Align Deep Features for Oriented Object Detection

Jiaming Han, Jian Ding, Jie Li, Gui-Song Xia

***Abstract*—The past decade has witnessed significant progress on detecting objects in aerial images that are often distributed with large scale variations and arbitrary orientations. However most of existing methods rely on heuristically defined anchors with different scales, angles and aspect ratios and usually suffer from severe misalignment between anchor boxes and axis-aligned convolutional features, which leads to the common inconsistency between the classification score and localization accuracy. To address this issue, we propose a *Single-shot Alignment Network* (S^2A-Net) consisting of two modules: a Feature Alignment Module (FAM) and an Oriented Detection Module (ODM). The FAM can generate high-quality anchors with an Anchor Refinement Network and adaptively align the convolutional features according to the anchor boxes with a novel Alignment Convolution. The ODM first adopts active rotating filters to encode the orientation information and then produces orientation-sensitive and orientation-invariant features to alleviate the inconsistency between classification score and localization accuracy. Besides, we further explore the approach to detect objects in large-size images, which leads to a better trade-off between speed and accuracy. Extensive experiments demonstrate that our method can achieve state-of-the-art performance on two commonly used aerial objects datasets (*i.e.*, DOTA and HRSC2016) while keeping high efficiency. The code is available at https://github.com/csuhan/s2anet.**

***Index Terms*—Object detection, aerial images, deep learning, feature alignment**

I. Introduction

OBJECT detection in aerial images aims at identifying the locations and categories of objects of interest (*e.g.*, planes, ships, vehicles). With the framework of Deep Convolutional Neural Networks (DCNNs), object detection in aerial images (ODAI) has made significant progress in recent years [1], [2], [3], [4], [5], where most of existing methods are devoted to cope with the challenges raised by the large scale variations and arbitrary orientations of crowded objects in aerial images.

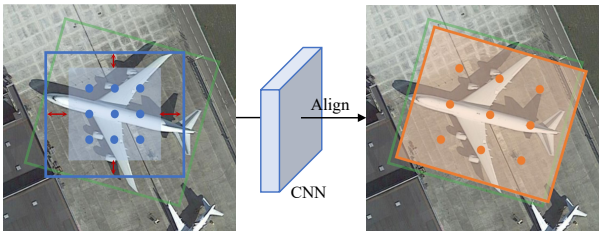
To achieve better detection performance, most state-of-the-art aerial object detectors [4], [5], [6], [7] rely on the complex R-CNN [8] frameworks, which consist of two parts: a Region Proposal Network (RPN) and an R-CNN detection head. In a general pipeline, the RPN is used to generate high-quality Region of Interests (RoIs) from horizontal anchors, and then an RoI Pooling operator is adopted to extract accurate features from RoIs. The R-CNN is finally employed to regress the bounding boxes and classify them into different categories.

J. Han, J. Ding are with the State Key Lab. LIESMARS and the Institute of Artificial Intelligence, Wuhan University, Wuhan, 430079, China.

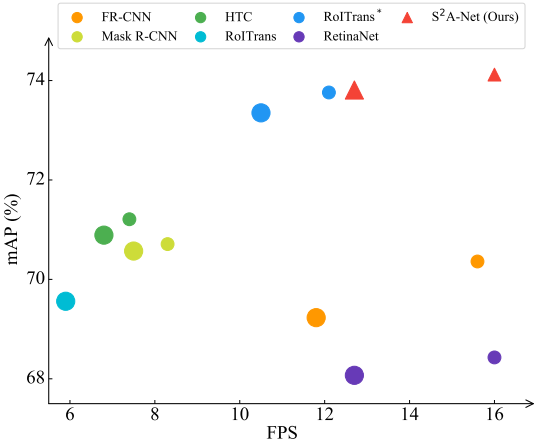
J. Li is with Shanghai Aerospace Electronic Technology Institute, Shanghai, China. E-mail: *trackerdsp@163.com*.

G.-S. Xia is with the School of Computer Science, the State Key Lab. LIESMARS, and also the Institute of Artificial Intelligence, Wuhan University, Wuhan, 430072 China. E-mail: *guisong.xia@whu.edu.cn*.

This study in this article is funded by the National Natural Science Foundation of China (NSFC) under the grant contracts No.61922065, No.61771350 and No.61771350.



(a) The misalignment issue and our solution



(b) Speed vs. accuracy (mAP) on DOTA `test-dev`

Fig. 1. **(a)** The misalignment (red arrows) between an anchor box (blue bounding box) and convolutional features (light blue rectangle). To alleviate this issue, we first refine the initial anchor into a rotated one (orange bounding box), and then adjust the feature sampling locations (orange points) with the guide of the refined anchor box to extract aligned deep features. The green box denotes the ground truth. **(b)** Performance comparisons of different methods under the same settings: ResNet50 (in small markers) and ResNet101 (in big markers) backbones, 1024 × 1024 input size of images, without data augmentation. FR-CNN [9], Mask R-CNN [10], RetinaNet [11], Hybird Task Cascade (HTC) [12] and RoI Transformer (RoITrans) [4] are tested. The speed of all methods is reported on V100 GPU. Note that Mask R-CNN, HTC and RoITrans are tested based on the `AerialDetection`[1] project. RoITrans* indicates an official re-implementation.

However, it is worth noticing that horizontal RoIs often result in severe misalignment between bounding boxes and oriented objects [4], [3]. For example, a horizontal RoI usually contains several instances due to oriented and densely packed objects in aerial images. A natural solution is employing oriented bounding boxes as anchors to alleviate this issue [2], [3]. As a consequence, well-designed anchors with different angles, scales and aspect ratios are required, which however leads to massive computations and memory footprint. Recently, RoI Transformer [4] was proposed to address this issue by transforming horizontal RoIs into rotated RoIs, avoiding a large number of anchors, but it still needs heuristically defined anchors and complex RoI operation.

[1] https://github.com/dingjiansw101/AerialDetection.

In contrast with R-CNN based detectors, one-stage detectors regress the bounding boxes and classify them directly with regular and densely sampling anchors. This architecture enjoys high computational efficiency but often lags behind in accuracy [3]. As shown in Fig. 1 (a), we argue that severe misalignment in one-stage detectors matters:

- Heuristically defined anchors are with low-quality and cannot cover the objects, leading a misalignment between objects and anchors. For example, the aspect ratio of a bridge usually ranges from $1/3$ to $1/30$, and only a few or even no anchors can be assigned to it. This misalignment usually aggravates the foreground-background class imbalance and hinders the performance.
- The convolutional features from the backbone network are usually axis-aligned with fixed receptive field, while objects in aerial images are distributed with arbitrary orientations and variant appearances. Even an anchor box is assigned to an instance with high confidence (*i.e.*, IoU), there is still a misalignment between anchor boxes and convolutional features. In other words, the corresponding features of an anchor box are hard to represent the whole object to some extent. As a result, the final classification score can not accurately reflect the localization accuracy, which also hinders the detection performance in post-processing phase (*e.g.*, non-maximum suppression (NMS)).

To address these issues in one-stage detectors, we propose a *Single-Shot Alignment Network* (S^2A-Net) which consists of two modules: a Feature Alignment Module (FAM) and an Oriented Detection Module (ODM). The FAM can generate high-quality anchors with an Anchor Refinement Network (ARN) and adaptively align the feature according to the corresponding anchor boxes (Fig 1(a)) with an Alignment Convolution (AlignConv). Different from other methods with densely sampling anchors, we employ only one squared anchor for each location in the feature map, and the ARN refines them into high-quality rotated anchors. Then the AlignConv, a variant of convolution, adaptively aligns the feature according to the shapes, sizes and orientations of its corresponding anchors. In the ODM, we first adopt active rotating filters (ARF) [13] to encode the orientation information and produce orientation-sensitive features, and then extract orientation-invariant features by pooling the orientation-sensitive features. Finally, we feed the features into a regression sub-network and a classification sub-network to yield the final predictions. Besides, we also explore the approach to detect objects on large-size images (*e.g.*, 4000×4000) rather than on chip images, which significantly reduces the overall inference time with negligible loss of accuracy. Extensive experiments on commonly used datasets, *i.e.*, DOTA [3] and HRSC2016 [14], demonstrate that our proposed method can achieve state-of-the-art performance while keeping high efficiency, see Fig 1 (b).

Our main contributions are summarized as follows:

- We propose a novel Alignment Convolution to alleviate the misalignment between axis-aligned convolutional features and arbitrary oriented objects in a fully convolutional way. Note AlignConv has negligible extra consuming time compared with standard convolution and can be embedded into many detectors with little modification.
- With the Alignment Convolution embedded, we design a light Single-Shot Alignment Network which enables us to generate high-quality anchors and aligned features for accurate object detection in aerial images.
- We report 79.42% mAP on the oriented object detection task on the DOTA dataset, achieving the state-of-the-art in both speed and accuracy.

II. Related Works

With the advance of machine learning, especially deep learning, object detection has made significant progress in recent years, which can be roughly divided into two groups: two-stage detectors and one-stage detectors. Two-stage detectors [8], [15], [9], [10] first generate a sparse set of RoIs in the first stage, and perform an RoI-wise bounding box regression and object classification in the second one. One-stage detectors, *e.g.*, YOLO [16] and SSD [17], detect objects directly and do not require the RoI generation stage. Generally, the performance of one-stage detectors usually lag behind two-stage detectors due to extreme foreground-background class imbalance. To address this problem, the *Focal Loss* [11] can be used, and anchor-free detectors [18], [19], [20] alternatively formulate object detection as a points detection problem to avoid complex computations related to anchors and usually run faster.

A. Object Detection in Aerial Images

Objects in aerial images are often crowded, distribute with large scale variations and appear at arbitrary orientations. Generic object detection methods with horizontal anchors [3] usually suffer from severe misalignment in such scenarios: one anchor/RoI may contain several instances. Some methods [2], [21], [22] adopt rotated anchors with different angles, scales and aspect ratios to alleviate this issue, while involving heavy computations related to anchors (*e.g.*, bounding box transform and ground truth matching). Ding *et al.* [4] propose RoI Transformer to transform horizontal RoIs into rotated RoIs, which avoids a large number of anchors and alleviates the misalignment issue. However, it still needs heuristically defined anchors and complex RoI operations. Instead of employing rotated anchors, Xu *et al.* [7] glide the vertex of the horizontal bounding box to accurately describe an oriented object. But the corresponding feature of a RoI is still horizontal and suffers from the misalignment issue. Recently proposed R^3Det [23] samples features from five locations (*e.g.*, center and corners) of the corresponding anchor box and sum them up to re-encode the position information. In contrast with the above methods, the proposed S^2A-Net in this paper gets ride of heuristically defined anchors and can generate high-quality anchors by refining horizontal anchors into rotated anchors. Besides, the proposed FAM module enables to achieve feature alignment in a fully convolutional way.

B. Feature Alignment in Object Detection

Feature alignment usually refers to the alignment between convolution features and anchor boxes/RoIs, which is important

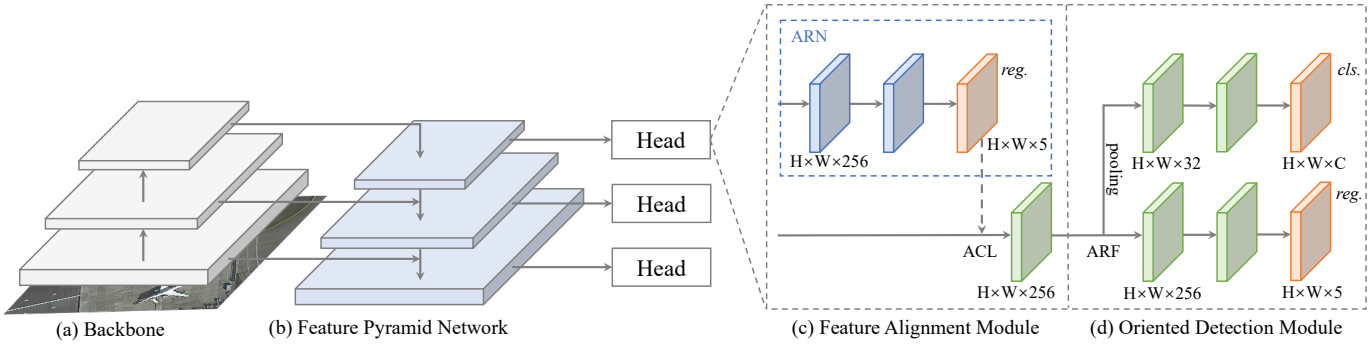


Fig. 2. Architecture of the proposed S^2A-Net. S^2A-Net consists of a backbone network, a Feature Pyramid Network [11], a Feature Alignment Module (FAM) and an Oriented Detection Module (ODM). The FAM and ODM make up the detection head which is applied to each scale the the feature pyramid. In FAM, the Anchor Refinement Network (ARN) is proposed to generate high-quality rotated anchors. Then we feed the anchors and input features into the Alignment Convolution Layer (ACL) to extract aligned features. Note we only visualize the regression (*reg.*) branch of ARN and ignore the classification (*cls.*) branch for simplification. In ODM, we first adopt active rotating filters (ARF) [13] to generate orientation-sensitive features, and pool the features to extract orientation-invariant features. Then a *cls.* branch and a *reg.* branch are applied to produce the final detections.

for both two-stage and one-stage detectors. Detectors relying on misaligned features are hard to obtain accurate detections. In two-stage detectors, an RoI operator (*e.g.*, RoIPooling [15], RoIAlign [10] and Deformable RoIPooling [24]) is adopted to extract fixed-length features inside the RoIs which can approximately represent the location of objects. RoIPooling first divides an RoI into a grid of sub-regions and then max-pools each sub-region into the corresponding output grid cell. However, RoIPooling quantizes the floating-number boundary of an RoI into integer, which introduces misalignment between the RoI and the feature. To avoid the quantization of RoIPooling, RoIAlign adopts bilinear interpolation to compute the extract values at each sampling location in sub-regions, significantly boosting the performance of localization. Meanwhile, Deformable RoIPooling adds an offset to each sub-region of an RoI, enabling adaptive feature selection. However, the RoI operator usually involves massive region-wise operation, *e.g.*, feature warping and feature interpolation, which becomes a bottleneck toward fast object detection.

Recently, Guided Anchoring [25] tries to align features with the guide of anchor shapes. It learns an offset field from the anchor prediction map and then guides the Deformable Convolution (DeformConv) to extract aligned features. AlignDet [26] designs an RoI Convolution to obtain the same effect as RoIAlign in one-stage detector. Both [25] and [26] achieve feature alignment in a fully convolutional way and enjoy high efficiency. These methods work well for objects in nature images but often lose their performance when detecting objects that are oriented and densely packed in aerial images, although some of them (*e.g.*, Rotated RoIPooling [22] and Rotated Position Sensitive RoIAlign [4]) have been adopted to achieve feature alignment in oriented object detection. Different from the aforementioned methods, our proposed method aims at alleviating the misalignment between axis-aligned convolutional features and arbitrary oriented objects, which adjusts the feature sampling locations with the guide of anchor boxes.

C. Inconsistency between Regression and Classification

An object detector usually consists of two parallel tasks: bounding-box regression and object classification, which share the same features from the backbone network. And the classification score is used to reflect the localization accuracy in a post-processing phase (*e.g.*, NMS). However, as discussed in [27] and [28], there is a common inconsistency between classification score and localization accuracy. Detections with high classification scores may produce bounding boxes with low localization accuracy. While other nearby detections with high localization accuracy may be suppressed in the NMS step. To address this issue, IoU-Net [27] proposed to learn to predict the IoU of a detection as the localization confidence and then combine the classification score and localization confidence as the final probability of a detection. Double-Head R-CNN [28] adopts different head architectures for different tasks, *i.e.*, fully connected head for classification and convolution head for regression. In our methods, we aim to improve the classification score by extracting aligned features for each instance. Especially when detecting densely packed objects in aerial images, accurate features are important to robust classification and precise localization. Besides, as discussed in [28], shared features from the backbone are not suitable for both classification and localization. Inspired by [13] and [29], we first adopt active rotating filters to encode the orientation information and then extract orientation-sensitive features and orientation-invariant features for regression and classification, respectively.

III. Proposed Method

In this section, we first enable RetinaNet for oriented object detection and select it as our baseline in Section III-A. We then detail the Alignment Convolution in Section III-B. The architectures of Feature Alignment Module and Oriented Detection Module are presented in Section III-C and Section III-D, respectively. Finally, we show details of the proposed S^2A-Net in both training and inference phase. The overall architecture is shown in Fig. 2. And the code is available at https://github.com/csuhan/s2anet.

A. RetinaNet as Baseline

We choose a representative single-shot detector, RetinaNet [11] as our baseline. It consists of a backbone network

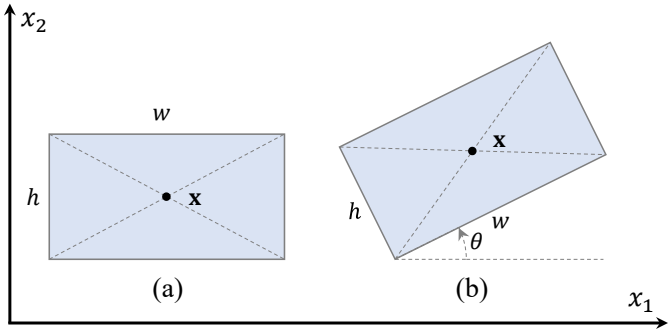


Fig. 3. Two types of bounding box. **(a)** Horizontal bounding box $\{(\mathbf{x}, w, h)\}$ with center point $\mathbf{x} = (x_1, x_2)$, width w and height h. **(b)** Oriented bounding box $\{(\mathbf{x}, w, h, \theta)\}$. $\mathbf{x}$ denotes the center point. w and h represent the long side and short side of a bounding box, respectively. θ means the angle from the position direction of x_1 to the direction of w where $\theta \in [-\frac{\pi}{4}, \frac{3\pi}{4}]$. And an oriented bounding box turns to a horizontal one when $\theta = 0$, *e.g.*, $(\mathbf{x}, w, h, 0)$.

and two task-specific subnetworks. Feature pyramid network (FPN) [30] is adopted as the backbone network to extract multi-scale features. Classification and regression subnetworks are fully convolutional networks with several (*i.e.*, 4) stacked convolution layers. Moreover, Focal loss is proposed to address the extreme foreground-background class imbalance during training.

Note that RetinaNet is designed for generic object detection, outputting horizontal bounding box (Fig. 3 (a)) represented as,

$$\{(\mathbf{x}, w, h)\},$$

with $\mathbf{x} = (x_1, x_2)$ as the center of the bounding box. In order to compatible with oriented object detection, we replace the regression output of the RetinaNet with oriented bounding box (Fig. 3 (b)) as,

$$\{(\mathbf{x}, w, h, \theta)\},$$

where $\theta \in [-\frac{\pi}{4}, \frac{3\pi}{4}]$ denotes the angle from the position direction of x_1 to the direction of the width w [4]. All other settings keep unchanged with original RetinaNet.

B. Alignment Convolution

In a standard 2D convolution, we first sample over the input feature map $\mathbf{X}$ defined on $\Omega = \{0, 1, \ldots, H-1\} \times \{0, 1, \ldots, W-1\}$ by a regular grid $\mathcal{R} = \{(r_x, r_y)\}$, and then sum up the sampled values weighted by $\mathbf{W}$. For example, the grid $\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}$ represents a kernel size 3×3 and dilation 1. For each location $\mathbf{p} \in \Omega$ on the output feature map $\mathbf{Y}$, we have

$$\mathbf{Y}(\mathbf{p}) = \sum_{\mathbf{r} \in \mathcal{R}} \mathbf{W}(\mathbf{r}) \cdot \mathbf{X}(\mathbf{p} + \mathbf{r}). \tag{1}$$

Compared with standard convolution, Alignment Convolution (AlignConv) adds an additional offset field $\mathcal{O}$ for each location $\mathbf{p}$, that is

$$\mathbf{Y}(\mathbf{p}) = \sum_{\mathbf{r} \in \mathcal{R};\, \mathbf{o} \in \mathcal{O}} \mathbf{W}(\mathbf{r}) \cdot \mathbf{X}(\mathbf{p} + \mathbf{r} + \mathbf{o}). \tag{2}$$

As shown in Fig. 4 (c) and (d), for location $\mathbf{p}$, the offset field $\mathcal{O}$ is calculated as the difference between anchor-based sampling locations and regular sampling locations (*i.e.*, $\mathbf{p} + \mathbf{r}$). Let $(\mathbf{x}, w, h, \theta)$ represent the corresponding anchor box at location $\mathbf{p}$. For each $\mathbf{r} \in \mathcal{R}$, the anchor-based sampling location $\mathbf{L}_\mathbf{p}^\mathbf{r}$ can be defined as

$$\mathbf{L}_\mathbf{p}^\mathbf{r} = \frac{1}{S}\big(\mathbf{x} + \frac{1}{k}(w, h) \cdot \mathbf{r}\big) R^T(\theta), \tag{3}$$

where k indicates the kernel size, S denotes the stride of the feature map, and $R(\theta) = \big(\cos\theta, -\sin\theta; \sin\theta, \cos\theta\big)^T$ is the rotation matrix, respectively. The offset field $\mathcal{O}$ at location $\mathbf{p}$ is

$$\mathcal{O} = \sum_{\mathbf{r} \in \mathcal{R}} (\mathbf{L}_\mathbf{p}^\mathbf{r} - \mathbf{p} - \mathbf{r}). \tag{4}$$

In this way, we can transform the axis-aligned convolutional features $\mathbf{X}(\mathbf{p})$ of a given location $\mathbf{p}$ into arbitrary oriented ones based on the corresponding anchor box.

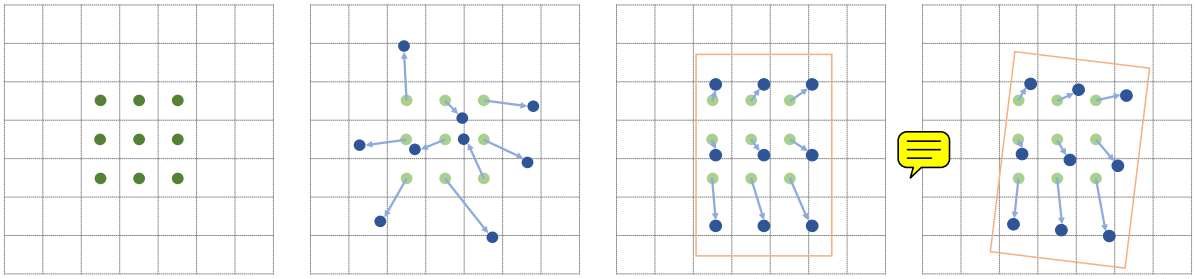
Comparisons with other convolutions. As shown in Fig. 4, standard convolution samples over the feature map by a regular grid. DeformConv learns an offset field to augment the spatial sampling locations. However, it may sample from wrong locations with weak supervision, especially for densely packed objects. Our proposed AlignConv extracts grid-distributed features with the guide of anchor boxes by adding an additional offset field. Different from DeformConv, the offset field in AlignConv is inferred from the anchor boxes directly. The examples in Fig. 4 (c) and (d) illustrate that our AlignConv can extract accurate features inside the anchor boxes.

C. Feature Alignment Module (FAM)

This section introduces the FAM that consists of an Anchor Refinement Network and an Alignment Convolution Layer, illustrated in Fig. 2 (c).

Anchor Refinement Network. The Anchor Refinement Network (ARN) is a light network with two parallel branches: an anchor classification branch (not shown in the figure) and an anchor regression branch. The anchor classification branch classifies anchors into different categories and the anchor regression branch refines horizontal anchors into rotated anchors with high-quality. By default, the classification branch is discarded in the inference phase to speed up the model. But for a fast version of S^{2}A-Net (see Section IV-D), the output of the classification branch is reserved to suppress false predictions in NMS. Following the one-to-one fashion in anchor-free detectors, we preset one squared anchor for each location in the feature map. And we do not filter out the predictions with low confidence because we notice that some negative predictions turn to positive in the final predictions.

Alignment Convolution Layer. With AlignConv embedded, we forms an Alignment Convolution Layer (ACL) which is shown in Fig. 5. Specifically, for an $H \times W \times 5$ anchor prediction map, we first decode the relative offset $(\Delta\mathbf{x}, \Delta w, \Delta h, \Delta\theta)$ into the absolute anchor boxes $(\mathbf{x}, w, h, \theta)$. Then the offset field calculated by Eq. (4) along with the input feature are fed into AlignConv to extract aligned features. Note that for each anchor box, we sample 9 (*i.e.*, 3 rows and 3 columns) points to obtain the offset field with a channel of 18 (*i.e.*, the horizontal and vertical offset of 9 points). It should be emphasized that ACL is a light convolution layer with negligible speed latency in offset field calculating.



(a) (b) (c) (d)

Fig. 4. Illustration of the sampling locations in different methods with 3×3 kernel. (a) is the standard 2D convolution. (b) is Deformable Convolution [24]. (c) and (d) are two examples of our proposed AlignConv with horizontal and rotated anchor box (orange rectangle), respectively.

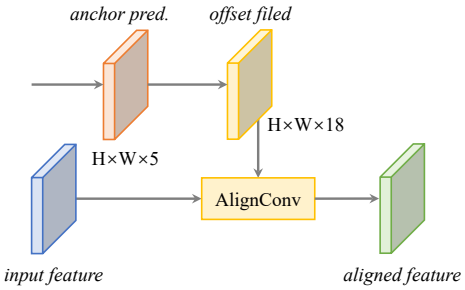


Fig. 5. Alignment Convolution Layer. It takes the input feature and the anchor prediction (*pred.*) map as input and output the aligned feature.

D. Oriented Detection Module (ODM)

As shown in Fig. 2 (d), the Oriented Detection Module (ODM) is proposed to alleviate the inconsistency between classification score and localization accuracy and then performs accurate object detection. We first adopt active rotating filters (ARF) [13] to encode the orientation information. An ARF is a $k \times k \times N$ filter that actively rotates $N-1$ times during convolution to produce a feature map with N orientation channels (N is 8 by default). For a feature map $\mathbf{X}$ and an ARF $\mathbf{F}$, the i-th orientation output of $\mathbf{Y}$ can be denoted as

$$\mathbf{Y}^{(i)} = \sum_{n=0}^{N-1} \mathbf{F}_{\theta_i}^{(n)} \cdot \mathbf{X}^{(n)}, \ \theta_i = i\frac{2\pi}{N}, \ i = 0, \ldots, N-1, \quad (5)$$

where $\mathbf{F}_{\theta_i}$ is the clockwise θ_i-rotated version of $\mathbf{F}$, $\mathbf{F}_{\theta_i}^{(n)}$ and $\mathbf{X}^{(n)}$ are the n-th orientation channel of $\mathbf{F}_{\theta_i}$ and $\mathbf{X}$, respectively. Applying ARF to the convolution layer, we can obtain orientation-sensitive feature with explicitly encoded orientation information. The bounding box regression task benefits from the orientation-sensitive feature, while the object classification task requires invariant features. Following [13], we aims to extract orientation-invariant feature by pooling the orientation-sensitive feature. This is simply done by choosing the orientation channel with the strongest response as the output feature $\hat{\mathbf{X}}$.

$$\hat{\mathbf{X}} = \max \mathbf{X}^{(n)}, \ 0 < n < N-1. \quad (6)$$

In this way, we can align the feature of objects with different orientations, toward robust object classification. Compared with the orientation-sensitive feature, the orientation-invariant feature is efficient with fewer parameters. For example, an $H \times W \times 256$ feature map with 8 orientation channels becomes $H \times W \times 32$ after pooling. Finally, we feed the orientation-sensitive feature and orientation-invariant feature into two subnetworks to regress the bounding boxes and classify the categories, respectively.

E. Single-Shot Alignment Network

We adopt RetinaNet as the baseline, including its network architecture and most parameter settings, and form S^{2}A-Net based on the combination of FAM and ODM. In the following, we detail S^{2}A-Net in both training and inference phase.

Regression targets. Following previous works, we give the parameterized regression targets as:

$$\begin{aligned} \Delta\mathbf{x}_g &= \big(\mathbf{x}_g - \mathbf{x}\big)R(\theta) \cdot \big(\frac{1}{w}, \frac{1}{h}\big), \\ (\Delta w_g, \Delta h_g) &= \log(w_g, h_g) - \log(w, h), \\ \Delta\theta_g &= \frac{1}{\pi}(\theta_g - \theta + k\pi), \end{aligned} \quad (7)$$

where $\mathbf{x}_g$, $\mathbf{x}$ are for the ground-truth box and the anchor box respectively (likewise for w, h, θ). And k is an integer to ensure $(\theta_g - \theta + k\pi) \in [-\frac{\pi}{4}, \frac{3\pi}{4}]$ (see Fig. 3). In FAM, we set $\theta = 0$ to represent a horizontal anchor. Then the regression targets can be expressed by Eq. (7). In ODM, we first decode the output of FAM and then re-compute the regression targets by Eq. (7).

Matching strategy. We adopt IoU as the metrics, and an anchor box can be assigned to positive (or negative) if its IoU greater than a foreground threshold (or less than a background threshold, respectively). Different from the IoU between horizontal bounding boxes, we calculate the IoU between two oriented bounding boxes. By default, we set foreground threshold as 0.5 and background threshold as 0.4 in both FAM and ODM.

Loss function. The loss of S^{2}A-Net is a multi-task one which consists of two parts, *i.e.*, the loss of FAM and the loss of ODM. For each part, we assign a class label to each anchor/refined anchor and regress its location. The loss function can be defined as:

$$\begin{aligned} \mathcal{L} = &\frac{1}{N_F}\Big(\sum_i \mathcal{L}_c(c_i^F, l_i^*) + \sum_i \mathbf{1}_{[l_i^* \geq 1]}\mathcal{L}_r(\mathbf{x}_i^F, \mathbf{g}_i^*)\Big) \\ &+ \frac{\lambda}{N_O}\Big(\sum_i \mathcal{L}_c(c_i^O, l_i^*) + \sum_i \mathbf{1}_{[l_i^* \geq 1]}\mathcal{L}_r(\mathbf{x}_i^O, \mathbf{g}_i^*)\Big), \end{aligned} \quad (8)$$

where λ is a loss balance parameter, $\mathbf{1}_{[\cdot]}$ is an indicator function, N_F and N_O are the numbers of positive samples in the FAM

and ODM respectively, i is the index of a sample in a mini-batch. c_i^F and $\mathbf{x}_i^F$ are the predicted category and refined locations of the anchor i in FAM. c_i^O and $\mathbf{x}_i^O$ are the predicted object category and locations of the bounding box in ODM. l_i^* and $\mathbf{g}_i^*$ are the ground-truth category and locations of the anchor i. The Focal loss [11] and smooth $L1$ loss are adopted as the classification loss $\mathcal{L}_c$ and the regression loss $\mathcal{L}_r$, respectively.

Inference. S^2A-Net is a fully convolutional network and we can simply forward an image through the network without complex RoI operation. Specifically, we pass the input image to the backbone network to extract pyramid features. Then the pyramid features are fed into FAM to produce refined anchors and aligned features. After that, ODM encodes the orientation information to produce the predictions with high confidence. Finally, we choose top-k (*i.e.*, 2000) predictions and adopt NMS to yield the final detections.

IV. Experiments and Analysis

A. Datasets

DOTA [3]. It is a large aerial image dataset for oriented objects detection which contains 2806 images with the size ranges from 800×800 to 4000×4000 and 188282 instances of 15 common object categories includes: Plane (PL), Baseball diamond (BD), Bridge (BR), Ground track field (GTF), Small vehicle (SV), Large vehicle (LV), Ship (SH), Tennis court (TC), Basketball court (BC), Storage tank (ST), Soccer-ball field (SBF), Roundabout (RA), Harbor (HA), Swimming pool (SP), and Helicopter (HC).

Both training and validation sets are used for training, and the testing set is used for testing. Following [3], we crop a series of 1024×1024 patches from original images with a stride of 824. We only adopt random horizontal flipping during training to avoid over-fitting and no other tricks are utilized if not specified. For fair comparison with other methods, we adopt data augmentation (*i.e.*, random rotation) in the training phase. For multi-scale experiments, we firstly resize original images at three scales (0.5, 1.0 and 1.5) and then crop them into 1024×1024 patches with a stride of 512.

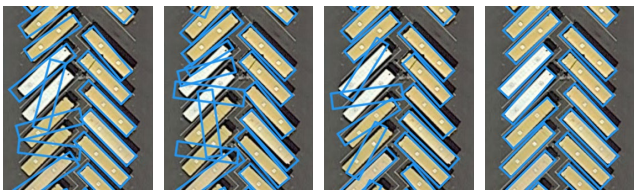
HRSC2016 [14]. It is a high resolution ship recognition dataset annotated with oriented bounding boxes which contains 1061 images, and the image size ranges from 300×300 to 1500×900. We use the training (436 images) and validation (181 images) sets for training and the testing set (444 images) for testing. All images are resized to $(800, 512)$ without changing the aspect ratio. Horizontal flipping is applied during training.

B. Implementation Details

We adopt ResNet101 FPN as the backbone network for fair comparison with other methods, and ResNet50 FPN is adopted for other experiments if not specified. For each level of pyramid features (*i.e.*, P_3 to P_7), we preset one squared anchor per location with a scale of 4 times the total stride size (*i.e.*, 32, 64, 128, 256, 512). The loss balance parameter λ is set to 1. The hyperparameters of Focal loss $\mathcal{L}_c$ are set to $\alpha = 0.25$ and $\gamma = 2.0$. We adopt the same training schedules as Detectron [31]. We train all models in 12 epochs for DOTA and 36 epochs for HRSC2016. SGD optimizer is adopted with an initial learning rate of 0.01 and the learning rate is divided by 10 at each decay step. The momentum and weight decay are 0.9 and 0.0001, respectively. We adopt learning rate warmup for 500 iterations. We use 4 V100 GPUs with a total batch size of 8 for training and a single V100 GPU for inference by default. The time of post-processing (*e.g.*, NMS) is included in all experiments.

TABLE I
Results of different RetinaNet on DOTA. Depth indicates the number of convolution layer in two subnetworks of RetinaNet.

	Model	#Anchor	Depth	mAP	Speed
(a)	RetinaNet	9	4	68.05	62 ms
(b)	RetineNet	9	2	67.64	58 ms
(c)	RetineNet	1	2	67.00	51 ms



Conv DeformConv GA-DeformConv AlignConv

Fig. 6. Qualitative comparison of different convolution methods. The blue bounding box indicates the prediction of large vehicle.

C. Ablation Studies

RetinaNet as baseline. As a single-shot detector, RetinaNet is fast enough. However, any module added to it will introduce more computations. We experiment different architectures and settings on RetinaNet. As shown in Table I (a), RetinaNet achieves a mAP of 68.05% in 62 ms, indicating that our baseline is solid. If the depth of RetinaNet head changes from 4 to 2, the mAP drops 0.41% and the inference time reduces 4 ms. Furthermore, if we set one anchor per location (Table I (c)), the inference time reduces 11% with a accuracy drop of 1.5% compared with Table I (a). The results show that a light detection head and few anchors can also achieve competitive performance and better speed-accuracy trade-off.

Effectiveness of AlignConv. As discussed in Section III-B, we compare AlignConv with other methods to validate its effectiveness. We only replace AlignConv with other convolution methods and keep other settings unchanged. Besides, we also add comparison with Guided Anchoring DeformConv (GA-DeformConv) [25]. Note that the offset field of GA-DeformConv is learned from the anchor prediction map in ARN by a 1×1 convolution.

As shown in Table II, AlignConv surpasses other methods by a big margin. Compared with the standard convolution, AlignConv improves about 3% mAP while only introduces 3ms speed latency. Besides, AlignConv improves the performance for almost all categories, especially for those categories with large aspect ratios (*e.g.*, bridge), densely distribution (*e.g.*, small vehicles and large vehicles) and fewer instances (*e.g.*, helicopters). On the contrary, DeformConv and GA-DeformConv only achieve 71.71% and 71.33% mAP, respectively. The qualitative comparison in Fig. 6 shows that AlignConv achieves

TABLE II
COMPARING ALIGNMENT CONVOLUTION (ALIGNCONV) WITH OTHER CONVOLUTION METHODS. WE COMPARE OUR ALIGNCONV WITH THE STANDARD CONVOLUTION (CONV), DEFORMABLE CONVOLUTION (DEFORMCONV) AND GUIDED ANCHORING DEFORMABLE CONVOLUTION (GA-DEFORMCONV).

Methods	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC	mAP	Speed
Conv	88.87	76.34	46.42	67.53	77.21	74.80	82.27	90.79	81.22	85.02	50.99	61.10	63.54	67.24	53.25	71.11	**59 ms**
DeformConv	88.96	80.23	45.92	67.51	77.10	74.23	84.28	90.81	81.47	85.56	54.19	**64.11**	64.85	68.13	48.34	71.71	60 ms
GA-DeformConv	88.72	79.56	46.19	65.41	76.86	74.96	79.44	90.78	80.99	84.73	55.31	63.17	62.07	67.69	54.12	71.33	60 ms
AlignConv	**89.11**	**82.84**	**48.37**	**71.11**	**78.11**	**78.39**	**87.25**	**90.83**	**84.90**	**85.64**	**60.36**	62.60	**65.26**	**69.13**	**57.94**	**74.12**	62 ms

TABLE III
ABLATION STUDIES. WE CHOOSE A LIGHT RETINANET (SHOWN IN TABLE I (C)) AS THE BASELINE, AND EXPERIMENT DIFFERENT SETTINGS OF S^{2}A-NET, *i.e.*, ANCHOR REFINEMENT NETWORK (ARN), ALIGNMENT CONVOLUTION LAYER (ACL) AND ACTIVE ROTATING FILTERS (ARF).

	Baseline	Different Settings of S^{2}A-Net				
ARN			✓	✓	✓	✓
ACL				✓		✓
ARF					✓	✓
mAP	67.00	68.26	71.17	73.24	71.11	74.12

TABLE IV
EXPERIMENTS OF DIFFERENT NETWORK DESIGNS. WE EXPLORE THE NETWORK DESIGN IN FAM AND ODM WITH DIFFERENT NUMBER OF CONVOLUTIONAL LAYERS. SETTING (C) IS THE DEFAULT SETTING OF OUR PROPOSED METHOD SHOWN IN FIG. 2.

	Model	FAM	ODM	mAP	Speed	Size
(a)	RetinaNet	-	-	68.05	62 ms	279 Mb
(b)	S^{2}A-Net	1	1	73.04	57 ms	257 Mb
(c)	S^{2}A-Net	2	2	74.12	62 ms	273 Mb
(d)	S^{2}A-Net	4	4	73.30	71 ms	304 Mb

accurate bounding box regression in detecting densely packed and arbitrary oriented objects, while other methods with implicit learning get poor performance.

Effectiveness of ARN and ARF. To evaluate the effectiveness of ARN and ARF, we experiment different settings of S^{2}A-Net. If ARN is discarded, then FAM and ODM share the same initial anchors without refinement. If ARF is discarded, we replace the ARF layer with standard convolution layer. As shown in Table III, without ARN, ACL and ARF, our method achieves 68.26% mAP, 1.26% mAP higher than the baseline method. This is mainly because we add supervision in both FAM and ODM. With the participation of ARN, we obtain 71.17% mAP, showing that anchor refinement is important to the final predictions in ODM.

Besides, we find the combination of ARN and ARF, which achieves 71.17% mAP, does nothing for performance improvement. However, if we put ACL and ARF together, there is an obvious improvement, from 73.24% to 74.12%. We argue that CNNs are not rotation-invariant, and even we can extract accurate features to represent the object, the corresponding features are still rotation-sensitive. So the participation of ARF augments the orientation information explicitly, which leads to better regression and classification.

Network design. As shown in Table IV, we explore different network designs in FAM and ODM. Compared with the baseline method in Table IV (a), we can conclude that S^{2}A-Net is not only an effective detector with high accuracy, but also an efficient detector in both speed and model size. The results in Table IV (b), (c) and (d) show that our proposed method is insensitive to the depth of the network and the performance improvements mainly come from our novel alignment mechanism. Besides, as the number of layers increases, there is a performance drop from Table IV (c) to (d). We hypothesize that deeper network can not benefit the small object detection which needs a smaller receptive field.

TABLE V
COMPARISON OF DIFFERENT SETTINGS DETECTING ON LARGE IMAGES IN DOTA. **Stride** IS THE CROPPING STRIDE REFERRED IN SECTION IV-A. **#Image** MEANS THE NUMBER OF IMAGES OR CHIPS. **Output** INDICATES THE MODULE (*i.e.*, FAM OR ODM) USED FOR TESTING. WE SHOW THE INFERENCE TIME REQUIRED FOR ENTIRE DATASET USING *FP32*/*FP16* WITH 4 V100 GPUS.

Input Size	Stride	#Image	Output	mAP	Time (s)
1024 × 1024	1024	8143	ODM	71.20	150 / 126
1024 × 1024	824	10833	ODM	74.12	246 / 160
1024 × 1024	512	20012	ODM	74.62	352 / 308
Original	-	937	ODM	74.01	120 / 103
Original	-	937	FAM	70.85	104 / 97

TABLE VI
COMPARING S^{2}A-NET WITH CLUSDET [32] ON DOTA. FOLLOWING [32], WE REPORT THE ACCURACY OF FIVE CATEGORIES (*i.e.*, PLANES, SMALL VEHICLES, LARGE VEHICLES, SHIPS AND HELICOPTERS) ON THE VALIDATION SET WITH DIFFERENT IOU THRESHOLDS (*i.e.*, $\text{MAP}_{.5}$, $\text{MAP}_{.75}$ AND $\text{MAP}_{.5-.95}$). THE RESULTS ARE CALCULATED FROM THE AXIS-ALIGNED BOUNDING BOXES OF THE OUTPUT OF S^{2}A-NET. #IMAGE MEANS THE NUMBER OF IMAGES OR CHIPS. † INDICATES THAT THE OUTPUT OF FAM IS ADOPTED FOR THE FINAL RESULTS.

Methods	#Image	$\text{mAP}_{.5-.95}$	$\text{mAP}_{.5}$	$\text{mAP}_{.75}$
ClusDet [32]	1055	32.2	47.6	39.2
S^{2}A-Net† (Ours)	**458**	42.7	72.7	45.3
S^{2}A-Net (Ours)	**458**	**43.9**	**75.8**	**46.3**

(a) Detection on chip images (b) Detection on large-size image

Fig. 7. Qualitative comparison of detection results. We crop a large-size image into 1024 × 1024 chip images with a stride of 824. The large-size image and chip images are fed into the same network to produce detection results (*e.g.*, planes in red boxes) without resizing. Instances with the same number are corresponding.

D. Detecting on large-size images

The size of aerial image often ranges from thousands to tens of thousands, which means more computations and memory footprint. Many previous works [3], [4] adopt a detection on chips strategy to alleviate this challenge, even if a chip

does not contain any object. ClusDet [32] tries to address this issue by generating clustered chips, while introducing more complex operations (*e.g.*, chip generation and results merge) and significant performance drop. As our proposed S^2A-Net is efficient and the architecture is flexible, we aims to detect objects on large-size images directly.

We first explore different settings of the input size and cropping stride, and report the mAP and overall time during inference (Table V). We first crop the images into 1024 × 1024 chips, and the mAP improves from 71.20% to 74.62% when the stride decreases from 1024 to 512. However, the number of chip images increases from 8143 to 20012, and the overall inference time increases about 135%. If we detect on the original large-size images without cropping, the inference time has reduced by 50% with negligible loss of accuracy. We argue that the cropping strategy makes it hard to detect objects around the boundary (Fig. 7). Besides, if we adopt the output of FAM for detection and Floating-Point 16 (FP16) to speed up the inference, we can reduce the inference time to 97 seconds with a mAP of 70.85%. Compared our S^2A-Net with ClusDet [32] (Table VI), our method only process 428 images and outperforms ClusDet by a large margin. If we adopt the output of FAM for evaluation, we still achieve 42.7% $mAP_{.5-.95}$ and 72.7% $mAP_{.5}$. The result demonstrates that out method is efficient and effective, and our detection strategy can achieve better speed-accuracy trade-off.

E. Comparisons with the State-of-the-art

In this section, we compare our proposed S^2A-Net with other state-of-the-art methods on two aerial detection datasets DOTA and HRSC2016. The settings have been introduced in Section IV-A and Section IV-B.

Results on DOTA[1]**.** Note all results are reported on ResNet101 backbone if not specified. We re-implement RetinaNet which is referred in Sec III-A. As shown in Table VII, we achieve 74.01% mAP in 22.6 FPS with ResNet-50-FPN backbone and without any data augmentation (*e.g.*, random rotation). Note that the FPS is a relative FPS and we obtain it by calculating the overall inference time and the number of chip images (*i.e.*, 10833). Besides, we achieve state-of-the-art 76.11% mAP with ResNet101 FPN backbone, outperforming all two-stage and one-stage methods. In multi-scale experiments, our S^2A-Net achieves 79.42% and 79.15% mAP with a ResNet-50-FPN and ResNet-101-FPN backbone, respectively. And we achieve the best result in 10 categories, especially those hard categories (*e.g.*, bridge, soccer-ball field, swimming pool, helicopter). Qualitative detection results of the baseline method (*i.e.*, RetinaNet) and our S^2A-Net are visualized in Fig 8. Compared with RetinaNet, our S^2A-Net produces less false predictions when detecting on the object with dense distribution and large scale variations. **Results on HRSC2016.** Note that DRN [35] and CenterMap-Net [34] are evaluated under PASCAL VOC2012 metrics while other methods are evaluated under PASCAL VOC2007 metrics, and the performance under VOC2012 metrics is better than that under VOC2007 metrics. As shown in Table VIII, our proposed S^2A-Net achieves 90.17% and 95.01% mAP under VOC2007 and VOC2012 metrics respectively, outperforming all other methods. The objects in HRSC2016 have large aspect ratios and arbitrary orientations. Previous methods often set more anchors for better performance, *e.g.*, 20 in RoI Trans. and 126 in R^3Det. Compared with the previous best result 89.26% (VOC2007) by R^3Det and 92.8% (VOC2012) by CenterMap-Net, we improve 0.91% and 2.21% mAP respectively with only one anchor, which effectively get ride of heuristically defined anchors. Some qualitative results are shown in Fig. 9.

[1]The result is available at https://captain-whu.github.io/DOTA/results.html with setting name `hanjiaming`. Note that, to concentrate on studying the algorithmic problem of ODAI, this setting is without using model fusions which can further improve the detection performance.

V. Conclusion

In this paper, we propose a simple and effective Single-Shot Alignment Network (S^2A-Net) for oriented object detection in aerial images. With the proposed Feature Alignment Module and Oriented Detection Module, our S^2A-Net realizes full feature alignment and alleviates the inconsistency between regression ans classification. Besides, we explore the approach to detect on large-size images for better speed-accuracy trade-off. Extensive experiments demonstrate that our S^2A-Net can achieve state-of-the-art performance on both DOTA and HRSC2016.

References

[1] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, 2016.

[2] Z. Liu, H. Wang, L. Weng, and Y. Yang, "Ship Rotated Bounding Box Space for Ship Extraction From High-Resolution Optical Satellite Images With Complex Backgrounds," *IEEE Geoscience and Remote Sensing Letters*, Aug. 2016.

[3] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "DOTA: A large-scale dataset for object detection in aerial images," in *CVPR*, 2018, pp. 3974–3983.

[4] J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu, "Learning roi transformer for oriented object detection in aerial images," in *CVPR*, 2019, pp. 2849–2858.

[5] G. Zhang, S. Lu, and W. Zhang, "Cad-net: A context-aware detection network for objects in remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, pp. 1–10, 2019.

[6] X. Yang, J. Yang, J. Yan, Y. Zhang, T. Zhang, Z. Guo, X. Sun, and K. Fu, "Scrdet: Towards more robust detection for small, cluttered and rotated objects," in *ICCV*, 2018, pp. 8231–8240.

[7] Y. Xu, M. Fu, Q. Wang, Y. Wang, K. Chen, G. Xia, and X. Bai, "Gliding vertex on the horizontal bounding box for multi-oriented object detection," *IEEE Trans. on PAMI*, 2020.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014, pp. 580–587.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. on PAMI*, no. 6, pp. 1137–1149, 2017.

[10] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017, pp. 2980–2988.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2980–2988.

[12] K. Chen, W. Ouyang, C. C. Loy, D. Lin, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, and J. Shi, "Hybrid task cascade for instance segmentation," 06 2019, pp. 4969–4978.

[13] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Oriented response networks," in *CVPR*, 2017, pp. 4961–4970.

[14] Z. Liu, L. Yuan, L. Weng, and Y. Yang, "A high resolution optical satellite image dataset for ship recognition and some new baselines," in *ICPRAM*, 2017, pp. 324–331.

TABLE VII

COMPARISONS WITH STATE-OF-THE-ART METHODS ON DOTA. R-101-FPN STANDS FOR RESNET 101 WITH FPN (LIKEWISE R-50-FPN), AND H-104 STANDS FOR HOURGLASS 104. † INDICATES TRAINING AND TESTING WITHOUT DATA AUGMENTATION. ‡ DENOTES THE INPUT IS THE ORIGINAL IMAGES OTHER THAN CHIP IMAGES. * MEANS MULTI-SCALE TRAINING AND TESTING.

Methods	Backbone	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC	mAP	FPS
two-stage:																		
FR-O [3]	R-101	79.42	77.13	17.70	64.05	35.30	38.02	37.16	89.41	69.64	59.28	50.30	52.91	47.89	47.40	46.30	54.13	-
Azimi *et al.* [33]	R-101-FPN	81.36	74.30	47.70	70.32	64.89	67.82	69.98	90.76	79.06	78.20	53.64	62.90	67.02	64.17	50.23	68.16	-
RoI Trans.* [4]	R-101-FPN	88.64	78.52	43.44	75.92	68.81	73.68	83.59	90.74	77.27	81.46	58.39	53.54	62.83	58.93	47.67	69.56	5.9
CADNet [5]	R-101-FPN	87.80	82.40	49.40	73.50	71.10	63.50	76.60	**90.90**	79.20	73.30	48.40	60.90	62.00	67.00	62.20	69.90	-
SCRDet [6]	R-101-FPN	**89.98**	80.65	52.09	68.36	68.36	60.32	72.41	90.85	**87.94**	86.86	65.02	66.68	66.25	68.24	65.21	72.61	-
Xu *et al.* [7]	R-101-FPN	89.64	**85.00**	52.26	77.34	73.01	73.14	86.82	90.74	79.02	86.81	59.55	**70.91**	72.94	70.86	57.32	75.02	10.0
CenterMap-Net [34]	R-50-FPN	88.88	81.24	53.15	60.65	78.62	66.55	78.10	88.83	77.80	83.61	49.36	66.19	72.10	72.36	58.70	71.74	-
CenterMap-Net* [34]	R-101-FPN	89.83	84.41	54.60	70.25	77.66	78.32	87.19	90.66	84.89	85.27	56.46	69.23	74.13	71.56	66.06	76.03	-
one-stage:																		
RetinaNet [11]	R-101-FPN	88.82	81.74	44.44	65.72	67.11	55.82	72.77	90.55	82.83	76.30	54.19	63.64	63.71	69.73	53.37	68.72	12.7
DRN [35]	H-104	88.91	80.22	43.52	63.35	73.48	70.69	84.94	90.14	83.85	84.11	50.12	58.41	67.62	68.60	52.50	70.70	-
DRN* [35]	H-104	89.71	82.34	47.22	64.10	76.22	74.43	85.84	90.57	86.18	84.89	57.65	61.93	69.30	69.63	58.48	73.23	-
R^3Det [23]	R-101-FPN	89.54	81.99	48.46	62.52	70.48	74.29	77.54	90.80	81.39	83.54	61.97	59.82	65.44	67.46	60.05	71.69	-
R^3Det [23]	R-152-FPN	89.49	81.17	50.53	66.10	70.92	78.66	78.21	90.81	85.26	84.23	61.81	63.77	68.16	69.83	67.17	73.74	-
S^2A-Net† (Ours)	R-50-FPN	89.11	82.84	48.37	71.11	78.11	78.39	87.25	90.83	84.90	85.64	60.36	62.60	65.26	69.13	57.94	74.12	16.0
S^2A-Net†‡ (Ours)	R-50-FPN	89.11	81.51	48.75	72.85	78.23	76.77	86.95	90.84	83.59	85.52	62.70	61.63	66.55	68.94	56.24	74.01	**22.6**
S^2A-Net (Ours)	R-101-FPN	88.70	81.41	54.28	69.75	78.04	80.54	88.04	90.69	84.75	86.22	65.03	65.81	76.16	73.37	58.86	76.11	12.7
S^2A-Net* (Ours)	R-50-FPN	88.89	83.60	**57.74**	**81.95**	79.94	**83.19**	89.11	90.78	84.87	**87.81**	70.30	68.25	78.30	77.01	**69.58**	**79.42**	16.0
S^2A-Net* (Ours)	R-101-FPN	89.28	84.11	56.95	79.21	**80.18**	82.93	**89.21**	90.86	84.66	87.61	**71.66**	68.23	**78.58**	**78.20**	65.55	79.15	12.7

TABLE VIII

COMPARISONS OF STATE-OF-THE-ART METHODS ON HRSC2016. #ANCHOR MEANS THE NUMBER OF ANCHORS AT EACH LOCATION OF THE FEATURE MAP. * INDICATES THAT THE RESULT IS EVALUATED UNDER PASCAL VOC2012 METRICS.

Methods	RC2 [36]	R^2PN* [37]	RRD [29]	RoI Trans. [4]	Xu *et al.* [7]	R^3Det [23]	DRN [35]	CenterMap-Net [34]	S^2A-Net (Ours)
#Anchor	-	24	13	20	20	126	-	15	**1**
mAP	75.7	79.6	84.3	86.2	88.2	89.26	92.7*	92.8*	**90.17 / 95.01***

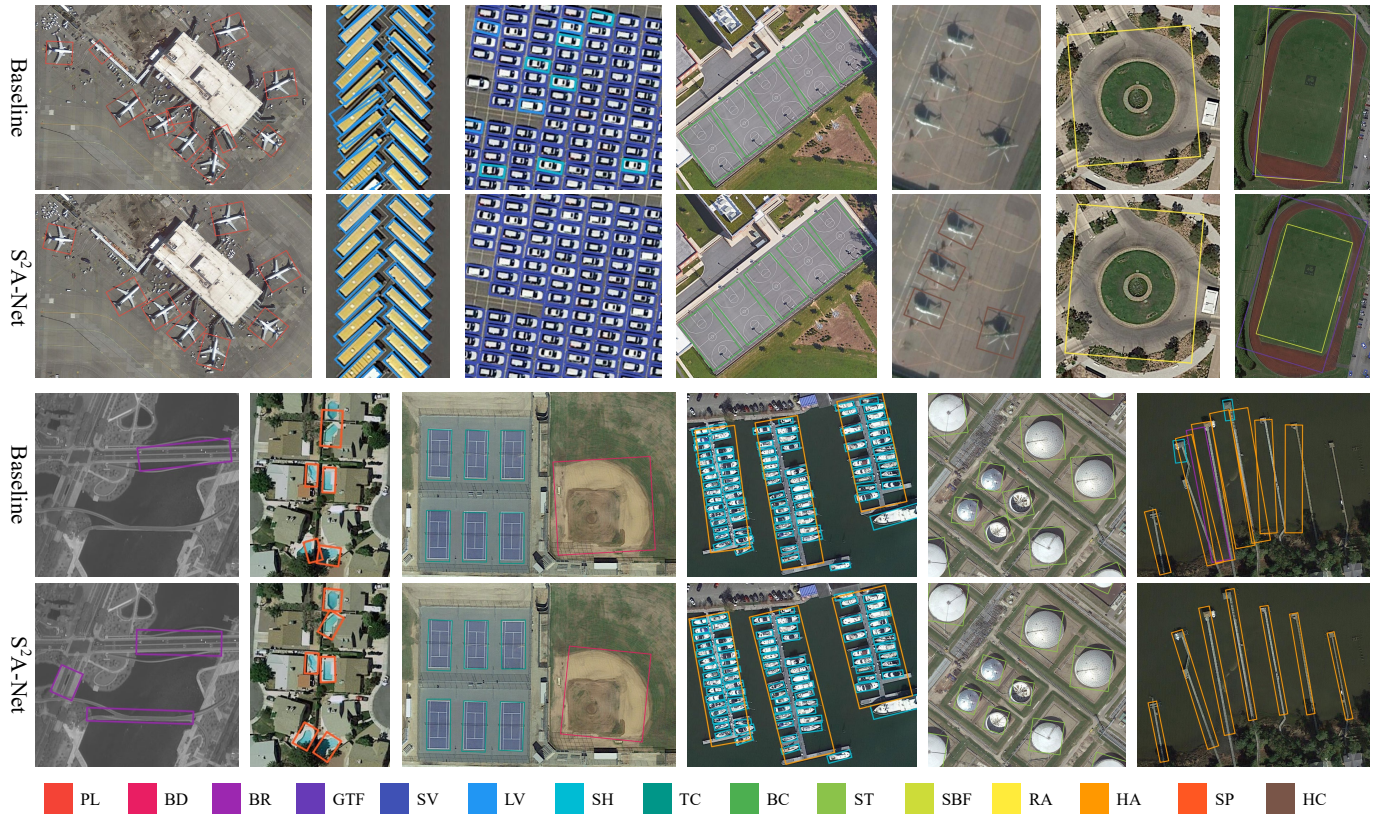


Fig. 8. Comparison of the detection results in DOTA with different methods. For each image pair, the upper image is the baseline method while the bottom is our proposed S^2A-Net.

[15] R. Girshick, "Fast R-CNN," in *ICCV*, 2015, pp. 1440–1448.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016, pp. 21–37.

[18] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *ECCV*, September 2018.

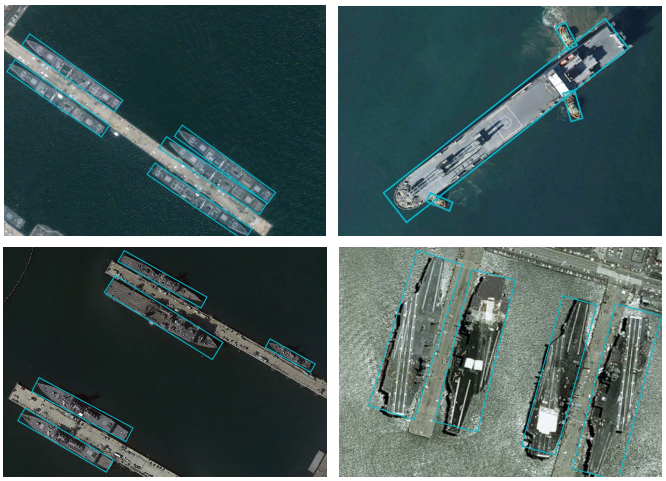


Fig. 9. Some detection results in HRSC2016 with our proposed S^2A-Net.

[19] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019.

[20] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "Reppoints: Point set representation for object detection," in *ICCV*, 2019, pp. 9656–9665.

[21] Z. Liu, J. Hu, L. Weng, and Y. Yang, "Rotated region based cnn for ship detection," 2017, pp. 900–904.

[22] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Trans. on Multimedia*, 2018.

[23] X. Yang, Q. Liu, J. Yan, A. Li, Z. Zhang, and G. Yu, "R3det: Refined single-stage detector with feature refinement for rotating object," *arXiv preprint arXiv:1908.05612*, 2019.

[24] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *ICCV*, 2017, pp. 764–773.

[25] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin, "Region proposal by guided anchoring," in *CVPR*, 2019, pp. 2960–2969.

[26] Y. Chen, C. Han, N. Wang, and Z. Zhang, "Revisiting feature alignment for one-stage object detection," *arXiv preprint arXiv:1908.01570*, 2019.

[27] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *ECCV*, September 2018.

[28] Y. Wu, Y. Chen, L. Yuan, Z. Liu, L. Wang, H. Li, and Y. Fu, "Rethinking classification and localization for object detection," *arXiv preprint arXiv:1904.06493*, 2019.

[29] M. Liao, Z. Zhu, B. Shi, G. Xia, and X. Bai, "Rotation-sensitive regression for oriented scene text detection," in *CVPR*, 2018, pp. 5909–5918.

[30] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017, pp. 2117–2125.

[31] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[32] F. Yang, H. Fan, P. Chu, E. Blasch, and H. Ling, "Clustered object detection in aerial images," in *ICCV*, vol. 2019-Octob, apr 2019, pp. 8310–8319.

[33] S. M. Azimi, E. Vig, R. Bahmanyar, M. Körner, and P. Reinartz, "Towards multi-class object detection in unconstrained remote sensing imagery," in *ACCV*, 2018, pp. 150–165.

[34] J. Wang, W. Yang, H.-C. Li, H. Zhang, and G.-S. Xia, "Learning center probability map for detecting objects in aerial images," *IEEE Transactions on Geoscience and Remote Sensing*, in press.

[35] X. Pan, Y. Ren, K. Sheng, W. Dong, H. Yuan, X. Guo, C. Ma, and C. Xu, "Dynamic refinement network for oriented and densely packed object detection," in *CVPR*, June 2020.

[36] L. Liu, Z. Pan, and B. Lei, "Learning a rotation invariant detector with rotatable bounding box," *arXiv preprint:1711.09405*, 2017.

[37] Z. Zhang, W. Guo, S. Zhu, and W. Yu, "Toward arbitrary-oriented ship detection with rotated region proposal and discrimination networks," *IEEE Geoscience and Remote Sensing Letters*, no. 99, pp. 1–5, 2018.

[38] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.

[39] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," 2014, pp. 740–755.

[40] J. Ding, Z. Zhu, G. Xia, X. Bai, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Icpr2018 contest on object detection in aerial images (odai-18)," in *ICPR*, 2018.

[41] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *NIPS*, 2016, pp. 4905–4913.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[43] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS - Improving Object Detection with One Line of Code," in *ICCV*, vol. 2017-Octob, 2017, pp. 5562–5570.

[44] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *CVPR*, 2018, pp. 6154–6162.

[45] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *CVPR*, 2018, pp. 4203–4212.

[46] T. X. Vu, H. Jang, T. X. Pham, and C. D. Yoo, "Cascade rpn: Delving into high-quality region proposal network with adaptive convolution," in *NIPS*, 2019.

[47] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *ICCV*, 2019, pp. 9626–9635.