Exploring Context with Deep Structured models for Semantic Segmentation

Guosheng Lin, Chunhua Shen, Anton van den Hengel, Ian Reid

Abstract—State-of-the-art semantic image segmentation methods are mostly based on training deep convolutional neural networks (CNNs). In this work, we proffer to improve semantic segmentation with the use of contextual information. In particular, we explore patch-patch context and patch-background context in deep CNNs. We formulate deep structured models by combining CNNs and Conditional Random Fields (CRFs) for learning the patch-patch context between image regions. Specifically, we formulate CNN-based pairwise potential functions to capture semantic correlations between neighboring patches. Efficient piecewise training of the proposed deep structured model is then applied in order to avoid repeated expensive CRF inference during the course of back propagation.For capturing the patch-background context, we show that a network design with traditional multi-scale image inputs and sliding pyramid pooling is very effective for improving performance. We perform comprehensive evaluation of the proposed method. We achieve new state-of-the-art performance on a number of challenging semantic segmentation datasets.

Index Terms—Semantic Segmentation, Convolutional Neural Networks, Conditional Random Fields, Contextual Models

CONTENTS

1 Introduction 2
1.1 Related work 2

2 Modeling semantic pairwise relations 3

3 Contextual Deep CRFs 4
3.1 Unary potential functions 4
3.2 Pairwise potential functions 4
3.2.1 Asymmetric pairwise potentials 5

4 Exploring background context 6

5 Network configurations 6

6 Prediction 7
6.1 Coarse-level prediction stage 7
6.2 Prediction refinement stage 8

7 CRF training 8
7.1 Piecewise training of CRFs 8

8 Implementation details 9
8.1 Efficient learning 9
8.2 Asynchronous gradient update 9

9 Experiments 10
9.1 Results on the NYUDv2 dataset 10
9.1.1 Component evaluation 11
9.1.2 Comparison with multi-unary ensemble 11
9.2 Results on the PASCAL VOC 2012 dataset 12
9.3 Results on the Cityscapes dataset 12
9.4 Results on the PASCAL-Context dataset 12
9.5 Results on the SUN-RGBD dataset . . . 13
9.6 Results on the COCO dataset 13
9.7 Results on the SIFT-flow dataset 13
9.8 Results on the KITTI dataset 14

10 Conclusions 14

References 14

- *G. Lin is with School of Computer Science and Engineering, Nanyang Technological University, Singapore. This work was done when G. Lin was with The University of Adelaide and Australian Centre for Robotic Vision. E-mail: guosheng.lin@gmail.com*
  C. Shen, A. van den Hengel and I. Reid are with School of Computer Science, The University of Adelaide, Australia; and Australian Centre for Robotic Vision. E-mail: {chunhua.shen, anton.vandenhengel, ian.reid}@adelaide.edu.au
- *C. Shen is the corresponding author.*

1 Introduction

Semantic image segmentation aims to predict a category label for every image pixel, which is an important yet challenging task for image understanding. Recent approaches have applied convolutional neural network (CNNs) [5], [17], [39] to this pixel-level labeling task and achieved remarkable success. Among these CNN-based methods, fully convolutional neural networks (FCNNs) [5], [39] have become a popular choice, because of their computational efficiency for dense prediction and end-to-end style learning.

Contextual relationships are ubiquitous and provide important cues for scene understanding tasks. Spatial context can be formulated in terms of semantic compatibility relations between one object and its neighboring objects or image patches (stuff), in which a compatibility relation is an indication of the co-occurrence of visual patterns. For example, a car is likely to appear over a road, and a glass is likely to appear over a table. Context can also encode incompatibility relations. For example, a boat is unlikely to appear on a road. These relations also exist at finer scales, for example, in object part-to-part relations, and part-to-object relations. In some cases, contextual information is the most important cue, particularly when a single object shows significant visual ambiguities. A more detailed discussion of the value of spatial context can be found in [26].

We explore two types of spatial context to improve the segmentation performance: patch-patch context and patch-background context. The patch-patch context is the semantic relation between the visual patterns of two image patches. Likewise, patch-background context the semantic relation between an image patch and a large background region.

Explicitly modeling the patch-patch contextual relations has not been well studied in recent CNN-based segmentation methods. In this work, we propose to explicitly model the contextual relations using conditional random fields (CRFs). We formulate CNN-based pairwise potential functions to capture semantic correlations between neighboring patches. Some recent methods combine CNNs and CRFs for semantic segmentation, e.g., the dense CRFs applied in [5], [8], [48], [60]. The purpose of applying the dense CRFs in these methods is to refine the upsampled low-resolution prediction to sharpen object/region boundaries. These methods consider Potts-model-based pairwise potentials for enforcing local smoothness. There the pairwise potentials are conventional log-linear functions. In contrast, here we learn more general pairwise potentials using CNNs to model the semantic compatibility between image regions. Our CNN pairwise potentials aim to improve the coarse-level prediction rather than merely encouraging local smoothness, and thus have a different purpose compared to Potts-model-based pairwise potentials. Given that these two types of potentials make different effects, they can be combined to improve segmentation results. Fig. 1 illustrates the prediction process of our method.

In contrast to patch-patch context, patch-background context is widely explored in the literature. For CNN-based methods, background information can be effectively captured by combining features from a multi-scale image network input, and has shown good performance in some recent segmentation methods [17], [40]. A special case of capturing patch-background context is considering the whole image as the background region and incorporating the image-level label information into learning. In our approach, to encode rich background information, we construct multi-scale networks and apply sliding pyramid pooling on feature maps. The traditional pyramid pooling (in a sliding manner) on the feature map is able to capture information from background regions of different sizes.

Incorporating general pairwise potentials usually involves computationally expensive inference, which brings challenges for CRF learning. To facilitate efficient learning we apply piecewise training of the CRF [53] to avoid repeated inference during back propagation training of the deep model.

Thus our main contributions are as follows.

- We formulate CNN-based general pairwise potential functions in CRFs to explicitly model patch-patch semantic relations.
- Deep CNN-based general pairwise potentials are challenging for efficient CNN-CRF joint learning. We perform approximate training, using piecewise training of CRFs [53], to avoid the repeated inference at every stochastic gradient descent iteration and thus achieve efficient learning.
- We explore background context by applying a network architecture with traditional multi-scale image input [17] and sliding pyramid pooling [31]. We empirically demonstrate the effectiveness of this network architecture for semantic segmentation.
- We set new state-of-the-art performance on a number of challenging semantic segmentation datasets, including NYUDv2, PASCAL VOC 2012, PASCAL-Context, SIFT-flow, SUN-RGBD, Cityscapes dataset and so on. In particular, we achieve an intersection-over-union score of 77.8 on the PASCAL VOC 2012 dataset.

1.1 Related work

Preliminary results of our work appeared in [33]. Exploiting contextual information has been widely studied in the literature (e.g., [11], [26], [46]). For example, early work of "TAS" [26] models different types of spatial context between *Things* and *Stuff* using a generative graphical model.

The most successful recent methods for semantic image segmentation are based on CNNs. CNN based methods have shown outstanding performance compared to traditional semantic segmentation methods like TextonBoost [49]. A number of these CNN-based methods are region-proposal-based methods [19], [24], which first generate region proposals and then assign category labels to each of them. Very recently, FCNNs [5], [8], [39] have become a popular choice for their efficient feature generation and end-to-end training. FCNNs have also been applied to a range of other dense-prediction tasks recently, such as image restoration [14], image super-resolution [12] and depth estimation [13], [15], [36]. The method that we propose here is also built upon fully convolution-style networks.

FCNN methods make use of the Image-Net trained CNN models (e.g., the VGG-16 model [51]) which takes

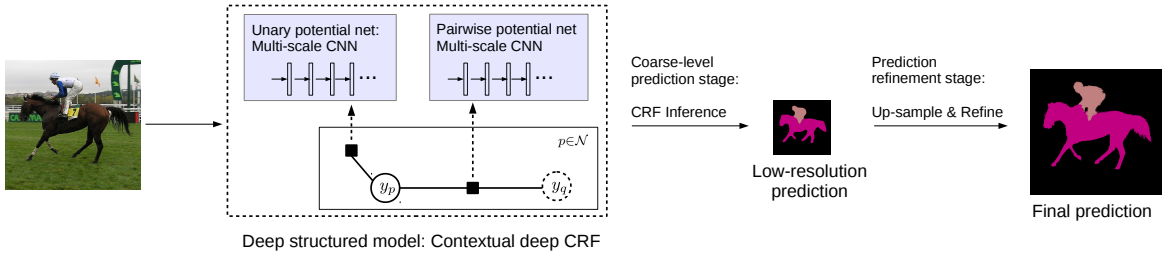


Fig. 1 – An illustration of the prediction process of our method. Both our unary and pairwise potentials are formulated as multi-scale CNNs for capturing semantic relations between image regions. Our method outputs low-resolution prediction after performing CRF inference, then the prediction is up-sampled and refined in a standard post-processing stage to output the final prediction.

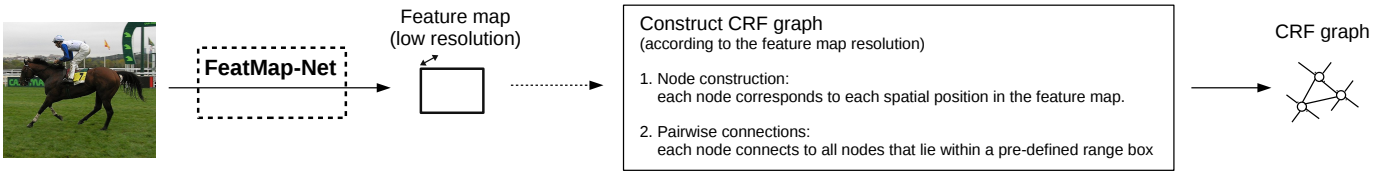


Fig. 2 – An illustration of generating a feature map with FeatMap-Net and constructing the CRF graph.

advantages of the large Image-Net dataset for learning deep models. For convolution and pooling layers, the resolution of the output feature map is down-sampled if the convolution/pooling stride is greater than 1. Usually a few such layers use a stride setting of 2, hence the direct predictions of FCNNs are typically in low resolution. To increase the prediction resolution, the naive method of directly reducing the strides for all layers is not able to address this down-sampled prediction for a deep network. Small strides result in prohibitively expensive computation for a deep network, and also reduce the view-of-field (the image region that a filter is able to "see") of the network layers. Network layers with insufficient view-of-field may not be able to capture high-level semantic patterns and thus degrade the performance.

To address this low-resolution prediction issue, a variety of FCNN based methods are proposed very recently which focus on refining the low-resolution prediction to obtain high resolution prediction. DeepLab-CRF [5] first applies atrous convolution to produce larger size feature maps and performs bilinear upsampling on the prediction score map to the input image size, then they apply the dense CRF method [30] to refine the object boundary by levering low-level (color contrast) information. They consider Potts-model based pairwise potential functions which enforce local smoothness. CRF-RNN [60] extends this approach by implementing the mean field CRF inference as recurrent layers for end-to-end learning of the dense CRF and FCNN network. The work in [42] learns deconvolution layers to upsample the low-resolution predictions. The depth estimation method [37] explores super-pixel pooling for building the gap between the low-resolution feature map and high-resolution final prediction. Eigen *et al.* [13] perform coarse-to-fine learning of multiple networks with different resolution outputs for refining the coarse prediction. The method FCN [39] and Hyper-column [23] explore mid-layer features (skip connections) for high-resolution prediction.

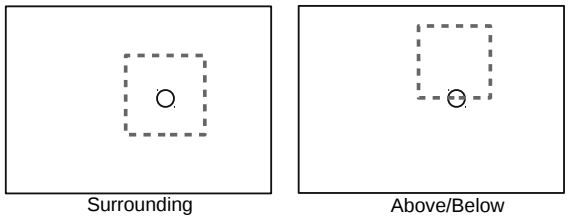


Fig. 3 – An illustration of constructing pairwise connections in a CRF graph. A node is connected to all other nodes which lie inside the range box (dashed box in the figure). Two types of spatial relations are described in the figure, which correspond to two types of pairwise potential functions.

Unlike these methods, our method focuses on improving the coarse (low-resolution) prediction by learning general CNN pairwise potentials to capture semantic relations between patches. These methods are complementary to our method.

Jointly learning CNNs and CRFs has also been explored in other applications apart from segmentation. Recent work in [36], [37] proposes to jointly learn *continuous* CRFs and CNNs for depth estimation from single monocular images. They focus on continuously-valued variable prediction, while our method is for discrete categorical label prediction. The work in [55] combines CRFs and CNNs for human pose estimation. The authors of [6] explore joint training of Markov random fields and deep neural networks for the tasks of predicting words from noisy images and multi-class classification. They require marginal inference for every gradient calculation which is computationally expensive for training deep models.

2 Modeling Semantic Pairwise Relations

We first describe how to build the CRF graph for modeling semantic pairwise relations. Given an image, we first apply a convolutional network to generate a feature map. We

refer to this network as 'FeatMap-Net', details of which are presented in Sec. 4 (Fig. 6 shows the overall architecture). With this feature map, we construct one node in the CRF graph corresponding to one spatial position of the feature map. Fig. 2 illustrates how we construct nodes and pairwise connections in a CRF graph.

Pairwise connections are constructed by connecting one node to all other nodes which lie within a spatial range box (the dashed box in Fig. 3). We consider different spatial relations by defining different types range boxes, and each type of spatial relation is modeled by a specific pairwise potential function. As shown in Fig. 3, our method models the "surrounding" and "above/below" spatial relations. For the surrounding relation, the range box is centered at the node. For the above/below relation, the bottom edge of the range box is centered at the node.

In our experiments, the size of the range box (dash box in the figure) size is $0.4a \times 0.4a$, where a is the length of the short edge of the feature map. It would be straightforward to construct more pairwise potentials, by varying either the sizes or positions of the connection range boxes, and our approach is not limited to connections within "boxes".

3 Contextual Deep CRFs

Here we present the details of our deep CRF model. We denote by $\boldsymbol{x} \in \mathcal{X}$ one input image and $\boldsymbol{y} \in \mathcal{Y}$ the labeling mask which describes the label configuration of each node in the CRF graph. The energy function is denoted by $E(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{\theta})$ which models the compatibility of the input-output pair, with a small output value indicating high confidence in the prediction $\boldsymbol{y}$. All network parameters are denoted by $\boldsymbol{\theta}$ which we need to learn. The conditional likelihood for one image is formulated as follows:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left[-E(\boldsymbol{y}, \boldsymbol{x})\right]. \tag{1}$$

Here Z is the partition function, defined as: $Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp\left[-E(\boldsymbol{y}, \boldsymbol{x})\right]$. The energy function is typically formulated by a set of unary and pairwise potentials:

$$E(\boldsymbol{y}, \boldsymbol{x}) = \sum_{U \in \mathcal{U}} \sum_{p \in \mathcal{N}_U} U(y_p, \boldsymbol{x}_p) + \sum_{V \in \mathcal{V}} \sum_{(p,q) \in \mathcal{S}_V} V(y_p, y_q, \boldsymbol{x}_{pq}). \tag{2}$$

Here U is a unary potential function. To make the exposition more general, we consider multiple types of unary potentials with $\mathcal{U}$ the set of all such unary potentials. $\mathcal{N}_U$ is a set of nodes for the potential U. Likewise, V is a pairwise potential function with $\mathcal{V}$ the set of all types of pairwise potential. $\mathcal{S}_V$ is the set of edges for the potential V. $\boldsymbol{x}_p$ and $\boldsymbol{x}_{pq}$ indicates the corresponding image regions which associate to the specified node and edge.

The potential function is constructed by a deep network for generating feature map (FeatMap-Net) and a shallow network (Unary-Net or Pairwise-Net) to generate the output of the potential function. Details are described in the following sections. An overview of our contextual deep structured model for prediction and training is shown in Fig. 4.

3.1 Unary potential functions

We formulate the unary potential function by stacking the FeatMap-Net for generating feature maps and a shallower fully connected network (referred to as Unary-Net) to generate the final output of the unary potential function. The unary potential function is written as follows:

$$U(y_p, \boldsymbol{x}_p; \boldsymbol{\theta}_U) = -z_{p,y_p}(\boldsymbol{x}; \boldsymbol{\theta}_U). \tag{3}$$

Here z_{p,y_p} is the output value of Unary-Net, which corresponds to the p-th node and the y_p-th class.

Fig. 4 shows an illustration of the Unary-Net and how it corporates with FeatMap-Net. Fig. 5 demonstrates the process for generating the feature vector for one node. The input of the Unary-Net is the node feature vector extracted from the feature map which is generated by FeatMap-Net. The feature vector for one CRF node is simply the corresponding feature vector in the feature map. The dimension of the Unary-Net output vector for one node is K, which is the same as the number of classes.

3.2 Pairwise potential functions

We formulate the unary potential function, analogous to the unary potentials, by stacking the FeatMap-Net for generating feature maps and a shallower fully connected network (referred to as Pairwise-Net) to generate the final output of the pairwise potential function. The pairwise potential function is written as follows:

$$V(y_p, y_q, \boldsymbol{x}_{pq}; \boldsymbol{\theta}_V) = -z_{p,q,y_p,y_q}(\boldsymbol{x}; \boldsymbol{\theta}_V). \tag{4}$$

Here z_{p,q,y_p,y_q} is the output value of Pairwise-Net. It is the confidence value for the node pair (p, q) when they are labeled with the class value (y_p, y_q), which measures the compatibility of the label pair (y_p, y_q) given the input image $\boldsymbol{x}$. $\boldsymbol{\theta}_V$ is the corresponding set of CNN parameters for the potential V, which we need to learn. The role of Pairwise-Net in our structured model is illustrated in Fig. 4. Fig. 5 describes the process for generating the feature vector for one pairwise connection. The input of Pairwise-Net is the edge feature vector which is generated from the feature map for two connected nodes. Following the work of [29], we concatenate the corresponding feature vectors of two connected nodes to obtain the CRF edge feature vector. The Pairwise-Net has K^2 output units to match the number of possible label combinations for a pair of nodes.

Our formulation of pairwise potentials is different from the Potts-model-based smoothness potentials in the existing methods of [5], [60]. The Potts-model-based pairwise potentials are a log-linear functions and employ a special formulation for enforcing neighborhood smoothness based on color contrast, and thus to sharpen object/region boundaries. In contrast, our pairwise potentials model the semantic compatibility relations between two nodes with the output for every possible value of the label pair (y_p, y_q) individually parameterized by CNNs. Clearly, these two types of pairwise potential formulations have different purposes and effects.

Most recent segmentation methods, e.g., the work in [5], [8], [48], [60], have applied the dense CRF method [30] in the prediction refinement stage for refining (sharpen object boundaries) the coarse (low-resolution) prediction. The

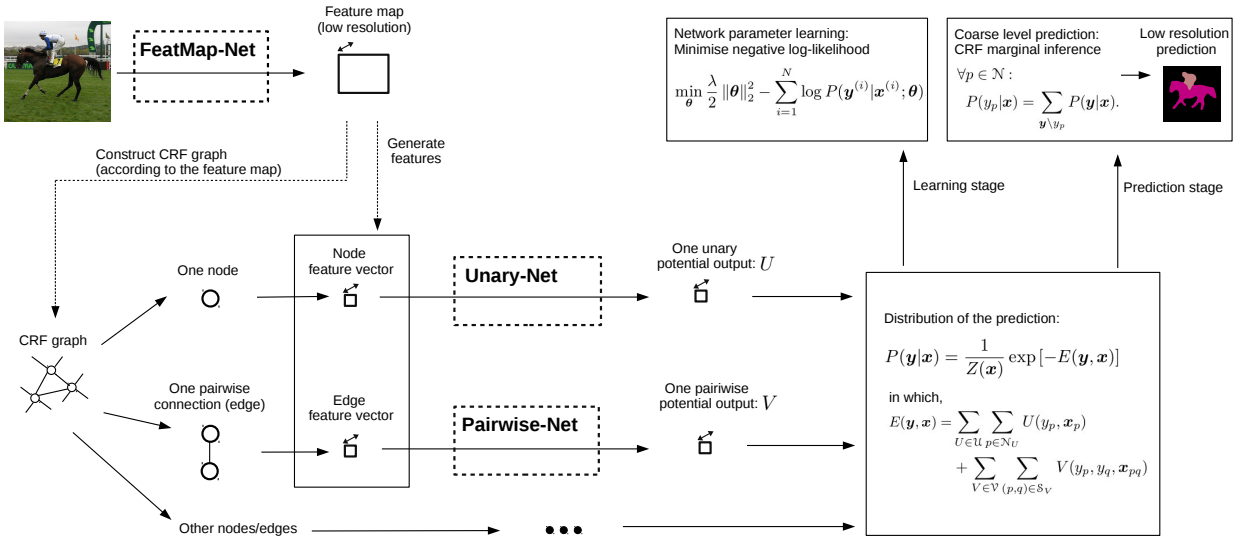


Fig. 4 – An overview of the proposed contextual deep structured model. Unary-Net and Pairwise-Net are shown here for generating potential function outputs.

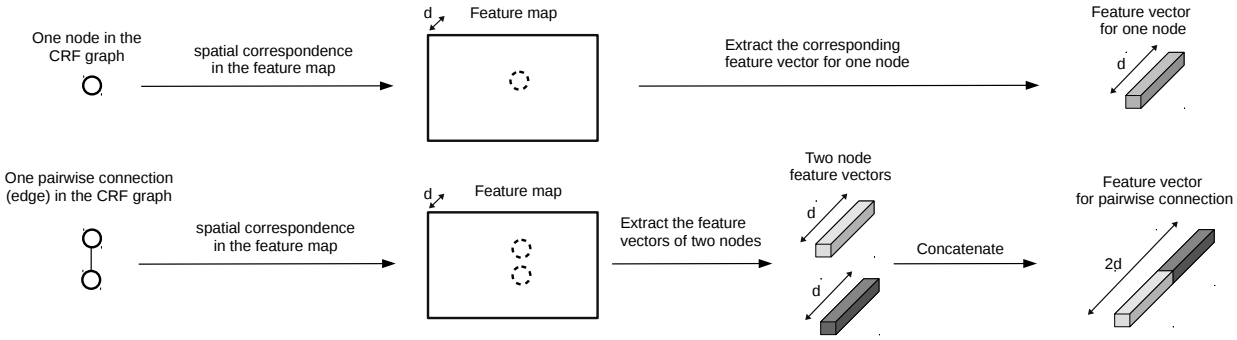


Fig. 5 – An illustration of generating feature vectors for CRF nodes and pairwise connections from the feature map output by FeatMap-Net. The symbol d denotes the feature dimension. We concatenate the corresponding features of two connected nodes in the feature map to obtain the CRF edge features.

dense CRF method is a Potts-model-based fully-connected CRF with pairwise potentials based on color contrast for local smoothness. It is important to clarify that, this smoothness CRFs and our contextual deep CRFs are working in different prediction stages. Our contextual CNN pairwise potentials are applied in the coarse prediction stage to improve the lower-resolution prediction, rather than applying in the boundary refinement stage.

In our framework, after obtaining the coarse level prediction, we still need to perform a refinement step to obtain the final high-resolution prediction (as shown in Fig. 1). Hence we also apply the dense CRF method [30], as in many other recent methods, in the prediction refinement step. Therefore, our method takes advantage of both contextual CNN potentials and the traditional smoothness potentials to improve the final result. More details for prediction can be found in Sec. 6.

3.2.1 Asymmetric pairwise potentials

As in [25], [57], modeling asymmetric relations requires learning asymmetric potential functions, the output of which should depend on the input order of a pair of nodes. In other words, the potential function is required to be capable of modeling different input orders. Typically we have the following case for asymmetric relations:

$$V(y_p, y_q, \boldsymbol{x}_{pq}) \neq V(y_q, y_p, \boldsymbol{x}_{qp}). \tag{5}$$

Ideally, the potential V is learned from the training data.

Here we discuss the asymmetric relation "above/below" as an example. We take advantage of the input pair order to indicate the spatial configuration of two nodes, thus the input $(y_p, y_q, \boldsymbol{x}_{pq})$ indicates the configuration that the node p is spatially lies above the node q. Clearly, the potential function is required to model different input orders.

The asymmetric property is readily achieved with our general formulation of pairwise potentials. The edge features for the node pair (p, q) are generated from a concatenation of the corresponding features of nodes p and q (as in [29]), in that order. The potential output for every possible pairwise label combination for (p, q) is individually parameterized by the pairwise CNNs. These factors ensure that the edge response is order dependent, easily satisfying the asymmetric requirement.

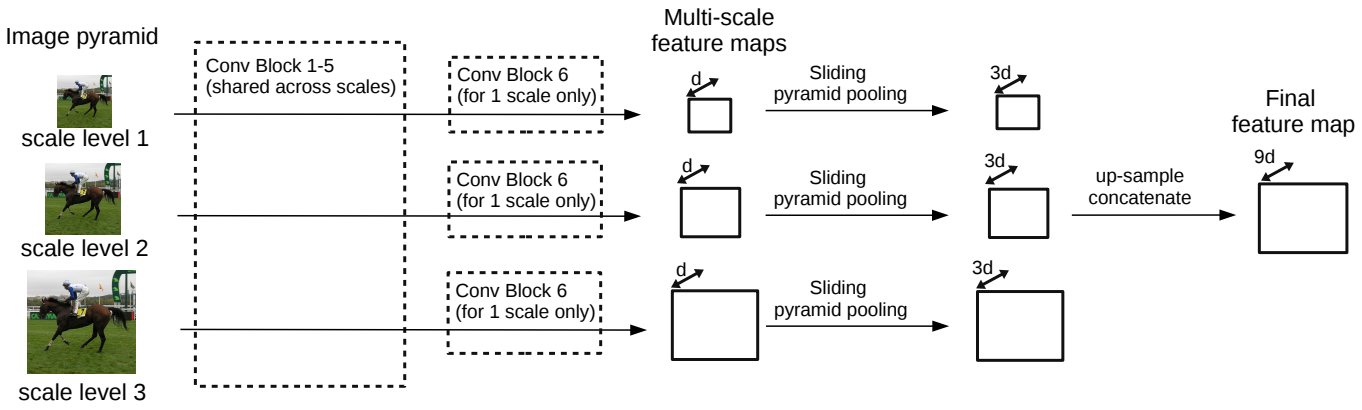


Fig. 6 – The details of our FeatMap-Net. An input image is first resized into 3 scales, then each resized image goes through 6 convolution blocks to output one feature map. Top 5 convolution blocks are shared for all scales. Every scale has a specific convolution block (Conv Block 6). We perform 2-level sliding pyramid pooling and concatenate the pooled feature map to the original feature map. The symbol d denotes the feature dimension.

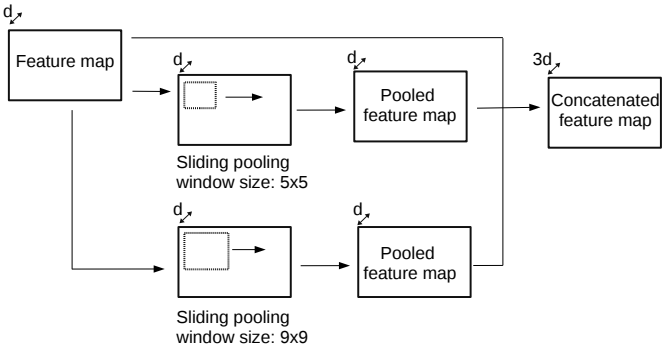


Fig. 7 – Details for sliding pyramid pooling. We perform 2-level sliding pyramid pooling on the feature map for capturing patch-background context, which encode rich background information and increase the field-of-view for the feature map.

4 Exploring Background Context

We develop multi-scale CNNs and sliding pyramid pooling in our FeatMap-Net to encode rich background information for capturing patch-background context. Fig. 6 shows the architecture of FeatMap-Net. Details are presented shortly in the squeal.

Applying CNNs on multi-scale images has shown improved performance in some recent segmentation methods, e.g., [17], [40]. In our multi-scale network, an input image is first resized into 3 scales, then each resized image goes through 6 convolution blocks to output one feature map. In our experiment, the 3 scales for the input image are set to 1.2, 0.8 and 0.4. All scales share the same top 5 convolution blocks. In addition, each scale has an exclusive convolution block ("Conv Block 6" in the figure) which captures scale-dependent information. The resulting 3 feature maps (corresponding to 3 scales) are of different resolutions, therefore we upscale the two smaller ones to the size of the largest feature map using bilinear interpolation. These feature maps are then concatenated to form one feature map.

We perform spatial pyramid pooling [31] (a modified version using sliding windows) on the feature map to capture information from background regions in multiple sizes. From another point of view, this increases the field-of-view for the feature map, for which feature vectors are able to encode information from a larger image region. Increasing the field-of-view generally helps to improve performance, which is also discussed in [5].

The details of spatial pyramid pooling are illustrated in Fig. 7. In our experiment, we perform 2-level pooling for each image scale. We define 5×5 and 9×9 sliding pooling windows with max-pooling to generate 2 sets of pooled feature maps. These pooled feature maps are then concatenated to the original feature map to construct the final feature map, and thus the resulting feature dimension is 512×3 for one image scale.

5 Network Configurations

We show the detailed network layer configuration for all networks in Fig. 8. For FeatMap-Net, the configuration of the convolution blocks is similar to the VGG-16 model [51]. The top 5 convolution blocks share the same configuration as the VGG-16 network. The first fully-connected layer in VGG-16 is converted into a convolution layer (see FCN in [39] for details) and merged into the 5-th convolution block. We only transfer the first fully-connected (FC) layer into our network rather than 2 FC layers. Note that transferring 2 FC layers is commonly applied in almost all recent FCN based methods [5], [39], [60]. The FC layer in the VGG-16 model contains a large number of filters (4096), thus our network which transfers only one FC layer is more efficient.

In FeatMap-Net, we add a new convolution block ("Conv Block 6" in the figure) which contains 2 convolution layers. This extra convolution block is not existed in the VGG-16 network. With this new convolution block, we are able to capture scale-dependent information and increase the abstraction level. We also have the consideration of increasing the field-of-view for the final feature map by adding this extra block.

As discussed in Sec. 1.1, The stride setting of the convolution and pooling layers will result in a feature map

which has a smaller resolution than the input image. For the convolution and pooling layers, the resolution of the output feature map is down-sampled if the stride is greater than 1. Note that there are a number of convolution/pooling layers in VGG-16 model which use the stride setting of 2. Therefore, for the original VGG-16 model, the resolution of the output feature map is 32 times smaller than the size of the input image (see FCN [39] for details). To increase the resolution of the feature map, almost all recent VGG-16 based methods [5], [39], [60] reduce the stride of the last two pooling layers to 1, which reduces the down-sampling factor from 32 to 8.

In our setting, we reduce the stride of the last max pooling layer (only one layer) in the VGG-16 network to 1, instead of reducing for two pooling layers in many other methods [5], [60]. The resolution of the resulting feature map is 16 times smaller than the size of the input image. For a 500×500 input image, the resolution of the resulting feature map is around 30×30.

Directly changing the stride inevitably degrades the performance of the learned filters since the field-of-view of the input feature map for some filters is changed. To preserve the field-of-view, recent work has proposed a number of approaches. For example, a straightforward approach is to increase the receptive field size of the filter (e.g., double the filter size). Large filter sizes will significantly increase the computation cost for convolution operations. This approach also brings the problem of how to upsample the filter weights. Probably a better approach is to apply the hole algorithm as in [5], which performs a skipping (sampled) dot-product calculation for filter convolution. Therefore, a large convolution window size can be applied without increasing the computation cost.

Different from existing approaches, here we apply a simple yet effective approach. We add extra two 3×3 convolution layers ("Conv Block 6") instead of increasing the filter size. These extra layers are able to enlarge the field-of-view and compensate the side-effect of reducing the stride in pooling layers.

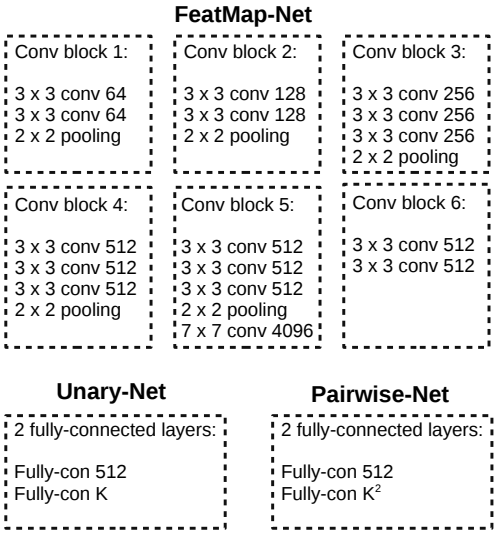


Fig. 8 – The detailed configuration of the networks: FeatMap-Net, Unary-Net and Pairwise-Net. K is the number of classes. The filter size for convolution and the number of filters are shown for all layers. For FeatMap-Net, the top 5 convolution blocks share the same configuration as the convolution blocks in the VGG-16 network. The stride of the last max pooling layer is 1, and for the other max pooling layers we use the same stride setting as the VGG-16 network.

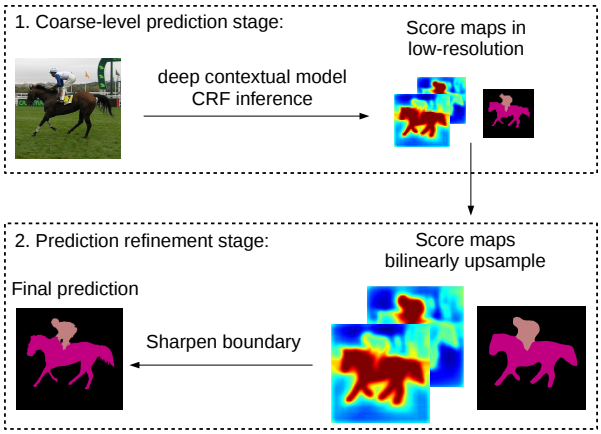


Fig. 9 – An illustration of our two-stage prediction process. The prediction process consists of two stages: the coarse-level prediction stage and the prediction refinement stage. We first perform CRF inference on our contextual model to generate a score map for coarse-level prediction, then we bilinearly unsmaple the score map and apply a boundary refinement method [30] to obtain the final prediction which has the same resolution as the input image.

6 Prediction

At the prediction stage, our deep structured model generates low-resolution prediction (as shown in Fig. 1), which is $1/16$ of the input image size. As discussed in Sec. 5, this is due to the stride setting of pooling layers. Therefore, we apply two prediction stages for obtaining the final high-resolution prediction: the coarse-level prediction stage and the prediction refinement stage. We first perform CRF inference on our contextual structured model to generate a score map for coarse-level prediction, then we bilinearly unsmaple the score map and apply a boundary refinement method [30] to obtain the final prediction which has the same resolution as the input image. This two-stage prediction process is illustrated in Fig. 9.

6.1 Coarse-level prediction stage

We perform CRF inference on our contextual structured model to obtain the coarse prediction of a test image. For example, we can solve the maximum a posteriori (MAP) problem: $\boldsymbol{y}^\star = \operatorname{argmax}_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x})$. Alternatively, we also can consider the marginal inference over nodes for prediction:

$$\forall p \in \mathcal{N}: \quad P(y_p|\boldsymbol{x}) = \sum_{\boldsymbol{y} \setminus y_p} P(\boldsymbol{y}|\boldsymbol{x}). \tag{6}$$

We obtain the marginal distribution for each node after performing this marginal inference. This marginal distribution can be further applied in the next prediction stage for boundary refinement. Details are shown in the next section.

Our CRF graph does not form a tree structure, nor are the potentials submodular, hence we need to an apply approximate inference. To address this we apply an efficient message passing algorithm which is based on the mean field approximation [43]. The mean field algorithm constructs a simpler distribution $Q(\boldsymbol{y})$, e.g., a product of independent marginals: $Q(\boldsymbol{y}) = \prod_{p\in\mathcal{N}} Q_p(y_p)$, which minimizes the KL-divergence between the distribution $Q(\boldsymbol{y})$ and $P(\boldsymbol{y})$. In our experiments, we perform 3 mean field iterations.

6.2 Prediction refinement stage

We generate the score map for the coarse prediction from the marginal distribution which we obtain from the mean-field inference. We first bilinearly up-sample the score map of the coarse prediction to the size of the input image. Then we apply a common post-processing method [30] (dense CRF) to sharpen the object boundary for generating the final high-resolution prediction. This post-processing method leverages low-level pixel intensity information (color contrast) for boundary refinement. Note that most recent work on image segmentation produce low-resolution prediction and have a upsampling and refinement process/model for the final prediction, e.g., [5], [8], [60].

In summary, we simply perform bilinear upsampling of the coarse score map and apply the boundary refinement post-processing. We argue that this stage can be further improved by applying more sophisticated refinement methods, e.g., training deconvolution networks [42] training multiple coarse to fine learning networks [13], and exploring middle layer features for high-resolution prediction [23], [39]. It is expected that applying better refinement approaches will gain further performance improvement. In the experiment part, we show an example of exploring the feature maps from middle layers to refine the coarse prediction. We apply this improved refinement approach on the dataset PASCAL VOC 2012. Refer to Sec. 9.2 for details.

7 CRF TRAINING

A common approach to CRF learning is to maximize the likelihood, or equivalently minimize the negative log-likelihood, which can be written for one image as:

$$-\log\, P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = E(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\theta}) + \log\, Z(\boldsymbol{x};\boldsymbol{\theta}). \tag{7}$$

Adding regularization to the CNN parameter $\boldsymbol{\theta}$, the optimization problem for CRF learning is:

$$\min_{\boldsymbol{\theta}} \frac{\lambda}{2}\left\|\boldsymbol{\theta}\right\|_2^2 - \sum_{i=1}^{N} \log\, P(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}). \tag{8}$$

Here $\boldsymbol{x}^{(i)}$, $\boldsymbol{y}^{(i)}$ denote the i-th training image and its segmentation mask; N is the number of training images; λ is the weight decay parameter. Substituting (7) into (8) yields:

$$\min_{\boldsymbol{\theta}} \frac{\lambda}{2}\left\|\boldsymbol{\theta}\right\|_2^2 + \sum_{i=1}^{N}\Big[E(\boldsymbol{y}^{(i)},\boldsymbol{x}^{(i)};\boldsymbol{\theta}) + \log\, Z(\boldsymbol{x}^{(i)};\boldsymbol{\theta})\Big]. \tag{9}$$

We can apply stochastic gradient (SGD) based methods to optimize the above problem for learning $\boldsymbol{\theta}$. The energy function $E(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\theta})$ is constructed from CNNs, and its gradient $\nabla_{\boldsymbol{\theta}} E(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\theta})$ easily computed by applying the chain rule as in conventional CNNs. However, the partition function Z brings difficulties for optimization. Its gradient is written:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{x};\boldsymbol{\theta}) \\
&\quad= \nabla_{\boldsymbol{\theta}} \log \sum_{\boldsymbol{y}} \exp\left[-E(\boldsymbol{y},\boldsymbol{x})\right] \\
&\quad= \sum_{\boldsymbol{y}} \frac{\exp\left[-E(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\theta})\right]}{\sum_{\boldsymbol{y}'}\exp\left[-E(\boldsymbol{y}',\boldsymbol{x};\boldsymbol{\theta})\right]} \nabla_{\boldsymbol{\theta}}[-E(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\theta})] \\
&\quad= -\,\mathbb{E}_{\boldsymbol{y}\sim P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} E(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\theta})
\end{aligned} \tag{10}$$

Generally the size of the output space $\mathcal{Y}$ is exponential in the number of nodes, which prohibits the direct calculation of Z and its gradient. The CRF graph we considered for segmentation here is a loopy graph (not tree-structured), in which a large number of nodes (more than 1000) and pairwise connections (more than 2×10^4) are involved for one image. For loopy graph with large number of nodes and edges, typically approximation is required for inference, and even this is generally computationally expensive.

More importantly, usually a large number of SGD iterations are required for training CNNs. Typically the number of iterations is in tens or hundreds of thousands. Thus performing inference at each SGD iteration is very computationally expensive.

7.1 Piecewise training of CRFs

Instead of directly solving the optimization in (9), we propose to apply an approximate CRF learning method. In the literature, there are two popular types of learning methods which approximate the CRF objective: pseudo-likelihood learning [2] and piecewise learning [53]. The main advantage of these methods in term of training deep CRF is that they do not involve marginal inference for gradient calculation, which significantly improves the efficiency of training. Decision tree fields [44] and regression tree fields [27] are based on pseudo-likelihood learning, while piecewise learning has been applied in the work [29], [53].

Here we develop this idea for the case of training the CRF with the CNN potentials. In piecewise training, the conditional likelihood is formulated as a number of independent likelihoods defined on potentials, written as:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{U\in\mathcal{U}}\prod_{p\in\mathcal{N}_U} P_U(y_p|\boldsymbol{x}) \prod_{V\in\mathcal{V}}\prod_{(p,q)\in\mathcal{S}_V} P_V(y_p,y_q|\boldsymbol{x}).$$

The likelihood $P_U(y_p|\boldsymbol{x})$ is constructed from the unary potential U. Likewise, $P_V(y_p,y_q|\boldsymbol{x})$ is constructed from the pairwise potential V. P_U and P_V are written as:

$$P_U(y_p|\boldsymbol{x}) = \frac{\exp\left[-U(y_p,\boldsymbol{x}_p)\right]}{\sum_{y'_p}\exp\left[-U(y'_p,\boldsymbol{x}_p)\right]}, \tag{11}$$

$$P_V(y_p,y_q|\boldsymbol{x}) = \frac{\exp\left[-V(y_p,y_q,\boldsymbol{x}_{pq})\right]}{\sum_{y'_p,y'_q}\exp\left[-V(y'_p,y'_q,\boldsymbol{x}_{pq})\right]}. \tag{12}$$

The log-likelihood for piecewise training is then:

$$\begin{aligned}
\log\, P(\boldsymbol{y}|\boldsymbol{x}) = &\sum_{U\in\mathcal{U}}\sum_{p\in\mathcal{N}_U} \log\, P_U(y_p|\boldsymbol{x}) \\
&+ \sum_{V\in\mathcal{V}}\sum_{(p,q)\in\mathcal{S}_V} \log\, P_V(y_p,y_q|\boldsymbol{x}).
\end{aligned} \tag{13}$$

The optimization problem for piecewise training is to minimize the negative log likelihood with regularization:

$$\min_{\boldsymbol{\theta}} \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 - \sum_{i=1}^{N} \Big[\sum_{U \in \mathcal{U}} \sum_{p \in \mathcal{N}_U^{(i)}} \log P_U(y_p | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}_U) + \sum_{V \in \mathcal{V}} \sum_{(p,q) \in \mathcal{S}_V^{(i)}} \log P_V(y_p, y_q | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}_V) \Big]. \quad (14)$$

Compared to the objective in (9) for direct maximum likelihood learning, the above objective does not involve the global partition function $Z(\boldsymbol{x}; \boldsymbol{\theta})$. To calculate the gradient of the above objective, we only need to calculate the gradient $\nabla_{\boldsymbol{\theta}_U} \log P_U$ and $\nabla_{\boldsymbol{\theta}_V} \log P_V$. With the definition in (11), P_U is a conventional softmax normalization function over only K (the number of classes) elements. Similar analysis can also be applied to P_V. Hence, we can easily calculate the gradient without involving expensive inference. Moreover, we are able to perform paralleled training of potential functions, since the above objective is formulated by a summation of independent log-likelihoods.

As previously discussed, CNN training usually involves a large number of gradient update iteration which prohibit the repeated expensive inference. Our piecewise approach here provides a practical solution for learning CRFs with CNN potentials on large-scale data.

8 Implementation Details

For the FeatMap-Net, the first 5 convolution blocks and the first convolution layer in the 6th convolution block are initialized from the VGG-16 network [51]. All remaining layers are randomly initialized. Note that VGG-16 network is widely applied in recent segmentation methods. All layers are trained using back-propagation/stochastic gradient descend (SGD). We apply simple data augmentation in the training stage. Specifically, we perform random scaling (from 0.7 to 1.2) and flipping of the images for training.

As illustrated in Fig. 3, we use 2 types of pairwise potential functions. In total, we have 1 type of unary potential function and 2 types of pairwise potential functions. We formulate one specific FeatMap-Net and potential network (Unary-Net or Pairwise-Net) for one type of potential function. In other words, one type of potential function is constructed by one FeatMap-Net and a shallow potential network. More details of FeatMap-Net and Unary/Pairwise-Net can be found in Fig. 4. There are two main benefits of modeling specific FeatMap-Net for each potential instead of sharing one FeatMap-Net across potentials. Different types of potentials have different focus and thus probably require separate feature maps. Using separate FeatMap-Net allows generating specific high-level features for the corresponding potential function. Moreover, with separate FeatMap-Net, we are able to parallel the training of different types of potentials, and thus ease the implementation and speed up the training.

8.1 Efficient learning

As previously discussed, one node in the CRF graph is connected to all nodes which lie within a predefined range box. Under this setting, the number of pairwise connections for one node can be a few hundred. For example, for an input image with a resolution of 500×500 pixels and 1.2x scaling, the resolution of the feature map after going through FeatMap-Net is around 35×35. In this case, the number of nodes are around 1200, and the number of connections is around 200 for one node. Hence for this image, we need to process 1200×200 pairwise relations for generating the edge features, passing forward the Pairwise-Net and back-propagating the gradients (in the training stage). These operations are considerably computationally expensive for such a large number of pairwise connections. If using high feature dimension for the feature map, these operations can even run out of the GPU memory.

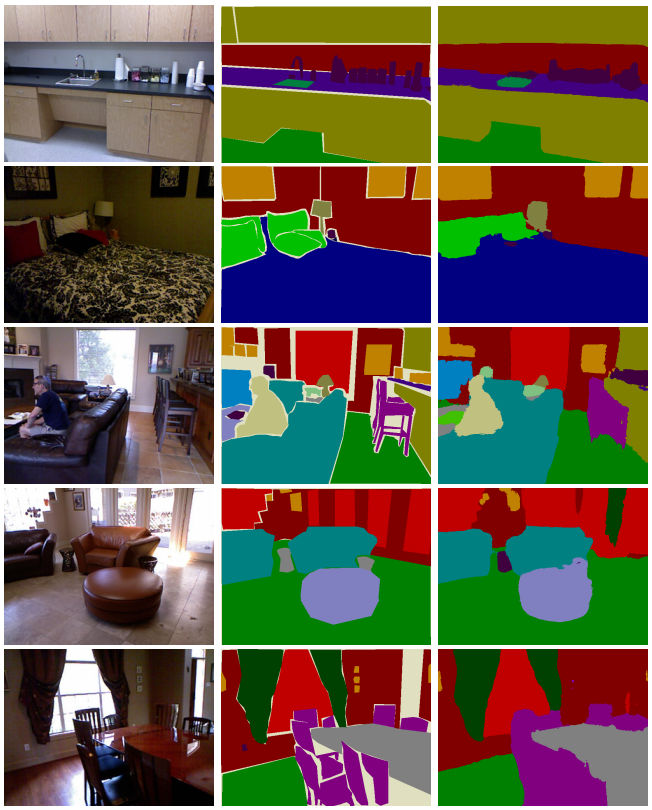
In our solution, to speed up the training and testing of the pairwise potentials, we perform sampling of pairwise connections for each node in the CRF graph. Since the feature map encodes redundant information in local regions, performing sampling can still preserve sufficient pairwise relations while removing redundancies. Specifically, we sample 24 neighboring nodes based on a regular 5×5 grid spanning the range box (excluding self-connection), and thus we have 24 pairwise connections for each node, which is an order of magnitude fewer connections than the original setting. We observe that this sampling setting, which reduces the number of pairwise connections significantly, speeds up the training without degrading the performance.

8.2 Asynchronous gradient update

The number of pairwise connections is still large even with sampling, which brings the problem of keeping the edge features in the GPU memory. Moreover, considering a large number of pairwise connections (more than 2×10^4) in one iteration for updating the parameters of Pairwise-Net might result in degraded gradients. This is similar to the case that using an extremely large batch size for gradient calculation in the training of a conventional classification network. An extremely large batch size for gradient update can significantly slow down the convergence and may decrease the performance [32]. Moreover, as discussed in [1], using small batch size may perform noise injection in the gradient calculation as is a form of regularization, which may lead to better parameter solutions. Overall, from both empirical observations and theoretical analysis, a appropriate setting of batch size is key to the network training. Therefore, we probably should not consider all pairwise connections in one gradient iteration for updating the Pairwise-Net.

To reduce the GPU memory consumption and improve the batch update for the Pairwise-Net, we perform asynchronous gradient update for training the FeatMap-Net and Pairwise-Net. With asynchronous gradient update, the gradient calculations for different parts of the network are not required in the same iteration, which breaks the dependency between different parts (or layers) of the networks. Asynchronous gradient update is widely applied in large-scale distributed network learning. For details one may refer to [10].

Specifically, in one stochastic gradient iteration, we perform multiple sub-iterations of gradient update for the Pairwise-Net and collect the gradients for the FeatMap-Net.



(a) Testing **(b)** Ground Truth **(c)** Prediction

Fig. 10 – Prediction examples on the NYUDv2 dataset.

TABLE 1 – Segmentation results on NYUDv2 dataset (40 classes). We compare to a number of recent methods. Our method significantly outperforms the existing methods.

method	training data	pixel accuracy	mean accuracy	IoU
Gupta et al. [21]	RGB-D	60.3	-	28.6
FCN-32s [39]	RGB	60.0	42.2	29.2
FCN-HHA [39]	RGB-D	65.4	46.1	34.0
ours	RGB	**70.0**	**53.6**	**40.6**

In each sub-iteration, a subset of pairwise connections is selected (e.g., 2000) for gradient calculation and the parameters of Pairwise-Net are updated. Clearly, in each sub-iteration we only process a small number of connections for updating the network parameters of Pairwise-Net, thus GPU consumption is low and the batch size for learning Pairwise-Net is reduced. This asynchronous approach addresses the problems of GPU memory and large batch size for training Pairwise-Net. After going through all pairwise connections, we collect the gradients for FeatMap-Net, and perform a conventional back-propagation gradient update to FeatMap-Net.

9 Experiments

We evaluate our method on 8 challenging semantic segmentation datasets: PASCAL VOC 2012, NYUDv2, PASCAL-Context, SIFT-flow, SUN-RGBD, KITTY, COCO and Cityscapes, which covers various types of scene images, including indoor/outdoor scene, street scene, etc. Our comprehensive experiments show that the proposed method

TABLE 2 – Ablation Experiments. The table shows the value added by the different system components of our method on the NYUDv2 dataset (40 classes).

method	pixel accuracy	mean accuracy	IoU
FCN-32s [39]	60.0	42.2	29.2
FullyConvNet Baseline	61.5	43.2	30.5
+ sliding pyramid pooling	63.5	45.3	32.4
+ multi-scales	67.0	50.1	37.0
+ boundary refinement	68.5	50.9	38.3
+ CNN contextual pairwise	70.0	53.6	40.6

TABLE 3 – Comparison with unary ensembles on the NYUDv2 dataset (40 classes). We compare our contextual CRF model to an ensemble of up to 4 unary-only networks. It clearly shows that using our CRF model with 1 pairwise potential (corresponding to the surrounding relation) and 1 unary potential outperforms the ensembles of multiple unary networks. Moreover, using an ensemble of 2 unary in our CRF model can further improve the performance ("1 pairwise + 2 unary"). These results verify the effectiveness of learning pairwise potentials.

settings	IoU score
1 unary	37.0
2 unary ensemble	37.8
3 unary ensemble	38.4
4 unary ensemble	38.7
1 pairwise + 1 unary	38.9
1 pairwise +2 unary	39.2

achieves new state-or-the-art performance on these datasets. For VGG pre-trained layers (Block 1 to Block 5 in FeatMap-Net), we use a small learning rate: 0.0001; for the remaining layers (Block 6 in FeatMap-Net, layers in Unary-Net and Pairwise-Net), we set a larger learning rate: 0.001. Our system is built on MatConvNet [56].

The segmentation performance is measured by the intersection-over-union (IoU) score [16], the pixel accuracy and the mean accuracy on categories [39]. We denote c_{ij} as an element in the confusion matrix, which is the number of pixels with the i-th category as the ground truth and the j-th category as the prediction; t_i is the total number of pixels for the i-th category in the ground truth; K is the number of categories. Pixel accuracy measures the portion of correctly predicted pixels: $\frac{\sum_i c_{ii}}{\sum_i t_i}$. Mean accuracy measures the per-category pixel accuracy: $\frac{1}{K}\sum_i \frac{c_{ii}}{t_i}$. IoU score calculates the portion of the intersection between the ground truth and the prediction: $\frac{1}{K}\sum_i \frac{c_{ii}}{t_i+\sum_j c_{ij}-c_{ii}}$.

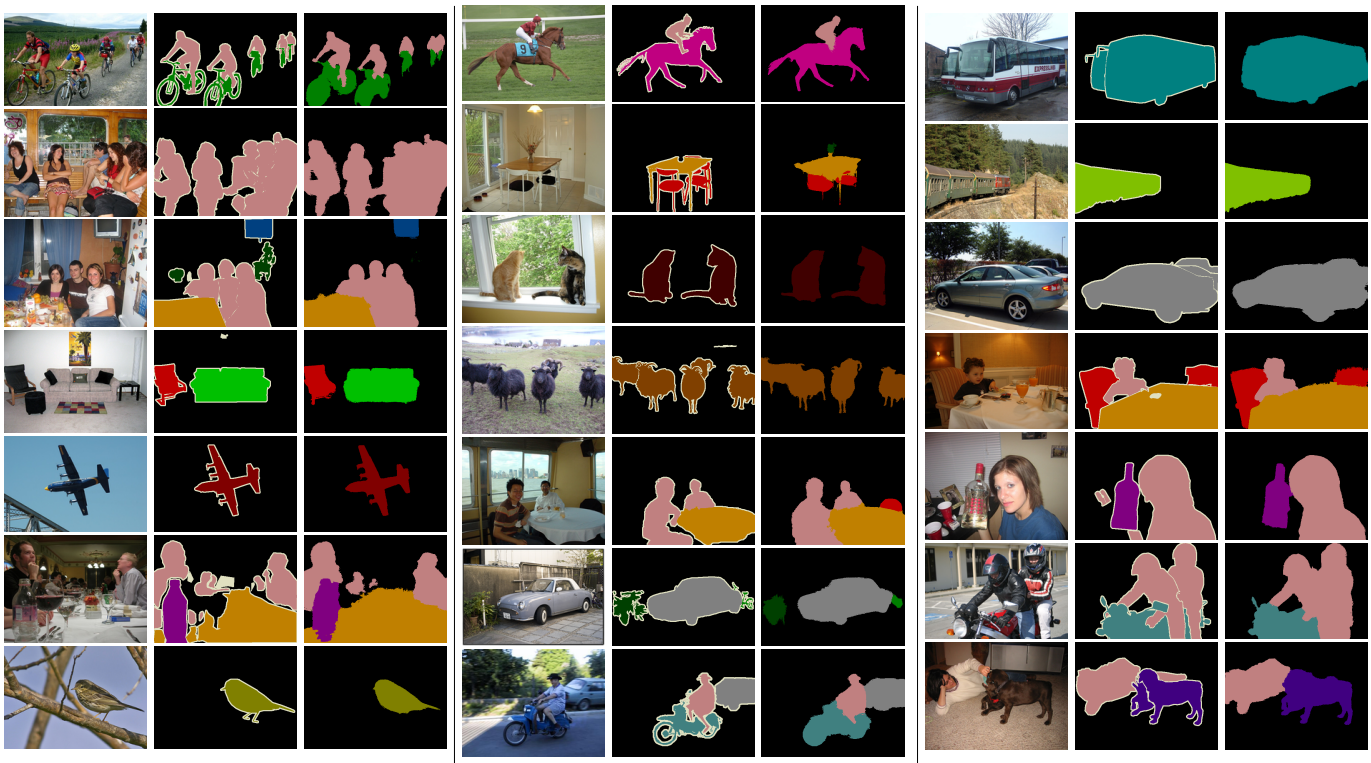
9.1 Results on the NYUDv2 dataset

We first evaluate our method on the NYUDv2 [50] dataset which has 1449 RGB-D indoor scene images. We use the segmentation labels provided in [20] for which the labels are processed into 40 classes. We use the standard training set which contains 795 images and the test set which contains 654 images. We train our models only on RGB images without using the depth information.

Results are shown in Table 1. Some prediction examples are shown in Fig. 10 Unless otherwise specified, our models are initialized using the VGG-16 network. VGG-16 is also used in the competing method FCN [39]. our contextual model with CNN pairwise potentials achieves

TABLE 4 – Individual category results on the PASCAL VOC 2012 test set (IoU scores). Our method performs the best

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	potted	sheep	sofa	train	tv	mean
Only using VOC training data																					
FCN-8s [39]	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
Zoom-out [40]	85.6	37.3	**83.2**	62.5	66.0	85.1	80.7	84.9	27.2	73.2	57.5	78.1	79.2	81.1	77.1	53.6	74.0	49.2	71.7	63.3	69.6
DeepLab [5]	84.4	54.5	81.5	63.6	65.9	85.1	79.1	83.4	30.7	74.1	59.8	79.0	76.1	83.2	80.8	59.7	82.2	50.4	73.1	63.7	71.6
CRF-RNN [60]	87.5	39.0	79.7	64.2	68.3	87.6	80.8	84.4	30.4	78.2	60.4	80.5	77.8	83.1	80.6	59.5	82.8	47.8	78.3	67.1	72.0
DeconvNet [42]	89.9	39.3	79.7	63.9	68.2	87.4	81.2	86.1	28.5	77.0	62.0	79.0	80.3	83.6	80.2	58.8	**83.4**	54.3	80.7	65.0	72.5
DPN [38]	87.7	**59.4**	78.4	64.9	70.3	89.3	83.5	86.1	31.7	79.9	**62.6**	81.9	80.0	83.5	82.3	60.5	83.2	53.4	77.9	65.0	74.1
ours	**90.6**	37.6	80.0	**67.8**	**74.4**	**92.0**	**85.2**	**86.2**	**39.1**	**81.2**	58.9	**83.8**	**83.9**	**84.3**	**84.8**	**62.1**	83.2	**58.2**	**80.8**	**72.3**	**75.3**
Using VOC+COCO training data																					
DeepLab [5]	89.1	38.3	88.1	63.3	69.7	87.1	83.1	85.0	29.3	76.5	56.5	79.8	77.9	85.8	82.4	57.4	84.3	54.9	80.5	64.1	72.7
CRF-RNN [60]	90.4	55.3	88.7	68.4	69.8	88.3	82.4	85.1	32.6	78.5	64.4	79.6	81.9	**86.4**	81.8	58.6	82.4	53.5	77.4	70.1	74.7
BoxSup [8]	89.8	38.0	**89.2**	**68.9**	68.0	89.6	83.0	87.7	34.4	83.6	**67.1**	81.5	83.7	85.2	83.5	58.6	84.9	55.8	**81.2**	70.7	75.2
DPN [38]	89.0	**61.6**	87.7	66.8	74.7	91.2	84.3	87.6	36.5	86.3	66.1	84.4	87.8	85.6	85.4	63.6	87.3	61.3	79.4	66.4	77.5
ours+	**94.1**	40.4	83.6	67.3	**75.6**	**93.4**	**84.4**	**88.7**	**41.6**	**86.4**	63.3	**85.5**	**89.3**	85.6	**86.0**	**67.4**	**90.1**	**62.6**	80.9	**72.5**	**77.8**



(a) Testing **(b)** Ground Truth **(c)** Prediction **(d)** Testing **(e)** Ground Truth **(f)** Prediction **(g)** Testing **(h)** Ground Truth **(i)** Predict

Fig. 11 – Some prediction examples of our method on the PASCAL VOC 2012 dataset.

the best performance, which sets new state-of-the-art result on the NYUDv2 dataset, Note that we do not use any depth information in our model.

9.1.1 *Component evaluation*

We evaluate the performance contribution of different components of the FeatMap-Net for capture patch-background context on the NYUDv2 dataset. We present the results of adding different components in FeatMap-Net, which are shown in Table 2. We start from a baseline setting of our FeatMap-Net ("FullyConvNet Baseline" in the result table), for which multi-scale and sliding pooling is removed. This baseline setting is the conventional fully convolution network for segmentation, which can be considered as our implementation of the FCN method in [39]. The result shows that our CNN baseline implementation ("FullyConvNet") achieves very similar performance (slightly better) than the FCN method. Applying multi-scale network design and sliding pyramid pooling significantly improve the performance, which clearly shows the benefits of encoding rich background context in our approach. Applying the dense CRF method [30] for boundary refinement gains further improvement. Finally, adding our contextual CNN pairwise potentials brings significant further improvement, for which we achieve the best performance in this dataset.

9.1.2 *Comparison with multi-unary ensemble*

We compare our CRF model with contextual pairwise potentials against the simple ensemble of multiple unary-only models. Four unary-only networks are independently trained in this experiment. Results are shown in Table 3. To clearly evaluate the effectiveness, we use 1 type of pairwise potential which corresponds to the surrounding relations in our CRF model. The result shows that using our CRF

model with 1 pairwise potential and 1 unary potential ("1 pairwise + 1 unary") outperforms the ensembles of multiple unary networks, which verifies the effectiveness of learning pairwise potentials. Moreover, using extra unary networks, i.e., an ensemble of 2 unary networks, in our CRF model can further improve the performance, as shown by the entry "1 pairwise + 2 unary" in the result table. It indicates that our pairwise potential is able to capture different information and complementary to the multi-unary ensemble.

9.2 Results on the PASCAL VOC 2012 dataset

PASCAL VOC 2012 [16] is a well-known segmentation evaluation dataset which consists of 20 object categories and one background category. This dataset is split into a training set, a validation set and a test set, which respectively contain 1464, 1449 and 1456 images. Following a conventional setting in [5], [24], the training set is augmented by extra annotated VOC images provided in [22], which results in 10582 training images. We verify our performance on the PASCAL VOC 2012 test set. We compare with a number of recent methods with competitive performance. Since the ground truth labels are not available for the test set, we evaluate our method through the VOC evaluation server.

The IoU scores are shown in the last column of Table 4. Prediction examples of our method are shown in Fig. 11. We first train our model only using the VOC images. We achieve an IoU score of 75.3, which is the best result amongst methods that only use the VOC training data.[1]

To improve the performance, following the setting in recent work [5], [8], we train our model with the extra images from the COCO dataset [34]. With these extra training images, we achieve an IoU score of 77.2.

As described in Sec. 6, our deep structured model generates low-resolution coarse prediction, which is 1/16 of the input image size. To obtain the final high-resolution prediction we apply a simple yet effective approach: we first perform bilinear upsampling of the coarse score map and then apply the boundary refinement post-processing [30]. To improving this simple approach, we exploit the feature maps from middle layers to refine the coarse prediction and produce high-resolution prediction, which is similar to the methods in [5], [23], [39]. With this improved refinement approach, we finally achieve an IoU score of 77.8, *which is best reported result on this challenging dataset.* [2]

The feature maps from the middle layers encode lower level visual information (from edge patterns to texture/object part patterns) and have higher resolution than the final output, thus it is expected that learning extra layers on these feature maps helps to predict details in the object boundaries. Specifically, we add refinement layers on top of the feature maps from the first 5 max-pooling layers and the score map of the coarse prediction (output by our deep structured model). Details are shown in Fig. 12. These refinement layers play a role of refining the coarse prediction by exploring middle layer features, which increase the resolution of the prediction from 1/16 (coarse prediction) to 1/2 of the input image. With this improved prediction, we perform boundary refinement using [30] to generate the final prediction.

1. The result link at the VOC evaluation server: http://host.robots.ox.ac.uk:8080/anonymous/KEFFM4.html

2. The result link at the VOC evaluation server: http://host.robots.ox.ac.uk:8080/anonymous/MVTNTX.html

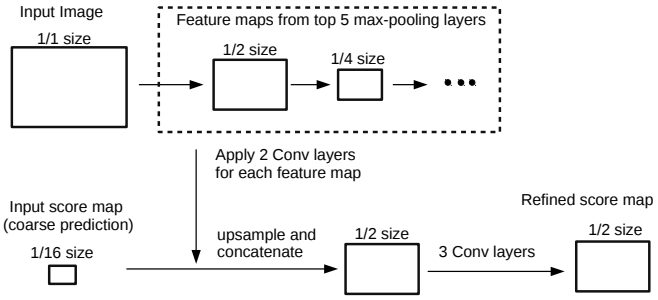


Fig. 12 – The illustration of exploiting the feature maps from middle layers to refine the low-resolution (1/16 of the input image) coarse prediction. The refined prediction has a resolution of 1/2 of the input image.

TABLE 5 – Segmentation results on the Cityscapes *test* set. our method achieves the best performance.

Method	IoU score
FCN-8s [39]	65.3
DPN [38]	66.8
Dilation10 [58]	67.1
DeepLab-CRF [5]	63.1
ours	**71.6**

The results for each category are shown in Table 4. We outperform comparing methods in most categories. For only using the VOC training set, our method outperforms the second best method, DPN [38], on 18 categories out of 20. For using VOC+COCO training set, our method outperforms DPN [38] on 15 categories out of 20.

9.3 Results on the Cityscapes dataset

The large scale outdoor image dataset Cityscapes [7] contains high-resolution street scene images from 50 different cities. This dataset provides pixel-level semantic segmentation labels of 5000 images for 25 classes including road, car, pedestrian, bicycle, sky etc. The provided "trainval" set has 3475 image. We use the training set (2975 images) for training. The ground truth of the test set is not available, and we evaluate our method through their evaluation server. We follow the provided protocol for dataset evaluation: 19 classes are valid for evaluation, and the remaining 6 classes are not considered in evaluation.

Results are shown in Table 5. As similar to the setting for the PASCAL VOC dataset, we train a refinement network which is described in Fig. 12 to obtain high resolution prediction. Here we set the output resolution in the refinement network as 1/4 of the input image size. The result clearly shows that our method outperforms other competing methods. Prediction examples on the validation set are shown in Fig. 13.

9.4 Results on the PASCAL-Context dataset

The PASCAL-Context [41] dataset provides the segmentation labels of the whole scene (including the "stuff" labels) for the PASCAL VOC images. We use the segmentation labels which contain 60 classes (59 classes plus the "

(a) Testing **(b)** Ground Truth **(c)** Prediction **(d)** Testing **(e)** Ground Truth **(f)** Prediction

Fig. 13 – Prediction examples of our method on Cityscapes dataset.

TABLE 6 – Segmentation results on PASCAL-Context dataset (60 classes). Our method performs the best.

method	pixel accuracy	mean accuracy	IoU
O2P [4]	-	-	18.1
CFM [9]	-	-	34.4
FCN-8s [39]	65.9	46.5	35.1
BoxSup [8]	-	-	40.5
ours	**71.5**	**53.9**	**43.3**

TABLE 7 – Segmentation results on SUN-RGBD dataset (37 classes). We compare to a number of recent methods. Our method significantly outperforms the existing methods.

method	training data	pixel accuracy	mean accuracy	IoU
Liu et al. [35]	RGB-D	–	10.0	–
Ren et al. [47]	RGB-D	–	36.3	–
Kendall et al. [28]	RGB	71.2	45.9	30.7
ours	RGB	**78.4**	**53.4**	**42.3**

TABLE 8 – Segmentation results on COCO dataset (80 classes). Our method significantly outperforms the fully convolution network ("FullyConvNet").

method	pixel accuracy	mean accuracy	IoU
FullyConvNet	84.2	56.9	37.2
FullyConvNet + refine	86.7	55.0	41.3
ours	**88.3**	**58.7**	**46.8**

TABLE 9 – Segmentation results on SIFT-flow dataset (33 classes). Our method performs the best.

method	pixel accuracy	mean accuracy	IoU
Liu et al. [35]	76.7	-	-
Tighe et al. [54]	75.6	41.1	-
Tighe et al. (MRF) [54]	78.6	39.2	-
Farabet et al. (balance) [17]	72.3	50.8	-
Farabet et al. [17]	78.5	29.6	-
Pinheiro et al. [45]	77.7	29.8	-
FCN-16s [39]	85.2	51.7	39.5
ours	**88.1**	**53.4**	**44.9**

background" class) for evaluation. We use the provided training/test splits. The training set contains 4998 images and the test set contains 5105 images.

Results are shown in Table 6. Prediction examples are shown in Fig. 14 Our method significantly outperform the competing methods. To our knowledge, *ours is the best reported result on this dataset*.

9.5 Results on the SUN-RGBD dataset

SUN-RGBD [52] is a segmentation dataset contains around 10,000 indoor images and provides pixel labeling masks of 37 classes, which is an extension of the NYUD dataset [50]. Results are shown in Table 7. Our method outperform the existing methods by a large margin, even though we does not make use of the depth information for training.

9.6 Results on the COCO dataset

The COCO dataset [34] contains more than 1 million images and provide segmentation labels for 80 classes. Since the test set is not available, we generate 2599 images for testing and the remaining images are for training. We select the test images on a class balance basis which ensures every category at lease appears in 50 images. Labeling regions which are smaller than 200 pixels are treated as "void" which are not considered in training and evaluation. Results are shown in Table 8. We compared to two baseline methods which are based on conventional fully convolution networks. The details of these baseline methods are the same as that for the Cityscapes dataset (see Sec. 9.3). The results shows that our method significantly outperforms the baselines.

9.7 Results on the SIFT-flow dataset

We further evaluate our method on the SIFT-flow dataset. This dataset contains 2688 images and provides the segmentation labels for 33 classes. We use the standard split for training and evaluation. The training set has 2488 images and the test set has 200 images. Since the images are in

TABLE 10 – Segmentation results on KITTI dataset (10 classes). We compare to a number of recent methods. Our method significantly outperforms the existing methods.

method	pixel accuracy	mean accuracy	IoU
Cadena et al. [3]	84.1	52.4	–
Zhang et al. [59]	89.3	65.4	–
ours	93.3	74.5	68.5
ours+	**94.3**	**75.9**	**70.3**

small sizes, we upscale the image by a factor of 2 for training. Results are shown in Table 9. We achieve the best performance on this dataset.

9.8 Results on the KITTI dataset

We perform further evaluation on the KITTI dataset [18] for road image segmentation. Zhang at el. [59] provide semantic segmentation labels of 10 classes for 252 images, in which 140 images are for training and the remaining 112 are for testing. We follow the provided training and testing splits for evaluation and report the results in Table 10. Clearly, our method performs the best. To further improve the performance, we perform pre-training on COCO images, for which the result is denoted by "ours+" in the result table.

10 Conclusions

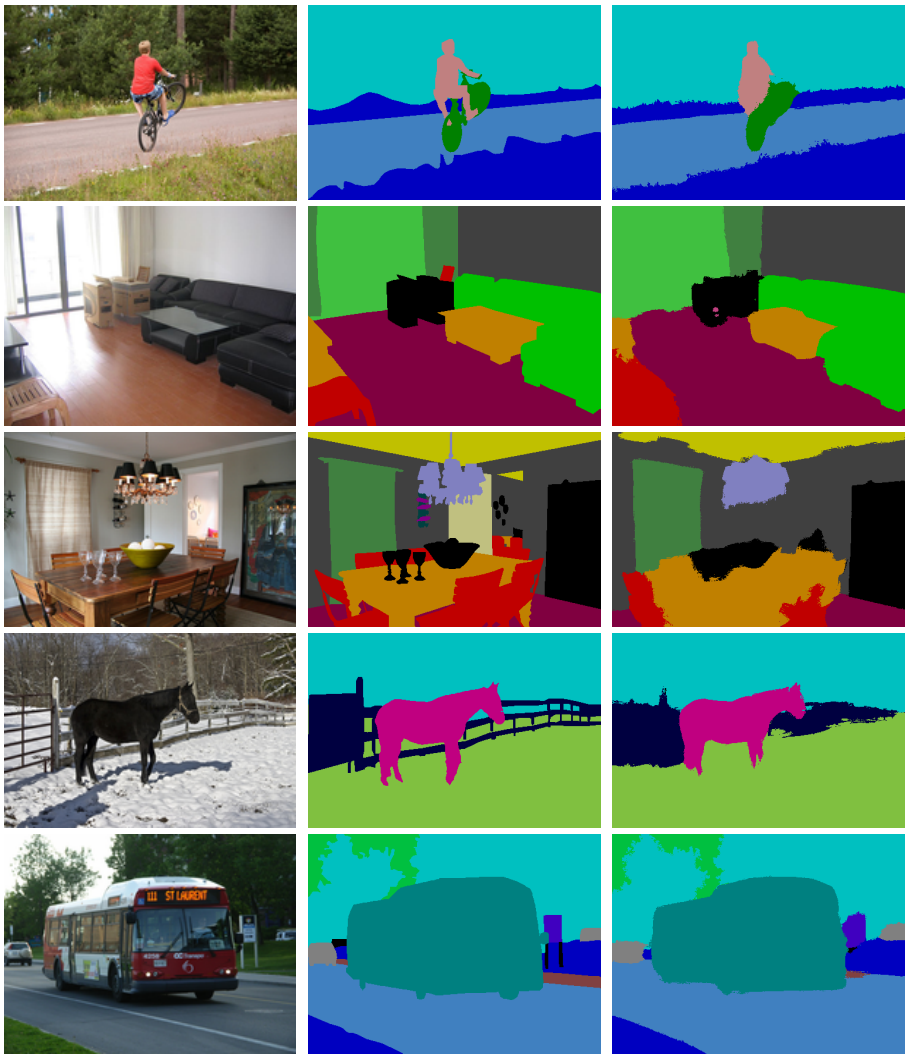
We have proposed a method which combines CNNs and CRFs to exploit *complex* contextual information for semantic image segmentation. Basically, we formulate CNN based pairwise potentials for modeling semantic relations between image regions. We have performed comprehensive experiments on 8 challenging segmentation datasets and we achieve state-of-the-art performance on all evaluated dataset including the PASCAL VOC 2012 dataset. The proposed method is potentially widely applicable to other tasks.

Acknowledgments

This research was supported by Australian Research Council through the ARC Centre for Robotic Vision (CE140100016). C. Shen's participation was in part supported by an ARC Future Fellowship (FT120100969). I. Reid's participation was in part supported by an ARC Laureate Fellowship (FL130100102).

References

[1] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012.
[2] J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 1977.
[3] C. Cadena and J. Kosecka. Semantic segmentation with heterogeneous sensor coverages. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
[4] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *Proc. Eur. Conf. Comp. Vis.*, 2012.
[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. Int. Conf. Learning Representations*, 2015.
[6] L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning deep structured models. In *Proc. Int. Conf. Machine Learn.*, 2015.
[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.
[8] J. Dai, K. He, and J. Sun. BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proc. Int. Conf. Comp. Vis.*, 2015.
[9] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2015.
[10] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, 2012.
[11] C. Doersch, A. Gupta, and A. A. Efros. Context as supervisory signal: Discovering objects with predictable context. In *Proc. European Conf. Computer Vision*, 2014.
[12] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Proc. Eur. Conf. Comp. Vis.*, 2014.
[13] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
[14] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *Proc. Int. Conf. Comp. Vis.*, 2013.
[15] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Proc. Adv. Neural Info. Process. Syst.*, 2014.
[16] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. In *Proc. Int. J. Comp. Vis.*, 2010.
[17] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE T. Pattern Analysis & Machine Intelligence*, 2013.
[18] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2012.
[19] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2014.
[20] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2013.
[21] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 2014.
[22] B. Hariharan, P. Arbelaez, L. D. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *Proc. Int. Conf. Comp. Vis.*, 2011.
[23] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2014.
[24] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Proc. European Conf. Computer Vision*, 2014.
[25] D. Heesch and M. Petrou. Markov random fields with asymmetric interactions for modelling spatial context in structured scene labelling. *Journal of Signal Processing Systems*, 2010.
[26] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *Proc. European Conf. Computer Vision*, 2008.
[27] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fieldsan efficient, non-parametric approach to image labeling problems. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2012.
[28] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *CoRR*, abs/1511.02680, 2015.
[29] A. Kolesnikov, M. Guillaumin, V. Ferrari, and C. H. Lampert. Closed-form training of conditional random fields for large scale image segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 2014.
[30] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Proc. Adv. Neural Info. Process. Syst.*, 2012.
[31] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2006.
[32] M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.



(a) Testing **(b)** Ground Truth **(c)** Prediction

Fig. 14 – Prediction examples on the PASCAL-Context dataset.

[33] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2016.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, 2014.

[35] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE T. Pattern Analysis & Machine Intelligence*, 2011.

[36] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2015.

[37] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields, 2015. http://arxiv.org/abs/1502.07411.

[38] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *Proc. Int. Conf. Comp. Vis.*, 2015.

[39] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2015.

[40] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2015.

[41] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, et al. The role of context for object detection and semantic segmentation in the wild. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2014.

[42] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. Int. Conf. Comp. Vis.*, 2015.

[43] S. Nowozin and C. Lampert. Structured learning and prediction in computer vision. *Found. Trends. Comput. Graph. Vis.*, 2011.

[44] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *Proc. Int. Conf. Comp. Vis.*, 2011.

[45] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. In *Proc. Int. Conf. Machine Learn.*, 2014.

[46] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *Proc. Int. Conf. Comp. Vis.*, 2007.

[47] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2012.

[48] A. G. Schwing and R. Urtasun. Fully connected deep structured networks, 2015. http://arxiv.org/abs/1503.02351.

[49] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[50] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. Eur. Conf. Comp. Vis.*, 2012.

[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. Int. Conf. Learning Representations*, 2015.

[52] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2015.
[53] C. A. Sutton and A. McCallum. Piecewise training for undirected models. In *Proc. Conf. Uncertainty Artificial Intelli*, 2005.
[54] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2013.
[55] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Proc. Adv. Neural Info. Process. Syst.*, 2014.
[56] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for matlab, 2014.
[57] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proc. IEEE Conf. Comp. Vis. Pattern Recogn.*, 2006.
[58] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, 2015.
[59] R. Zhang, S. Candra, K. Vetter, and A. Zakhor. Sensor fusion for semantic segmentation of urban scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
[60] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *Proc. Int. Conf. Comp. Vis.*, 2015.