Planck Institute for Intelligent Systems, Tübingen, 72076, Germany

Abstract

This paper introduces an efficient, non-linear image adaptive filtering as a generalization of the standard spatial convolution of convolutional neural networks (CNNs). We build on the bilateral filtering operation, a commonly used edge-aware image processing technique. Our implementation of bilateral filters uses specialized data structures, and in this paper we demonstrate how these lead to generalizations and make the filters amendable to learning. This development enriches the convolutional operation found in CNNs, which becomes image adaptive, can process sparse input data, and produce continuous output. Our result also generalizes a class of densely connected graphical models with tractable mean field inference. This view also has consequences for the bilateral filter itself. We remove the restriction to isotropic and separable kernels, allow more general filter classes and demonstrate how to learn them. As we show, the class of densely connected graphical models for which mean field inference is tractable can be largely increased from Gaussian edge potentials to highly parameterized pairwise potential functions. The insight we present in this paper generalizes the result in (Krähenbühl & Koltun, 2012).

The view that we take in this paper is motivated from a signal theoretic perspective. A continuous convolution of the signal \( v \) with a filter \( w \) at position \( x \) is defined as:

\[
(v * w)(x) = \int_{\mathbb{R}^d} v(\tau)w(x - \tau)d\tau, \quad (1)
\]

and when dealing with images this is typically approximated with a discrete filtering operation. The convolution that we present in this paper has two immediate consequences. First we allow a more general class of filters \( w \), supporting convolutions not only on the image grid but in general feature spaces, for example using radiometric differences such as color. Second, the signal \( v \) can be represented at a sparse set of points. For example, even if not all pixels of an image are available, the convolution we propose is still well defined and can be used to process the image. Likewise, one can compute the result of the con-
convolution at arbitrary output positions \( x \), not necessarily restricted to the points where the input signal is presented. In the experimental section we will present some use cases for these properties.

In this paper we report on several contributions that are a consequence of the proposed general permutohedral lattice convolution:

- A generalization of the spatial convolutional layer in CNNs to an image dependent layer that is able to process sparse features and has an image adaptive receptive field.

- The ability to work with different input representations and the implicit learning of a de-convolution of the output. This enables a scale free handling of input and output data.

- An extension of the permutohedral blur from radial and separable to arbitrary filters and from fixed (Gaussian) parameters to learned parameters.

- The generalization of the result from (Krähenbühl & Koltun, 2012) from Gaussian edge potentials to more highly parameterized models.

2. Related Work

Processing image data has been one of the primary use cases for convolutional neural networks, starting with document recognition (LeCun et al., 1998). In recent years the success of CNNs in image classification (Krizhevsky et al., 2012) has sparked an interest in learning-based image representations in the computer vision community and CNNs have found wide-spread use. These include diverse problems such as image classification (Krizhevsky et al., 2012), object detection (Girshick et al., 2013), structured prediction problems like human pose estimation (Tompson et al., 2014) and image denoising (Burger et al., 2012).

All aforementioned works consider architectures where the convolutional layer is a standard discrete convolution in the image domain. In (Lin et al., 2014) a network in network (NiN) is proposed to generalize the linear filtering operation of the convolution to a multi-layer perceptron. In effect the filtering becomes non-linear in the receptive field which increases the discriminability of the network of a single unit. This NiN change leads to performance increases on benchmark datasets compared to common linear CNNs. This is similar in spirit to what the permutohedral convolution achieves, the permutohedral convolution depends on the input patch it operates on.

Our work is inspired by the bilateral filtering operation (Aurich & Weule, 1995; Smith & Brady, 1997; Tomasi & Roberto, 1998), an influential image processing operation. The bilateral filter is a generalization of the simple Gaussian convolution for smoothing an image. A discrete bilateral filtering corresponds to the following operation:

\[
v_i = \sum_j \exp \left( -\frac{1}{2\sigma^2} \| f_i - f_j \|^2 \right) v_j, \quad f \in \mathbb{R}^d, \quad v \in \mathbb{R},
\]

with \( f_i, f_j \) being five-dimensional vectors of positional (2 dimensions) and color values (3 dimensions). The main benefit of this operation is its image adaptivity: inputs are averaged only if they are close in position and color value. This results in a filter that preserves sharp edges, a desired property for any image processing technique. Ever since its inception, the bilateral filter has found wide-spread use in applications of image processing, such as image denoising and texture/illumination separation, among others. A comprehensive survey of theory and applications of the bilateral filter can be found in (Paris et al., 2008). Because of its practical importance, much effort has been devoted to devise computational methods for fast evaluation of (2).

We build on the permutohedral lattice data structure introduced in (Adams et al., 2010) but note that the use of Gaussian k-d trees (Adams et al., 2009) may also be considered, especially in the case of higher dimensional features \( f \).

The work of (Krähenbühl & Koltun, 2012) highlights a link between the bilateral filtering operation and a dense conditional random field. A mean field iteration for a densely connected CRF with Gaussian edge potentials can be implemented as a bilateral filtering operation. This makes mean field inference tractable for an expressive class of models. More recent work has shown how parameters in this class of models can be learned from data (Krähenbühl & Koltun, 2013; Kiefel & Gehler, 2014) but both are limited to Gaussian filterings. The general permutohedral convolution can be understood as a generalization of (Krähenbühl & Koltun, 2012). The potential functions of the dense CRF are not restricted to be Gaussians but can be parameterized and more importantly, be learned from training data, significantly enhancing the expressiveness of the model class. The result of (Krähenbühl & Koltun, 2012) has found many use cases in computer vision applications (Vineet et al., 2013; 2012; Campbell et al., 2013).

3. Permutohedral Lattice Convolution

We propose a generalized convolution operation on a \( d \)-dimensional input space. Let us begin with the discrete spatial convolution of a regular CNN. This operation corresponds to evaluate

\[
v'_i = \sum_{j \in N(i)} w_{i,j} v_j,
\]
Figure 1. The permutohedral convolution consists of three steps: first the samples are splatted onto the lattice, then a convolution operates on the lattice considering a margin of \( s = 2 \) neighbors for each grid point, and finally the result of the convolution is transformed back to the output samples.

Let us consider a direct generalization of this operation to also include color information. When encoding RGB values, the features \( f_i, f_j \) are five dimensional vectors. For example, in the case of the bilateral filtering with standard deviation \( \sigma \) in Eq. (2), we can identify the filter value as

\[
w(f_i, f_j) = \exp(-\frac{1}{2\sigma^2} \| f_i - f_j \|^2).
\]

The convolution now operates on a five-dimensional space of position and color. A direct implementation of such an operation using common building blocks of a CNN library would map each pixel into the cross product of space and color, i.e. \( \mathbb{Z}^2 \times \mathbb{Z}^3 \) with the set of all integers \( \mathbb{Z} \). We can now define a filter as a five-dimensional tensor to convolve the signal. Reading out the non-zero grid points and mapping them back to their original image positions reconstructs the blurred result.

In general, this product space is populated sparsely which results in an expensive convolution that needs to consider every grid point in the product space. Also, in general the input features \( f \) may not lie on a regular \( d \) dimensional grid of the form \( \mathbb{Z}^d \) but be specified through continuous values in \( \mathbb{R}^d \). Hence, an interpolation method is needed to map the samples onto a regular grid. Every sample is influencing the corners of the hyper-cube it lies in, whereas the number of corners grows exponentially with the dimensionality of the input space. In the bilateral case the five dimensional cube has 32 corners.

We now turn to an implementation that solves the aforementioned problems. Input data is a tuple \((f_i, v_i)\) of feature locations \( f_i \in \mathbb{R}^d \) and corresponding signal values \( v_i \in \mathbb{R} \). Importantly, this does not assume the feature locations \( f_i \) to be sampled on a regular grid, e.g. \( \mathbb{Z}^d \), like position and RGB. Similarly to the naive implementation of the bilateral convolution operator we map the input signal to a regular structure, that we call a lattice. A convolution operates on the constructed lattice and the result is mapped back to the output space. The entire operation consists of three stages (see Figure 1): splatting (the mapping to the lattice space), convolution and slicing (the mapping back from the lattice).

This strategy has been used to implement fast Gaussian filtering (Paris & Durand, 2009; Adams et al., 2010; 2009). Here we use it for arbitrary convolutions.

The desired properties of the underlying lattice are problem specific and depend on the filter and dimensionality of the input space. Here, we suggest to use the permutohedral lattice which has been introduced to compute approximations to Eq. (2) (Adams et al., 2010). This data structure is suited for feature dimensions between 4 and 10 and small standard deviations \( \sigma \). For other settings faster approximations exist, see Adams et al. (2009) for a discussion.

3.1. Review of Permutohedral Lattice

The permutohedral lattice is the result of the projection of the set \( \mathbb{Z}^{d+1} \) onto a plane defined by the normal vector \( \mathbf{1} \in \mathbb{R}^{d+1} \). This \( d \) dimensional plane is embedded into \( \mathbb{R}^{d+1} \). An example of a lattice of two dimensions is depicted in Figure 1. The lattice points tessellate the subspace with regular cells. Given a point from the embedding space, it is efficient to find the enclosing simplex of the projection onto the plane. We will represent a sparse set of points from \( \mathbb{R}^d \) by a sparse set of simplex corners in the lattice. Importantly, the number of corners does not grow exponentially with the dimension \( d \) as it would for an axis-align simplex representation. We continue to describe the different parts of the permutohedral convolution for completeness.

The splat and slice operations take the role of an interpolation between the different signal representations. There exists an efficient \( O(d) \) time one-to-one mapping between the input space \( \mathbb{R}^d \) and the lattice plane that is embedded in \( \mathbb{R}^{d+1} \). Finding the enclosing simplex of a point on the lattice plane can be accomplished in \( O(d^2) \).

The splat operation (cf. left-most image in Fig. 1) now proceeds by finding the enclosing simplex on the lattice of a given point from the input space and distributes its value onto the corners according to the barycentric coordinates inside the simplex. The value at a lattice point \( l_k \) is computed by summing over all enclosed input points. This can be defined formally using the index set \( C(k) \) and the barycentric coordinates \( b_{k,i} \) of input point \( i \) with re-
spect to the corner \( k \) of the enclosing simplex as \( l_k = \sum_{i \in C(k)} b_{k,i} v_i \).

The reverse slice operation (cf. right-most image in Fig. 1) computes an output value \( v'_i \) by using its barycentric coordinates \( b_{k,i} \) and sums over the corner points \( k \) whenever the input point \( i \) belongs to the index set \( C(k) \), \( v'_i = \sum_{k \in C(k)} b_{k,i} v_i \).

The trick of the implementation of the bilateral blur from the work of (Adams et al., 2010) is to only allocate those simplices that are inhabited by at least one input point and thus this data structure is input dependent.

### 3.2. General Permutohedral Convolutions

The algorithm presented in (Adams et al., 2010) implements a separable Gaussian blur on the lattice structure. This approximates the bilateral filtering operation in Eq. (2). The contribution of this work is to extend this fixed Gaussian convolution to a more general one and combine the advantages of the feature flexibility of the input points and the expressive power of the commonly used spatial convolution of CNNs.

A hyper-hexagon substitutes the hyper-cube of the spatial convolution. The entries of the hexagonal filter are accessed according to the difference of the involved lattice points. An example of a two-dimensional permutohedral filter is shown in Fig. 1. It is defined by kernel values \( w_n \) to compute \( l'_k = \sum_{n,k'}(n,k') \in N(k) w_n l_{k'} \). The convolution kernel \( w_n \) is problem specific and its domain is restricted to the set of neighboring lattice points \( N(k) \). In a typical use, a Gaussian filter to defines and fixes the values of \( w_n \), but it can be understood as a general filtering operation in the permutohedral lattice space with free parameters. This viewpoint has non-trivial consequences for CNNs, bilateral filters, and probabilistic graphical models that we discuss in more detail in Section 4.

The size of the neighborhood \( N(k) \) plays a similar role as the filter size (spatial extent) of a grid-based CNN. The filtering kernel of a common spatial convolution that considers \( s \) points to either side in all dimensions has \( (2s + 1)^d \in \mathcal{O}(s^d) \) parameters. A comparable filter on the permutohedral lattice with a \( s \) neighborhood is specified by \( 1 + (d + 1)(s^d + s) \in \mathcal{O}(s^d) \) elements.

All the operations described above are linear with respect to the input values \( v_i \) and in the filter weights \( w_n \). The derivation of the gradients with respect to those is hence straightforward and gradient computation is again a convolution in the permutohedral space.

The convolution in the permutohedral space can be used as a building block for a convolutional layer in a CNN architecture. Parameters can be learned using backpropagation in the same way that spatial filters are learned. However, this convolution is image dependent; in other words, the receptive field changes from patch to patch and this behaviour can be controlled by specifying features \( f_i \).

#### Runtime.

The permutohedral lattice convolution is parallelizable, and scales with the filter size. Specialized implementations run in real time in image processing applications. The slicing and splatting operations incur extra computational cost over the standard convolution. We implemented the permutohedral lattice convolution in the Caffe software package (Jia et al., 2014). When comparing to a 3×3 spatial convolution implemented in Caffe we found an increase in runtime of about a factor of two. Our implementation does not rely on BLAS and is tailored for fast gradient estimation which can be omitted for test time. Therefore we expect similar runtimes with an optimized version. The code will be made available upon acceptance.

### 4. Discussion of Permutohedral Lattice Convolutions

The previous section motivated a generalized convolution operation based on a data structure developed for the bilateral filter. Here we describe several aspects of this convolution operation when used as a building block for common computer vision tasks. We will also discuss possible applications and the implications on bilateral filtering for densely connected CRFs.

#### Substituting a Spatial Convolution.

The permutohedral lattice convolution can be used as a replacement of the common spatial convolution. When using only positional information as features \( f \) a permutohedral convolution closely resembles the common spatial convolution. When used in CNNs we find empirically similar results for standard problems such as image and digit classification. Since the permutohedral lattice convolution incurs extra computational cost by splatting and slicing, the use of spatial kernels is only advised in scenarios of varying input and output spaces, as discussed next. In the case of using position features only, the interpolation into the lattice has a similar effect as a Gaussian blur on the input signal as noted by (Adams et al., 2010). This blur can have a degrading effect on the input signal which may lead to a worse prediction performance of the CNN.

#### Using Varying Input and Output Spaces.

The permutohedral convolution decouples the signal sampling from the convolution. Since the filter is defined on the lattice points directly, it can process varying sized inputs. We can understand an image as a signal that is being sampled and represented at the pixel locations. The number of pixels determines the image resolution. Whether the image...
data is splatted into the lattice at full resolution, at lower resolution, or using a random selection of pixels, is independent from the convolution that acts on the lattice points and therefore remains well defined. The use of more pixels approximates the “true” signal more faithfully and leads to a lower approximation error of the continuous convolution Eq.(1). In other words, the permutohedral filter implicitly defines an interpolation of the input signal. This can be exploited in cases where the input sampling is sparse or where the output is coarser or finer depending on the application. As an example we train a CNN for digit recognition but present the input images only by sub-sampling pixels at random positions, see Section 5.1.

The same reasoning is true for output signals: we can read out the result of the convolution at arbitrary points. Consider the problem of semantic segmentation using a CNN. The use of spatial pooling layers in a CNN reduces the resolution of the output signal, which in turn is only a coarse prediction. This can be fixed in several ways, and different methods have been proposed to learn a de-convolution as a final layer in the network that recovers the full resolution of the input image (Long et al., 2014; Chen et al., 2014). In case the last layer is a permutohedral convolutional layer, an output can be produced at any desired resolution and there is no need to devise more specialized architectures. This experiment will be discussed in Section 5.3.

**Generalized Bilateral Filtering.** The original use of bilateral filters is to perform image denoising. Throughout the image processing literature bilateral filtering always corresponds to the use of Gaussian kernels. This has an image adaptive behaviour in the way Eq.(2) prescribes. However, as we note in this paper this restriction can be removed and the view as a parameterized convolutional operation immediately suggests a generalization. The bilateral filter can be learned, also using a multi-layer architecture, while retaining its edge-aware behaviour that was found to be advantageous for image processing applications. We conduct a illustrative experiment to compare standard CNNs and permutohedral layer CNNs that use image information, see experiment 5.2 for details. The application to other image processing problems that use bilateral filters are an interesting direction to explore.

**Potential Functions in Dense CRF.** The work of (Krähenbühl & Koltun, 2012) shows that a mean field approximation of a special class of densely connected CRFs can be implemented as bilateral filtering. Although mean field is an approximate inference method, this result allowed to use a rich class of models and was found to be of practical value in several computer vision applications (Krähenbühl & Koltun, 2012; Vineet et al., 2013; 2012; Campbell et al., 2013). The papers (Krähenbühl & Koltun, 2012; Vineet et al., 2013; 2012; Campbell et al., 2013). The work testing direction to explore.

| Method     | Original | 100% | 60%  | 20%   |
|------------|----------|------|------|-------|
| LeNet      | 0.9919   | 0.9660| 0.9348| 0.6434|
| PCNN       | 0.9903   | 0.9844| 0.9534| 0.5767|
| LeNet 100%| 0.9856   | 0.9809| 0.9678| 0.7386|
| PCNN 100% | 0.9900   | 0.9863| 0.9699| 0.6910|
| LeNet 60% | 0.9789   | 0.9821| 0.9740| 0.8151|
| PCNN 60%  | 0.9885   | 0.9864| 0.9771| 0.8214|
| LeNet 20% | 0.9763   | 0.9754| 0.9665| 0.8928|
| PCNN 20%  | 0.9728   | 0.9735| 0.9701| 0.9042|

Table 1. Classification accuracy on MNIST. We compare the LeNet (LeCun et al., 1998) implementation that is part of Caffe (Jia et al., 2014) to the network with the first layer replaced by a permutohedral convolution layer (PCNN). Both are trained on the original image resolution (first two rows). Three more PCNN and CNN models are trained with randomly subsampled images (100%, 60% and 20% of the pixels). An additional bilinear interpolation layer samples the input signal on a spatial grid for the CNN model.

Koltun, 2013; Kiefel & Gehler, 2014 discuss learning of parameters in this CRF, but both use a Gaussian filter and only show how to learn scaling parameters.

The permutohedral convolution generalizes the class of dense CRF models for which the result of (Krähenbühl & Koltun, 2012) holds, in effect we remove the parametric restriction to Gaussian edge potentials. In our case the parameters of the filter in the permutohedral lattice define the pairwise potential functions and mean field inference still corresponds to a convolution. The values can then be learned by backpropagation through the mean field iterations. This view of the model with unrolled inference is advocated in (Domke, 2013; Stoyanov et al., 2011) and suggested to be of practical value in the limited computation setting in (Wainwright, 2006). In the experiments we include a dense CRF for which parameters are learned.

## 5. Experiments

We evaluate the permutohedral CNN in different scenarios to validate its performance and also to highlight different potential use cases. These include a diverse set of experiments, where we start with illustrative examples and conclude with a challenging semantic segmentation problem.

### 5.1. Subsampling Signals

One of the strengths of the proposed method is that it does not depend on a regular grid sampling as the traditional convolution operators. We highlight this feature with the following experiment using the MNIST 10 class classification problem. We sample continuous points uniformly at random in the input image, use their interpolated values as signal and continuous positions as features. This mimics sub-sampling of a high-dimensional signal. Additionally,
Sparse Convolutional Networks using the Permutohedral Lattice

| Method         | Test MSE   |
|----------------|------------|
| 2-layer CNN    | 1.66623 · 10^{-5} |
| 2-layer PCNN   | 1.29221 · 10^{-5} |

Table 2. MSE results of a denoising task using the BSDS500 dataset (Arbelz et al., 2011). Reported is the mean squared distance to the ground truth image averaged over all pixels.

we interpolate the low-dimensional input signal at grid positions to compare against a spatial convolution.

We take a reference implementation of LeNet (LeCun et al., 1998) that is part of the Caffe project (Jia et al., 2014) and compare it against the same architecture but replacing the first convolutional layer with a permutohedral lattice convolution (PCNN). The filter size and numbers are adjusted to get a comparable number of parameters (5 × 5 for LeNet, 2-neighborhood for PCNN).

The results are shown in Table 1. We see that training on the original MNIST data (column Original, LeNet vs. PCNN) leads to a slight decrease in performance of the PCNN (99.03%) compared to LeNet (99.19%). The PCNN can be trained and evaluated on sparse signals, and we resample the image as described above for 100%, 60% and 20% of the pixels at random locations. The methods are also evaluated on test images that are subsampled in the same way. Note that we can train and test with different subsampling rates. We introduce an additional bilinear interpolation layer for the LeNet architecture to train on the same data. Both models do a spatial interpolation and we expect them to yield a similar classification accuracy.

5.2. Image Denoising

The main application that inspired the development of the bilateral filtering operation is image denoising (Aurich & Weule, 1995). We conducted an experiment to compare the performance of two different two-layer CNN architectures on a denoising task. Both networks are trained with patches of size 50 × 50 for a full-convolution output. We use images from the Berkeley segmentation dataset (BSDS500) (Arbelz et al., 2011), converted to single-channel, and corrupted with Gaussian noise with a standard deviation of 0.05 with respect to pixel from the domain [0, 1]. Each network is trained using Euclidean loss and 20 patches from the training images per iteration.

The first network uses two spatial convolution layers (2-layer CNN in Table 2). The first convolutional layer has a kernel size of 5 and outputs 17 values and the second layer has a convolution bank with kernel size of 5 and outputs the scalar gray-scale value. This network is trained with a fixed learning rate of 10^{-4}, momentum weight of 0.9 and a weight decay of 5 · 10^{-4} for 100k steps.

| Method                             | Validation-IoU | Test-IoU |
|------------------------------------|----------------|----------|
| Chen et al. (2014)-DeepLab         | 59.80          | -        |
| Chen et al. (2014)-CRF             | 63.74          | 66.4     |
| Long et al. (2014)                 | -              | 62.2     |
| Mostajabi et al. (2014)            | -              | 64.4     |
| Harirhan et al. (2014)             | -              | 51.6     |
| Ion et al. (2014)                  | -              | 47.8     |
| Dai et al. (2014)                  | -              | 57.0     |
| ours-DeepLab                       | 60.27          | 61.4     |

Table 3. IoU results on Pascal VOC12. Comparison to the literature and results obtained by cross-validation with a Gaussian kernel / learning. ours-DeepLab result corresponds to our re-implementation of (Chen et al., 2014).

In the second network all convolutions are performed in the permutohedral lattice (2-layer PCNN). Every pixel that is used during the computation is also annotated with the gray-scale intensity information in the image which results in a two-stage bilateral filtering setup. Since the neighborhood of the permutohedral filters is set to 2 (to get a comparable spatial extent to the CNN counterpart) we reduce the number of outputs after the first layer to get a comparable number of free parameters (868 for the 2-layer CNN and 831 for the 2-layer PCNN). The PCNN is trained for 100k steps and fixed learning rate of 10^{-5}.

We evaluate the mean squared error on full images from the validation set and see a slightly better performance of the bilateral network with trained filters, see Table 2. Admittedly this setup is rather simple, but it validates that the generalized bilateral filtering has an advantage when applied to image data. A complete image denoising system would employ RGB color information and also needs to be properly adjusted in network size. Multi-layer perceptrons have obtained state-of-the-art denoising results (Burger et al., 2014) and the permutohedral lattice layer can readily be used in such an architecture.

5.3. Semantic Segmentation

Semantic segmentation can be understood as a pixel-classification task, that is given an image patch, predict the class of the center pixel. In principle any classifier can be trained for this input/output relationship and then be applied by sliding it over the image to obtain a classification for every image pixel. This sliding window evaluation is inefficient as it requires multiple calls to the classification system. Convolutional networks benefit from shared computation at the convolutional stages but spatial pooling layers reduce the resolution of the produced output.

In this section we apply the permutohedral convolutional layer for this task. Over the last year the state-of-the-art
performance in semantic segmentation was considerably increased due to the adoption of CNN architectures in conjunction with the use of additional label information. Convolutional networks that are adopted versions from image classification systems are dominating the leaderboards on most image datasets (Chen et al., 2014; Long et al., 2014; Mostajabi et al., 2014; Dai et al., 2014; Hariharan et al., 2014).

We adopt the “DeepLab” network described in (Chen et al., 2014). The CNN from (Simonyan & Zisserman, 2014) (VGG) that obtained the best result on the recent ImageNet 2012 challenge in classification and localization was used as a starting point. All fully-connected layers are substituted with convolutions to generate a full image prediction. The output layer is reduced to a 21 class prediction. As reported by Chen et al. (2014) due to spatial pooling, this leads to a network with an output stride of 32. This can be further reduced to a stride of 8 with the atrous convolution (Mallat, 2008).

Based on the pre-trained weights from the VGG model, the parameters of this network were fine-tuned for $200 \cdot 10^3$ steps with a batch size of 10 patches. The learning rate was set to be $10^{-4}$ for the first $100 \cdot 10^3$ iterations and successively reduced by 0.1 after every $50 \cdot 10^3$ steps. The weight decay and momentum were fixed to $5 \cdot 10^{-4}$ and 0.9.

The Pascal VOC segmentation dataset (Everingham et al., 2012) is the de-facto standard benchmark dataset for this task. It comprises 21 classes and comes with a pre-defined split into training (1464 images), validation (1449 images) and test data (1456 images). Following the training protocol of Chen et al. (2014) we augment the training set with images from Hariharan et al. (2011) to a size of 10,582 training samples. The performance of all following models is measured in the intersection over union score.

Table 3 confirms the score of DeepLab proposed by Chen et al. (2014). Compared to the original publication, which reports 59.80, our training resulted in 60.27. We use a bilinear interpolation to upscale the downsamplied prediction of the DeepLab model. A denser evaluation of the network would remove the need for output interpolation. Nevertheless, this also incurs a significantly higher computational cost since the image needs to be feed-in with several offsets. The permutohedral convolution can be computed at higher output resolutions and thus can directly be used to circumvent this problem. We discuss two variants.

**Bilateral Infilling.** We can augment the architecture by a permutohedral convolution layer that processes the output of the DeepLab model. As explained in the previous sections, this layer can deal with the different sampling granularity of the input compared to the denser output. To better guide the infilling of class label information we make use of the color information along with the position for each pixel. We call this the “Bilateral” model. An initialization of the filter weights that correspond to a Gaussian convolution already leads to an improvement by 1.0%. Piece-wise training of the filter weights further improves this performance by 0.25% (see Table 3).

**Learning Mean Field Inference.** We are inspired by Chen et al. (2014) who run a dense-CRF inference on using the DeepLab output as unary predictions. Their model uses the mean field approximation of a dense-CRF presented in Krähenbühl & Koltun (2012). Here we propose to directly stack additional permutohedral convolutional layers on top of the DeepLab network. Each of these additional layers mimic the iterative computational steps of a mean field approximation. The one-layered extension “MF (1 step)” – initialized with a Gaussian kernel – leads to a higher performance compared to the base model and the bilateral extension. Adding another such layer continues this trend. Mean field update steps require that the same...
Sparse Convolutional Networks using the Permutohedral Lattice

Figure 3. We feed a low resolution image (b) into the DeepLab network to produce a coarse semantic prediction (c). The permutohedral lattice convolution can be computed at any desired output resolution. Therefore we can use a higher resolution version of the input image (a) to produce a high resolution semantic prediction (d). Although the intermediate DeepLab prediction is much coarser, it is almost identical to the output (e) obtained by processing the high resolution image (a) directly.

Visual results. Using color information considerably improves the visual quality of the predictions. In Figure 2 we show qualitative results of the learned mean field inference method compared to the linear interpolated result of DeepLab. The label predictions follow closely the object boundaries and carve out fine detail of the depicted objects.

In Fig. 3 we show another use case of computing the output of the permutohedral lattice at higher resolutions. Consider that we have a high resolution image (Fig. 3(Left)) and evaluate DeepLab at every second pixel only, thus reducing the effective resolution of the image. The semantic output is on a very coarse scale. We feed this result into the “MF (2 steps)” model. Since the last layer is a permutohedral lattice, the result can be read out at a higher resolution than what was used as input for DeepLab, here the original input image. This produces a sharp result that interpolates using the learned filter and the input image at high resolution. In contrast Figure 3 (right) is the result when using the high resolution image directly as input to DeepLab. This result is not considerably different from the prediction using a subsampled input image and the high resolution only for generating the output.

Discussion. The problem of generating a denser output for full-convolutional networks has also been approached by Long et al. (2014) who learns a de-convolution on the output of the network. Our work is similar in the sense that the learned permutohedral convolution is also trained to reconstruct the image labels as well as possible but in contrast to Long et al. (2014) can use color information.

6. Conclusion

This paper presents a generalization of the spatial convolutional layer in CNNs to sparse input signals. The presented method also generalizes the result from (Krähenbühl & Koltun, 2012) and allows the use of densely connected graphs with parameterized potential functions, not restricted to the class of Gaussian edge potentials. We highlight different possible use cases. Given the wide-spread use of both CNNs and the dense model of (Krähenbühl & Koltun, 2012) we envision that the presented extension will prove useful in different applications.

We are not advocating the use of the permutohedral lattice convolution as a general replacement of spatial convolutions in CNNs. The latter can of course be tuned to achieve similar effects, especially by increasing the number of filters per layer. As we discuss in this paper, there are several important use cases where a generalized bilateral convolution is advantageous. Applications include semantic segmentation, denoising, but also other applications of structured output prediction like the human pose estimation model of (Kiefel & Gehler, 2014).

The gained flexibility of sparse feature information can be exploited in different ways. In this paper we presented experiments using color information as in the original bilateral filtering operation and used in (Krähenbühl & Koltun, 2012). Several extensions are possible, for example the non-local means (Buades et al., 2005) method uses PCA coefficients obtained from image patches as features. We plan to encode scale invariance and unary uncertainty into the convolution. For CNN architectures, scale invariance is typically implemented by presenting multiple versions of an image to several branches of a network. A similar effect could be obtained by using a special lattice structure. Another route is to attempt to learn the features \( f \) that define the lattice. We leave this possibility as future work.

Prior knowledge about the observed signal can be used to define the domain of the convolution. The spatial filter of CNNs is a particular type of prior knowledge, in this work we demonstrate that this can be generalized to sparse signals and show several consequences of this viewpoint.
References

Adams, Andrew, Gelfand, Natasha, Dolson, Jennifer, and Levoy, Marc. Gaussian kd-trees for fast high-dimensional filtering. In ACM SIGGRAPH, 2009.

Adams, Andrew, Baek, Jongmin, and Davis, Myers Abraham. Fast high-dimensional filtering using the permutohedral lattice. Comput. Graph. Forum, 29(2):753–762, 2010.

Arbelez, Pablo, Maire, Michael, Fowlkes, Charless, and Malik, Jitendra. Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 33(5):898–916, May 2011.

Aurich, Volker and Weule, Jörg. Non-linear Gaussian filters performing edge preserving diffusion. In DAGM, pp. 538–545, 1995.

Buades, A., Coll, B., and Morel, J.-M. A non-local algorithm for image denoising. In Computer Vision and Pattern Recognition (CVPR), volume 2, pp. 60–65 vol. 2, June 2005.

Burger, Harold C., Schuler, Christian J., and Harmeling, Stefan. Image denoising: Can plain neural networks compete with bm3d? In CVPR, 2012.

Campbell, Neill D. F., Subr, Kartic, and Kautz, Jan. Fully-Connected CRFs with Non-Parametric Pairwise Potentials. In International Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, and Yuille, Alan L. Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062v1, 2014.

Dai, Jifeng, He, Kaiming, and Sun, Jian. Convolutional feature masking for joint object and stuff segmentation. arXiv preprint arXiv:1412.1283, 2014.

Domke, Justin. Learning graphical model parameters with approximate marginal inference. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI), 2013.

Everingham, Mark, Gool, Luc Van, Williams, C.K., Winn, John, and Zisserman, Andrew. The pascal voc2012 challenge results. 2012.

Farabet, Clement, Couprie, Camille, Najman, Laurent, and LeCun, Yann. Learning hierarchical features for scene labeling. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2013.

Girshick, Ross B., Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR, abs/1311.2524, 2013.

Hariharan, Bharath, Arbelaez, Pablo, Bourdev, Lubomir, Maji, Subhransu, and Malik, Jitendra. Semantic contours from inverse detectors. In International Conference on Computer Vision (ICCV), 2011.

Hariharan, Bharath, Arbelaez, Pablo, Girshick, Ross, and Malik, Jitendra. Simultaneous detection and segmentation. In Computer Vision–ECCV 2014, pp. 297–312. Springer, 2014.

Ion, Adrian, Carreira, João, and Sminchisescu, Cristian. Probabilistic joint image segmentation and labeling by figure-ground composition. IJCV, pp. 40–57, 2014.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014.

Kiefel, Martin and Gehler, Peter. Human pose estimation with fields of parts. In European Conference on Computer Vision (ECCV), 2014.

Krähenbühl, Philipp and Koltun, Vladlen. Efficient inference in fully connected crfs with gaussian edge potentials. NIPS, 2012.

Krähenbühl, Philipp and Koltun, Vladlen. Parameter learning and convergent inference for dense random fields. In ICML, 2013.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Neural Information Processing Systems (NIPS), 2012.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

Lin, M., Chen, Q., and Yan, S. Network in network. In International Conference on Learning Representations (ICLR), 2014.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. arXiv preprint arXiv:1411.4038v1, 2014.

Mallat, Stéphane. A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way. Academic Press, 3rd edition, 2008. ISBN 0123743702, 9780123743701.
Mostajabi, Mohammadreza, Yadollahpour, Payman, and Shakhnarovich, Gregory. Feedforward semantic segmentation with zoom-out features. *arXiv preprint arXiv:1412.0774*, 2014.

Paris, Sylvain and Durand, Frédéric. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81(1):24–52, January 2009.

Paris, Sylvain, Kornprobst, Pierre, Tumblin, Jack, and Durand, Frédéric. Bilateral filtering: Theory and applications. *Foundations and Trends in Computer Graphics and Vision*, 4(1):1–73, 2008. ISSN 1572-2740.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Smith, Stephen M. and Brady, J. Michael. SUSAN – a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, May 1997. ISSN 0920-5691.

Stoyanov, Veselin, Ropson, Alexander, and Eisner, Jason. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Tomasi, Carlo and Roberto, Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision (ICCV)*, pp. 839–846, 1998.

Tompson, Jonathan, Jain, Arjun, LeCun, Yann, and Bregler, Christoph. Joint training of a convolutional network and a graphical model for human pose estimation. In *Neural Information Processing Systems (NIPS)*, 2014.

Vineet, Vibhav, Warrell, Jonathan, and Torr, Philip H.S. Filter-based mean-field inference for random fields with higher order terms and product label-spaces. In *ECCV*, 2012.

Vineet, Vibhav, Sheasby, Glenn, Warrell, Jonathan, and Torr, Philip H.S. Posefield: An efficient mean-field based method for joint estimation of human pose, segmentation and depth. In *EMMCVPR*, 2013.

Wainwright, Martin J. Estimating the “wrong” graphical model: Benefits in the computation-limited setting. *J. Mach. Learn. Res.*, 7:1829–1859, December 2006. ISSN 1532-4435.