IMPROVING OBJECT DETECTION WITH RELATION GRAPH INFERENCE

Chen-Hang He, Shun-Cheung Lai and Kin-Man Lam

Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hong Kong

ABSTRACT

Many classic object detection approaches have proven that detection performance can be improved by adding the object's context information. However, only a few methods have attempted to exploit the object-to-object relationship during learning. The reason for this is that objects may appear at different locations in an image, with an arbitrary size and scale. This makes it difficult to model the objects in a unified way within a network. Inspired by Graph Convolutional Network (GCN), we propose a detection algorithm that can infer the relationship among multiple objects during the inference, achieved by constructing a relation graph dynamically with a self-adopted attention mechanism. The relation graph encodes both the geometric and visual relationship between objects. This can enrich the object feature by aggregating the information from the object and its relevant neighbors. Experiments show that our proposed module can efficiently improve the detection performance of existing object detectors.

Index Terms— Object detection, Graph convolutional neural network

1. INTRODUCTION

In many computer vision tasks, feature fusion is often applied to improve the efficiency of an algorithm. An example is the layer-level fusion, such as in the feature pyramid network [1] and the lateral network [2], which incorporates the bottom layer with fine-grained details and the top layer with strong semantics into a single layer. Another example is the spatial-level fusion, such as in [3], which uses the nonlocal mean for image denoising, and in [4], which introduces background information to enhance local features. In addition, Convolutional Neural Networks (CNNs) also fuse the features from earlier layers by a sequence of convolution and pooling operations, with the size of the receptive field increased for feature extraction. These fusion methods are usually implicit and low-level, and cannot be generalized to different tasks. Many empirical studies [5, 6, 7] have shown that, by incorporating object-to-object relationships, the performance of recognition algorithms can be easily improved. This motivated us to explore an explicit, high-level feature fusion technique for object-detection tasks, in which object-to-object relations are used to achieve feature fusion at the instance level. Inspired by the graph convolutional network (GCN) [8], we propose to enhance the local feature representations, by constructing an object relation graph on top of the proposal-based object detectors, e.g. [9] and [10]. Each graph node represents the feature of a region proposal, while the edges represent the object relations between each pair of proposals. Thus, we can enrich the features of each proposal by aggregating the features from all other related proposals through the graph convolution. The edge weight is a measure of the relevance of the current proposal and other proposals, and can be learned in an adaptive self-attention manner. Our proposed method can be considered as a variant of Faster RCNN [10]. In the first stage, we use the region proposal network (RPN) to generate redundant object proposals, which serve as the graph nodes. Then, we apply two different modules to learn the object relations between these proposals. In the second stage, we adopt two graph convolutional layers, which take the relation matrix and the Region-of-Interest (RoI) features as input and produce the bounding box offsets and classes. In the following sections, we will briefly review Faster RCNN and GCN, and also describe our proposed detector in detail.

2. RELATED WORK

2.1. Faster RCNN

Faster RCNN is a two-stage detector. It integrates feature extraction, proposal extraction, bounding-box regression, and classification into a unified network, which greatly improves the overall performance, especially in terms of detection speed. Faster RCNN consists of four basic modules. The feature-extraction layer uses a set of basic operations: Convolution+ReLU+pooling layers, to extract high-resolution feature maps from an input image. The feature maps are shared by the subsequent Region Proposal Network (RPN) layers and RoI pooling layer. RPN uses softmax to generate class-agnostic region proposals by correcting the anchors from the predicted offsets. The RoI pooling layer, which collects the high-resolution feature maps and proposals, produces RoI

Acknowledgements: The work described in this paper was supported by a project from CNERC of the Hong Kong Polytechnic University, Hong Kong (Project No. 1-BBYT).

features, which are then fed to the subsequent classification and regression layers. The classification layer uses the RoI features to determine the category of the proposals, and the regression layer generates the refined position of the object bounding boxes.

2.2. Spectral Graph Convolutions

In this section, we firstly define the notions used for the graph convolutions. Given a graph signal $x \in \mathbb{R}^N$, the normalized graph Laplacian is denoted as $\mathcal{L} = I_N - D^{-1/2}AD^{-1/2}$, where A is the symmetric adjacency matrix and D is the diagonal degree matrix. The graph convolution is equivalent to the multiplication of the graph signal and the filtering kernel g_θ parameterized by $\theta \in \mathbb{R}^N$ in the spectrum domain, and taking the inverse Fourier transform afterwards. In the matrix-vector form, this can be written as follows:

$$x * g_\theta = UG_\theta(\Lambda)U^T x = G_\theta(U\Lambda U^T)x = G_\theta(\mathcal{L})x, \quad (1)$$

where U is a matrix whose columns are the Eigenbases after the Eigendecomposition of $\mathcal{L} = U\Lambda U^T$, and $G_\theta(\Lambda)$ is a diagonal matrix, whose diagonal elements are a function of the eigenvalues parameterized by θ. Based on this property, [8] proposed a first-order approximation of (1), in terms of the Chebyshev expansion of the graph Laplacian, as follows:

$$x * g_\theta \approx \theta(I_N + D^{-1/2}AD^{-1/2})x. \quad (2)$$

The generalized matrix-vector form of the graph convolutional layer can then be expressed as follows:

$$X^{(l+1)} = \tilde{A}X^{(l)}\Theta^{(l)}. \quad (3)$$

This equation demonstrates that the graph inference can be efficiently presented by a simple layer-wise multiplication. $X^{(l)}$ and $X^{(l+1)}$ are the input and output of a graph convolutional layer, respectively, and $\tilde{A}$ is a normalized adjacency matrix, with self-connection added. It is worth noting that the learned filters for graph convolution depend on the Laplacian eigenbases. In other words, the model is trained on a specific graph structure, which cannot be applied to a graph with a different structure. In [11], a more flexible model, known as graph attention network, was proposed, which can adaptively learn the graph structure, i.e. $\tilde{A}$ in (3), by performing self-attention without relying on the pre-defined adjacency matrix.

3. METHODOLOGY

3.1. Object Relation Graph Modeling

In our proposed model, an object relation graph is built on top of any region-based detector. Consider an arbitrary graph signal $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of the region proposals, such that $|\mathcal{V}| = N$, and $\mathcal{E}$ is a set of edges with $|\mathcal{E}| = M$. Define $A^{relation}$, a self-adapted adjacency matrix of $\mathcal{G}$, which

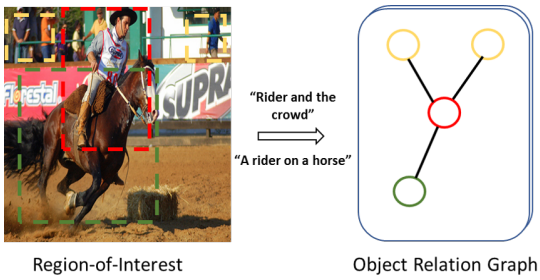

Fig. 1. The object detection task is regarded as a graph inference schedule, in which each node represents a RoI feature and each connection indicates the relativeness between the region proposals.

reflects the relations between each pairs of proposals by performing self-attention. Thus, following (3), we can generate our graph inference as follows:

$$X^{(l+1)} = \sigma(\tilde{A}^{relation}X^{(l)}\Theta^{(l)}), \quad (4)$$

$$\tilde{A}^{relation}_{ij} = \frac{\exp(A^{relation}_{ij})}{\sum_k \exp(A^{relation}_{ik})}, \quad (5)$$

where $\tilde{A}^{relation}$ is a softmax-normalized matrix so that the coefficients are comparable across edges, and $\sigma(\cdot)$ is an activation function. To this end, we aggregate the feature information of each node in a graph, as well as its one-hop neighbors, based on the node-to-node connections. We assume that the connection between two neighboring nodes is governed by an edge weight, which reflects the objects' relationship and can be adaptively learned during the graph inference. This allows the aggregation to focus more on the relevant nodes that have "stronger" relationships. The relationship is described by the geometric relationship R^g_{ij} and the visual relationship R^p_{ij} between the i^{th} and j^{th} proposals. We will show the specific definition of R^g and R^p in the following section.

3.2. Geometric Relationship

We propose a geometric relation module, which aims to model the spatial relationship between two RoIs, e.g. "a person **on top** of a horse", "a cup **on** a table", and "a river **under** a bridge" by calculating an geometric relation feature. This module takes the RoIs' relative geometric features as the input, and then projects them into a subspace to measure how well they are related, by multiplying with a geometric weight $W_g \in \mathbb{R}^{d_g \times 8}$, where d_g is the dimension of geometric relation feature, as follows:

$$R^g_{ij} = \text{ReLU}(W_g r^g_{ij}). \quad (6)$$

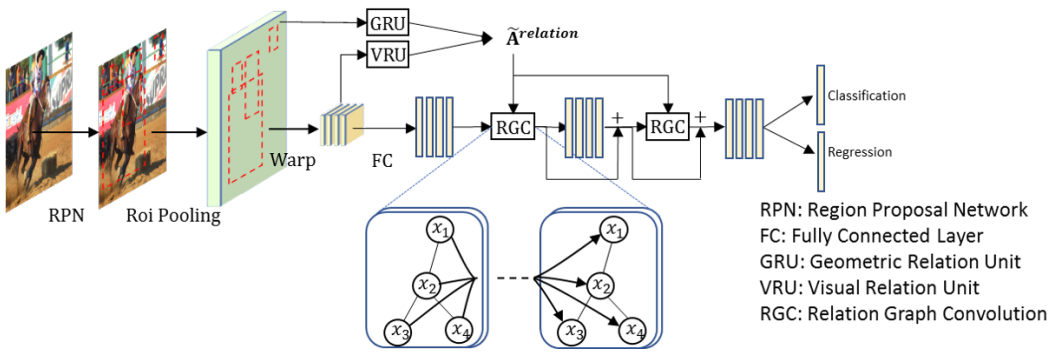

Fig. 2. The architecture of the proposed object detector. First, we use a region proposal network to extract a fixed number of RoIs. Each RoI is transformed into a fixed-size feature by performing RoI pooling. Then, two relation modules, GRU and VRU, are applied to these RoIs and produce a relation matrix. Next, we parse the RoI feature with relation graph inference, which in fact performs feature aggregation based on their relativeness. The output features are more representative compared to the traditional fully connected network.

In order to make the geometric relation invariant to scale and shifting, we define the relative geometric features as follows:

$$r_{ij}^{g} = [w_i, h_i, w_j, h_j, \frac{\|cx_i - cx_j\|}{w_j}, \frac{\|cy_i - cy_j\|}{h_j}, \log(\frac{w_i}{w_j}), \log(\frac{h_i}{h_j})]^{\mathsf{T}}, \tag{7}$$

where w_i and h_i are the width and height of the i^{th} RoI, which is normalized by the image scale, so that $0 < w_i, h_i < 1$. (cx_i, cy_i) is the center position of the i^{th} RoI. ReLU is applied to truncate the feature response by zero, so that it restricts the relations between objects with certain geometric structures.

3.3. Visual Relationship

It is also necessary to model the co-occurrence of pairs of RoIs, e.g. "**a boat** on **a river**" and "**a mouse** nearby **a laptop**". In other words, the visual relationships of the RoIs are modeled, by measuring the impact from one RoI to another one based on their visual cues. This can be achieved by performing self-attention. The module input is a set of RoI features, $X = \{x_1, x_2, ..., x_N\}$, $x_i \in \mathbb{R}^F$, where N is the number of RoIs and F is the feature dimension. In order to obtain sufficient representative power to describe the relations between two RoIs, we concatenate the features from these two RoIs, and transform the resulting feature into another high-dimensional space. To this end, a sharable, learnable transformation matrix $W_v \in \mathbb{R}^{d_v \times 2F}$ is applied to each pair of RoI features, where d_v is the dimension of visual relation features. It can then be calculated as follows:

$$R_{ij}^{v} = \text{LeakyReLU}(W_v[x_i || x_j]), \tag{8}$$

where

$$\text{LeakyReLU}(x) = \begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{otherwise,} \end{cases} \tag{9}$$

and $||$ denotes the concatenation operation. These features exhibit the significance of the j^{th} RoI to the i^{th} RoI based on visual information, instead of structural information. For example, R_{ij}^{v} should be large if x_i and x_j are the features from the "chair" and the "table" RoIs, but small if they are those from the "aeroplane" and the "boat" RoIs. Furthermore, those RoI pairs with negative correlation should be suppressed because the detector may generate false positives. Therefore, LeakyReLU is used as the activation function.

3.4. Object Relation Graph Convolutional Network

The object relation coefficients are computed as the weighted sum of the geometric and visual relation features as follows:

$$A_{ij}^{relation} = \frac{R_{ij}^{g} \cdot \exp(R_{ij}^{v})}{\sum_m R_{mj}^{g} \cdot \exp(R_{mj}^{v})}. \tag{10}$$

Given the object relation coefficients $A_{ij}^{relation}$ between each pair of nodes, we can form the feature aggregation among the RoIs via a Relation Graph Convolutional (RGC) layer, which allows every node to be impacted by every other node, based on their object relationships. Similar to [11], we extend our self-attention module in a multi-head manner. Specifically, we execute the graph inference in (4) K times independently, and concatenate the features to form the final output. To this end, we can formulate our object relation graph convolutional layer with "Algorithm 1". In order to avoid gradient explosion, we employ skip connections [13] for each graph convolutional layer. The overall proposed architecture is illustrated in Fig. 2.

4. EXPERIMENT RESULTS

In this section, we evaluate the effectiveness of our proposed RGC network on top of the Faster RCNN detector. We used VGG-16 [14] as the feature extractor, and extracted 128 RoI

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster RCNN (baseline)	68.95	68.31	77.06	65.22	**54.73**	53.32	**76.06**	79.52	80.31	48.23	72.92	**64.73**	77.86	80.52	75.02	**76.17**	39.42	65.23	64.77	**75.66**	71.25
Ours	**70.83**	**77.44**	**79.09**	**71.55**	53.79	**59.61**	74.94	**85.82**	**82.76**	**52.74**	**75.07**	62.01	**81.89**	**85.79**	**78.14**	75.87	41.16	**66.93**	**67.38**	72.96	**71.74**

Table 1. Detection results on the VOC 2007 test set, trained on 07 trainval.

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast RCNN [9]	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	**74.8**	80.4	70.4
SSD500 [12]	75.1	**79.8**	79.5	74.5	63.4	51.9	84.9	85.6	**87.2**	56.6	80.1	70.0	**85.4**	84.9	**80.9**	78.2	49.0	**78.4**	72.4	**84.6**	75.5
ION [4]	75.6	79.2	**83.1**	**77.6**	65.6	54.9	85.4	85.1	87.0	54.4	**80.6**	**73.8**	85.3	82.2	82.2	74.4	47.1	75.8	72.7	84.2	**80.4**
Faster RCNN (baseline)	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Ours	**76.1**	79.0	79.6	74.9	**65.9**	**63.1**	**86.0**	**87.4**	86.7	**61.3**	80.0	73.2	83.6	**85.5**	79.4	**79.2**	**52.1**	74.3	72.6	82.2	75.1

Table 2. Detection results on the VOC 2007 test set, trained on 07 trainval + 12 trainval.

Algorithm 1 Relation Graph Convolution Layer. The number of RoIs is N. The number of heads is H. Geometric transformation matrix: $W_g^l \in \mathbb{R}^{d_g}$. Visual transformation matrix $W_v^l \in \mathbb{R}^{d_v}$. Layer transformation matrix $\Theta^{(l)} \in \mathbb{R}^{d_{out}/H}$.

Input: input feature: $X^{(l)} \in \mathbb{R}^{N \times d_{in}}$, RoIs position $\{w_i, h_i, cx_i, cy_i\}_{i=0...N-1}$

1: **for** h in $(0, 1, ..., H)$ **do**
2: Calculate $R^g \in \mathbb{R}^{N \times d_g}$ using (6)
3: Calculate $R^v \in \mathbb{R}^{N \times d_v}$ using (8)
4: Calculate $A^{relation} \in \mathbb{R}^{N \times N}$ using (9)
5: Normalization: $\tilde{A}^{relation} = softmax(A^{relation})$
6: Calculate head output $X_h^{(l+1)} \in \mathbb{R}^{d_{out}/H}$ using (4)
7: **end for**
8: Concatenation:$X^{(l+1)} = [X_0^{(l+1)}, ..., X_{H-1}^{(l+1)}]$

Output: $X^{(l+1)} \in \mathbb{R}^{d_{out}}$

features from an input image. Then, we replace the last two fully connected layers with the proposed RGC layers. This replacement introduces less than 5% of parameters, which is negligible. We use the default hyper-parameters for Faster RCNN, except that the learning rate for the proposed layers is increased to 5e-3. Our method is implemented with PyTorch [15], and our source code is available at `https://github.com/skyhehe123/RGC.pytorch`.

4.1. Overall performance

We evaluate our detector on the PASCAL VOC [16] dataset, which has 20 classes. VOC 2007 contains around 5K trainval images and 5K test images, and VOC 2012 contains around 11K trainval images and 11K test images. We first compare our algorithm to Faster RCNN, which is our baseline, on the VOC 2007 test set, and trained on the "07 trainval" split. From Tab. 1, we can see that the performance is improved by a mAP of 1.88% after using our proposed module. We have also conducted another experiment, which adds more training data from the "trainval" split VOC 2012. As shown in Tab. 2, our proposed method can achieve an excellent mAP of 76.1% on VOC 2007, which exceeds most of the state-of-the-art methods.

4.2. Object Relation Visualization

Fig. 3 shows the detected objects with large relation coefficients, after performing the non-maximum suppression. As shown in the left image, the most related object to the "table" is the "chair", while "person" also exhibits high relativeness to the "table". The most related objects to the "bicycle" are the "person" riding on the bicycle and another "bicycles" nearby. It is worth noting that the coefficients obviously reflect the co-occurrence between instances, as well as the spatial relation, because an instance pair with more reasonable spatial distance tends to give larger coefficients.

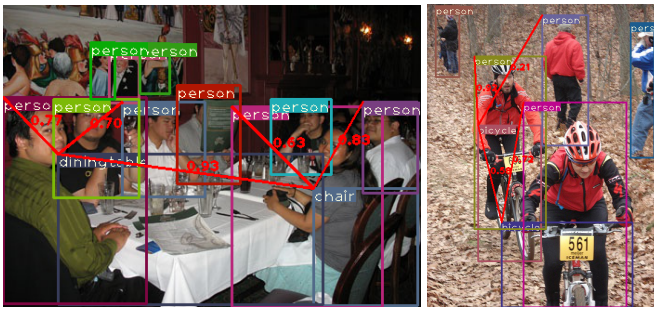

Fig. 3. Object relation visualization. The red line indicates the existence of an object relationship between two bounding-boxes. The coefficients reflect the relativeness of the object pairs. For better visualization, we only show some of the relation pairs with high coefficients.

5. CONCLUSION

In this paper, we proposed an enhancement module for object detection, whose structure is based on an object relation graph. The graph is constructed based on both the objects' geometric and visual relations, so that the features of a potential object region can be enriched by aggregating the information from other regions. Our experiments have shown the effectiveness of our proposed module, and achieved a gain in terms of a mAP by 1.88% on VOC 2007. Furthermore, we have validated that our method can achieve better performance with more training data.

6. REFERENCES

[1] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 1, p. 4.

[2] C. H. He and K. M. Lam, "Fast vehicle detection with lateral convolutional neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2341–2345.

[3] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005, vol. 2, pp. 60–65.

[4] S. Bell, Z. Lawrence, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2874–2883.

[5] Z. Tu and X. Bai, "Auto-context and its application to high-level vision tasks and 3d brain image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 32, no. 10, pp. 1744–1757, 2010.

[6] X. Chen and A. Gupta, "Spatial memory for context reasoning in object detection," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4086–4096.

[7] C. Galleguillos, A. Rabinovich, and S. Belongie, "Object categorization using co-occurrence, location and appearance," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2016.

[9] R. Girshick, "Fast r-cnn," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems (NIPS)*, 2015, pp. 91–99.

[11] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2018.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Springer European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[15] "Pytorch," `http://pytorch.org/`.

[16] M. Everingham, L. V. Cool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.