MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Deep Hierarchical Parsing for Semantic Segmentation

Sharma, A.; Tuzel, O.; Jacobs, D.

TR2014-125 March 2015

Abstract

This paper proposes a learning-based approach to scene parsing inspired by the deep Recursive Context Propagation Network (RCPN). RCPN is a deep feed-forward neural network that utilizes the contextual information from the entire image, through bottom-up followed by top-down context propagation via random binary parse trees. This improves the feature representation of every super-pixel in the image for better classification into semantic categories. We analyze RCPN and propose two novel contributions to further improve the model. We first analyze the learning of RCPN parameters and discover the presence of bypass error paths in the computation graph of RCPN that can hinder contextual propagation. We propose to tackle this problem by including the classification loss of the internal nodes of the random parse trees in the original RCPN loss function. Secondly, we use an MRF on the parse tree nodes to model the hierarchical dependency present in the output. Both modifications provide performance boosts over the original RCPN and the new system achieves state-of-the-art performance on Stanford Background, SIFT-Flow and Daimler urban datasets.

arXiv

Deep Hierarchical Parsing for Semantic Segmentation

Abhishek Sharma
Computer Science Department
University of Maryland
bhokaal@cs.umd.edu

Oncel Tuzel
MERL
Cambridge
oncel@merl.com

David W. Jacobs
Computer Science Department
University of Maryland
djacobs@umiacs.umd.edu

Abstract

This paper proposes a learning-based approach to scene parsing inspired by the deep Recursive Context Propagation Network (RCPN). RCPN is a deep feed-forward neural network that utilizes the contextual information from the entire image, through bottom-up followed by top-down context propagation via random binary parse trees. This improves the feature representation of every super-pixel in the image for better classification into semantic categories. We analyze RCPN and propose two novel contributions to further improve the model. We first analyze the learning of RCPN parameters and discover the presence of bypass error paths in the computation graph of RCPN that can hinder contextual propagation. We propose to tackle this problem by including the classification loss of the internal nodes of the random parse trees in the original RCPN loss function. Secondly, we use an MRF on the parse tree nodes to model the hierarchical dependency present in the output. Both modifications provide performance boosts over the original RCPN and the new system achieves state-of-the-art performance on Stanford Background, SIFT-Flow and Daimler urban datasets.

1. Introduction

Semantic segmentation refers to the problem of labeling every pixel in an image with the correct semantic category. Handling the immense variability in the appearance of semantic categories requires the use of context to achieve human-level accuracy, as shown, for example, by [24, 14, 13]. Specifically, [14, 13] found that human performance in labeling a super-pixel is worse than a computer when both have access to that super-pixel only. Effectively using context presents a significant challenge, especially when a *real-time* solution is required.

An elegant deep recursive neural network approach for semantic segmentation was proposed in [19], referred to as RCPN. The main idea was to facilitate the propagation of contextual information from each super-pixel to every other super-pixel through random binary parse trees. First, a *semantic mapper* mapped visual features of the super-pixels into a semantic space. This was followed by a recursive combination of semantic features of two adjacent image regions, using a *combiner*, to yield the holistic feature vector of the entire image, termed the root feature. Next, the global information contained in the root feature was disseminated to every super-pixel in the image, using a *decombiner*, followed by classification of each super-pixel via a *categorizer*. The parameters were learned by minimizing the classification loss of the super-pixels by back-propagation through structure [5]. RCPN was shown to outperform recent approaches in terms of per-pixel accuracy (PPA) and mean-class accuracy (MCA). Most interestingly, it was almost two orders of magnitude faster than competing algorithms.

RCPN's speed and state-of-the-art performance motivate us to carefully analyze it. In this paper we show that it still has some weaknesses and we show how to remedy them. In particular, the direct path from the semantic mapper to the categorizer gives rise to bypass errors that can cause RCPN to bypass the combiner and decombiner assembly. This can cause back-propogation to reduce RCPN to a simple multi-layer neural network for each super-pixel. We propose modifications to RCPN that overcome this problem

1. **Pure-node RCPN** - We improve the loss function by adding the classification loss of those internal nodes of the random parse trees that correspond to a single semantic category, referred to as pure-nodes. This serves three purposes. a) It provides more labels for training, which results in better generalization. b) It encourages stronger gradients deep in the network. c) Lastly, it tackles the problem of bypass errors, resulting in better use of contextual information.
2. **Tree MRF RCPN** - Pure-node RCPN also provides us with reliable estimates of the internal node label distributions. We utilize the label distribution of the internal nodes to define a tree-style MRF on the parse tree to model the hierarchical dependency between the nodes.

The resulting architectures provide promising improvements over the previous state-of-the-art on three semantic segmentation datasets: Stanford background [6], SIFT flow [11] and Daimler urban [16].

The next section describes some of the related works followed by a brief overview of RCPN in Sec. 3. We describe our proposed methods in Sec. 4 followed by experiments in Sec. 5. Finally, we conclude in Sec. 6.

2. Related Work

The previous work on semantic segmentation roughly follows two major themes: learning-based and non-parametric models.

Learning-based models learn the appearance of semantic categories, under various transformations, and the relations among them using parametric models. CRF based image models have been quite successful in jointly modeling the appearance and structure of an image; [6, 15, 14, 13] use CRFs to combine unary potentials obtained from the visual features of super-pixels with the neighborhood constraints. The differences among these approaches are mainly in terms of the visual features, form of the N-ary potentials and the the CRF modeling. A joint-CRF on multiple levels of an image segmentation hierarchy is formulated in [10]. It achieves better results than a flat-CRF owing to the utilization of higher order contextual information coming in the form of a segmentation hierarchy. Multi-scale convolution neural networks are used in [2] to learn visual feature extractors from raw-image/label training pairs. It achieved impressive results on various datasets using gPb, purity-cover and CRF on top of the learned features. It was extended in [17] by feeding in the per-pixel predicted labels using a CNN classifier to the next stage of the same CNN classifier. However, the propagation structure is not adaptive to the image content and only propagating label information did not improve much over the prior work.

A type of learning based model was proposed in [21] that aims at learning a mapping from the visual features to a semantic space followed by classification. The semantic mapping is learned by optimizing a structure prediction cost on the ground-truth parse trees of training images with the hope that such a training would embed the visual features in a semantically meaningful space, where classification would be easier. However, our experiments using the code provided by the authors show that semantic space mapping is actually no better than a simple 2-layer neural network on the visual features directly.

Recently, a lot of successful non-parametric approaches for natural scene parsing have been proposed [23, 11, 20, 4, 22, 25]. These approaches are instances of sophisticated template matching to retrieve images that are visually similar to the query, from a database of labeled images. The matching step is followed by super-pixel label transfer from the retrieved images to the query image. Finally, a structured prediction model such as CRF is used to jointly utilize the unary potentials with plausible image models. These approaches differ in terms of the retrieval of candidate images or super-pixels, transfer of label from the retrieved candidates to the query image, and the form of the structured prediction model. These approaches are based on nearest-neighbor retrieval that introduces a critical performance/accuracy trade-off. Theoretically, these approaches can utilize a huge amount of data with ever increasing accuracy. But a very large database would require large retrieval-time, which limits the scalability of these methods.

3. Background Material

In this section, we provide a brief overview of the RCPN based semantic segmentation framework, please refer to [19] for details.

3.1. Overview

RCPN formulates the problem of semantic segmentation as labeling each super-pixel into desired semantic categories. The complete pipeline starting from the input image to the final pixel-wise labels is shown in Fig. 1. It starts with the super-segmentation of the image followed by the extraction of visual features for each super-pixel; [19] used the Multi-scale CNN [2] to extract per pixel features that are then averaged over super-pixels. RCPN then constructs random binary parse trees obtained using the adjacency information between super-pixels. The leaf-nodes correspond to the initial super-pixels and successive random merger of two adjacent super-pixels builds the internal nodes up to the root node, which corresponds to the entire image. The super-pixel features along with a parse tree are passed through an assembly of four modules: (*semantic mapper*, *combiner*, *decombiner* and *categorizer*, in order) that outputs labels for each super-pixel. Multiple random parse trees can be used, both during training and testing. At test time, each parse tree can gives rise to different labels for the same super-pixel, therefore, voting is used to decide the final label.

Notation: Throughout this article - $\mathbf{v}_i$ denotes visual features of i^{th} super-pixel, $\mathbf{x}_i$ denotes semantic feature of i^{th} super-pixel and $\tilde{\mathbf{x}}_i$ denotes enhanced super-pixel features.

Semantic mapper is a neural network that maps visual features of each super-pixel to a d_{sem} dimensional semantic feature

$$\mathbf{x}_i = F_{sem}(\mathbf{v}_i; W_{sem}) \tag{1}$$

here, F_{sem} is the network and W_{sem} are the layer weights.

Combiner: Combiner is a neural network that recursively maps two child node features (x_i and x_j) to their

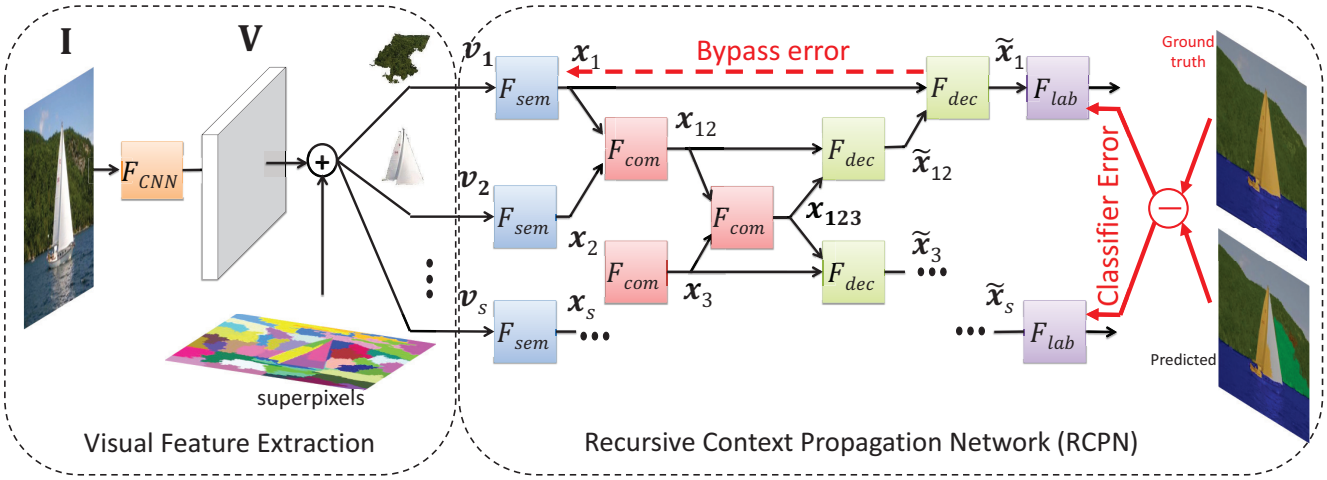


Figure 1: Complete flow diagram of RCPN for semantic segmentation.

parent feature ($x_{i,j}$). Intuitively, the combiner network attempts to aggregate the semantic content of the children features such that the parent's features become representative of the children. The root features represent the entire image.

$$\mathbf{x}_{i,j} = F_{com}([\mathbf{x}_i, \mathbf{x}_j]; W_{com}). \tag{2}$$

here, F_{com} is the network and W_{com} are the layer weights.

Decombiner is a neural network that recursively disseminates the context information from a parent node to its children through the parse tree. This network maps the semantic features of the child node and its parent to the contextually enhanced feature of the child node. This top-down contextual propagation starts from the root feature and the decombiner is applied recursively up to the enhanced super-pixel features. Therefore, it is expected that every super-pixel feature contains the contextual information aggregated from the entire image.

$$\tilde{\mathbf{x}}_i = F_{dec}([\mathbf{x}_i, \tilde{\mathbf{x}}_{i,j}]; W_{dec}). \tag{3}$$

here, F_{dec} is the network and W_{dec} are the layer weights.

Categorizer is the final network, which maps the context enhanced semantic features ($\tilde{\mathbf{x}}_i$) of each super-pixel to one of the semantic category labels; it is a Softmax classifier

$$\mathbf{y}_j = F_{cat}(\tilde{\mathbf{x}}_i; W_{cat}). \tag{4}$$

Together, all the parameters of RCPN are denoted as $W_{rcpn} = \{W_{sem}, W_{com}, W_{dec}, W_{cat}\}$. Let's assume there are S super-pixels in an image I and denote a set of R random parse trees of I as $\mathcal{T}$. Then, the loss function for I is

$$\mathcal{L}(I) = \frac{1}{RS} \sum_{r=1}^{R} \sum_{s=1}^{S_i} L(\mathbf{y}_{r,s}, t_s; \mathcal{T}_r, W_{rcpn}) \tag{5}$$

here, $\mathbf{y}_{r,s}$ is the predicted class-probability vector and t_s is the ground-truth label for the s^{th} super-pixel for random parse tree $\mathcal{T}_r$ and $L(\mathbf{y}_s, t)$ is the cross-entropy loss function. Network parameters, W_{rcpn}, are learned by minimizing $\mathcal{L}(I)$ for all the images in the training data.

4. Proposed Approach

In this section, we study the RCPN model, discover potential problems with parameter learning and propose useful modifications to the learning and the model. Our first modifications tackle a potential pitfall during training that stems from the special architecture of RCPN and can reduce it to a simple multi-layer NN. The second modification extends the model by building an MRF on top of the parse trees to utilize the hierarchical dependency between the nodes.

4.1. Pure-node RCPN

Here we propose a model that will handle bypass errors. At the same time, this model solves a problem of gradient attenuation, and also multiplies the training data. For the ease of understanding all our discussions will be limited to 1-layer modules. This result in each of the W_{sem}, W_{com}, W_{dec} and W_{cat} as matrices. Like most deep networks, RCPN also suffers from vanishing gradients for the lower layers. This stems from the vanishing error signal, because the gradient ($\mathbf{g}_l$) for the l^{th} layer depends on the error signal ($\mathbf{e}_{l+1}$) from the layer above -

$$\mathbf{g}_l = \mathbf{e}_{l+1}\mathbf{x}_l^T \tag{6}$$

here, $\mathbf{x}_l$ is the input to the l^{th} layer. For RCPN, vanishing gradients are more of a problem because of very deep parse trees due to recursion. For instance, a 100 super-pixel image will lead to a minimum of ($log_2(100) \times 2 + 2 > 14$) layers under the strong assumption of perfectly balanced binary parse trees. In practice, we can only create roughly balanced binary trees that often lead to ~ 30 layers.

We show that the internal nodes of the parse tree can be used to alleviate these problem. Each node in the parse tree corresponds to a connected region in the image. The leaf nodes correspond to the initial super-pixels and the internal nodes correspond to the merger of two or more connected regions, referred to as merged-region. We use the term *pure nodes* to refer to the internal nodes of the parse tree associated with the merger of two or more regions of the same semantic category. Therefore, the merged-regions corresponding to the pure nodes can serve as additional labeled samples during training. We empirically found that roughly 65% of all the internal nodes are pure-nodes for all three datasets. We include the classification loss of the pure-nodes in the loss function (Eqn. 5) for training and refer to the new procedure as *pure-node RCPN* or PN-RCPN for short. The classification loss, $\mathcal{L}^p(I)$, now becomes -

$$\mathcal{L}^p(I) = \mathcal{L}(I) + \frac{1}{\sum P_r} \sum_{r=1}^{R} \sum_{p=1}^{P_r} L(\mathbf{y}_{r,p}, t_{r,p}; \mathcal{T}_r, W_{rcpn}) \tag{7}$$

here, P_r is the number of pure-nodes for the r^{th} random parse tree $\mathcal{T}_r$ and subscripts (r, p) map to the p^{th} pure-node for the r^{th} random parse tree. Note that different parse trees for the same image can have different pure nodes.

In order to understand the benefits of PN-RCPN and contrast it with RCPN, we make use of an illustrative example depicted with the help of Fig. 2. The left-half of a random parse tree for an image I with 5 super-pixels, annotated with various variables involved during one forward-backward propagation through RCPN are PN-RCPN are shown in Fig. 2a and 2b, respectively. We denote, $\mathbf{e}_i^{cat}$ (a $C \times 1$ vector) as the error at enhanced super-pixel nodes; $\mathbf{e}_k^{dec}$ (a $2d_{sem} \times 1$ vector) as the error at the decombiner; $\mathbf{e}_k^{com}$ (a $2d_{sem} \times 1$ vector) as the error at the combiner and $\mathbf{e}_i^{sem}$ (a $d_{sem} \times 1$ vector) as the error at the semantic mapper. Subscripts bp and $total$ indicate bypass and the sum total error at a node, respectively. We assume a non-zero categorizer error signal for the first super-pixel only, ie $\mathbf{e}_{i\neq 1}^{cat} = \mathbf{0}$. These assumptions facilitate easier back-propagation tracking through the parse tree, but the conclusions drawn will hold for general cases as well.

The first obvious benefit of using pure-nodes is more labeled samples from the same training data that can improve generalization. The second advantage of PN-RCPN can be understood by contrasting the back-propagation signals for a sample image for RCPN and PN-RCPN, with the help of Fig. 2a (RCPN) and 2b (PN-RCPN). Note that in the case of RCPN, the back-propagated training signal was generated at the enhanced leaf-node features and progressively attenuates as it back-propagates through the parse tree, shown with the help of variable thickness solid red arrows. On the other hand, pure-node RCPN has an internal node (shown as a green color node) that injects a strong error signal deep into the parse tree, resulting in stronger gradients even in the deeper layers. Moreover, PN-RCPN *explicitly* forces the combiner to learn meaningful combination of two super-pixels, because incorrect classification of the combined features is penalized.

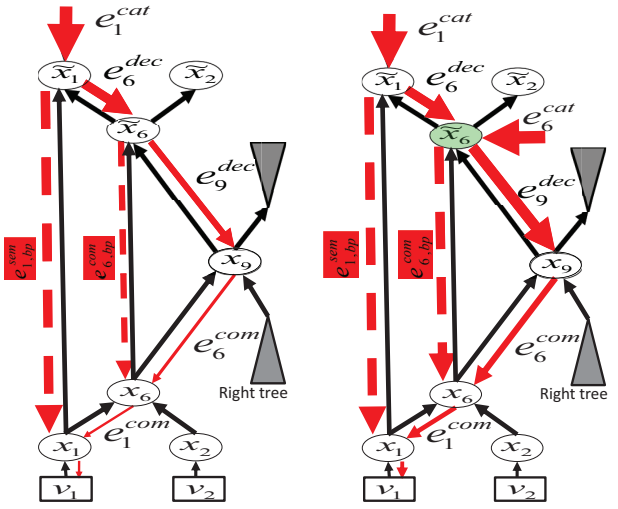
Now, we come to the third benefit of the PN-RCPN architecture. In what follows, we describe a subtle yet potentially serious problem related to RCPN learning, provide empirical evidence that this problem exists, and argue that PN-RCPN can offer a solution to this problem.

4.1.1 Understanding the Bypass Error

During the minimization of the loss functions (Eqn. 5 or 7), typically, more effective parameters in bringing down the objective function receive stronger gradients and reach their stable state early. Due to the presence of multiple layers of non-linearities and complex connections, the loss function is highly non-convex and the solution inevitably converges to a local minimum. It was shown in [19] that the combiner and decombiner assembly is the most important constituent of the RCPN model. Therefore, we expect the learning process to pay more attention to W_{com} and W_{dec}. Unfortunately, the RCPN architecture introduces short-cut paths in the computation graph from the semantic mapper to the categorizer during the forward propagation that gives rise to *bypass errors* during back-propagation. Bypass errors severely affect the learning by reducing the effect of the combiner on the overall loss function, thereby favoring a non-desirable local minimum.

In order to understand the effect of bypass error, we again make use of the example in Fig. 2 to show that bypass paths allow the back-propagated error signals from the categorizer ($\mathbf{e}_i^{cat}$) to reach the semantic mapper through one layer only. On the other hand, $\mathbf{e}_i^{cat}$ goes through multiple layers before reaching the combiner. Therefore, the gradient g_{com} for the combiner is weaker than the gradient for the semantic mapper (g_{sem}).

From the Fig. 2a we can see that there are two possible paths for $\mathbf{e}_1^{cat}$ to reach the combiner. One of them requires 2 layers ($\tilde{\mathbf{x}_1} \rightarrow \tilde{\mathbf{x}_6} \rightarrow \mathbf{x}_6$) and the other requires 3 layers ($\tilde{\mathbf{x}_1} \rightarrow \tilde{\mathbf{x}_6} \rightarrow \mathbf{x}_9 \rightarrow \mathbf{x}_6$). Similarly, $\mathbf{e}_1^{cat}$ can reach $\mathbf{x}_1$ through a 1 layer bypass path ($\tilde{\mathbf{x}_1} \rightarrow \mathbf{x}_1$) or a several layers path through the parse tree. Due to gradient attenuation, the smaller the number of layers the stronger the back-propagated signal, therefore, bypass errors lead to $g_{sem} \geq g_{com}$. This can potentially render the combiner network inoperative and guide the training towards a network that effectively consists of a $N_{sem} + N_{dec} + N_{cat}$ layer network from the visual feature ($\mathbf{v}_i$) to the super-pixel label (y_i). This results in little or no contextual information exchange between the super-pixels. In the worst case $W_{dec} = [W \;\; 0]$; this removes the effect of parents on



(a) (b)

Figure 2: Back-propagated error tracking to visualize the effect of bypass error. The variables follow the notation introduces in Sec. 3. Forward propagation and back-propagation are shown by solid black and red arrows, respectively. The attenuation of the error signal is shown by variable **width** red arrows. The bypass errors are shown with dashed red arrows. (a) RCPN: Error signal from $\tilde{\mathbf{x}}_1$ reaches to $\mathbf{x}_1$ in just one step, through the bypass path. (b) PN-RCPN introduces pure-nodes classification loss (for $\tilde{\mathbf{x}}_6$), thereby, forcing the network to learn meaningful internal node representation via combiner, thereby, promoting effective contextual propagation.

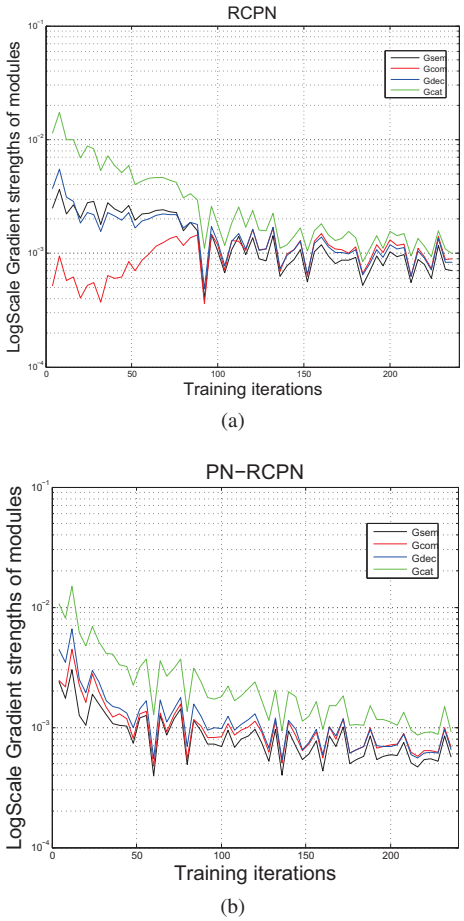


Figure 3: Comparison of gradient strengths of different modules of (a) RCPN and (b) PN-RCPN during training.

their children features during top-down contextual propagation through the decombiner, thereby completely removing the affect of the combiner from RCPN. Practically, the random initialization of the parameters ensures that they will not converge to such a pathological solution. However, we show that a better local minimum can be achieved by tackling the bypass errors.

In order to see that $\mathbf{g}_{sem} \geq \mathbf{g}_{com}$, we compute the gradient strengths of each module (g_{sem}, g_{com}, g_{dec}, g_{cat}) during training. The gradient strengths of different modules for RCPN and PN-RCPN are normalized by the number of parameters and plotted in Fig. 3a and Fig. 3b, respectively. As expected, g_{cat} is the strongest, because it is closest to the initial error signal. Surprisingly, for RCPN g_{sem} is slightly stronger than g_{dec} and significantly stronger than g_{com} during the initial phase of training. Normally, we would expect g_{sem}, which is the farthest away from the error signal, to be the weakest due to vanishing gradients. This observation suggests that the initial training phase favors a multi-layer NN. However, we also observe that during the later stages of training, g_{com} is comparable to other gradients. Unfortunately, it has been conclusively established, by many empirical studies, that the initial phase of training is crucial for determining the final values of the network parameters, and thereby their performance [1]. From the figure we see that the combiner catches up with the other modules during later stages of training, but by then the parameters are already in the attraction basin of a poor solution.

On the other hand, the gradients for PN-RCPN (Fig 3b) follow the natural order of strength, which gives more importance to the combiner and decombiner than the semantic mapper during the initial training. Fig. 2b provides an intuitive explanation by showing the categorizer error signal ($\mathbf{e}_6^{cat}$) for $\tilde{\mathbf{x}}_6$ that reaches to the combiner through one layer only ($\mathbf{e}_{6,bp}^{com}$). To further investigate which of the three aforementioned benefits play the biggest role in improving the performance of PN-RPCN over RPCN, we trained PN-RCPN on SIFT flow under the same setting as Table 2, but we removed as many leaf node labels from the classification loss as the number of pure-nodes. This makes the number

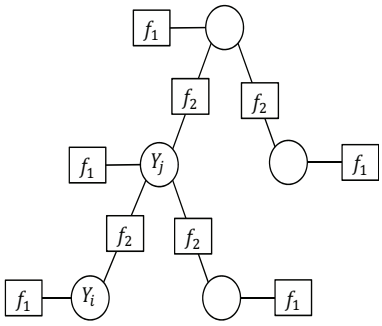


Figure 4: Factor graph representation of the MRF model.

of labeled samples equal in both RCPN and PN-RCPN, but leaf-nodes are replaced with pure-nodes. As expected, it still improves PPA and MCA score for PN-RCPN (80.5% and 35.3%) vs. RCPN (79.6% and 33.6%). This last experiment confirms that inclusion of pure-nodes does not only provide more samples but also helps in overcoming the discussed shortcomings of RCPN.

4.2. Tree MRF Inference

The pure node extension of RCPN provides the label distributions over merged-regions associated with the internal nodes in addition to individual super-pixel labels. In this section, we describe a Markov Random Field (MRF) structure to model the output label dependencies of the super-pixels while leveraging the internal node label distributions for hierarchical consistency. The proposed MRF uses the same trees structure as that of the parse trees used for RCPN inference. A factor graph representation of this MRF is shown in Figure 4. The variables Y_i are L-dimensional binary label vectors associated with each region (merged or single super-pixel) of the image, L is the number of possible labels. The k^{th} dimension of Y_i is set according to the presence (1) or absence (0) of the k^{th} class super-pixel in the region that leads to a $2^L - 1$ dimensional state space.

Let $\mathbf{y}$ be an L-dimensional label assignment for an image region corresponding to Y_i, then unary potentials f_1 are given by the label distributions predicted by the RCPN and defined as -

$$f_1(Y_i = \mathbf{y}) = \frac{-\mathbf{y}^T \log(\mathbf{p}_i)}{\|\mathbf{y}\|_1} \tag{8}$$

where $\mathbf{p}_i$ is the softmax output of the categorizer network for super-pixel i. If the probabilities given by RCPN are not degenerate, the unary potential prefers to assign a single label, that of the node with the highest probability.

The pairwise potentials f_2 are introduced to impose consistency between a pair of child and parent regions. The parent region *must* include all the labels assigned to its children regions, which is a hard constraint:

$$f_2(Y_i = \mathbf{y}_1, Y_j = \mathbf{y}_2) = \begin{cases} \infty, & \text{if } \mathcal{S}(\mathbf{y}_1) \setminus \mathcal{S}(\mathbf{y}_2) \neq \emptyset. \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

where node j is the parent node of i and $\mathcal{S}(\mathbf{y})$ is the set of labels in $\mathbf{y}$.

The unary potentials f_1 utilize all levels of the tree simultaneously and prefer purer nodes, whereas pairwise potentials, f_2 enforce consistency across the tree hierarchy. This design allows for spatial smoothness at lower levels and mixed labeling at the higher levels. The tree structure of the MRF affords exact decoding using max-product belief propagation. The size of the state space is exponential in the number of labels. However, in practice there are rarely more than a handfull of different object classes within an image. Therefore, to reduce the size of the state space, we first identify different labels predicted by the RCPN and only retain the 9 most frequently occurring super-pixel labels per image.

5. Experimental analysis

In this section we evaluate the performance of proposed methods for semantic segmentation on three different datasets: Stanford Background, SIFT Flow and Daimler Urban. Stanford background dataset contains 715 color images of outdoor scenes, it has 8 classes and the images are approximately 240×320 pixels. We used the 572 train and 143 test image split provided by [21] for reporting the results. SIFT Flow contains 2688, 256×256 color images with 33 semantic classes. We experimented with the train/test (2488/200) split provided by the authors of [23]. Daimler Urban dataset has 500, 400×1024 images captured from a moving car in a city, it has 5 semantic classes. We trained the model using 300 images and tested on the rest of the 200 images, the same split-ratio has been used by previous work on this dataset.

5.1. Visual feature extraction

We use a Multi-scale convolution neural network (Multi-scale CNN) [2] to extract pixel-wise features using publicly available library Caffe [7]. We follow [19] and use the same CNN structure with similar preprocessing (subtracting 0.5 from each channel at each pixel location in the RGB color space) at 3 different scales (1,1/2 and 1/4) to obtain the visual features. The CNN architecture has three convolutional stages with $8 \times 8 \times 16$ $conv \rightarrow 2 \times 2$ $maxpool \rightarrow 7 \times 7 \times 64$ $conv \rightarrow 2 \times 2$ $maxpool \rightarrow 7 \times 7 \times 256$ $conv$ configuration, each max-pooling is non-overlapping. Therefore, every image scale gives a 256 dimensional output map. The outputs from each scale are concatenated to get the final feature map. Note that the $256 \times 3 = 768$ dimensional

concatenated output feature map is still 1/4th of the height and width of the input image due to the max-pooling operations. In order to obtain the input size per-pixel feature map we simply scale-up each feature map by a factor of 4 in height and width using Bilinear interpolation.

We use the publicly available implementation of [12] to obtain 100 (same as RCPN) and 800 super-pixels per image for SIFT Flow and Daimler Urban, respectively. Daimler uses more super-pixels due to its larger size. For Stanford background, we have used the super-pixels provided by [21].

5.2. Model Selection

Unlike most of the previous works that rely on careful hand-tuning and expert knowledge for setting the model parameters, we only need to set one parameter, namely d_{sem}, after we have fixed the modules to be 1-layer neural networks. This affords a generic approach to semantic segmentation that can be easily trained on different datasets. For the sake of strict comparison with the original RCPN architecture, we also use 1-layer modules with $d_{sem} = 60$ in all our experiments. *Plain-NN* refers to training a 2-layer NN with 60 hidden nodes, on top of visual features for each super-pixel. *RCPN* refers to the original RCPN model [19]. *PN-RCPN* refers to pure-node RCPN and *TM-RCPN* refers to tree-MRF RCPN.

5.3. Evaluation metrics

We have used four standard evaluation metrics -

- **Per pixel accuracy (PPA):** Ratio of the correct pixels to the total pixels in the test images, while ignoring the background.
- **Mean class accuracy (MCA):** Mean of the category wise pixel accuracy.
- **Intersection over Union (IoU):** Ratio of true positives to the sum of true positive, false positive and false negative, averaged over all classes. This is a popular measure for semantic segmentation of objects because it penalizes both over- and under-segmentation.
- **Time per image (TPI):** Time required to label an image on GPU and CPU.

The results from previous works are taken directly from the published articles. Some of the previous works do not report all four evaluation metrics; we leave the corresponding entry blank in the comparison tables.

5.4. Stanford Background

We report our results with CNN features extracted from the original scale only, because multi-scale CNN features overfit, perhaps due to small training data, as observed in [19]. We use 10 and 40 random trees for training and testing, respectively. The results are shown in Table 1. From the comparison, it is clear that our proposed approaches outperform previous methods. We observe that PN-RCPN significantly improves the results in terms of MCA and IoU over RCPN. We observe a marginal improvement offered by TM-RCPN over PN-RCPN.

Table 1: Stanford background result.

Method	PPA	MCA	IoU	TPI (s) CPU/GPU
Gould, [6]	76.4	NA	NA	30 – 600 / NA
Munoz, [15]	76.9	NA	NA	12 / NA
Tighe, [23]	77.5	NA	NA	4 / NA
Kumar, [8]	79.4	NA	NA	$\leq$ 600 / NA
Socher, [21]	78.1	NA	NA	NA / NA
Lempitzky, [10]	81.9	72.4	NA	$\geq$ 60 / NA
Singh, [20]	74.1	62.2	NA	20 / NA
Farabet, [2]	81.4	76.0	NA	60.5 / NA
Eigen, [4]	75.3	66.5	NA	16.6 / NA
Pinheiro, [17]	80.2	69.9	NA	10 / NA
Plain-NN	80.1	69.7	56.4	1.1/0.4
RCPN [19]	81.8	73.9	61.3	1.1/0.4
PN-RCPN	82.1	79.0	64.0	1.1/0.4
TM-RCPN	**82.3**	**79.1**	**64.5**	1.6–6.1/0.9–5.9

5.5. SIFT Flow

We report our results using multi-scale CNN features at three scales (1,1/2 and 1/4), as in [19]. Some of the classes in SIFT Flow dataset have a very small number of training instances, therefore, we also trained with balanced sampling to compensate for rare occurrence, referred to as *bal.* prefix. We use 4 and 20 random trees for training and testing, respectively. The results for SIFT flow dataset are shown in Table 2. PN-RCPN led to significant improvement in all three measures over RCPN and balanced training led to significant boost in MCA. The use of TM-RCPN does not affect the results much compared to PN-RCPN. We observe a strong trade-off between PPA and MCA on this dataset. Our overall best model in terms of both PPA and MCA (*bal. TM-RCPN*) looks equivalent to the work in [25]; PPA: 76.4 vs. 79.8, MCA: 52.6 vs. 48.8.

5.6. Daimler Urban

We report our results using multi-scale CNN features with balanced training in Table 3. The previous results are based on the predicted labels provided by the authors of [18]. The authors, in their paper [18], have reported the results with background as one of the classes, but the ground-truth labels for this dataset have portions of foreground classes labeled as the background. Therefore, even a correct labeling is penalized. All the results in Table 3, including [9, 18], ignore the background class for a fair eval-

Table 2: SIFT Flow result.

Method	PPA	MCA	IoU	TPI (s) CPU/GPU
Tighe, [23]	77.0	30.1	NA	8.4 / NA
Liu, [11]	76.7	NA	NA	31 / NA
Singh, [20]	79.2	33.8	NA	20 / NA
Eigen, [4]	77.1	32.5	NA	16.6 / NA
Farabet, [2]	78.5	29.6	NA	NA / NA
(Balanced), [2]	72.3	50.8	NA	NA / NA
Tighe, [22]	78.6	39.2	NA	$\geq$ 8.4 / NA
Pinheiro, [17]	77.7	29.8	NA	NA / NA
Yang, [25]	79.8	48.7	NA	$\leq$ 12/NA
Plain-NN	76.3	32.1	24.7	1.1/0.36
RCPN, [19]	79.6	33.6	26.9	1.1/0.4
bal. RCPN, [19]	75.5	48.0	28.6	1.1/0.4
PN-RCPN	**80.9**	39.1	30.8	1.1/0.4
bal. PN-RCPN	75.5	**52.8**	30.2	1.1/0.4
TM-RCPN	80.8	38.4	30.7	1.6–6.1/0.9–5.4
bal. TM-RCPN	76.4	52.6	**31.4**	1.6–6.1/0.9–5.8

uation. **IoU Dyn** is the IoU for dynamic objects ie cars, pedestrians and bicyclists. We would like to underscore that the previous approaches ([9, 18]) use stereo, depth, visual odometry and multi-frame temporal information that relies on the fact that the images are coming from a moving vehicle whereas, we only use an independent single visual image and still obtain similar or better performance. We observe significant improvements in terms of IoU with the use of PN-RCPN over RCPN and Plain-NN which could be due to the well structured image semantics of this dataset that allows it to learn the structure very effectively and utilize the context in a much better way than the other two datasets. Some of the representative segmentation results are shown in Fig. 5. We have also submitted a complete video of semantic segmentation for all the test images for Daimler urban in the supplementary material.

5.7. Segmentation Time

In this section we provide the timing details for the experiments. Only the Multi-CNN feature extraction is executed on a GPU for our Plain-NN and RCPN variants. Due to similar image sizes, SIFT flow and Stanford Background took almost the same computation per image except while using TM-RCPN, because of the difference in label state-space size. The time break-up for SIFT flow (same for Stanford) in seconds is 0.3 (super-pixellation) + 0.08/0.8 (GPU/CPU visual feature) + 0.01 (PN-RCPN) + 0.5–5 (TM-MRF). For Daimler, the corresponding timings are 2.4 + 0.4/3.5 + 0.09 + 6 seconds. Therefore, the bottleneck for our system is the super-pixellation time for PN-RCPN and MRF inference for TM-RCPN. Fortunately, there are real-time super-pixellation algorithms, such as [3], that can help us achieve state-of-the-art semantic segmentation within 100 milliseconds on an NVIDIA Titan Black GPU.

Table 3: Daimler result. Numbers in *italics* indicate the use of stereo, depth and multi-frame temporal information.

Method	PPA	MCA	IoU	IoU Dyn	TPI (s) CPU/GPU
Joint, [9, 18]	***94.5***	***91.0***	***86.0***	***74.5***	*111 / NA*
Stix., [18]	*92.8*	*87.5*	*80.6*	*72.3*	***0.05*** */ NA*
bal. Plain-NN	91.4	83.2	75.8	56.2	5.9 / 2.8
bal. RCPN	93.3	87.6	80.9	66.0	6.0 / 2.8
bal. PN-RCPN	**94.5**	90.2	84.5	73.8	6.0 / 2.8
bal. TM-RCPN	**94.5**	90.1	84.5	73.8	12 / 8.8

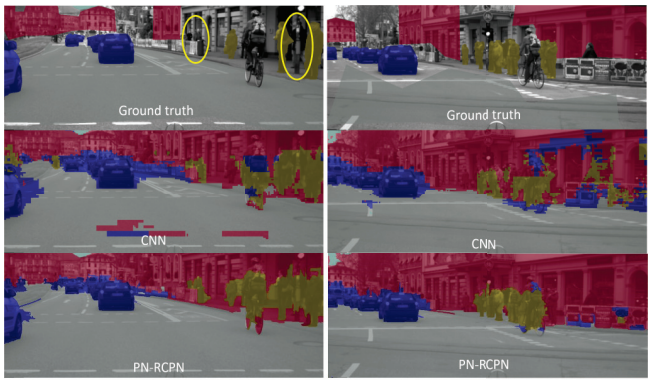


Figure 5: Some representative image segmentation results on Daimler Urban dataset. Here, CNN refers to direct per-pixel classification resulting from the multi-scale CNN. The ground-truth images are only partially labeled and we have shown the unlabeled pedestrians by yellow ellipses.

6. Conclusion

We analyzed the recursive contextual propagation network, referred to as RCPN [19] and discovered potential problems with the learning of it's parameters. Specifically, we showed the existence of bypass errors and explained how it can reduce the RCPN model to an effective multi-layer neural network for each super-pixel. Based on our findings, we proposed to include the classification loss of pure-nodes to the original RCPN formulation and demonstrated it's benefits in terms of avoiding the bypass errors. We also proposed a tree MRF on the parse tree nodes to utilize the pure-node's label estimation for inferring the super-pixel labels. The proposed approaches lead to state-of-the-art performance on three segmentation datasets: Stanford background, SIFT flow and Daimler urban.

References

[1] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, 2010. 5

[2] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE TPAMI*, August 2013. 2, 6, 7, 8

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004. 8

[4] R. Fergus and D. Eigen. Nonparametric image parsing using adaptive neighbor sets. *IEEE CVPR*, 2012. 2, 7, 8

[5] C. Goller and A. Kchler. Learning task-dependent distributed representations by backpropagation through structure. *Int Conf. on Neural Network*, 1995. 1

[6] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. *IEEE ICCV*, 2009. 2, 7

[7] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013. 6

[8] M. P. Kumar and D. Koller. Efficiently selecting regions for scene understanding. *IEEE CVPR*, 2010. 7

[9] L. Ladick, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. Torr. Joint optimization for object class segmentation and dense stereo reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012. 7, 8

[10] V. Lempitsky, A. Vedaldi, and A. Zisserman. A pylon model for semantic segmentation. *NIPS*, 2011. 2, 7

[11] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *IEEE TPAMI*, 33(12), Dec 2011. 2, 8

[12] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. *IEEE CVPR*, 2011. 7

[13] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. *IEEE CVPR*, 2014. 1, 2

[14] R. Mottaghi, S. Fidler, J. Yao, R. Urtasun, and D. Parikh. Analyzing semantic segmentation using hybrid human-machine crfs. *IEEE CVPR*, 2013. 1, 2

[15] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. *ECCV*, 2010. 2, 7

[16] D. Pfeiffer, S. K. Gehrig, and N. Schneider. Exploiting the power of stereo confidences. *CVPR*, 2013. 2

[17] P. H. O. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. *ICML*, 2014. 2, 7, 8

[18] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth. Stixmantics: A medium-level model for real-time semantic scene understanding. *ECCV*, 2014. 7, 8

[19] A. Sharma, O. Tuzel, and M. Y. Liu. Recursive context propagation network for semantic segmentation. *NIPS*, 2014. 1, 2, 4, 6, 7, 8

[20] G. Singh and J. Kosecka. Nonparametric scene parsing with adaptive feature relevance and semantic context. *IEEE CVPR*, 2013. 2, 7, 8

[21] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. *ICML*, 2011. 2, 6, 7

[22] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. *IEEE CVPR*, 2013. 2, 8

[23] J. Tighe and S. Lazebnik. Superparsing. *Int. J. Comput. Vision*, 101(2):329–349, 2013. 2, 6, 7, 8

[24] A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-based vision system for place and object recognition. *IEEE CVPR*, 2003. 1

[25] J. Yang, B. Price, S. Cohen, and M.-H. Yang. Context driven scene parsing with attention to rare classes. *CVPR*, pages 3294–3301, 2014. 2, 7, 8