Abstract

Most current semantic segmentation methods rely on fully convolutional networks (FCNs). However, the use of large receptive fields and many pooling layers, cause blurring and low spatial resolution inside the deep layers, which often lead to spatially fragmented FCN predictions. In this work, we address this problem by introducing Convolutional Random Walk Networks (RWNs) that combine the strengths of FCNs and random walk based methods. Our proposed RWN jointly optimizes pixelwise affinity and semantic segmentation learning objectives, and combines these two sources of information via a novel random walk layer that enforces consistent spatial grouping in the deep layers of the network. We show that such a grouping mechanism improves the semantic segmentation accuracy when applied in the deep low spatial resolution FCN layers. Our proposed RWN fully integrates pixelwise affinity learning and the random walk process. This makes it possible to train the whole network in an end-to-end fashion with the standard back-propagation algorithm. Additionally, our RWN needs just 131 additional parameters compared to the state-of-the-art DeepLab network, and yet it produces an improvement of 1.5% according to the mean IOU evaluation metric on Pascal SBD dataset.

1 Introduction

The recent introduction of fully convolutional networks (FCNs) \[14\] has led to the significant developments in the semantic image segmentation task. Adopting the FCN approach, many subsequent methods have achieved even better performance \[5\, 20\, 6\, 9\, 12\, 5\, 20\, 6\, 13\, 16\]. However, at the core of their architecture, all of these methods still rely on traditional FCNs, which suffer from several limitations. Large receptive fields in the convolutional layers and the presence of pooling layers lead to low spatial resolution outputs in the deepest FCN layers. Since traditional FCNs do not have any explicit pixel grouping mechanism, pixel-level relationships inside the deepest layers become inconsistent and cause spatially fragmented outputs. In this work, we introduce a novel random walk based network architecture that incorporates a spatial grouping mechanism inside the network. We show that enforcing spatial consistency at low resolution deep layers improves the semantic segmentation accuracy compared to the traditional FCN based approaches.

Inspired by random walk methods \[15\, 1\, 17\], we propose a novel alternative to FCNs, which we refer to as a Random Walk Network (RWN). Our proposed RWN architecture jointly optimizes (1) pixelwise affinity and (2) semantic segmentation learning objectives. The two branches are linked via our proposed random walk layer, which enforces the spatial consistency in the deepest layer of
Figure 1: The architecture of our proposed RWN architecture (best viewed in color). RWN architecture is composed of three distinct parts: (1) the segmentation prediction branch, which is based on the state-of-the-art DeepLab [5], architecture, (2) the affinity learning branch, that takes pairwise L1 distances in the RGB, conv1_1, and conv1_2 space, and learns pixel-level affinities by minimizing the Euclidean loss to the ground truth similarity labels, and (3) a random walk layer that combines the information from the previous two branches and enforces spatial consistency among the pixels inside the deepest low-resolution network layer. During training, the entire network is jointly optimized for both affinity learning and semantic segmentation in an end-to-end fashion.

We introduce the new model of Random Walk Networks that combines the strengths of traditional FCNs and random walk based methods to enforce a consistent pixelwise grouping inside the deepest low resolution layers of the network.

Our RWN architecture jointly optimizes pixelwise affinity and semantic segmentation learning objectives, and combines the information from both branches via a novel random walk layer.

Contrary to the prior methods that use hand constructed color-based affinities, RWN automatically learns pixelwise affinities from low-level deep features, which we show to be more useful for semantic segmentation than the hand-crafted color affinities.

One of the benefits of our proposed RWN architecture is that the entire network can be jointly optimized in an end-to-end fashion using the traditional back-propagation algorithm, which as we show improves the overall semantic segmentation accuracy.

RWN needs just 131 more learning parameters than the base DeepLab [5] architecture and yet it outperforms DeepLab system by 1.5% according to the mean IOU metric on the Pascal SBD dataset [8]. The RWN learned unary potentials also lead to better globalization results.

2 Related Work

The recent introduction of the fully convolutional networks (FCNs) [14] has led to remarkable results in a semantic image segmentation task. However, due to the large receptive fields and many pooling layers, the FCN typically suffer from low spatial resolution predictions, which cause inconsistent relationships between the neighboring pixels inside the deep layers. Recently there has been some attempts to address these problems. All of this subsequent work can be divided into several groups.

The work in [3][5][18] used the FCN learned unary potentials, in the separate globalization framework to refine the original FCN results. One of the disadvantages related to these methods is that the learning of the unary potentials and the globalization framework is completely disjoint, and not integrated into the learning. To address this issue, several recent methods [20][13][12] proposed to integrate a CRF-like inference procedure into the network, which allowed to train such models in the end-to-end fashion and achieve CRF-like performance. However, one of the disadvantages of
these methods is a dramatically increased complexity of the model. For instance, the method in [20],
requires to cast the original FCN into a Recurrent Neural Network (RNN), which is much more
computationally expensive. The work in [13] proposes to use local convolutional layers, which lead
to a significantly larger number of overall parameters. Similarly, the method in [12] proposes to
model unary and pairwise potentials by the separate multi-scale network branches, which essentially
doubles the number of parameters compared to the traditional FCN architectures. An increased
number of parameters can be especially problematic, since the number of parameters in the traditional
FCN is already relatively high (∼ 15M).

In addition to the above methods it is also worth mentioning the deconvolution networks [16, 9],
which use deconvolution and unpooling layers to recover the fine object details from the coarse FCN
predictions in the end-to-end fashion. However, in order to effectively recover the fine details one
must employ almost the same number of deconvolution layers as there are convolution layers, which
results in an largely increased number of parameters as well.

All of the above methods utilize traditional FCN at the core of their architecture. While these methods
have been able to circumvent the spatial inconsistency problem inside the deep layers of traditional
FCNs, their solution did not fix the core problem; i.e. the lack of a consistent grouping mechanism in
the deep low spatial resolution layers inside the network. Our main goal, in this work, is to show that
such a spatial grouping mechanism can improve semantic segmentation results.

To address this problem, we propose a novel RWN approach that enforces consistent pixel grouping in
the deepest network layer via the means of a novel random walk layer. To the best of our knowledge,
we are the first to present a model that combines the strengths of the FCNs and the random walk
process for the semantic segmentation task. Our proposed RWN architecture jointly optimizes pixel
affinity and semantic segmentation learning objectives, which are then integrated via the random
walk layer. One of the advantages of our RWN architecture is that all the operations in every layer
are differentiable, and unlike the methods in [3, 5, 18], our entire network can be optimized via the
standard back-propagation algorithm in the end-to-end fashion.

Additionally, unlike the methods employing a large number of learnable parameters [20] [13] [12] [16] [9],
our RWN needs only 131 additional parameters compared to the DeepLab [5] architecture. This
number of extra parameters comprises just 0.0008% of the overall network of a DeepLab network and
yet it outperforms DeepLab by 1.5% according to the mean IOU metric on Pascal SBD dataset [8].
Additionally, in the experimental section, we show that compared to the DeepLab learned unary
potentials, RWN learned unary potentials are more beneficial for various globalization techniques.

3 Background: Random Graph Walks

Random walks are one of the most widely known and used methods in graph theory [15]. Most
notably, the concept of random walks led to the development of PageRank [17] and Personalized
PageRank [11] algorithms, which are still widely used for many applications. Let \( G = (V, E) \) denote
an undirected graph with a set of vertices \( V \) and a set of edges \( E \). Then a random walk in such graph
can be characterized by the transition probabilities between its vertices. If there exists an edge \( E_{ij} \)
connecting the vertices \( i \) and \( j \), then the transition probability between these two vertices can be
written as \( a_{ij} = \frac{1}{d_i} \) where \( d_i \) denotes the degree of vertex \( i \). If there is no edge between these nodes
the transition probability between them is set to 0.

Now, let \( W \) be the symmetric \( n \times n \) affinity matrix, where \( W_{ij} \in [0, 1] \) denotes how similar the
nodes \( i \) and \( j \) are, and where \( n \) denotes the number of nodes in the graph. Then let \( D \) indicate a
diagonal \( n \times n \) matrix, which stores the degree values for each node: \( D_{ii} = \sum_{j=1}^{n} W_{ij} \) for all \( j \)
except \( i \neq j \). Then we can express our random walk transition matrix as \( A = D^{-1}W \).

Given this setup, the main goal is to model how the information in the graph spreads if we start at
a particular node, and perform a random walk on this graph. Let \( y_t \) be a \( n \times 1 \) vector denoting a
node distribution at time \( t \). In the context of PageRank algorithm, \( y_t \) may indicate the rank estimates
associated with each of the \( n \) web pages at time \( t \). Then, according to the random walk theory, we
could spread the rank information in the graph by performing a one step random walk process. This
process could be expressed as \( y_{t+1} = Ay_t \), where \( y_{t+1} \) denotes a newly obtained rank distribution
after one random walk step, the matrix \( A \) indicates random walk transition probabilities, and \( y_t \)
illustrates the rank distribution at time step \( t \). Thus, we can observe, that the information among
Figure 2: A figure illustrating the probability maps corresponding to the majority object class by DeepLab (DL-MSC) and our RWN methods (best viewed in color). Note that the RWN diffuses the segmentation information across the regions corresponding to the entire object, which leads to spatially smoother probabilistic maps than DeepLab predictions.

the nodes can be diffused, by simply multiplying the random walk transition probability matrix $A$, with the rank distribution $y_t$ at a particular time $t$. This process can be repeated multiple times, until convergence is reached. For a more detailed survey please see [15, 17].

4 Convolutional Random Walk Networks

In the context of a semantic segmentation problem, each pixel in the image can be viewed as a separate node in the graph, where the similarity between two nodes can be evaluated according to some arbitrary metric (e.g. color, texture, etc). In this work, we aim to integrate a random walk process and the FCN architecture to diffuse the semantic segmentation predictions among the pixels that are similar to each other. Such a process introduces an explicit grouping mechanism, which helps to maintain spatial consistency among the pixels inside the low-resolution deep layers of the network. Such a scheme also makes the network predictions more stable, because each prediction is made considering the information from the neighboring nodes as well.

Our proposed approach can be broadly decomposed into three parts: (1) a branch producing semantic segmentation potentials (2) a branch learning pixel-level similarity metric, and (3) a mechanism to diffuse the semantic segmentation information among the nodes. For the semantic segmentation branch, we employ the state-of-the-art DeepLab-LargeFOV FCN architecture [5]. To learn a pixelwise affinity metric, RWN uses a separate affinity branch that optimizes a pixel-level affinity learning criterion. The semantic segmentation and the affinity branches are then combined via the random walk layer, which encourages spatial consistency among similar pixels. The entire RWN can be jointly optimized in an end-to-end fashion, which improves the semantic segmentation accuracy. The detailed illustration of our proposed RWN architecture is presented in Fig. 1. We now describe each of the components of the RWN architecture in more detail.

4.1 Semantic Segmentation Branch

For the semantic segmentation branch, we use a the state-of-the-art DeepLab-LargeFOV FCN architecture [5], which we refer to as DeepLab. DeepLab is a fully convolutional adaptation of the VGG [19] architecture, which contains 16 convolutional layers. It has been shown to yield strong semantic segmentation results [5]. However, due to the use of large receptive fields in the convolutional layers and several pooling layers, the final segmentation predictions are made at a resolution that is 8 times lower than the original image. Such a dramatic decrease in the resolution exacerbates the problem of spatial inconsistency inside the deepest network layers.

We note that even though we use a DeepLab architecture in our experiments, other architectures could be integrated into our framework as well. Most current semantic segmentation methods rely on the VGG based FCN component, which exhibits spatial inconsistency problem in its deep layers. Thus, in this work, our main goal is to show, that enforcing spatial consistency in the low spatial resolution layers can improve the overall semantic segmentation performance.
Figure 3: A figure illustrating semantic segmentation results. Images in columns two and three represent DeepLab-msc and RWN predictions respectively. Compared to the DeepLab system, RWN does not have as many false positive predictions. Also observe that in the second row, RWN correctly identifies the sofa on the right, which was missed by a human annotator. While RWN correctly captures the entire sofa, DeepLab incorrectly labels parts of it as a ‘chair’.

4.2 Pixel-Level Affinity Branch

To learn the pairwise pixel-level affinities, we employ a separate affinity learning branch with its own learning objective (See Fig. 1). The affinity branch is connected with the input RGB image, and low-level conv\textsubscript{1\_1} and conv\textsubscript{1\_2} layers. The feature maps corresponding to these layers have the same height and width dimensions but a different number of channels (3, 64, and 64 respectively). We then apply pooling to all of these representations with a stride equal to \( z \). In this case, we want the pooled height and width to match the height and width of the activations inside the fc8 layer of the network, which is 8 times lower than the original input image. Thus, we set \( z = 8 \). The pooled RGB, conv\textsubscript{1\_1}, and conv\textsubscript{1\_2} are then concatenated to obtain a new representation \( C \), which has dimensions \( n \times n \times k \), where \( n \) depicts the height and width after the pooling, and \( k \) denotes the number of concatenated channels. Then, let \( F \) be a sparse \( n^2 \times n^2 \times k \) matrix that stores L1 distances between the individual \( n^2 \) pixels and all of their neighbors within the radius \( R \). Note that the distances are not summed up across all \( k \) channels, but are rather stored separately for each channel. The resulting matrix \( F \) is then used as an input to the affinity branch as illustrated in Figure 1.

The affinity branch consists of a \( 1 \times 1 \times 1 \) convolutional layer and an exponential layer. The output of the exponential layer is then attached to the Euclidean loss layer and is optimized to predict the ground truth pixel affinities, which are obtained from the original semantic segmentation annotations. Note that \( F \), which is used as an input to the affinity branch, is a sparse matrix, as only a small fraction of all the entries in \( F \) are populated with non-zero entries. The rest of the entries are ignored during the computation. Also note that we only use features from RGB, conv\textsubscript{1\_1} and conv\textsubscript{1\_2} layers, because they are not affected by the pooling, and thus, preserve the original spatial layout.

4.3 Random Walk Layer

To integrate the semantic segmentation potentials and our learned pixel-level affinities, we introduce a novel random walk layer, which diffuses the semantic segmentation information based on the learned affinities. The random walk layer is connected to the two bottom layers: (1) the fc8 layer containing the semantic segmentation potentials, and (2) the affinity layer that outputs a sparse \( n^2 \times n^2 \) row-normalized affinity matrix \( \hat{A} \) (also known as the random walk transition matrix). Then, let \( f \) denote the activation values from the fc8 layer, reshaped to the dimensions of \( n^2 \times m \), where \( n^2 \) refers to the number of activation values per channel inside the fc8 layer, and \( m \) is the number of object classes in the dataset. A single random walk layer implements one step of the random walk process, which can be performed as \( \hat{y} = \hat{A}f \), where \( \hat{y} \) indicates the diffused segmentation predictions, and \( \hat{A} \) denotes the random walk transition matrix. In order to simulate \( t \) random walk steps, we can simply substitute \( A \) with \( \hat{A}^t \), where \( \hat{A}^t \) denotes \( A \) raised to the power of \( t \).

The random walk layer is then attached to the softmax loss layer, and is optimized to predict ground truth semantic segmentations. One of the advantages of our proposed random walk layer is that it is implemented as a matrix multiplication, which makes it possible to back-propagate the gradients to both (1) affinity and (2) segmentation branches. Let the softmax-loss gradient be an \( n^2 \times m \) matrix
Table 1 lists the semantic segmentation results, which indicate that our RWN method consistently outperforms DeepLab-LargeFOV baseline by 1.5% and 1.4% according to per-class mean IOU, and overall IOU metrics in the single-scale setting respectively. Additionally, RWN outperforms the DeepLab system by 0.7% according to both metrics in the multi-scale setting.

$L$, where $n^2$ is the number of pixels in the fc8 layer, and $m$ is the number of predicted object classes. Then the gradients, which are back-propagated to the semantic segmentation branch are computed as $G_{ss} = A^T L$, where $A^T$ is the transposed random walk transition matrix. Also, the gradients, that are back-propagated to the affinity branch are computed as $G_{aff} = L f^T$, where $f^T$ is a $m \times n^2$ matrix that contains transposed activation values from the fc8 layer of the segmentation branch. We note that $G_{aff}$ is a sparse $n^2 \times n^2$ matrix, which means that the above matrix multiplication only considers the pixel pairs that correspond to the non-zero entries in the random walk transition matrix $A$.

### 4.4 Implementation Details

We jointly train our RWN for the affinity learning and semantic segmentation tasks, in an end-to-end fashion for about 2000 iterations, with a learning rate of $10^{-5}$, 0.9 momentum, the weight decay of $5 \cdot 10^{-5}$, and 15 samples per batch. For the RWN parameters, we use the random walk transition matrix connectivity radius equal to $R = 5$. This means that every pixel is connected to its neighbors that are within the distance of $R = 5$. For all of our experiments, we use a Caffe library [10]. It takes about 9 hours to train the network on NVIDIA GeForce GTX Titan X GPU. The training is done on a Pascal SBD dataset [8], which contains 8055 training and 2857 testing images. We employ data augmentation techniques such as cropping, and mirroring. Testing a single $514 \times 514 \times 3$ image takes about 1 second. However, we believe that both training and testing could be sped up considerably. Currently, the Caffe library does not support sparse matrix operations, and thus, we used external sparse matrix library Eigen [7]. Allocating the memory for the sparse matrix creation ($\approx 350K$ non-zero entries per matrix) is the main bottleneck, and we believe that with a more efficient sparse matrix library, our method could run almost as fast as the original FCNs.

### 5 Experimental Results

In this section, we present our results for semantic segmentation on the SBD [8] dataset, which contains objects and their per-pixel annotations for 20 Pascal VOC classes (excluding the background class). We evaluate semantic segmentation results using the standard metric of the pixel intersection-over-union (IOU) averaged per pixels across the 20 classes. We also include the class-agnostic overall pixel intersection-over-union score, which measures the per-pixel IOU across all 20 classes.

Since our segmentation branch is based on the DeepLab-LargeFOV [5] architecture, we use DeepLab system as the main baseline throughout our experiments. We also run experiments with a slightly more advanced multi-scale DeepLab architecture. In Subsection 5.1, we show that our proposed RWN architecture consistently outperforms both DeepLab baselines. In Subsection 5.2, we compare the DeepLab and RWN learned unary potentials in the context of 3 globalization methods, and demonstrate that RWN learned unary potentials lead to higher segmentation accuracy. Finally, in Subsection 5.3, we perform a number of ablation experiments related to our RWN architecture.

#### 5.1 Comparing the Learned Segments

Table 1 lists the semantic segmentation results, which indicate that our RWN method consistently outperforms both single-scale and multi-scale DeepLab baselines. In the single-scale setting RWN outperforms DeepLab system by 1.5% and 1.4% according to the mean per class IOU, and overall IOU metrics respectively, achieving better accuracy in 19 out of 20 object classes. We also observe that RWN outperforms DeepLab by 0.7% according to both metrics in the multi-scale setting.

In addition to the quantitative, we also present qualitative results in Figures 2 and 3. In Figure 2, we compare the probabilistic maps of a majority object class produced by the RWN and DeepLab-MSC.
networks. Note that RWN probabilistic maps are spatially smoother across the regions that correspond to the objects whereas DeepLab often produces spatially fragmented predictions.

In Figure 3, we also compare the final segmentation predictions. Based on these qualitative results, we observe, that RWN produced segmentation do not produce as many false positives as the DeepLab system. Additionally, we also notice that RWN produces spatially consistent segmentations, whereas DeepLab may assign separate parts of the objects to different labels.

5.2 Comparing the Performance in the Globalization Framework

In the previous subsection, we showed that our introduced random walk grouping mechanism improves the semantic segmentation accuracy compared to the DeepLab alternative. However, since the random walk layer is applied in the layer that exhibits such a low resolution (8 times lower than the original image), there still may be a need to do some post-processing afterwards.

In this section, we use RWN and DeepLab learned potentials in the framework of three globalization techniques, which include Dense-CRF [11], Boundary Neural Fields (BNF) [3], and Graph Cuts [4]. DenseCRF employs a fully connected CRF, with color-based affinities to refine the initial segmentation predictions. The BNF is a spectral boundary-based method that refines the initial segmentations. Finally, the Graph Cuts method employs a max-flow based approach to segment the objects, given the initial unary potentials and arbitrary pixel-level similarity metric. For both BNF and Graph Cuts methods, we use HFL boundaries [2], which were shown to work well for object segmentation tasks. In Table 2, we list these results, which suggest that RWN learned unary potentials produce better results than the DeepLab learned potentials.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean IOU</th>
<th>Overall IOU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>74.01</td>
<td>75.26</td>
</tr>
<tr>
<td>Dense-CRF [11]</td>
<td>75.79</td>
<td>76.20</td>
</tr>
<tr>
<td>BNF [3]</td>
<td>76.30</td>
<td>76.32</td>
</tr>
<tr>
<td>Graph Cuts [4]</td>
<td>75.25</td>
<td>76.39</td>
</tr>
</tbody>
</table>

Table 2: Globalization results when using RWN versus DeepLab learned unary potentials. The results are evaluated using the per class mean IOU, and overall IOU metrics. We show that in all cases, RWN learned unary potentials lead to better globalization results than the DeepLab learned potentials.

5.3 Ablation Experiments

In Table 3, we quantitatively analyze the design factors of our RWN architecture. The checkmarks indicate the components that were used in the particular experiment, whereas the columns illustrate the specific experiments and their corresponding quantitative performance. All the results are evaluated using the same mean IOU, and overall IOU metrics for semantic segmentation.

<table>
<thead>
<tr>
<th>Affinity Learning</th>
<th>Gaussian Affinity</th>
<th>conv1_1 Feats</th>
<th>conv1_2 Feats</th>
<th>Random Walk</th>
<th>Joint Training</th>
<th>mean class IOU</th>
<th>overall pixel IOU</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>69.20</td>
<td>71.19</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>73.81</td>
<td>74.21</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>74.27</td>
<td>74.31</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>74.88</td>
<td>74.95</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>75.26</td>
<td>75.36</td>
</tr>
</tbody>
</table>

Table 3: We present our experiments with different RWN architectures. The checkmarks indicate the components that were used in the particular experiment, whereas the columns depict the actual experiments and their corresponding quantitative performance. All the results are evaluated using the same mean IOU, and overall IOU metrics for semantic segmentation.

Is the Affinity Learned by RWN Better than the Hand-Crafted Color Affinities? Most of the methods that require pixel-level affinities still rely on hand-crafted Gaussian color affinities of the form \( \exp \left( \frac{(x-y)^2}{\sigma^2} \right) \), where \( x \) and \( y \) denote color-based pixel intensity values, and \( \sigma \) is a hand-picked smoothness parameter. In columns 3 and 8, we compare RWN results when using the hand-crafted Gaussian color affinity (where \( \sigma \) is selected via the validation) and when using the RWN learned affinity. The results show that RWN learned affinity is beneficial since it yields an improvement of 1.05%, and 1.07% over the hand-crafted Gaussian affinity according to the two metrics respectively.

Which Affinity Cues are Most Beneficial for Segmentation? The RWN affinity branch learns to predict pixel-level based affinities based on the RGB, \( conv1_1 \) and \( conv1_2 \) cues. In columns 4, 5, 7 we study how these affinity cues influence segmentation accuracy. First, we note that just using the
RGB cues to learn the affinity, still leads to slightly better performance than using the hand-crafted Gaussian color affinity (see columns 3 and 4). Second, when using the cues separately from just one of the RGB, $conv_1$ or $conv_2$ layers, we observe that using the cues from $conv_1$ leads to the best segmentation results, outperforming the other two types of cues by at least 0.64% and 0.71% according to the both metrics respectively. Finally, we note that combining all of these cues to learn the affinities further improves the results by 0.31% and 0.27% respectively (see columns 7 and 8).

**Is Affinity Learning Necessary?** The random walk layer in our RWN architecture is trained to predict semantic segmentation masks. However, since this layer gets one of its inputs from the affinity branch, the gradients from the segmentation loss objective, are also back-propagated to the affinity branch (See Figure 1). Therefore, the question becomes whether the affinity learning objective is necessary? In columns 1 and 8, we compare the RWN without and with the affinity learning objective. These results indicate that the affinity learning objective is a crucial component of our RWN architecture since it leads to an improvement of 6.06%, and 6.21% over the architecture without the affinity learning objective.

**Does Joint Training Help?** During the back-propagation, the random walk layer back-propagates the gradients to the segmentation and affinity branches. We want to quantify the contribution of the segmentation loss gradients to the affinity branch during the training process. Our final objective is to produce accurate segmentation maps. Thus, presumably the segmentation loss gradients should be beneficial to the affinity branch, as the affinities would be adjusted to minimize the segmentation loss. In order to validate this hypothesis, we consider an RWN training scheme, in which the segmentation loss gradients from the random walk layer are not back-propagated to the affinity branch. The results in columns 6 and 8 confirm our intuition: training two branches independently does not perform as well as the joint training scheme, where we back-propagate the softmax-loss gradients back to the affinity layer as well. These results suggest that the joint training strategy is beneficial as it yields the improvements of 0.38% and 0.35% respectively.

**How Many Random Walk Steps Are Necessary?** In Figure 4 we illustrate how the IOU accuracy changes when we use a different number of random walk steps. We observe that a single random walk step provides the best balance between the original network activations and our proposed random walk based grouping scheme.

![Accuracy versus the Number of Random Walk Steps](image)

Figure 4: The figure illustrating how the IOU accuracy changes as we apply more random walk steps. From the curve, we observe that a single random walk step provides the best balance between the original network activations and our proposed random walk based grouping scheme.

6 Conclusion

In this work, we introduced Random Walk Networks (RWNs), an effective alternative to traditional fully convolutional networks (FCNs). RWNs combine the strengths of traditional FCNs and the random graph walk process, by employing a spatial grouping mechanism, which is implemented via the means of a novel random walk layer. Unlike many popular methods, which employ hand-crafted color based pixel affinities, our proposed RWN jointly optimizes the affinity and semantic segmentation learning objectives. This allows it to learn affinities that are well suited for the semantic segmentation task. One of the advantages of our RWN architecture is that the entire network can be jointly optimized in an end-to-end fashion via the regular back-propagation algorithm, which improves the segmentation accuracy. Compared to the state-of-the-art DeepLab system, our RWN achieves an improvement of 1.5% at the cost of only 131 additional learnable parameters, which comprises only 0.0008% of original number of the parameters in the network. We also show that in

<table>
<thead>
<tr>
<th>Number of Random Walk Steps</th>
<th>IOU Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.735</td>
</tr>
<tr>
<td>1</td>
<td>0.74</td>
</tr>
<tr>
<td>2</td>
<td>0.745</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>0.755</td>
</tr>
<tr>
<td>5</td>
<td>0.76</td>
</tr>
<tr>
<td>6</td>
<td>0.765</td>
</tr>
<tr>
<td>7</td>
<td>0.77</td>
</tr>
<tr>
<td>8</td>
<td>0.775</td>
</tr>
</tbody>
</table>

6 Conclusion

In this work, we introduced Random Walk Networks (RWNs), an effective alternative to traditional fully convolutional networks (FCNs). RWNs combine the strengths of traditional FCNs and the random graph walk process, by employing a spatial grouping mechanism, which is implemented via the means of a novel random walk layer. Unlike many popular methods, which employ hand-crafted color based pixel affinities, our proposed RWN jointly optimizes the affinity and semantic segmentation learning objectives. This allows it to learn affinities that are well suited for the semantic segmentation task. One of the advantages of our RWN architecture is that the entire network can be jointly optimized in an end-to-end fashion via the regular back-propagation algorithm, which improves the segmentation accuracy. Compared to the state-of-the-art DeepLab system, our RWN achieves an improvement of 1.5% at the cost of only 131 additional learnable parameters, which comprises only 0.0008% of original number of the parameters in the network. We also show that in
comparison to the DeepLab learned unary potentials, RWN potentials lead to higher accuracy in the framework of various globalization approaches.

The main goal of this work was to show that our proposed random walk based spatial grouping mechanism can improve the segmentation accuracy when it is applied in the deep low resolution network layers. In this work, we used an RWN architecture that employs a single random walk layer at the end of the network. However, alternative network architectures with more random walk layers may produce an even better performance. We plan to investigate such architectures in future work.

References


