Learning Affinity via Spatial Propagation Networks

Sifei Liu
UC Merced, Nvidia

Shalini De Mello
Nvidia

Jinwei Gu
Nvidia

Guangyu Zhong
Dalian University of Technology

Ming-Hsuan Yang
UC Merced, Nvidia

Jan Kautz
Nvidia

Abstract

In this paper, we propose spatial propagation networks for learning the affinity matrix for vision tasks. We show that by constructing a row/column linear propagation model, the spatially varying transformation matrix exactly constitutes an affinity matrix that models dense, global pairwise relationships of an image. Specifically, we develop a three-way connection for the linear propagation model, which (a) formulates a sparse transformation matrix, where all elements can be the output from a deep CNN, but (b) results in a dense affinity matrix that effectively models any task-specific pairwise similarity matrix. Instead of designing the similarity kernels according to image features of two points, we can directly output all the similarities in a purely data-driven manner. The spatial propagation network is a generic framework that can be applied to many affinity-related tasks, including but not limited to image matting, segmentation and colorization, to name a few. Essentially, the model can learn semantically-aware affinity values for high-level vision tasks due to the powerful learning capability of the deep neural network classifier. We validate the framework on the task of refinement for image segmentation boundaries. Experiments on the HELEN face parsing and PASCAL VOC-2012 semantic segmentation tasks show that the spatial propagation network provides a general, effective and efficient solution for generating high-quality segmentation results.

1 Introduction

An affinity matrix is a generic matrix that determines how close, or similar, two points are in a space. In computer vision tasks, it is a weighted graph that regards each pixel as a node, and connects each pair of pixels by an edge [25, 16, 15, 10, 29]. The weight on that edge should reflect the pairwise similarity with respect to different tasks. For example, for low-level vision tasks such as image filtering, the affinity values should reveal the low-level coherence of color and texture [29, 28, 10, 9]; for mid to high-level vision tasks such as image matting and segmentation [16, 22], the affinity measure should reveal the semantic-level pairwise similarities. Most techniques explicitly or implicitly assume a measurement or a similarity structure over the space of configurations. The success of such algorithms depends heavily on the assumptions made to construct these affinity matrices, which are generally not treated as part of the learning problem.

In this paper, we show that the problem of learning the affinity matrix can be equivalently expressed as learning a group of small row/column-wise, spatially varying linear transformation matrices. Since a linear transformation can be easily implemented as a differentiable module in a deep neural network, the transformation matrix can be learned in a purely data-driven manner as opposed to being constructed by hand. Specifically, we adopt an independent deep CNN to output all entities of the matrix with the input of the original RGB images, such that the affinity is learned from a deep model conditioned on the specific inputs. We show that using a three-way connection, instead of the fully connection, is sufficient for learning a dense affinity matrix and requires much fewer output channels

of a deep CNN. Therefore, instead of using designed features and kernel tricks, our network outputs all entities of the affinity matrix in a data-driven manner.

The advantages of learning affinity matrix in a data-driven manner are multifold. First, the assumption of a similarity matrix based on distance metric in certain space (e.g., RGB or Euclidean [10, 25, 5, 36, 14]) may not describe the pairwise relations in mid-to-high-level feature space. To apply such designed pairwise kernels to tasks such as semantic segmentation, multiple iterations is required [14, 5, 36] for a satisfactory performance. In contrast, the proposed method learns and outputs all entities of an affinity matrix under direct supervision of ultimate loss functions, where no iteration, specific design or assumption of kernel function is needed. Second, we can learn the high-level affinity measure initializing from hierarchical deep features from the VGG [26] and ResNet [11] where conventional metrics and kernels may not be applied. Due to the above properties, especially the first one, the framework is far more efficient than the related graphical models, such as Dense CRF.

Our proposed architecture, namely spatial propagation network (SPN), contains a deep CNN that learns the affinity entities and a spatial linear propagation module. Images or general 2D matrix are fed into the module, and propagated under the guidance of the learned affinity. All modules are differentiable and jointly trained using stochastic gradient descent (SGD) method. The spatial linear propagation module is computationally efficient for inference due to the linear time complexity of the recurrent architecture.

2   Related Work

Numerous methods explicitly design affinity matrices for image filtering [29, 10], colorization [15], matting [16] and image segmentation [14] based on the physical nature of the problem. Other methods, such as total variation (TV) [23] and learning to diffuse [18] improve the modeling of pairwise relationships by utilizing different objectives, or incorporating more priors into diffusion partial differential equations (PDEs). However, due to the lack of an effective learning strategy, it is still challenging to produce a learning based affinity for complex visual analysis problems. Recently, Maire et al. [22] trained a deep ConvNet to directly predict the entities of an affinity matrix, which demonstrated good performance on image segmentation. However, since the affinity is followed by a solver of spectral embedding as an independent part, it is not directly supervised for the classification/prediction task. Bertasius et al. [2] introduced a random walk network that optimizes the objectives of pixel-wise affinity for semantic segmentation. Differently, the affinity matrix is additionally supervised by a ground-truth sparse pixel similarities, which limits the potential connections between pixels.

On the other hand, many graphical model-based methods have successfully improved the performance of image segmentation. In the deep learning framework, the conditional random fields (CRFs) with efficient mean field inference is frequently used [14, 36, 17, 5, 24, 1] to model the pairwise relations in the semantic labeling space. Some of them use it as a post-processing module [5], while others integrate it as a jointly-trained part [36, 17, 24, 1]. While both dense CRFs and the proposed method describe the densely connected pairwise relationships, dense CRFs relies on designed kernels, while ours utilizes propagation structure with directly learned pairwise links, which is much less explored before, but more suitable as an embedded deep-learning module. Since in this paper, SPN is trained as a universal segmentation refinement module, we specifically compare it with one of the methods [5] that relies on the dense CRF [14] as a post-processing strategy. Our architecture is also related to the multi-dimensional RNN or LSTM [30, 3, 8]. However, both the standard RNN and LSTM contain multiple non-linear units and thus do not fit into the proposed affinity framework.

3   Proposed Approach

In this work, we construct a spatial propagation network (SPN) that can transform a two-dimensional (2D) map (*e.g.*, the result of coarse image segmentation) to a new one with desired properties (*e.g.*, a new segmentation map with significantly refined details). With spatially varying parameters that supports the propagation process, we show theoretically in Section 3.1 that this module is equivalent to the standard anisotropic diffusion process [32, 18]. As proved, the transformation of maps is controlled by a Laplacian matrix that is constituted by the parameters of the spatial propagation module. Since the propagation module is differentiable, those parameters can be learned by any

type of neural network (*e.g.*, a typical deep CNN) that is connected to this module through joint training. We introduce the propagation network in Section 3.2, and specifically analyze the properties of different types of connections within the framework for learning the affinity matrix.

3.1 Linear Propagation as Spatial Diffusion

We apply a linear transformation by means of the spatial propagation network, where a matrix is scanned row/column-wise in four fixed directions: left-to-right, top-to-bottom, and verse-vise. This strategy is used widely in [8, 30, 19, 4]. We take the left-to-right direction as an example for the following discussion. Other directions are processed independently in the same manner.

We denote X and H as two 2D maps of size $n \times n$, with exactly the same dimensions as the matrix before and after spatial propagation, where x_t and h_t, respectively, represent their t^{th} columns with $n \times 1$ elements each. We linearly propagate information from left-to-right between adjacent columns using an $n \times n$ linear transform matrix w_t as:

$$h_t = (I - d_t)\, x_t + w_t h_{t-1}, \quad t \in [2, n] \tag{1}$$

where I is the $n \times n$ identity matrix, the initial condition $h_1 = x_1$, and $d_t(i, i)$ is a diagonal matrix, where the ith element is the sum of all the elements of the ith row of w_t:

$$d_t(i, i) = \sum_{j=1, j \neq i}^{n} w_t(i, j). \tag{2}$$

As shown, the matrix H, where $\{h_t \in H, t \in [1, n]\}$, is updated in a column-wise manner recursively. For each column, h_t is a linear, weighted combination of the previous column h_{t-1}, and the corresponding column x_t in X.

When the recursive scanning is finished, the updated 2D matrix H can be expressed with an expanded formulation of Eq. (1):

$$H_v = \begin{bmatrix} I & 0 & \cdots & \cdots & 0 \\ w_2 & \lambda_2 & 0 & \cdots & \cdots \\ w_3 w_2 & w_3 \lambda_2 & \lambda_3 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \lambda_n \end{bmatrix} X_v = G X_v, \tag{3}$$

where G is a lower triangular, $N \times N (N = n^2)$ transformation matrix, which relates X and H. H_v and X_v are vectorized versions of X and H, respectively, with the dimension of $N \times 1$. Specifically, they are created by concatenating h_t and x_t along the same, single dimension, *i.e.*, $H_v = \left[h_1^T, ..., h_n^T \right]^T$ and $X_v = \left[x_1^T, ..., x_n^T \right]^T$. All the parameters $\{\lambda_t, w_t, d_t, I\}, t \in [2, n]$ are $n \times n$ sub-matrices, where $\lambda_t = I - d_t$.

In the following section, we validate that Eq. (3) can be expressed as a spatial anisotropic diffusion process, with the corresponding propagation affinity matrix constituted by all $w_t, t \in [2, n]$.

Theorem 1. *The summation of elements in each row of G equals to one.*

Since G contains $n \times n$ sub-matrices, each representing the transformation between the corresponding columns of H and X, we denote all the weights used to compute h_t as the tth block-row G_t. On setting $\lambda_1 = I$, the kth constituent $n \times n$ sub-matrix of G_t is:

$$G_{tk} = \begin{cases} \prod_{\tau=k+1}^{t} w_\tau \lambda_k, & k \in [1, t-1] \\ \lambda_k, & k = t \end{cases} \tag{4}$$

To prove that the summation of any row in G equals to one, we instead prove that for $\forall t \in [1, n]$, each row of G_t has the summation of one.

Proof. Denoting $E = [1, 1, ..., 1]^T$ as an $n \times 1$ vector, we need to prove that $G_t [1, ..., 1]_{N \times 1}^T = E$. Equivalently $\sum_{k=1}^{t} G_{tk} E = E$, because G is a lower triangular matrix. In the following part, we first prove that when $m \in [1, t-1]$, we have $\sum_{k=1}^{m} G_{tk} E = \prod_{\tau=m+1}^{t} w_t E$ by mathematical induction .

Initial step. When $m = 1$, $\sum_{k=1}^{m} G_{tk}E = G_{t1}E = \prod_{\tau=2}^{t} w_\tau E$, which satisfies the assertion.

Inductive step. Assume there is a $n \in [1, t-1]$, such that $\sum_{k=1}^{n} G_{tk}E = \prod_{\tau=n+1}^{t} w_t E$, we must prove the formula is true for $n + 1 \in [1, t-1]$.

$$
\begin{aligned}
\sum_{k=1}^{n+1} G_{tk}E &= \sum_{k=1}^{n} G_{tk}E + G_{t(n+1)}E \\
&= \prod_{\tau=n+1}^{t} w_\tau E + \prod_{\tau=n+2}^{t} w_\tau \\
&= \prod_{\tau=n+2}^{t} w_\tau \left[(w_{n+1} + I - d_{n+1}) E \right].
\end{aligned}
\tag{5}
$$

According to the formulation of the diagonal matrix in Eq. (2) we have $\sum_{k=1}^{n+1} G_{tk}E = \prod_{\tau=n+2}^{t} w_\tau E$. Therefore, the assertion is satisfied. When $m = t$, we have:

$$
\begin{aligned}
\sum_{k=1}^{t} G_{tk}E &= \sum_{k=1}^{t-1} G_{tk}E + G_{tt}E \\
&= \prod_{\tau=t}^{t} w_\tau E + \lambda_t E \\
&= w_\tau E + (I - d_t) E = E,
\end{aligned}
\tag{6}
$$

which yields the equivalence of Theorem 1. □

Theorem 2. *We define the evolution of a 2D matrix as a time sequence $\{U\}_T$, where $U(T = 1) = U_1$ is the initial state. When the transformation between any two adjacent states follows Eq. (3), the sequence is a diffusion process expressed with a partial differential equation (PDE):*

$$
\partial_T U = -LU
\tag{7}
$$

where $L = D - A$ is the Laplacian matrix, D is the degree matrix composed of d_t in Eq. (2), and A is the affinity matrix composed by the off-diagonal elements of G.

Proof. We substitute the X and H as two consecutive matrices U_{T+1} and U_T in (3). According to Theorem 1, we ensure that the sum of each row $I - G$ is 0 that can formulate a standard Laplacian matrix. Since G has the diagonal sub-matrix $I - d_t$, we can rewrite (3) as:

$$
\begin{aligned}
U_{T+1} &= (I - D + A) U_T \\
&= (I - L) U_T
\end{aligned}
\tag{8}
$$

where $G = (I - D + A)$, D is an $N \times N$ diagonal matrix containing all the d_t and A is the off-diagonal part of G. It then yields $U_{T+1} - U_T = -LU_T$, a discrete formulation of (7) with the time discretization interval as one. □

Theorem 2 shows the essential property of the row/column-wise linear propagation in Eq. (1): it is a standard diffusion process where L defines the spatial propagation and A, the affinity matrix, describes the similarities between any two points. Therefore, learning the image affinity matrix A in Eq. (8) is **equivalent** to learning a group of transformation matrices w_t in Eq. (1).

In the following section, we show how to build the spatial propagation (1) as a differentiable module that can be inserted into a standard feed-forward neural network, so that the affinity matrix A can be learned in a data-driven manner.

3.2 Learning Data-Driven Affinity

Since the spatial propagation in Eq.(1) is differentiable, the transformation matrix can be easily configured as a row/column-wise fully-connected layer. However, we note that since the affinity matrix indicates the pairwise similarities of a specific input, it should also be conditioned on the content of this input (*i.e.*, different input images should have different affinity matrices). Instead of

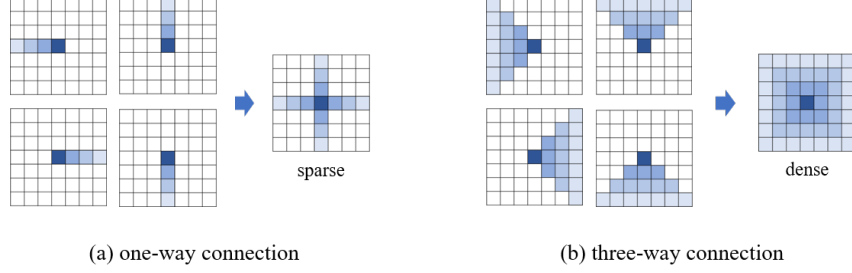(a) one-way connection   (b) three-way connection

Figure 1: Different propagation ranges for (a) one-way connections; and (b) three-way connections. Each pixel (node) receives information from a single line with one-way connection, and from a 2 dimensional plane with three-way connection. Integration of four directions w.r.t. (a) results in global, but sparsely connected pairwise relations, while (b) formulates global and densely connected pairwise relations.

setting the w_t matrices as fixed parameters of the module, we design them as the outputs of a deep CNN, which can be directly conditioned on an input image.

One simple way is to set the output of the deep CNN to use the same size as the input matrix. When the input has c channels (*e.g.*, an RGB image has $c = 3$), the output needs $n \times c \times 4$ channels (there are n connections from the previous row/column per pixel per channel, and with four different directions). Obviously, this is too many (*e.g.*, an $128 \times 128 \times 16$ feature map needs an output of $128 \times 128 \times 8192$) to be implemented in a real-world system. Instead of using full connections between the adjacent rows/columns, we show that certain local connections, corresponding to a sparse row/column-wise transform matrix, can also formulate a densely connected affinity. Specifically, we introduce the (a) one-way connection and the (b) three-way connection as two different ways to implement Eq. (1).

One-way connection. The one-way connection enables every pixel to connect to only one pixel from the previous row/column (see Figure 1(a)). It is equivalent to an one dimensional (1D), linear recurrent propagation that scans each row/column independently as an 1D sequence. Following Eq. (1), we denote $x_{k,t}$ and $h_{k,t}$ as the kth pixels in the tth column, where the left-to-right propagation for one-way connection is:

$$h_{k,t} = (1 - p_{k,t}) \cdot x_{k,t} + p_{k,t} \cdot h_{k,t-1}, \tag{9}$$

where p is a scaler weight indicating the propagation strength between the pixels at $\{k, t-1\}$ and $\{k, t\}$. Equivalently, w_t in Eq. (1) is a diagonal matrix, with the elements constituted by $p_{k,t}, k \in [1, n]$.

The one-way connection is a direct extension of sequential recurrent propagation [8, 31, 13]. The exact formulation of Eq. (9) has been used previously for semantic segmentation [4] and for learning low-level vision filters [19]. In [4], Chen *et al.*explain it by domain transform, where in semantic segmentation, p corresponds to the object edges. Liu *et al.* [19] explain it by arbitrary-order recursive filters, where p corresponds to more general image properties (*e.g.*, low-level image/color edges, missing pixels, etc.). Both of these can be explained as the same linear propagation framework of Eq. (1) with one-way connection.

Three-way connection. We propose a novel three-way connection in this paper. It enables each pixel to connect to three pixels from the previous row/column, *i.e.*, the left-top, middle and bottom pixels from the previous column for the left-to-right propagation direction (see Figure. 2(b)). With the same notations, we denote $\mathbb{N}$ as the set of these three pixels. Then the propagation for the three-way connection is:

$$h_{k,t} = \left(1 - \sum_{k \in \mathbb{N}} p_{k,t}\right) x_{k,t} + \sum_{k \in \mathbb{N}} p_{k,t} h_{k,t-1} \tag{10}$$

Equivalently, w_t forms a tridiagonal matrix, with $p_{:,k}, k \in \mathbb{N}$ constitute the three non-zero elements of each row/column.

Relations to the affinity matrix. As introduced in Theorem 2, the affinity matrix A with linear propagation is composed of the off-diagonal elements of G in Eq. (3). The one-way connection formulates a spares affinity matrix, since each sub-matrix of A has nonzero elements only along its

5

diagonal, and the multiplication of several individual diagonal matrices will also results in a diagonal matrix. On the other hand, the three-way connection, also with a sparse w_t, can form a relatively dense A with the multiplication of several different tridiagonal matrices. It means pixels can be densely and globally associated, by simply increasing the number of connections of each pixel during spatial propagation from one to three. As shown in Figures 2(a) and 2(b), the propagation of one-way connections is restricted to a single row, while the three-way connections can expand the region to a triangular 2D plane with respect to each direction. The summarization of the four directions result in dense connections of all pixels to each other (see Figure. 2(b)).

Stability of linear propagation. Model stability is of critical importance for designing linear systems. In the context of spatial propagation (Eq. 1), it refers to restricting the responses or errors that flow in the module from going to infinity, and preventing the network from encountering the vanishing of gradients in the backpropagation process [37]. Specifically, the norm of the temporal Jacobian $\partial h_t \setminus \partial h_{t-1}$ should be equal to or less than one. In our case, it is equivalent to regularizing each transformation matrix w_t with its norm satisfying

$$\|\partial h_t \setminus \partial h_{t-1}\| = \|w_t\| \leq \lambda_{max}, \tag{11}$$

where λ_{max} denotes the largest singularity value of w_t. This condition, $\lambda_{max} \leq 1$ provides a sufficient condition for stability. In the supplementary material, we show the requirements with respect to the elements in w_t.

Theorem 3. *Let* $\left\{p_{t,k}^K\right\}_{k \in \mathbb{N}}$ *be the weight in* w_t*, the model can be stabilized if* $\sum_{k \in \mathbb{N}} \left|p_{t,k}^K\right| \leq 1$.

Proof. See supplementary material. Let λ be the eigenvalue of matrix w_t and λ_{max} be the largest one. According to Gershgorin's Theorem [7], where every eigenvalue of a square matrix w_t satisfies:

$$|\lambda - p_{t,t}| \leq \sum_{k=1, k \neq t}^{n} |p_{k,t}|, \quad t \in [1, n] \tag{12}$$

then $|\lambda - p_{t,t}| + |p_{t,t}| \leq \sum_{k=1}^{n} |p_{k,t}|$. According to the triangle inequality, and since $\sum_{k=1, t \neq k}^{n} |p_{k,t}| \leq 1$, we have

$$\lambda_{max} \leq |\lambda - p_{t,t}| + |p_{t,t}| \leq \sum_{k=1}^{n} |p_{k,t}| \leq 1 \tag{13}$$

which satisfies the model stability condition. □

Theorem 3 shows that the stability of a linear propagation model can be maintained by regularizing the all weights of each pixel in the hidden layer H, with the summation of their absolute values less than one. For the one-way connection, Chen *et al.* [4] limited each scalar output p to be within $(0, 1)$. Liu *et al.* [19] extended the range to $(-1, 1)$, where the negative weights showed preferable effects for learning image enhancers. It indicates that the affinity matrix is not necessarily restricted to be positive/semi-positive definite (*e.g.*, the setting is also applied in [16].) For the three-way connection, we simply regularize the three weights (the output of a deep CNN) according to Theorem 3 without restriction to be any positive/semi-positive definite.

4 Implementation

We describe the implementation of the three-way connection-based network. We specify two separate branches: (a) a deep CNN, namely the guidance network that outputs all elements of the transformation matrix, and (b) a linear propagation module that outputs the propagation result (see Figure 2). The structure of a guidance network can be any regular deep CNN, which is designed for the task at hand. Examples of this network are described in Section 5. The propagation module receives an input map and output a refined or transformed result. It also takes the weights learned by the deep CNN guidance network as the second input.

The guidance network takes, as input, any 2D matrix that can help with learning the affinity matrix (*e.g.*, typically an RGB image). It outputs all the weights that constitute the transformation matrix w_t.
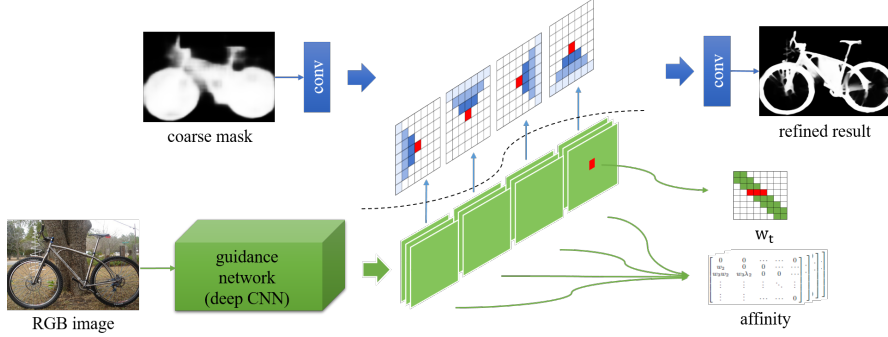
Figure 2: We illustrate the general architecture of the SPN using a three-way connection for segmentation refinement. The network, divided by the black dash line, contains a propagation module (upper) and a guidance network (lower). The guidance network outputs all entities that can constitute four affinity matrices, where each sub-matrix w_t is a tridiagonal matrix. The propagation module, being guided by the affinity matrices, deforms the input mask to a desired shape. All modules are differentiable and jointly learned via SGD.

The linear propagation module takes, as inputs, a 2D map that needs to be propagated (*e.g.*, a coarse segmentation mask), and the weights generated by the guidance network. Suppose that we have a map of size $n \times n \times c$ that is input into the propagation module, the guidance network needs to output a weight map with the dimensions of $n \times n \times c \times (3 \times 4)$, *i.e.*, each pixel in the input map is paired with 3 scalar weights per direction, and 4 directions in total. The propagation module contains 4 independent hidden layers for the different directions, where each layer combines the input map with its respective weight map using Eq. (10). All submodules are differentiable and jointly trained using stochastic gradient descent (SGD). We use node-wise max-pooling, similarly to [19], to integrate the hidden layers and to obtain the final propagation result.

5 Experimental Results

The SPN can be trained jointly with any segmentation model by being inserted on top of the last layer that outputs probability maps, or trained separately as a segmentation refinement model. In this paper we choose the second option. Given a coarse image segmentation mask as the input to the spatial propagation module, we show that the SPN can produce higher-quality masks with significantly refined details at object boundaries (see Figure 2). Many models [21, 5] generate low-resolution segmentation masks with coarse boundary shapes to seek a balance between computational efficiency and semantic accuracy. In specific, producing an original high-resolution segmentation mask usually requires the network to neither reduce the size of the input nor that of the output. In such settings, configuring a network with both sufficient capacity and a global receptive field is usually impractical due to the huge model size. Such problem is often solved in some sacrifice of the output resolution. The majority of work [21, 5, 36] choose to firstly produce an output probability map with $8\times$ smaller resolution, and then refine the result using either post-processing [5] or jointly trained modules [36]. It is a non-trivial task for producing high-quality segmentation results. In this work, we train only one SPN model for each task, and treat it as an universal refinement tool for different public available segmentation models.

We carry out the refinement of segmentation on two tasks: (a) generating high-resolution segmentation results on the HELEN face parsing dataset [27]; and (b) refining generic object segmentation on top of a pretrained model (e.g., VGG and ResNet based models [21, 5]. For the HELEN dataset, we directly use low-resolution RGB face images to train a baseline parser, which successfully catches the global semantic information. The SPN is then trained on top of the coarse segmentation to generate high-resolution output. For the Pascal VOC dataset, we train the SPN on top of the coarse segmentation results generated by the FCN-8s [21], and directly generalize it to any other pretrained model. We implement the network with a modified CAFFE [12]. The SPN is parallelized during propagating each row-column to the next one with CUDA. We used SGD optimizer, and set the base learning rate to 0.0001. In general, we train the HELEN and VOC segmentation tasks for about 40 and 100 epochs, respectively. The inference time (we do not use cuDNN) of SPN on HELEN and Pascal VOC is about 7ms and 84ms for an image of 512×512 resolution, respectively. In comparison, the dense CRF costs about 1s [14], 3.2s [5] and 4.4s [36] with different versions of publicly available

implementations (CPU only). We note that the majority time is spend on the guidance network, which can be accelerated by utilizing various existing network compressing strategies, applying smaller models, or sharing weights with the segmentation model if they are jointly trained. During inference, a single $64 \times 64 \times 32$ SPN hidden layer takes 1.3ms with the same computational settings.

General network settings. For the HELEN dataset, we train the SPN with smaller patches cropped from the original high-resolution images, their corresponding coarse segmentation maps produced by our baseline parser, and with the corresponding high-resolution ground-truth segmentation masks for supervision. All coarse segmentation maps are obtained by applying the baseline (for HELEN) or pre-trained (for Pascal VOC) image segmentation CNNs on their standard training splits [6, 5]. Since the baseline HELEN parser produces low-resolution segmentation results, we upsample them using a bi-linear filter to be of the same size as the desired higher output resolution. For the Pascal VOC dataset, we use the original output image segmentation probability maps produced by the pre-trained CNN models as its input. These CNN models contain upsampling layers, that typically upsample the internal feature representations by $8\times$ (*e.g.*, in [21, 5]) and produce output segmentation masks that are of the same size as that of the input images. We set the SPN as a patch refinement model on top of the coarse map with basic semantic information. We fix the size of our input patches to 128×128, use the *softmax* loss, and use the SGD solver for all the experiments. During training, the patches are sampled from image regions that contain more than one ground-truth segmentation label (*e.g.*, a patch with all pixels labeled as "background" will not be sampled). During testing for the VOC dataset, we restrict the classes in the refined results to be contained within the corresponding coarse input.

We combine the guidance network and the spatial propagation module similarly to [19]. We use two propagation units (*e.g.*, the bottom part in Figure. 2 is one propagation unit) with cascaded connections to achieve better results. Differently, we feed in the integrated hidden map of the first unit to the second unit, instead of cascading each direction separately and integrate them at the end of the second unit. We use two more convolutional layers with 32 channels before and after the propagation units to transfer the input map to an intermediate feature map, to make it compatible with the node-wise max-pooling. In addition, we maintain a smaller size of the propagation layer to make the model more efficient w.r.t computational speed and memory. This is carried out by bi-linearly downsampling/upsampling after the two convolutional layers, so that the hidden maps of propagation module is with a smaller dimension of 64×64. Note that to compare the one-way with the three-way connection, we use exactly the same structure except the propagation units. We do not apply any configuration used by [4] or [19].

HELEN Dataset. The HELEN dataset provides high-resolution photography-style face images (2330 in total), with high-quality manually labeled facial components including eyes, eyebrows, nose, lips, and jawline, which makes the high-resolution segmentation tasks applicable. All prevIoUs work utilize low-resolution parsing output as their final results for evaluation. Although many [27, 33, 20] achieve preferable performance, their results cannot be directly adopted by high-quality facial image editing applications. We use the setting that splits 100 samples for test following [33, 20]. We still take the hair region as one category, but do not evaluate it for fair comparisons with the state-of-the work [20]. We use similarity transformation according to the results of 5-keypoint detection [35] to align all face images to the center. Keeping the original resolution, we then crop or pad them to the size of 1024×1024.

We first train a baseline CNN with a symmetric downsample/upsample structure. The input image is $8\times$ downsampled from the original version. The downsampling part of the network is equipped with five consecutive conv+relu+max-pooling (with stride of 2) layers. Starting from 32, each one has double the number of channels, resulting in a $4 \times 4 \times 512$ feature maps at the bottleneck. In order to use the information at different levels of image resolution, we add skipped-links by summing features maps of the same dimensions from the corresponding upsample and dowsample layers. The upsample part has symmetric configurations, except that the max-pooling is replaced with bilinear upsampling, and the last sub-module has 11 channels for the 11 classes. We apply the multi-objective loss as [20] to improve the accuracy along the boundaries. We note that the symmetric structure is powerful, since the results we obtained for the baseline CNN are comparable (see Table. 1) to that of [20], who apply a much larger model (38 MB vs. 12 MB) in comparison. We then train a SPN on top of the baseline CNN results, with patches of input RGB image and coarse segmentations masks sampled from the preprocessed high-resolution image. For the guidance network, we use the same structure as that of the baseline segmentation network, except that its upsampling part ends at a resolution of 64×64, and its output layer has $32 \times 12 = 384$ channels. In addition, we train another

Table 1: Quantitative evaluation results on the HELEN dataset. We denote the upper and lower lips as "U-lip" and "L-lip", and overall mouth part as "mouth", respectively. The label definitions follow [20].

Method	skin	brows	eyes	nose	mouth	U-lip	L-lip	in-mouth	overall
Liu *et al.* [20]	90.87	69.89	74.74	90.23	82.07	59.22	66.30	81.70	83.68
baseline-CNN	90.53	70.09	74.86	89.16	83.83	55.61	64.88	71.72	82.89
Highres-CNN	91.78	71.84	74.46	89.42	81.83	68.15	72.00	71.95	83.21
SPN (one-way)	92.26	75.05	85.44	91.51	88.13	77.61	70.81	79.95	87.09
SPN (three-way)	**93.10**	**78.53**	**87.71**	**92.62**	**91.08**	**80.17**	**71.63**	**83.13**	**89.30**

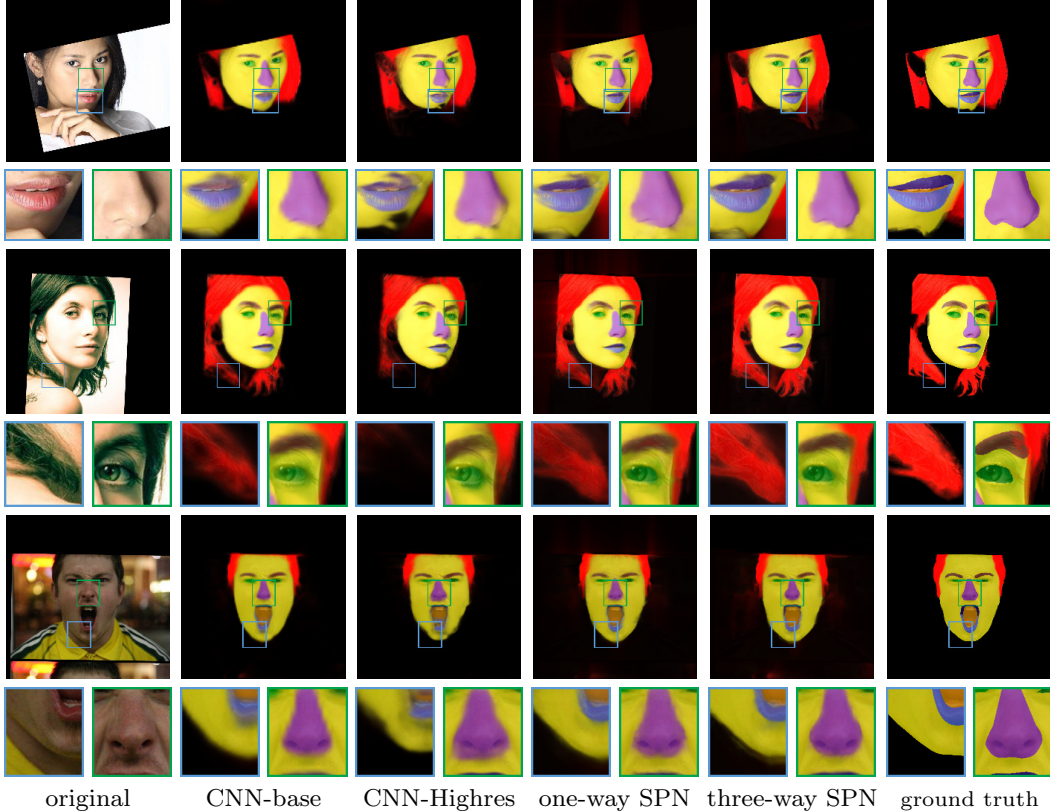original   CNN-base   CNN-Highres   one-way SPN   three-way SPN   ground truth

Figure 3: Results of face parsing on the HELEN dataset with detailed regions cropped from the high resolution images. (Images are all with high resolution and can be viewed by zoom-in.)

face parsing CNN with 1024×1024 sized inputs and outputs (CNN-Highres) for better comparison. It has three more sub-modules at each end of the baseline network, where all are configured with 16 channels to process higher resolution images.

We show quantitative and qualitative results in Table. 1 and 3 respectively. We compared the one/three way connection SPNs with the baseline, the CNN-Highres and the most relevant state-of-the-art technique for face parsing [20]. Note that the results of baseline and [20][1] are bi-linearly upsampled to 1024×1024 before evaluation. Overall, both SPNs outperform the other techniques with a significant margin of over 6 intersection-over-union (IoU) points, especially for the smaller facial components (*e.g.*, eyes and lips) where with smaller resolution images, the segmentation network performs poorly. We note that the one-way connection-based SPN is quite successful on relatively simple tasks such as the HELEN dataset, but fails for more complex tasks, as revealed by the results of Pascal VOC dataset in the following section.

Pascal VOC Dataset. The PASCAL VOC 2012 segmentation benchmark [6] involves 20 foreground object classes and one background class. The original dataset contains 1464 training, 1499 validation and 1456 testing images, with pixel-level annotations. The performance is mainly measured in terms

---

[1]The original output (also for evaluation) size it $250 * 250$.

| RGB | pretrained | with densecrf | with three-way SPN | ground truth |

Figure 4: Visualization of Pascal VOC segmentation results (left) and object probability (by $1 - P_b$, P_b is the probability of background). The "pretrained" denotes the base Deeplab ResNet-101 model, while the rest 4 columns show the base model combined with the dense CRF [5] and the proposed SPN, respectively.

Table 2: Quantitative evaluation results on the Pascal VOC dataset. We compare the two connections of SPN with the corresponding pre-trained models, including: (a) FCN-8s (F), (b) Deeplab VGG (V) and (c) Deeplab ResNet-101 (R). AC denotes accuracy, "+" denote added on top of the base model.

Model	F	+1 way	+3 way	V	+1 way	+3 way	R	+1 way	+3 way
overall AC	91.22	90.64	**92.90**	92.61	92.16	**93.83**	94.63	94.12	**95.49**
mean AC	77.61	70.64	**79.49**	80.97	73.53	**83.15**	84.16	77.46	**86.09**
mean IoU	65.51	60.95	**69.86**	68.97	64.42	**73.12**	76.46	72.02	**79.76**

of pixel IoU averaged across the 21 classes. We train our SPNs on the train split with the coarse segmentation results produced by the FCN-8s model [21]. The model is fine-tuned on a pre-trained VGG-16 network, where different levels of features are upsampled and concatenated to obtain the final, low-resolution segmentation results ($8\times$ smaller than the original image size). The guidance network of the SPN also fine-tunes the VGG-16 structure from the beginning till the *pool5* layer as the downsampling part. Similar to the settings for the HELEN dataset, the upsampling part has a symmetric structure with skipped links until the feature dimensions of 64×64. The spatial propagation module has the same configuration as that of the SPN that we employed for the HELEN dataset. The model is applied on the coarse segmentation maps of the validation and test splits generated by any image segmentation algorithm without fine-tuning. We test the refinement SPN on three base models: (a) FCN-8s [21], (b) the atrous spatial pyramid pooling (ASPP-L) network fine-tuned with VGG-16, denoted as Deeplab VGG, and (c) the ASPP-L: a multi-scale network fine-tuned with ResNet-101 [11] (pre-trained on the COCO dataset), denoted as Deeplab ResNet-101. Among them, (b) and (c) are the two basic models from [5], which are then refined with dense CRF [14] conditioned on the original image.

Table 2 shows that through the three-way SPN, the accuary of segmentation is significantly improved over the coarse segmentation results for all the three baseline models. It has strong capability of generalization and can successfully refine any coarse maps from different pretrained models by a large margin. Different with the Helen dataset, the one-way SPN fails to refine the segmentation, which is probably due to its limited capability of learning preferable affinity with a sparse form, especially when the data distribution gets more complex. Table 4 shows that by replacing the dense CRF module with the same refinement model, the perfor-

Table 4: Quantitative comparison (mean IoU) with dense CRF-based refinement [5] on Deeplab pre-trained models.

mIoU	CNN	+dense CRF	+SPN
VGG	68.97	71.57	**73.12**
ResNet	76.40	77.69	**79.76**

Table 3: Quantitative evaluation results on the Pascal VOC dataset. We refine the base models proposed with dilated convolutions [34]. "+" denote added on top of the "Front end" model.

Model	Front end	+3 way	+Context	+Context+3 way
overall AC	93.03	93.89	93.44	**94.35**
mean AC	80.31	83.47	80.97	**83.98**
mean IoU	69.75	73.14	71.86	**75.28**

mance is boosted by a large margin, without fine-tune. One the test split, the DeepNet ResNet-101 based SPN achieves the mean IoU of **80.22**, while the dense CRF gets 79.7. The three-way SPN produces fine visual results, as shown in the red bounding box of Figure 4. By comparing the probability maps (column 3 versus 7), SPN exhibits fundamental improvement in object details, boundaries, and semantic integrity.

In addition, we show in table 3 that the same refinement model can also be generalize to dilated convolution based networks [34]. It significantly improves the quantitative performance on top of the "Front end" base model, as well as adding a multi-scale refinement module, denoted as "+Context". Specifically, the SPN improves the base model with much larger margin compared to the context aggregation module (see "+3 way" vs "+Context" in table 3).

6 Conclusion

We propose spatial propagation networks for learning the affinity matrix for vision tasks. The spatial propagation network is a generic framework that can be applied to numerous tasks, and in this work we demonstrate the effectiveness in semantic ßobject segmentation. Experiments on the HELEN face parsing and PASCAL VOC object semantic segmentation tasks show that the spatial propagation network is general, effective and efficient for generating high-quality segmentation results.

| RGB | pretrained (R) | with dense CRF | with 3-way SPN | ground truth |

Figure 5: Based on model R, we visualization of Pascal VOC segmentation results (left) and object probability (by $1 - P_b$, where P_b denotes the probability of the background region).

References

[1] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr. Higher order conditional random fields in deep neural networks. In *ECCV*. Springer, 2016.

[2] G. Bertasius, L. Torresani, S. X. Yu, and J. Shi. Convolutional random walk networks for semantic image segmentation. *arXiv preprint arXiv:1605.07681*, 2016.

[3] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[4] L. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. *arXiv preprint arXiv:1511.03328*, 2015.

[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[6] M. Everingham, S. A. Eslami, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[7] S. Geršgorin. Uber die abgrenzung der eigenwerte einer matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 1931.

[8] A. Graves, S. Fernández, and J. Schmidhuber. Multi-dimensional recurrent neural networks. In *ICANN*, 549–558, 2007.

[9] K. He, J. Sun, and X. Tang. Single image haze removal using dark channel prior. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2341–2353, 2011.

[10] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1397–1409, 2013.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] N. Kalchbrenner, I. Danihelka, and A. Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[14] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[15] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics (ToG)*, 23(3):689–694, 2004.

[16] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008.

[17] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel. Deeply learning the messages in message passing inference. *arXiv preprint arXiv:1506.02108*, 2015.

[18] R. Liu, G. Zhong, J. Cao, Z. Lin, S. Shan, and Z. Luo. Learning to diffuse: A new perspective to design pdes for visual analysis. *IEEE transactions on pattern analysis and machine intelligence*, 38(12):2457–2471, 2016.

[19] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision*, 2016.

[20] S. Liu, J. Yang, C. Huang, and M.-H. Yang. Multi-objective convolutional learning for face labeling. In *CVPR*, 2015.

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[22] M. Maire, T. Narihira, and S. X. Yu. Affinity CNN: learning pixel-centric pairwise relations for figure/ground embedding. *CoRR*, abs/1512.02767, 2015.

[23] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[24] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[25] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

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

[27] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang. Exemplar-based face parsing. In *CVPR*, 2013.

[28] J. A. Suykens, J. D. Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48(1):85–105, 2002.

[29] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[30] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[31] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.

[32] J. Weickert. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

[33] T. Yamashita, T. Nakamura, H. Fukui, Y. Yamauchi, and H. Fujiyoshi. Cost-alleviative learning for deep convolutional neural network-based facial part labeling. *IPSJ Transactions on Computer Vision and Applications*, 7:99–103, 2015.

[34] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[35] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, 2014.

[36] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *IEEE International Conference on Computer Vision*, 2015.

[37] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*, 2016.