Rotation-Insensitive and Multi-class Object Detection in Remote Sensing Images

Zhonghan Chang (changzhonghan16@mails.ucas.ac.cn), Zhirui Wang,Xian Sun, Fun Fu

I. DOAI Information

- Username: czh
- Institute: IECAS
- Emailadress: changzhonghan16@mails.ucas.ac.cn
- TeamMembers: Zhonghan Chang

II. Proposed Work

A. Overview

We propose an effective object detection framework for multiple class and arbitrary-oriented object detection in remote sensing images. Given a satellite image, the detector first uses a standard ConvNets to compute feature maps on a satellite image. A feature-fusion architecture is used to process the feature maps and generate multi-scale feature representations that are shared with RPN and detection subnetworks. Then, we use the RPN to predict region proposals, which are defined as minimum bounding boxes around candidate objects. For each object proposals, RoI Pooling is performed on the feature maps to extract its features with a fixed size and the features are fed into subnetworks for classification and rotated bounding box regression. Finally, we apply non-maximum suppression on the detection boxes for post-progression.

B. Feature-Fusion Architecture

Our detection framework is built upon ImageNet-pretrained ConvNet architectures which learns a coarse, highly semantic feature representation. Multi-scale object detection in satellite images requires both high-level semantic representations and low-level finer details. We handle this issue by incorporate a feature-fusion architecture into the object detection framework.

In order to build a feature-fusion architecture, we first augment the backbone network with a top-down architecture which learns to incorporate semantic information from high-level feature maps into the low-level, finer details features. We divide backbone network (e.g, VGG, ResNets) into several ConvNet blocks as C_i, where each block consists of multiple layers generating feature maps of the same size. These ConvNet blocks progressively decrease the resolution of the output with a stride of 2 in a forward pass. The top-down module is represented by several top-down blocks (defined as T_i). In the top-down pathway, each block T_i connects to the preceding top-down block T_{i-1} as well as to the last layer of matching ConvNet block C_i via lateral connections. A top-down block takes the feature maps from backbone network as input, along with refined features in top-down path, and generate new up-sampled feature maps. All the top-down blocks process the features in an iterative procedure and restore the spatial size of feature maps with a factor of 2. After building a top-down module, a bottom-up-module is aggregated to the network Similar to top-down module, the bottom-up module is composed of several stacked blocks, defined as (defined as D_i). The first block D_2 connects to the final output of top-down module. In the next, the output of bottom-up module D_i and the corresponding top-down block T_i are fed into the following block D_{i+1}. Similar to the top-down blocks, the bottom-up blocks work in an iterative fashion and propagate high-resolution, refined feature maps into the highly semantic but lacking-details feature maps from high layers. In the bottom-up module, the spatial size of output feature map is $2\times$ smaller than the input, which is different from the characteristic of the top-down blocks.

In the previous work an object detector is built upon a top-down architecture and only makes prediction on the final top-down feature maps with highest resolution. The top-down module learns to refine the backbone features with high-level feature maps, ensuring that the whole network get necessary context information for object recognize. We argue that the high-level feature maps are not suitable for accurate object localization due to the lack of finer details information. In our work, the bottom-up module is responsible for incorporating finer details from low-level feature maps into the detection network. With the top-down module and bottom-up module working in a cascaded version, the feature-fusion architecture ensures the object detector to learn a feature hierarchy with both rich semantic information and finer details simultaneously. In this way, all level feature maps of the network are utilized to exploit object information for object detection in our work.

C. Region Proposal Network

Ours detection framework is a region-based objector based on Faster R-CNN. In Faster R-CNN, horizontal bounding boxes is used to localize object coarsely. But we use oriented bounding box provide a precise spatial location for arbitrary oriented object. In our method, the location of a target object is denoted as an oriented bounding box by a five-tuples $b = (x, y, w, h, \theta)$, where (x, y) specifies the center coordinates of the matching bounding box, (w, h) represents long side and short side of the bounding box respectively, and θ represents the angle of the long side from the horizontal direction.

Following Faster R-CNN, RPN is applied on feature maps in a sliding window fashion, generating objectness scores and bounding box regression offsets for anchor boxes. Faster R-CNN predicting proposals on a single level feature are not enough for multi-scale object detection. In this work, the RPN operates on the feature hierarchy from feature-fusion architecture. Sincerely, we define anchor boxes with default scales of $\{16, 32, 64, 128, 256\}$ on feature maps $\{P_2, P_3, P_4, P_5, P_6\}$ independently. In order to handle the large diversity of object sizes, we introduce anchors of additional scales of $\{2^{0/k}, 2^{1/k}, \dots, 2^{k-1/k}\}$ with respect to the default scales in each level. The parameter k controls the number and scales of extra anchors. By default, it's assigned of $k = 3$ and the anchors across all the level can cover 15 scales range from 16 to 407 pixels. In precious methods, anchors are defined as horizontal boxes with multiple scales and aspect ratios at each position which fails to adapt to oriented boxes regression. We adopt several modifications of anchors. First, we argument the anchors by introducing multiple angles including $\left(0, \frac{\pi}{2}\right)$ to involve prior orientation information. Second, the aspect ratio of anchor is also adjusted to $(1{:}1.5, 1{:}3)$ to cover objects of different shapes. Therefore, we assign 12 anchor boxes with 2 angles, 3 aspect ratios and a set of increasing scales for the matching feature maps. This paradigm makes the anchor boxes have dense distribution spatially and leads to more matching boxes for object with arbitrary orientations, scales and aspect ratios.

Benefited from the multi-scale feature hierarchy generated by feature-fusion architecture, the RPN is not necessary to handle the anchors with all scales on any specific level. Instead, the anchor box with larger scales is assign to the high-level feature map and the RPN needs to predict region proposals of different scales on the matching level. Although there are anchor boxes with 15 scales, 2 aspect ratios and 3 angles in total, the RPN only has to handle 12 anchors per level. The RPN is constructed by append a ConvNets architecture with two fully connected layers. The output of classification layer is connected to sigmoid activation functions and make binary classification for each anchor independently. The regression layer outputs 5 parameterized encodings for each anchor. Therefore, the RPN is trained to generate 60 outputs (12×5) for anchor boxes regression, and 24 outputs (12×2) for region classification at each sliding-window position.

D. *Object Detection*

For the object detection, we first adopt the backbone network (with feature-fusion module) to process the image, producing a set of feature maps, then RoI-Pooling operation is involved to extract a set of features from the feature maps for each region proposal. In faster R-CNN [], the RoI pooling layer is implemented to extract feature map inside a horizonal rectangle window and is not appropriate for oriented object detection. In order to remedy this issue, the RoI pooling layer is adapt to handle arbitrary-oriented region proposals. In the first stage of detection, the RPN generate a number of region proposals, each one of them can be represented as an oriented bounding box. We introduce the proposed RoI-pooling operation starting from extracting region feature from a feature map f and one region proposal denoted by b here. The feature map f has a stride of S with respect to the input image. For a region proposal, a set of evenly spaced points are sampled within the oriented rectangle window over a specified interval. The coordinates of the set of sampling points on the input image is denoted as $\mathbf{p} = \{p_{u,v}\}_{1\le u\le K, 1\le v\le K}$, here, the parameter u and v denotes the index of sampling points along the long side and short side of the region proposal respectively. The parameter K controls the number of sampling points for a proposal. Then the region proposal is projected onto the feature map as an oriented window. The relative coordinates of the sampling point $p'_{u,v}$ on the feature map f is computed as $q_{u,v} = q_{u,v}/S$. As the coordinates of sampling points may not be exactly dividable by the stride factor S, the data-type of transformed coordinates is float-number. For a sampling point $q_{u,v} = (x_{u,v}, y_{u,v})$, there are four neighborhood cell of feature map defined by $\{q_{u,v}^{11}, q_{u,v}^{12}, q_{u,v}^{21}, q_{u,v}^{22}\}$ surrounding its location. Then we use bilinear interpolation to compute the value of the feature map at the sampling location $q_{u,v}$ and it is defined as:

$$f(q_{u,v}) = \sum_{1\le i\le 2}\sum_{1\le j\le 2} \omega(q_{u,v}^{ij}, q_{u,v}) \cdot f(q_{u,v}^{ij}) \tag{6}$$

Here, the function $f(\cdot)$ denotes the value of feature map f at the given location. The function $\omega(a,b)$ denotes the bilinear interpolation weight:

$$\omega(a,b) = (1-(x_a - x_b)) \cdot (1-(y_a - y_b)) \tag{7}$$

Therefore, every sampling points within an oriented proposal can be computed by bilinear interpolation. And it's straightforward to extend the proposed RoI-pooling to the practical case, where a lot of RoIs should be processed. The RoI-pooling operation doesn't involve any complex operation or neural networks, while introducing nearly negligible computation cost. In our experiment, we set the number of sampling points $K = 14$ by default for saving computation cost. After bilinear interpolation, the extracted feature values inside each proposal are aggregated by max pooling with a stride of 2. Finally, we can extract feature representation of fixed spatial size 7×7 for each region proposal. Benefit to the bilinear interpolation, the modified RoI-pooling is sensitive to the alteration of sampling position and ensures the network obtain the accurate feature value of each RoI.

In the R-CNN subnetwork, we first adopt two fully-connected layers with 1024 channels to process the RoI feature maps. The first two layers are responsible for the selecting appropriate component from the multi-level features and computing a shared feature for following layers. We append two parallel output layers to the end of network. The first output layer has a softmax function and predicts the probability over K classes, denoted as $p = (p_0, \dots, p_K)$. The second later outputs bounding box regression for each of the K classes, denoted as $t_k = (t_x^k, t_y^k, t_w^k, t_h^k, t_\theta^k)$ and k is the class index.