An End-to-end Tree based approach for Instance Segmentation

KV Manohar[⋆]
kvmanohar22@gmail.com
IIT Kharagpur

Yusuke Niitani
niitani@preferred.jp
Preferred Networks inc., Japan

Abstract. This paper presents an approach for bottom-up hierarchical instance segmentation. We propose an end-to-end model to estimate energies of regions in an hierarchical region tree. To this end, we introduce a Convolutional Tree-LSTM module to leverage the tree-structured network topology. For constructing the hierarchical region tree, we utilize the accurate boundaries predicted from a pre-trained convolutional oriented boundary network. We evaluate our model on PASCAL VOC 2012 dataset showing that we obtain good trade-off between segmentation accuracy and time taken to process a single image.

1 Introduction

In this work we address the task of *instance segmentation* which involves segmenting each individual instance of a semantic class in an image. Many top-down approaches to this problem are based on object detection pipelines [1], [2] and each box is refined to generate a segmentation. Further, these methods do not consider entire image but rather independent proposals and as a result cannot handle occlusions between different objects. Since these methods are based on initial detections, they cannot recover from false detections motivating an approach that reasons globally.

A key aspect of our approach is to leverage the hierarchical segmentation trees [3] to sample potential object instances. To this end, we propose a new bottom-up approach to parse the regions in an hierarchical region tree. At the core of our approach lies Convolutional Tree-LSTM module which estimates the energies of the regions taking into account the entire image and tracking temporal relations across regions through different levels of the tree. Unlike MCG [4], that uses hand engineered features to generate object candidates, we exploit rich features learnt by Convolutional Neural Networks to sample object instances. Further, MCG involves complex pipeline involving proposal generation and ranking. The resulting system is very slow and takes more than 9.9 seconds for candidate generation alone. Ours on the other hand is trained end-to-end and on average takes 0.06 seconds at test time.

Our paper is outlined as follows. We begin by reviewing related work in section 2. In section 3 we describe the details of our approach. In section 4, we dwell into implementation details. We investigate the performance of our method both qualitatively and quantitatively in section 5. Finally, we conclude in section 6.

[⋆] work done when the author was an intern at Preferred Networks inc., Japan

2 Related Work

Our work is closely related to bottom-up methods exploiting superpixels [5]. Pham et. al. [6] proposed a dynamic programming based approach to image segmentation by constructing a hierarchical segmentation tree. An unified energy function jointly quantifies geometric goodness-of-fit and objectness measure. A top-down traversal through the tree comparing the energies of the current node and its subtree results in optimal tree cut. Kirillov et. al. [7] impose graph structure on the superpixels and formulate instance estimation as a MultiCut problem. One of the limitations of this method however is that, it cannot find instances that are formed by disconnected regions in the image. Unlike these methods, by training our model end-to-end we can find such instances as discussed in section 6.

3 Method

Given an input image $\mathcal{I}$, our goal is to segment the image into semantically meaningful non-overlapping regions. Fig 1 depicts the overview of our method. Henceforth, we adopt the following notation. For a given $\mathcal{I}$, let $\mathcal{T}$, $L = \{1, 2, \ldots, l_{max}\}$, $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$, $\mathcal{F} = \{F_{r_1}, F_{r_2}, \ldots, F_{r_N}\}$ and $\mathcal{C} = \{C_{r_1}, C_{r_2}, \ldots, C_{r_N}\}$ represent the hierarchical tree, set of distinct levels, set of regions in the tree, corresponding features for the regions and children of the regions in the tree respectively. For each level $0 < l \leq l_{max}$, we denote the set of regions, corresponding features and the threshold at this level as $\mathcal{R}_l = \{r^l_1, r^l_2, \ldots, r^l_{N_l}\} \subseteq \mathcal{R}$, $\mathcal{F}_l = \{F_{r^l_1}, F_{r^l_2}, \ldots, F_{r^l_{N_l}}\}$ and α_l respectively. Tree cut at a level $l^{'}$ for a horizontal cut-threshold $\lambda_{cut} = \alpha_{l^{'}}$ results in a new set of levels $L^{'} = \{l | l \;\; l^{'}\}$.

3.1 Feature Extraction

We first extract features $\mathbf{F}$ by passing input image $\mathcal{I}$ through a series of convolutions. For a given region $r \in \mathcal{R}$ in the tree, we generate a tightest bounding box b_r covering the non-linear boundary of r. We then extract a fixed spatial dimensional feature map F_r (e.g., 7×7) from $\mathbf{F}$ corresponding to b_r. Our approach in extracting F_r is similar to ROIAlign layer [1]. Additionally, we mask out the features corresponding to the region $b_r \setminus r$ giving rise to the final feature map F_r.

3.2 Convolutional Tree-LSTM module

The motivation behind the method is to estimate how the probabiliy distribution over the categories change when a new region is added to the region under consideration in the subsequent levels. The model implicitly learns the temporal

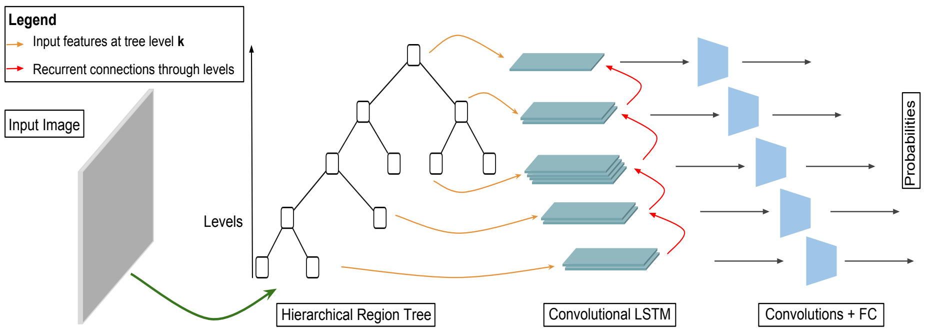

Fig. 1: Overview of our method. We (1) construct hierarchical region tree using Ultrametric Contour Map (UCM), (2) estimate energies of each region in the tree starting from level 1 at the bottom and all the way to the top, and (3) threshold the regions based on the energies.

relations which lead to the formation of a given region.
We process the hierarchical tree T starting from level l^{0} which corresponds to the initial cut-threshold $\lambda_{cut} = \alpha_{l^0}$ using Convolutional Tree-LSTM predicting softmax probabilities for each region $r\ 2\ R_l$ at all the levels $l\ 2\ L^{0}$ in order. Input to the LSTM at each level l are the features F_l. Eqs. 1-7 summarizes the forward propagation through the LSTM module. For jth region at level l,

$$\tilde{h}_j^l = \sum_{k \in C_{r_j^l}} h_k^l; \tag{1}$$

$$i_j^l = \quad (W^i \quad F_{r_j^l} + U^i \quad \tilde{h}_j^l + b^i); \tag{2}$$

$$f_{jk}^l = \quad (W^f \quad F_{r_j^l} + U^f \quad h_k^l + b^f) \quad 8k\ 2\ C_{r_j^l}; \tag{3}$$

$$o_j^l = \quad (W^o \quad F_{r_j^l} + U^o \quad \tilde{h}_j^l + b^o); \tag{4}$$

$$u_j^l = \tanh(W^u \quad F_{r_j^l} + U^u \quad \tilde{h}_j^l + b^u); \tag{5}$$

$$c_j^l = i_j^l \quad u_j^l + \sum_{k \in C_{r_j^l}} f_{jk}^l \quad c_k^l; \tag{6}$$

$$h_j^l = o_j^l \quad \tanh(c_j^l); \tag{7}$$

where $\ ,\ $ denote convolution operation and Hadamard product respectively. We do the above for each region j and $8\ l\ 2\ L^{0}$. For a region j at level l, $c_k^l, h_k^l 8k\ 2\ C_{r_j^l}$ are initialized to zeros provided they are the leaves of the tree and for the rest of the regions, c_k^l, h_k^l are governed by the Eqs. 6 and 7 respectively. Fig 2 depicts analysis on variation of sequence length and number of regions considered for different horizontal cuts.
On top of the LSTM module, we apply series of convolutions and fully connected layers which take input as h_j^l and predict probabilities.

Fig. 2: Variation of number of regions considered and sequence length for di erent initial horizontal cut thresholds.

3.3 Objective formulation

For a given image I, let $M = \{m_1; m_2; \ldots; m_M\}$; $L^G = \{l_1; l_2; \ldots; l_M\}$ be the set of ground truth masks and one-hot labels respectively. For each mask m_i, we construct the positive set $P_i^+ = \{p_1^i; p_2^i; \ldots; p_{N_i}^i\}$ which consists of probabilities of regions from R whose IoU with m_i is greater than $_{+}$. Similarly, we construct $P = \{p_1; p_2; \ldots; p_N\}$ consisting of probabilities of regions from R whose IoU with all m_i is less than . We then formulate the loss as follows,

$$L = \frac{1}{M} \sum_{i=1}^{M} \sum_{r=1}^{|P_i^+|} l_i^T \log(p_r^i) \quad \sum_{r=1}^{|P|} \sum_{c=1}^{C} l_c^b \log(p_r); \tag{8}$$

where l_c^b is 1 if class c corresponds to the background label b and T represents the transpose of vector. The hyperparameter in Eq. 8 controls the balance between positive and negative regions.

4 Implementation Details

4.1 Network Architecture

We use the pre-trained COB network for estimating contours which is a ResNet50 model. Features F are extracted from res3 layer of ResNet50 model having spatial resolution of $28 \quad 28$. ROIAlign extracts features having a xed spatial resolution of $7 \quad 7$. All the convolutions within the LSTM have kernel size of $3 \quad 3$, stride 1 and use zero-padding. On top of convolutional LSTM, we have 2 $3 \quad 3$ convolutions and 2 fully connected layers predicting softmax probabilities.

4.2 Training details

We set the parameters $_{+}$, , to 0.7, 0.3 and 0.2 respectively in all our experiments. We train Convolutional LSTM and subsequent layers from scratch with a batch size of 1, initial learning rate of 0.001 and decay it by a factor of 0.1 after every 20 epochs. We experiment over various initial cut-thresholds from $_{cut} = 0.3$ to $_{cut} = 0.9$ in steps of 0.1.

Table 1: Time taken to process a single image in seconds.

Method	Hierarchical Segmentation[a]	Candidate Generation[b]	Total
MCG [4]	24.4 3:6	9.9 3:5	34.3 6:2
SCG [4]	3.2 0:4	1.5 0:5	4.7 0:7
Scene-cut [6]	0.79	3.76	4.55
Ours	0.79	0.06	0.85

[a] Involves estimating contours, UCM and constructing hierarchical region tree
[b] Involves estimating energies of regions and optimal tree cut

Fig. 3: Precision-recall curves for all categories in VOC 2012 val dataset.

5 Experiments

We use the pretrained COB network to predict the contours which was trained on PASCAL Context dataset. We train our Convolutional Tree-LSTM and subsequent layers on PASCAL VOC 2012 dataset. We evaluate our model on PASCAL VOC 2012 val dataset using average precision, Jaccard Index and time taken to process an image as evaluation metrics. Table 1 compares the time taken to process a single image by di erent methods. Fig 3 denotes the precision-recall curves for all the classes.

Table 2: Variation of average precision for di erent tree cut thresholds.

Cut threshold	Plane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	MBike	Person	Plant	Sheep	Sofa	Train	TV	mAP
0.5	0.72	0	0.5	0.44	0.16	0.73	0.46	0.63	0.01	0.26	0.12	0.37	0.32	0.53	0.29	0.15	0.39	0.15	0.71	0.36
0.6	0.72	0	0.49	0.45	0.15	0.74	0.44	0.62	0.01	0.27	0.12	0.37	0.29	0.55	0.29	0.16	0.39	0.15	0.69	0.36
0.7	0.79	0	0.75	0.54	0.14	0.68	0.63	0.76	0	0.5	0.24	0.61	0.47	0.74	0.36	0.28	0.49	0.17	0.93	0.47
0.8	0.81	0	0.7	0.67	0.29	0.78	0.65	0.79	0	0.4	0.29	0.47	0.46	0.65	0.38	0.31	0.39	0.2	0.92	0.48
0.9	0.68	0	0.47	0.38	0.04	0.5	0.39	0.49	0	0.22	0.03	0.19	0.18	0.42	0.29	0.1	0.44	0.11	0.66	0.23

On the VOC 2012 val set, our best performing model scores 48% mAP. Our model struggles on categories like bicycle, chair. However on categories like train and plane, our model achieves higher performance. Table 2 summarizes the average precision for all the categories. We further compare Jaccard Index with MCG and is presented in table 3.