Semantic Object Parsing with Graph LSTM

Xiaodan Liang[1]   Xiaohui Shen[4]   Jiashi Feng[3]   Liang Lin[1]*   Shuicheng Yan [2,3]

[1] Sun Yat-sen University   [2] 360 AI Institue   [3] National University of Singapore
[4] Adobe Research

xdliang328@gmail.com, xshen@adobe.com, elefjia@nus.edu.sg,
linliang@ieee.org, eleyans@nus.edu.sg

Abstract. By taking the semantic object parsing task as an exemplar application scenario, we propose the Graph Long Short-Term Memory (Graph LSTM) network, which is the generalization of LSTM from sequential data or multi-dimensional data to general graph-structured data. Particularly, instead of evenly and fixedly dividing an image to pixels or patches in existing multi-dimensional LSTM structures (e.g., Row, Grid and Diagonal LSTMs [1][2]), we take each arbitrary-shaped superpixel as a semantically consistent node, and adaptively construct an undirected graph for each image, where the spatial relations of the superpixels are naturally used as edges. Constructed on such an adaptive graph topology, the Graph LSTM is more naturally aligned with the visual patterns in the image (e.g., object boundaries or appearance similarities) and provides a more economical information propagation route. Furthermore, for each optimization step over Graph LSTM, we propose to use a confidence-driven scheme to update the hidden and memory states of nodes progressively till all nodes are updated. In addition, for each node, the forgets gates are adaptively learned to capture different degrees of semantic correlation with neighboring nodes. Comprehensive evaluations on four diverse semantic object parsing datasets well demonstrate the significant superiority of our Graph LSTM over other state-of-the-art solutions.

Keywords: Object Parsing, Graph LSTM, Recurrent Neural Networks

1 Introduction

Beyond traditional image semantic segmentation, semantic object parsing aims to segment an object within an image into multiple parts with more fine-grained semantics and provide full understanding of image contents, as shown in Fig. 1. Many higher-level computer vision applications [3, 4] can benefit from a powerful semantic object parser, including person re-identification [5], human behavior analysis [6][7][8][9].

Recently, Convolutional Neural Networks (CNNs) have demonstrated exciting success in various pixel-wise prediction tasks such as semantic segmentation and detection [10][11], semantic part segmentation [12][13] and depth prediction [14]. However, the pure convolutional filters can only capture limited local context while the precise inference for semantic part layouts and their interactions requires a global perspective of the image. To consider the global structural context, previous works thus use dense

---

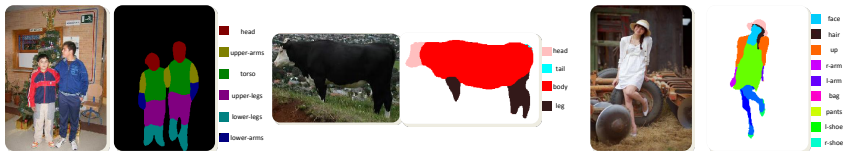* The corresponding author is Liang Lin.

Fig. 1. Examples of semantic object parsing results by the proposed Graph LSTM model. It parses an object into multiple parts with different semantic meanings. Best viewed in color.

pairwise connections (Conditional Random Fields (CRFs)) upon pure pixel-wise CNN classifiers [15][10][16][17]. However, most of them try to model the structure information based on the predicted confidence maps, and do not explicitly enhance the feature representations in capturing global contextual information, leading to suboptimal segmentation results under complex scenarios.

An alternative strategy is to exploit long-range dependencies by directly augmenting the intermediate features. The multi-dimensional Long Short-Term Memory (LSTM) networks have produced very promising results in modeling 2D images [18][19][20][21], where long-range dependencies, which are essential to object and scene understanding, can be well memorized by sequentially functioning on all pixels. However, in terms of the information propagation route in each LSTM unit, most of existing LSTMs [1] [2] [21] have only explored pre-defined fixed topologies. As illusrated in the top row of Fig. 2, for each individual image, the prediction for each pixel by those methods is influenced by the predictions of fixed neighbors (e.g., 2 or 8 adjacent pixels or diagonal neighbors) in each time-step. The natural properties of images (e.g., local boundaries and semantically consistent groups of pixels) have not be fully utilized to enable more meaningful and economical inference in such fixed locally factorized LSTMs. In addition, much computation with the fixed topology is redundant and inefficient as it has to consider all the pixels, even for the ones in a simple plain region.

In this paper, we propose a novel Graph LSTM model that extends the traditional LSTMs from sequential and multi-dimensional data to general graph-structured data, and demonstrate its superiority on the semantic object parsing task. Instead of evenly and fixedly dividing an image into pixels or patches as previous LSTMs did, Graph LSTM takes each arbitrary-shaped superpixel as a semantically consistent node of a graph, while the spatial neighborhood relations are naturally used to construct the undirected graph edges. The adaptive graph topology can thus be constructed where different nodes are connected with different numbers of neighbors, depending on the local structures in the image. As shown in the bottom row of Fig. 2, instead of broadcasting information to a fixed local neighborhood following a fixed updating sequence as in the previous LSTMs, Graph LSTM proposes to effectively propagate information from one adaptive starting superpixel node to all superpixel nodes along the adaptive graph topology for each image. It can effectively reduce redundant computational costs while better preserving object/part boundaries to facilitate global reasoning.

Together with the adaptively constructed graph topology of an image, we propose a confidence-driven scheme to subsequently update the features of all nodes, which is
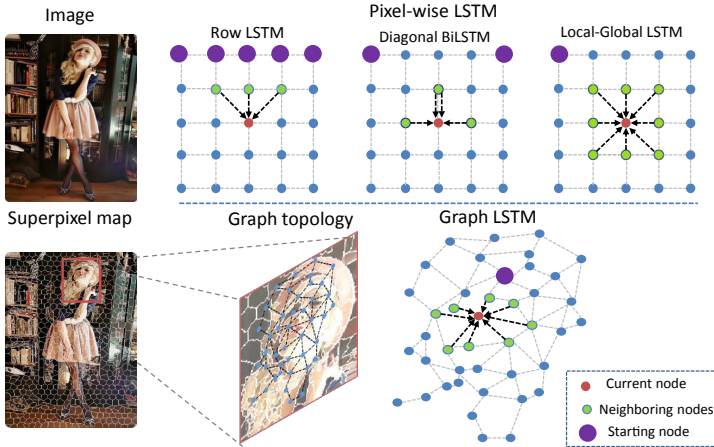
Fig. 2. The proposed Graph LSTM structure. 1) The top row shows the traditional pixel-wise LSTM structures, including Row LSTM [2], Diagonal BiLSTM [2] [1] and Local-Global LSTM [21]. 2) The bottom row illustrates the proposed Graph LSTM that is built upon the superpixel over-segmentation map for each image.

inspired by the recent visual attention models [22][23]. Previous LSTMs [1][2] often simply start at pre-defined pixel or patch locations and then proceed toward other pixels or patches following a fixed updating route for different images. In contrast, we assume that starting from a proper superpixel node and updating the nodes following a certain content-adaptive path can lead to a more flexible and reliable inference for global context modelling, where the visual characteristics of each image can be better captured.

As shown in Fig. 3, the Graph LSTM, as an independent layer, can be easily appended to the intermediate convolutional layers in a Fully Convolutional Neural Network [24] to strengthen visual feature learning by incorporating long-range contextual information. The hidden states represent the reinforced features, and the memory states recurrently encode the global structures.

Our contributions can be summarized in the following four aspects. 1) We propose a novel Graph LSTM structure to extend the traditional LSTMs from sequential and multi-dimensional data to general graph-structured data, which effectively exploits global context by following an adaptive graph topology derived from the content of each image. 2) We propose a confidence-driven scheme to select the starting node and sequentially update all nodes, which facilitates the flexible inference while preserving the visual characteristics of each image. 3) In each Graph LSTM unit, different forget gates for the neighboring nodes are learned to dynamically incorporate the local contextual interactions in accordance with their semantic relations. 4) We apply the proposed Graph LSTM in semantic object parsing, and demonstrate its superiority through comprehensive comparisons on four challenging semantic object parsing datasets (i.e., PASCAL-Person-Part dataset [25], Horse-Cow parsing dataset [26], ATR dataset [27] and Fashionista dataset [28]).
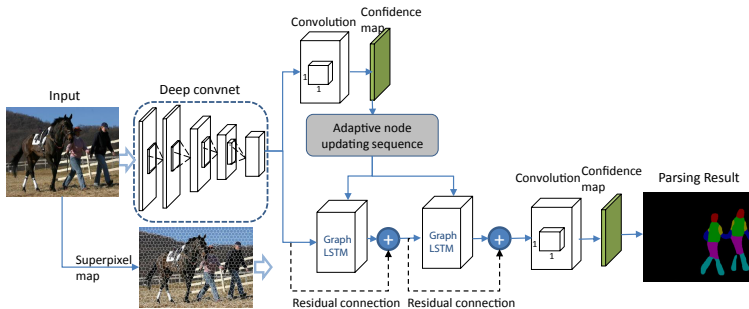
Fig. 3. Illustration of the proposed network architecture for semantic object parsing.

2 Related Work

LSTM on Image Processing: Recurrent neural networks have been first introduced to address the sequential prediction tasks [29] [30] [31], and then extended to multi-dimensional image processing tasks [18] [19].Benefiting from the long-range memorization of LSTM networks, they can obtain considerably larger dependency fields by sequentially performing LSTM units on all pixels, compared to the local convolutional filters. Nevertheless, in each LSTM unit, the prediction of each pixel is affected by a fixed factorization (e.g., 2 or 8 neighboring pixels [1][32][21][33] or diagonal neighborhood [2][19]), where diverse natural visual correlations (e.g., local boundaries and homogeneous regions) have not been considered. Meanwhile, the computation is very costly and redundant due to the sequential computation on all pixels. Tree-LSTM [34] introduces the structure with tree-structured topologies for predicting semantic representations of sentences. Compared to Tree-LSTM, Graph LSTM is more natural and general for 2D image processing with arbitrary graph topologies and adaptive updating schemes.

Semantic Object Parsing: There has been increasing research interest on the semantic object parsing problem including the general object parsing [26][16][35][25][36], person part segmentation [13][12] and human parsing [28][37][38][39][40][41][42]. To capture the rich structure information based on the advanced CNN architecture, one common way is the combination of CNNs and CRFs [15][10][43][16], where the CNN outputs are treated as unary potentials while CRF further incorporates pairwise or higher order factors. Instead of learning features only from local convolutional kernels as in these previous methods, we incorporate the global context by the novel Graph LSTM structure to capture long-distance dependencies on the superpixels. The dependency field of Graph LSTM can effectively cover the entire image context.

3 The Proposed Graph LSTM

We take semantic object parsing as its application scenario, which aims to generate pixel-wise semantic part segmentation for each image. Fig. 3 illustrates the designed

network architecture based on Graph LSTM. The input image first passes through a stack of convolutional layers to generate the convolutional feature maps.The proposed Graph LSTM takes the convolutional features and the adaptively specified node updating sequence for each image as the input, and then efficiently propagates the aggregated contextual information towards all nodes, leading to enhanced visual features and better parsing results. To both increase convergence speed and propagate signals more directly through the network, we deploy residual connections [44] after one Graph LSTM layer to generate the input features of the next Graph LSTM layer. Note that residual connections are performed to generate the element-wise input features for each layer, which would not destroy the computed graph topology. After that, several 1×1 convolution filters are employed to produce the final parsing results. The following subsections will describe the main innovations inside Graph LSTM, including the graph construction and the Graph LSTM structure.

3.1   Graph Construction

The graph is constructed on superpixels that are obtained through image over-segmentation using SLIC [45][1]. Note that, after several convolutional layers, the feature maps of each image have been down-sampled. Therefore, in order to use the superpixel map for graph construction in each Graph LSTM layer, one needs to upsample the feature maps into the original size of the input image.

The superpixel graph $\mathcal{G}$ for each image is then constructed by connecting a set of graph nodes $\{v_i\}_{i=1}^{N}$ via the graph edges $\{\mathcal{E}_{ij}\}$. Each graph node v_i represents a superpixel and each graph edge $\mathcal{E}_{ij}$ only connects two spatially neighboring superpixel nodes. The input features of each graph node v_i are denoted as $\mathbf{f}_i \in \mathbb{R}^d$, where d is the feature dimension. The feature $\mathbf{f}_i$ is computed by averaging the features of all the pixels belonging to the same superpixel node v_i. As shown in Fig. 3, the input states of the first Graph LSTM layer come from the previous convolutional feature maps. For the subsequent Graph LSTM layers, the input states are generated after the residual connections [44] for the input features and the updated hidden states by the previous Graph LSTM layer. To make sure that the number of the input states for the first Graph LSTM layer is compatible with that of the following layers and that the residual connections can be applied, the dimensions of hidden and memory states in all Graph LSTM layers are set the same as the feature dimension of the last convolutional layer before the first Graph LSTM layer.

3.2   Graph LSTM

Confidence-driven Scheme. The node updating scheme is more important yet more challenging in Graph LSTM than the ones in traditional LSTMs [2][1] due to its adaptive graph topology. To enable better global reasoning, Graph LSTM specifies the adaptive starting node and node updating sequence for the information propagation of each image. Given the constructed undirected graph $\mathcal{G}$, we extensively tried several schemes

---

[1] Other over-segmentation methods such as entropy rate-based approach [41] could also be used, and we did not observe much difference in the final results in our experiments.

to update all nodes in a graph in the experiments, including the Breadth-First Search (BFS), Depth-First Search (DFS) and Confidence-Driven Search (CDS). We find that the CDS achieves better performance. Specifically, as illustrated in Fig. 3, given the top convolutional feature maps, the 1×1 convolutional filters can be used to generate the initial confidence maps with regard to each semantic label. Then the confidence of each superpixel for each label is computed by averaging the confidences of its contained pixels, and the label with highest confidence could be assigned to the superpixel. Among all the foreground superpixels (i.e., assigned to any semantic part label), the node updating sequence can be determined by ranking all the superpixel nodes according to the confidences of their assigned labels.

During updating, the $(t + 1)$-th Graph LSTM layer determines the current states of each node v_i that comprises the hidden states $\mathbf{h}_{i,t+1} \in \mathbb{R}^d$ and memory states $\mathbf{m}_{i,t+1} \in \mathbb{R}^d$ of each node. Each node is influenced by its previous states and the states of neighboring graph nodes as well in order to propagate information to the whole image. Thus the inputs to Graph LSTM units consist of the input states $\mathbf{f}_{i,t+1}$ of the node v_i, its previous hidden states $\mathbf{h}_{i,t}$ and memory states $\mathbf{m}_{i,t}$, and the hidden and memory states of its neighboring nodes $v_j, j \in \mathcal{N}_{\mathcal{G}}(i)$.

Averaged Hidden States for Neighboring Nodes. Note that with an adaptive updating scheme, when operating on a specific node in each Graph LSTM layer, some of its neighboring nodes have already been updated while others may have not. We therefore use a visit flag q_j to indicate whether the graph node v_j has been updated, where q_j is set as 1 if updated, and otherwise 0. We then use the updated hidden states $\mathbf{h}_{j,t+1}$ for the visited nodes, i.e., $q_j = 1$ and the previous states $\mathbf{h}_{j,t}$ for the unvisited nodes. The $\mathbb{1}(\cdot)$ is an indicator function. Note that the nodes in the graph may have an arbitrary number of neighboring nodes. Let $|\mathcal{N}_{\mathcal{G}}(i)|$ denote the number of neighboring graph nodes. To obtain a fixed feature dimension for the inputs of the Graph LSTM unit during network training, the hidden states $\bar{\mathbf{h}}_{i,t}$ used for computing the LSTM gates of the node v_i are obtained by averaging the hidden states of neighboring nodes, computed as:

$$\bar{\mathbf{h}}_{i,t} = \frac{\sum_{j \in \mathcal{N}_{\mathcal{G}}(i)}(\mathbb{1}(q_j = 1)\mathbf{h}_{j,t+1} + \mathbb{1}(q_j = 0)\mathbf{h}_{j,t})}{|\mathcal{N}_{\mathcal{G}}(i)|}. \tag{1}$$

Adaptive Forget Gates. Note that unlike the traditional LSTMs [1][46], the Graph LSTM specifies different forget gates for different neighboring nodes by functioning the input states of the current node with their hidden states, defined as $\bar{g}_{ij}^f, j \in \mathcal{N}_{\mathcal{G}}(i)$. It results in the different influences of neighboring nodes on the updated memory states $\mathbf{m}_{i,t+1}$ and hidden states $\mathbf{h}_{i,t+1}$. The memory states of each neighboring node are also utilized to update the memory states $\mathbf{m}_{i,t+1}$ of the current node. The shared weight metrics U^{fn} for all nodes are learned to guarantee the spatial transformation invariance and enable the learning with various neighbors. The intuition is that each pair of neighboring superpixels may be endowed with distinguished semantic correlations compared to other pairs.

Graph LSTM Unit. The Graph LSTM consists of four gates: the input gate g^u, the forget gate g^f, the adaptive forget gate $\bar{g}^f$, the memory gate g^c and the output gate g^o. The W^u, W^f, W^c, W^o are the recurrent gate weight matrices specified for input fea-

tures while U^u, U^f, U^c, U^o are those for hidden states of each node. $U^{un}, U^{fn}, U^{cn}, U^{on}$ are the weight parameters specified for states of neighboring nodes. The hidden and memory states by the Graph LSTM can be updated as follows:

$$
\begin{aligned}
g_i^u &= \delta(W^u \mathbf{f}_{i,t+1} + U^u \mathbf{h}_{i,t} + U^{un} \bar{\mathbf{h}}_{i,t} + b^u), \\
\bar{g}_{ij}^f &= \delta(W^f \mathbf{f}_{i,t+1} + U^{fn} \mathbf{h}_{j,t} + b^f), \\
g_i^f &= \delta(W^f \mathbf{f}_{i,t+1} + U^f \mathbf{h}_{i,t} + b^f), \\
g_i^o &= \delta(W^o \mathbf{f}_{i,t+1} + U^o \mathbf{h}_{i,t} + U^{on} \bar{\mathbf{h}}_{i,t} + b^o), \\
g_i^c &= \tanh(W^c \mathbf{f}_{i,t+1} + U^c \mathbf{h}_{i,t} + U^{cn} \bar{\mathbf{h}}_{i,t} + b^c), \qquad (2) \\
\mathbf{m}_{i,t+1} &= \frac{\sum_{j \in \mathcal{N}_{\mathcal{G}}(i)} (\mathbb{1}(q_j = 1)\bar{g}_{ij}^f \odot \mathbf{m}_{j,t+1} + \mathbb{1}(q_j = 0)\bar{g}_{ij}^f \odot \mathbf{m}_{j,t})}{|\mathcal{N}_{\mathcal{G}}(i)|} \\
&\quad + g_i^f \odot \mathbf{m}_{i,t} + g_i^u \odot g_i^c, \\
\mathbf{h}_{i,t+1} &= \tanh(g_i^o \odot \mathbf{m}_{i,t+1}).
\end{aligned}
$$

Here δ is the logistic sigmoid function, and $\odot$ indicates a point-wise product. The memory states $\mathbf{m}_{i,t+1}$ of the node v_i are updated by combining the memory states of visited nodes and those of unvisited nodes by using the adaptive forget gates. Let $\mathbf{W}, \mathbf{U}$ denote the concatenation of all weight matrices and $\{\mathbf{Z}_{j,t}\}_{j \in \mathcal{N}_{\mathcal{G}}(i)}$ represent all related information of neighboring nodes. We can thus use G-LSTM$(\cdot)$ to shorten Eqn. (2) as

$$
(\mathbf{h}_{i,t+1}, \mathbf{m}_{i,t+1}) = \text{G-LSTM}(\mathbf{f}_{i,t+1}, \mathbf{h}_{i,t}, \mathbf{m}_{i,t}, \{\mathbf{Z}_{j,t}\}_{j \in \mathcal{N}_{\mathcal{G}}(i)}, \mathbf{W}, \mathbf{U}, \mathcal{G}). \qquad (3)
$$

The mechanism acts as a memory system, where the information can be written into the memory states and sequentially recorded by each graph node, which is then used to communicate with the hidden states of subsequent graph nodes and previous Graph LSTM layer. The back propagation is used to train all the weight metrics.

4 Experiments

Dataset: We evaluate the performance of the proposed Graph LSTM structure on semantic object parsing on four challenging datasets.

PASCAL-Person-Part dataset [25]. The public PASCAL-Person-part dataset concentrates on the human part segmentation annotated by Chen et al. [25] from PASCAL VOC 2010 dataset. The dataset contains detailed part annotations for every person. Following [12][13], the annotations are merged to be Head, Torso, Upper/Lower Arms and Upper/Lower Legs, resulting in six person part classes and one background class. 1,716 images are used for training and 1,817 for testing.

Horse-Cow parsing dataset [26]. The Horse-Cow parsing dataset is a part segmentation benchmark introduced in [26]. For each class, most observable instances from PASCAL VOC 2010 benchmark [47] are manually selected, including 294 training images and 227 testing images. Each image pixel is elaborately labeled as one of the four part classes, including head, leg, tail and body.

ATR dataset [27] and Fashionista dataset [28]. Human parsing aims to predict every pixel of each image with 18 labels: face, sunglass, hat, scarf, hair, upper-clothes,

Table 1. Comparison of object parsing performance with four state-of-the-art methods over the PASCAL-Person-Part dataset [26].

Method	head	torso	u-arms	l-arms	u-legs	l-legs	Bkg	Avg
DeepLab-LargeFOV [15]	78.09	54.02	37.29	36.85	33.73	29.61	92.85	51.78
HAZN [12]	80.79	59.11	43.05	42.76	38.99	34.46	93.59	56.11
Attention [13]	-	-	-	-	-	-	-	56.39
LG-LSTM [21]	**82.72**	60.99	45.40	**47.76**	42.33	37.96	88.63	57.97
Graph LSTM	82.69	**62.68**	**46.88**	47.71	**45.66**	**40.93**	**94.59**	**60.16**

left-arm, right-arm, belt, pants, left-leg, right-leg, skirt, left-shoe, right-shoe, bag, dress and null. Originally, 7,700 images are included in the ATR dataset [27], with 6,000 for training, 1,000 for testing and 700 for validation. 10,000 real-world human pictures are further collected by [42] to cover images with more challenging poses, occlusion and clothes variations. We follow the training and testing settings used in [42].

Evaluation metric: The standard intersection over union (IOU) criterion and pixelwise accuracy are adopted for evaluation on PASCAL-Person-Part dataset and Horse-Cow parsing dataset, following [26][36][13]. We use the same evaluation metrics as in [37][27][42] for evaluation on two human parsing datasets, including accuracy, average precision, average recall, and average F-1 score.

Network architecture: For fair comparison with [16][12][13], our network is based on the publicly available model, "DeepLab-CRF-LargeFOV" [15] for the PASCAL-Person-Part and Horse-Cow parsing dataset, which slightly modifies VGG-16 net [48] to FCN [24]. For fair comparing with [21][42] on two human parsing datasets, the basic "Co-CNN" structure proposed in [42] is utilized due to its leading accuracy. Our networks based on "Co-CNN" are trained from the scratch following the same setting in [42].

Training: We use the same data augmentation techniques for the object part segmentation and human parsing as in [16] and [42], respectively. The scale of the input image is fixed as 321×321 for training networks based on "DeepLab-CRF-LargeFOV". Based on "Co-CNN", the input image is rescaled to 150×100 as in [42]. We use the SLIC over-segmentation method [45] to generate averagely 1,000 superpixels for each image. Two training steps are employed to train the networks. First, we train the convolutional layer with 1×1 filters to generate initial confidence maps that are used to produce the starting node and the update sequence for all nodes in Graph LSTM. Then, the whole network is fine-tuned based on the pretrained model to produce final parsing results. In each step, the learning rate of the newly added layers, including Graph LSTM layers and convolutional layers is initialized as 0.001 and that of other previously learned layers, is initialized as 0.0001. All weight matrices used in the Graph LSTM units are randomly initialized from a uniform distribution of [-0.1, 0.1]. The Graph LSTM predicts the hidden and memory states with the same dimension as in the previous convolutional layers. We only use two Graph LSTM layers for all models since only slight improvements are observed by using more Graph LSTM layers, which also consumes more computation resources. We fine-tune the networks on "DeepLab-CRF-LargeFOV" for roughly 60 epochs and it takes about 1 day. For training based on "Co-CNN" from scratch, it takes

Table 2. Comparison of object parsing performance with five state-of-the-art methods over the Horse-Cow object parsing dataset [26].

				Horse				
Method	Bkg	head	body	leg	tail	Fg	IOU	Pix.Acc
SPS [26]	79.14	47.64	69.74	38.85	-	68.63	-	81.45
HC [36]	85.71	57.30	77.88	51.93	37.10	78.84	61.98	87.18
Joint [16]	87.34	60.02	77.52	58.35	51.88	80.70	65.02	88.49
LG-LSTM [21]	89.64	66.89	84.20	60.88	42.06	82.50	68.73	90.92
HAZN [12]	90.87	70.73	84.45	63.59	51.16	-	72.16	-
Graph LSTM	**91.73**	**72.89**	**86.34**	**69.04**	**53.76**	**87.51**	**74.75**	**92.76**
				Cow				
Method	Bkg	head	body	leg	tail	Fg	IOU	Pix.Acc
SPS [26]	78.00	40.55	61.65	36.32	-	71.98	-	78.97
HC [36]	81.86	55.18	72.75	42.03	11.04	77.04	52.57	84.43
Joint [16]	85.68	58.04	76.04	51.12	15.00	82.63	57.18	87.00
LG-LSTM [21]	89.71	68.43	82.47	53.93	19.41	85.41	62.79	90.43
HAZN [12]	90.66	**75.10**	83.30	57.17	28.46	-	66.94	-
Graph LSTM	**91.54**	73.88	**85.92**	**63.67**	**35.22**	**88.42**	**70.05**	**92.43**

Table 3. Comparison of human parsing performance with seven state-of-the-art methods when evaluating on ATR dataset [27].

Method	Acc.	F.g. acc.	Avg. prec.	Avg. recall	Avg. F-1 score
Yamaguchi et al. [28]	84.38	55.59	37.54	51.05	41.80
PaperDoll [37]	88.96	62.18	52.75	49.43	44.76
M-CNN [41]	89.57	73.98	64.56	65.17	62.81
ATR [27]	91.11	71.04	71.69	60.25	64.38
Co-CNN [42]	95.23	80.90	81.55	74.42	76.95
Co-CNN (more) [42]	96.02	83.57	84.95	77.66	80.14
LG-LSTM [21]	96.18	84.79	84.64	79.43	80.97
LG-LSTM (more) [21]	96.85	87.35	85.94	82.79	84.12
CRFasRNN (more) [10]	96.34	85.10	84.00	80.70	82.08
Graph LSTM	97.60	91.42	84.74	83.28	83.76
Graph LSTM (more)	**97.99**	**93.06**	**88.81**	**87.80**	**88.20**

about 4-5 days. In the testing stage, one image takes 0.5 second on average except for the superpixel extraction step.

4.1 Results and Comparisons

We compare the proposed Graph LSTM structure with several state-of-the-art methods on four public datasets.

PASCAL-Person-Part dataset [26]: We report the results and the comparisons with four recent state-of-the-art methods [15][12][13][21] in Table 1. The results of "DeepLab-LargeFOV" were originally reported in [12]. The proposed Graph LSTM structure substantially outperforms these baselines in terms of average IoU metric. In particular, for the semantic parts with more likely confusions such as upper-arms and lower-arms, the Graph LSTM provides considerably better prediction than baselines, e.g., 4.95% and 6.67% higher over [12] for lower-arms and upper-legs, respectively. This superior performance achieved by Graph LSTM demonstrates the effectiveness of exploiting global context to boost local prediction.

Table 4. Comparison of human parsing performance with five state-of-the-art methods on the test images of Fashionista [28].

Method	Acc.	F.g. acc.	Avg. prec.	Avg. recall	Avg. F-1 score
Yamaguchi et al. [28]	87.87	58.85	51.04	48.05	42.87
PaperDoll [37]	89.98	65.66	54.87	51.16	46.80
ATR [27]	92.33	76.54	73.93	66.49	69.30
Co-CNN [42]	96.08	84.71	82.98	77.78	79.37
Co-CNN (more) [42]	97.06	89.15	87.83	81.73	83.78
LG-LSTM [21]	96.85	87.71	87.05	82.14	83.67
LG-LSTM (more) [21]	97.66	91.35	89.54	85.54	86.94
Graph LSTM	97.93	92.78	88.24	87.13	87.57
Graph LSTM (more)	**98.14**	**93.75**	**90.15**	**89.46**	**89.75**

Table 5. Per-Class Comparison of F-1 scores with six state-of-the-art methods on ATR [27].

Method	Hat	Hair	S-gls	U-cloth	Skirt	Pants	Dress	Belt	L-shoe	R-shoe	Face	L-leg	R-leg	L-arm	R-arm	Bag	Scarf
Yamaguchi et al. [28]	8.44	59.96	12.09	56.07	17.57	55.42	40.94	14.68	38.24	38.33	72.10	58.52	57.03	45.33	46.65	24.53	11.43
PaperDoll [37]	1.72	63.58	0.23	71.87	40.20	69.35	59.49	16.94	45.79	44.47	61.63	52.19	55.60	45.23	46.75	30.52	2.95
M-CNN [41]	80.77	65.31	35.55	72.58	77.86	70.71	81.44	38.45	53.87	48.57	72.78	63.25	68.24	57.40	51.12	57.87	43.38
ATR [27]	77.97	68.18	29.20	79.39	80.36	79.77	82.02	22.88	53.51	50.26	74.71	69.07	71.69	53.79	58.57	53.66	57.07
Co-CNN [42]	72.07	86.33	72.81	85.72	70.82	83.05	69.95	37.66	76.48	76.80	89.02	85.49	85.23	84.16	84.04	81.51	44.94
Co-CNN more [42]	75.88	89.97	81.26	87.38	71.94	84.89	71.03	40.14	81.43	81.49	92.73	88.77	88.48	89.00	88.71	83.81	46.24
LG-LSTM (more) [21]	81.13	90.94	81.07	88.97	80.91	91.47	77.18	60.32	83.40	83.65	93.67	92.27	92.41	90.20	90.13	85.78	51.09
Graph LSTM (more)	**85.30**	90.47	72.77	**95.11**	**97.31**	**96.58**	**96.43**	**68.55**	**85.27**	**84.35**	92.70	91.13	**93.17**	**91.20**	81.00	**90.83**	**66.09**

Horse-Cow Parsing dataset [26]: Table 2 shows the comparison results with five state-of-the-art methods on the overall metrics. The proposed Graph LSTM gives a huge boost in average IOU. For example, Graph LSTM achieves 70.05%, 7.26% better than LG-LSTM [21] and 3.11% better than HAZN [12] for the cow class. Large improvement, i.e. 2.59% increase by Graph LSTM in IOU over the best performing state-of-the-art method, can also be observed from the comparisons on horse class.

ATR dataset [27]: Table 3 and Table 5 report the comparison performance with seven state-of-the-arts on overall metrics and F-1 scores of individual semantic labels, respectively. The proposed Graph LSTM can significantly outperform these baselines, particularly, 83.76% vs 76.95% of Co-CNN [42] and 80.97% of LG-LSTM [21] in terms of average F-1 score. Following [42], we also take the additional 10,000 images in [42] as extra training images and report the results as "Graph LSTM (more)". The "Graph LSTM (more)" can also improve the average F-1 score by 4.08% over "LG-LSTM (more)". We show the F-1 score for each label in Table 5. Generally, our Graph LSTM shows much higher performance than other baselines. In addition, our "Graph LSTM (more)" significantly outperforms "CRFasRNN (more)" [10], verifying the superiority of Graph LSTM over the pair-wise terms in CRF in capturing global context. The results of "CRFasRNN (more)" [10] are obtained by training the network using their public code.

Fashionista dataset [28]: Table 4 gives the comparison results on the Fashionista dataset. Following [27], we only report the performance by training on the same large ATR dataset [27] and then testing on the 229 images of the Fashionista dataset. Our Graph LSTM architecture can substantially outperform the baselines by a large gain.

Table 6. Performance comparisons of using different LSTM structures and taking the superpixel smoothing as the post-processing step when evaluating on PASCAL-Person-Part dataset.

Method	head	torso	u-arms	l-arms	u-legs	l-legs	Bkg	Avg
Grid LSTM [1]	81.85	58.85	43.10	46.87	40.07	34.59	85.97	55.90
Row LSTM [2]	82.60	60.13	44.29	47.22	40.83	35.51	87.07	56.80
Diagonal BiLSTM [2]	82.67	60.64	45.02	47.59	41.95	37.32	88.16	57.62
LG-LSTM [21]	82.72	60.99	45.40	47.76	42.33	37.96	88.63	57.97
Diagonal BiLSTM [2] + superpixel smoothing	82.91	61.34	46.01	48.07	42.56	37.91	89.21	58.29
LG-LSTM [21] + superpixel smoothing	**82.98**	61.58	46.27	**48.08**	42.94	38.55	89.66	58.58
Graph LSTM	82.69	**62.68**	**46.88**	47.71	**45.66**	**40.93**	**94.59**	**60.16**

Table 7. Performance comparisons with different node updating schemes when evaluating on PASCAL-Person-Part dataset.

Method	head	torso	u-arms	l-arms	u-legs	l-legs	Bkg	Avg
BFS (location)	**83.00**	61.63	46.18	**48.01**	44.09	38.71	93.82	58.63
BFS (confidence)	82.97	62.20	46.70	48.00	44.02	39.00	90.86	59.11
DFS (location)	82.85	61.25	45.89	48.02	42.50	38.10	89.04	58.23
DFS (confidence)	82.89	62.31	46.76	48.04	44.24	39.07	91.18	59.21
Graph LSTM (confidence-driven)	82.69	**62.68**	**46.88**	47.71	**45.66**	**40.93**	**94.59**	**60.16**

4.2   Discussions

Graph LSTM vs locally fixed factorized LSTM. To show the superiority of the Graph LSTM compared to previous locally fixed factorized LSTM [2][21][1], Table 6 gives the performance comparison among different LSTM structures. These variants use the same network architecture and only replace the Graph LSTM layer with the traditional fixedly factorized LSTM layer, including Row LSTM [2], Diagonal BiLSTM [2], LG-LSTM [21] and Grid LSTM [1]. The experimented Grid LSTM [1] is a simplified version of Diagnocal BiLSTM [2] where only the top and left pixels are considered. Their basic structures are presented in Fig. 2. It can be observed that using richer local contexts (i.e., number of neighbors) to update the states of each pixel can lead to better parsing performance. In average, there are six neighboring nodes for each superpixel node in the constructed graph topologies in Graph LSTM. Although the LG-LSTM [21] has employed eight neighboring pixels to guide local prediction, its performance is still worse than our Graph LSTM.

Graph LSTM vs superpixel smoothing. In Table 6, we further demonstrate that the performance gain by Graph LSTM is not just from using more accurate boundary information provided by superpixels. The superpixel smoothing can be used as a post-processing step to refine confidence maps by previous LSTMs. By comparing "Diagonal BiLSTM [2] + superpixel smoothing" and "LG-LSTM [21] + superpixel smoothing" with our "Graph LSTM", we can find that the Graph LSTM can still bring more performance gain benefiting from its advanced information propagation based on the graph-structured representation.

Node updating scheme. Different node updating schemes to update the states of all nodes are further investigated in Table 7. The Breadth-first search (BFS) and Depth-first search (DFS) are the traditional algorithms to search graph data structures. For one par-

Table 8. Performance comparisons of using the confidence-drive scheme based on confidences on different foreground labels when evaluating on PASCAL-Person-Part dataset.

Foreground label	head	torso	u-arms	l-arms	u-legs	l-legs	Avg
Avg IoU	61.03	61.45	60.03	59.23	60.49	59.89	60.35

ent node, selecting different children nodes to first update may lead to different updated hidden states for all nodes. Two ways of selecting first children nodes for updating are thus evaluated: "BFS (location)" and "DFS (location)" choose the spatially leftmost node among all children nodes to update first while "BFS (confidence)" and "DFS (confidence)" select the child node with maximal confidence on all foreground classes. We find that using our confidence-driven scheme can achieve better performance than other alternative ones. The possible reason may be that the features of superpixel nodes with higher foreground confidences embed more accurate semantic meanings and thus lead to more reliable global reasoning.

Note that we use the ranking of confidences on all foreground classes to generate the node updating scheme. In Table 8, we extensively test the performance of using the initial confidence maps of different foreground labels to produce the node updating sequence. In average, only slight performance differences are observed when using the confidences of different foreground labels. In particular, using the confidences of "head" and "torso" leads to improved performance over using those of all foreground classes, i.e., 61.03% and 61.45% vs 60.16%. It is possible because the segmentation of head/torso are more reliable in the person parsing case, which further verifies that the reliability of nodes in the updating order is important. It is difficult to determine the best semantic label for each task, hence we just use the one over all the foreground labels for simplicity and efficiency in implementation.

Adaptive forget gates. In Graph LSTM, adaptive forget gates are adopted to treat the local contexts from different neighbors differently. The superiority of using adaptive forget gates can be verified in Table 9. "Identical forget gates" shows the results of learning identical forget gates for all neighbors and simultaneously ignoring the memory states of neighboring nodes. Thus in "Identical forget gates", the g_i^f and $\mathbf{m}_{i,t+1}$ in Eqn. (2) can be simply computed as

$$
\begin{aligned}
g_i^f &= \delta(W^f \mathbf{f}_{i,t+1} + U^f \mathbf{h}_{i,t} + U^{fn} \bar{\mathbf{h}}_{i,t} + b^f), \\
\mathbf{m}_{i,t+1} &= g_i^f \odot \mathbf{m}_{i,t} + g_i^u \odot g_i^c.
\end{aligned}
\tag{4}
$$

It can be observed that learning adaptive forgets gates in Graph LSTM shows better performance over learning identical forget gates for all neighbors on the object parsing task, as diverse semantic correlations with local context can be considered and treated differently during the node updating. Compared to Eqn. (4), no extra parameters is brought to specify adaptive forget gates due to the usage of the shared parameters U^{fn} in Eqn. (2).

Superpixel number. The drawback of using superpixels is that superpixels may introduce quantization errors whenever pixels within one superpixel have different ground

Table 9. Comparisons of parsing performance by the version with or without learning adaptive forget gates for different neighboring nodes when evaluating on PASCAL-Person-Part dataset.

Method	head	torso	u-arms	l-arms	u-legs	l-legs	Bkg	Avg
Identical forget gates	**82.89**	62.31	46.76	**48.04**	44.24	39.07	91.18	59.21
Graph LSTM (dynamic forget gates)	82.69	**62.68**	**46.88**	47.71	**45.66**	**40.93**	**94.59**	**60.16**
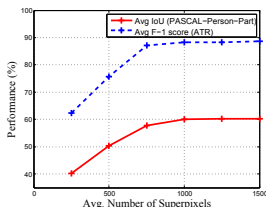
Fig. 4. Performance comparisons with six averaged numbers of superpixels when evaluating on PASCAL-Person-Part and ATR datasets, including 250, 500, 750, 1000, 1250, 1500 .

truth labels. We thus evaluate the performance of using different average numbers of superpixels to construct the graph structure. As shown in Fig. 4, there are slight improvements when using over 1,000 superpixels. We thus use averagely 1,000 superpixels for each image in all our experiments.
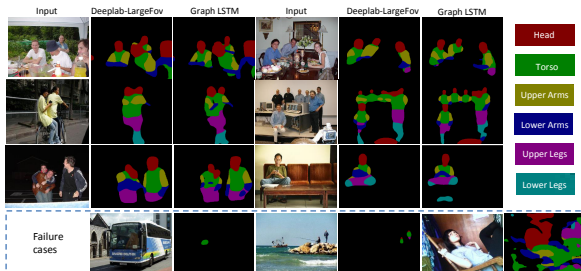

Fig. 5. Comparison of parsing results of our Graph LSTM and the baseline "DeepLab-LargeFov" and some failure cases by our Graph LSTM on PASCAL-Person-Part.

Residual connections. Residual connections were first proposed in [44] to better train very deep convolutional layers. The version in which the residual connections are eliminated achieves 59.12% in terms of Avg IoU on PASCAL-Person-Part dataset. It demonstrates that residual connections between Graph LSTM layers can also help boost the performance, i.e., 60.16% vs 59.12%. Note that our Graph LSTM version without using residual connections is still significantly better than all baselines in Table 1.

4.3   More Visual Comparison and Failure cases

The qualitative comparisons of parsing results on PASCAL-Person-Part and ATR dataset are visualized in Fig. 5 and Fig. 6, respectively. In general, our Graph-LSTM outputs more reasonable results for confusing labels by effectively exploiting global context to assist the local prediction. We also show some failure cases on each dataset.
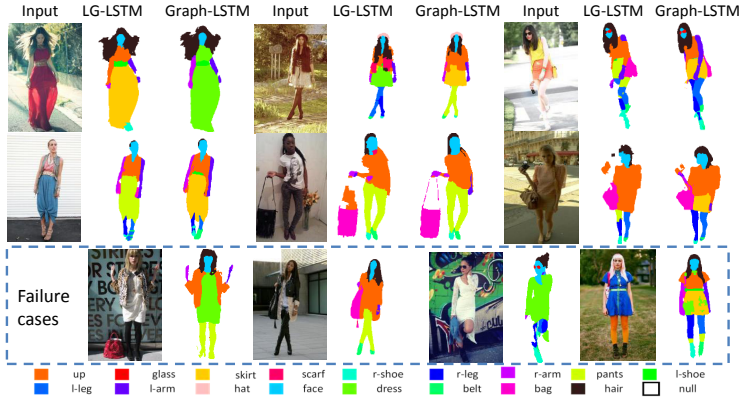

Fig. 6. Parsing result comparisons of our Graph LSTM and the LG-LSTM [21] and some failure cases by our Graph LSTM on ATR dataset.

5   Conclusion and Future Work

In this work, we proposed a novel Graph LSTM network to address the fundamental semantic object parsing task. Our Graph LSTM generalizes the existing LSTMs into the graph-structured data. The adaptive graph topology for each image is constructed by connecting the arbitrary-shaped superpixels nodes via their spatial neighborhood connections. The confidence-driven scheme is used to adaptively select the starting node and determine the node updating sequence. The Graph LSTM can thus sequentially update the states of all nodes. Comprehensive evaluations on four public semantic object parsing datasets well demonstrate the significant superiority of our graph LSTM. In future, we will explore how to dynamically adjust the graph structure to directly produce the semantic masks according to the connected superpixel nodes.

References

1. Kalchbrenner, N., Danihelka, I., Graves, A.: Grid long short-term memory. In: ICLR. (2016)
2. van den Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: ICML. (2016)
3. Zhang, H., Kim, G., Xing, E.P.: Dynamic topic modeling for monitoring market competition from online text and image data. In: ACM SIGKDD, ACM (2015) 1425–1434
4. Zhang, H., Hu, Z., Wei, J., Xie, P., Kim, G., Ho, Q., Xing, E.: Poseidon: A system architecture for efficient gpu-based deep learning on multiple machines. arXiv preprint arXiv:1512.06216 (2015)
5. Zhao, R., Ouyang, W., Wang, X.: Unsupervised salience learning for person re-identification. In: CVPR. (2013) 3586–3593
6. Wang, Y., Tran, D., Liao, Z., Forsyth, D.: Discriminative hierarchical part-based models for human parsing and action recognition. JMLR **13**(1) (2012) 3075–3102
7. Gan, C., Wang, N., Yang, Y., Yeung, D.Y., Hauptmann, A.G.: Devnet: A deep event network for multimedia event detection and evidence recounting. In: CVPR. (2015) 2568–2577
8. Gan, C., Lin, M., Yang, Y., de Melo, G., Hauptmann, A.G.: Concepts not alone: Exploring pairwise relationships for zero-shot video activity recognition. In: AAAI. (2016)
9. Liang, X., Wei, Y., Shen, X., Yang, J., Lin, L., Yan, S.: Proposal-free network for instance-level object segmentation. arXiv preprint arXiv:1509.02636 (2015)
10. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.: Conditional random fields as recurrent neural networks. In: ICCV. (2015)
11. Liang, X., Liu, S., Wei, Y., Liu, L., Lin, L., Yan, S.: Towards computational baby learning: A weakly-supervised approach for object detection. In: ICCV. (2015) 999–1007
12. Xia, F., Wang, P., Chen, L.C., Yuille, A.L.: Zoom better to see clearer: Huamn part segmentation with auto zoom net. In: ECCV. (2016)
13. Chen, L.C., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: Scale-aware semantic image segmentation. In: CVPR. (2016)
14. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV. (2015)
15. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In ICLR (2015)
16. Wang, P., Shen, X., Lin, Z., Cohen, S., Price, B., Yuille, A.: Joint object and part segmentation using deep learned potentials. In: ICCV. (2015)
17. Gadde, R., Jampani, V., Kiefel, M., Gehler, P.V.: Superpixel convolutional networks using bilateral inceptions. In: ICLR. (2016)
18. Byeon, W., Liwicki, M., Breuel, T.M.: Texture classification using 2d lstm networks. In: ICPR. (2014) 1144–1149
19. Theis, L., Bethge, M.: Generative image modeling using spatial lstms. In: NIPS. (2015)
20. Byeon, W., Breuel, T.M., Raue, F., Liwicki, M.: Scene Labeling with LSTM Recurrent Neural Networks. In: CVPR. (2015) 3547–3555
21. Liang, X., Shen, X., Xiang, D., Feng, J., Lin, L., Yan, S.: Semantic object parsing with local-global long short-term memory. In: CVPR. (2016)
22. Sharma, S., Kiros, R., Salakhutdinov, R.: Action recognition using visual attention. arXiv preprint arXiv:1511.04119 (2015)
23. Mnih, V., Heess, N., Graves, A., et al.: Recurrent models of visual attention. In: Advances in Neural Information Processing Systems. (2014) 2204–2212
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. (2015)

25. Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., et al.: Detect what you can: Detecting and representing objects using holistic models and body parts. In: CVPR. (2014) 1979–1986

26. Wang, J., Yuille, A.: Semantic part segmentation using compositional model combining shape and appearance. In: CVPR. (2015)

27. Liang, X., Liu, S., Shen, X., Yang, J., Liu, L., Dong, J., Lin, L., Yan, S.: Deep human parsing with active template regression. TPAMI (2015)

28. Yamaguchi, K., Kiapour, M., Ortiz, L., Berg, T.: Parsing clothing in fashion photographs. In: CVPR. (2012)

29. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: NIPS. (2009) 545–552

30. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS. (2014) 3104–3112

31. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R.S., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML. (2015) 2048–2057

32. Graves, A., Fernandez, S., Schmidhuber, J.: Multi-dimensional recurrent neural networks. In: ICANN. (2007)

33. Stollenga, M.F., Byeon, W., Liwicki, M., Schmidhuber, J.: Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. arXiv preprint arXiv:1506.07452 (2015)

34. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075 (2015)

35. Lu, W., Lian, X., Yuille, A.: Parsing semantic parts of cars using graphical models and segment appearance consistency. In: BMVC. (2014)

36. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: CVPR. 447–456

37. Yamaguchi, K., Kiapour, M., Berg, T.: Paper doll parsing: Retrieving similar styles to parse clothing items. In: ICCV. (2013)

38. Dong, J., Chen, Q., Xia, W., Huang, Z., Yan, S.: A deformable mixture parsing model with parselets. In: ICCV. (2013)

39. Wang, N., Ai, H.: Who blocks who: Simultaneous clothing segmentation for grouping images. In: ICCV. (2011)

40. Simo-Serra, E., Fidler, S., Moreno-Noguer, F., Urtasun, R.: A High Performance CRF Model for Clothes Parsing. In: ACCV. (2014)

41. Liu, S., Liang, X., Liu, L., Shen, X., Yang, J., Xu, C., Lin, L., Cao, X., Yan, S.: Matching-CNN Meets KNN: Quasi-Parametric Human Parsing. In: CVPR. (2015)

42. Liang, X., Xu, C., Shen, X., Yang, J., Liu, S., Tang, J., Lin, L., Yan, S.: Human parsing with contextualized convolutional neural network. In: ICCV. (2015)

43. Schwing, A.G., Urtasun, R.: Fully connected deep structured networks. arXiv preprint arXiv:1503.02351 (2015)

44. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)

45. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels. Technical report (2010)

46. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997) 1735–1780

47. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge 2010 (voc2010) results (2010)

48. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)