RAWNet：A Learnable Multiflow Refinement Network with Rotated Attention Wise Object Detectors

Appendix:

Performance of Single object detection methods in deep learning is limited by high-resolution, complex backgrounds and uneven size and quantity distribution of training samples. current models need oversize occupation of memory and computation cost for quantity time. In this paper, an accuracy-speed balanced real-time object detector based on the rotated attention wise network is proposed, which utilizes multipath refinement flow network and pass wise connection to extract multiscale features and employs learnable attention wise modules with rotated bounding boxes to search and locate right semantic and location features simultaneously. We not only explore various architecture efficiency but also fine-tune the training process by adopting various backbones and data augmentation strategies to increase the diversity of training data and overcome the size of restrictions of input images. Extensive experiments and comprehensive evaluations on daily-life COCO and large-scale DOTA datasets demonstrate the effectiveness of the proposed framework, which achieves state-of-the-art performance in real-time object detection methods.

1. Implement

1.1. The Whole Network Architecture

The whole network architecture is in Fig. 4. We propose an intensively reconstructed network called RAWNet. It employs the multipath refinement flow network as the backbone, which makes it easier to extract multilevel scale features from patches. In addition, we not only consider the efficiency between the backbone and neck through designing the MRFNet, but also enhance the fusion effect of extract features aided by a passing wise connection (PWC) strategy. At last, this framework is learnable for multiclass and multiscale object, due to we design various attention wise modules to make the training and validation more reasonable and extract features in a global perspective. Also, we modify the above model with rotated bounding boxes and combine different kinds of loss functions into one comprehensive loss function.

1.2. Multipath Refinement Flow Network Architecture

Multipath refinement flow network architecture is shown in Fig.2. MRFNet combines each convolution layer and data flow branches. Specifically, there are two path channels $a_0 = [a_0', a_0'']$, one is linked to the end of the stage directly, another will go through the whole layers. Every stage has a downsample and fusion layer, $[a_0'', a_1, \cdots, a_k]$

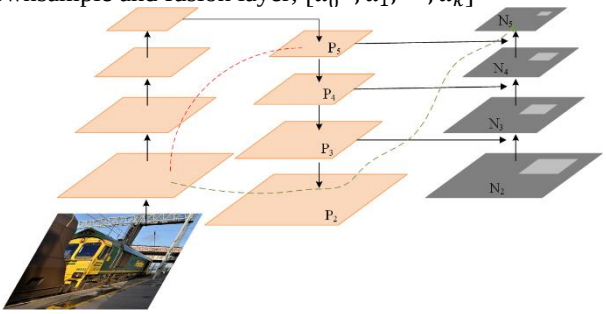


Fig. 2 *Pass wise connection benefit the feature extraction and fusion.*

will be downsampled to lower dimensions and larger output numbers respectively. Then the output of this refinement transfer results a_τ, is concatenated with a_0' and undergo another transition layer. Finally, the output a_U is generated. The feed-forward pass and weight updating of MRFNet are shown in Equations 1 and 2, respectively.

$$a_k = w_k * [a_0', a, \cdots, a_{k-1}]$$

$$\begin{aligned} a_T &= w_T * [a_0'', a_1, \cdots, a_k] \\ a_U &= w_U * [a_0', a_T] \end{aligned} \quad (1)$$

$$\begin{aligned} w_K' &= f_k(w_k, \{g_0'', g_1, \cdots, g_{k-1}\}) \\ w_T' &= f_T(w_T, \{g_0'', g_1, \cdots, g_k\}) \\ w_U' &= f_U(w_U, \{g_0', g_T\}) \end{aligned} \quad (2)$$

We compare our method with the mainstream CNN architectures such as ResNet, ResNext, DenseNet, the outputs of which are usually a linear or non-linear combination of the outputs of intermediate layers. Therefore, the output of a k-layer CNN can be expressed as follows:

$$\begin{aligned} y &= F(a_0) = a_k \\ &= H_k(a_{k-1}, H_{k-1}(a_{k-2}), H_{k-2}(a_{k-3}), \cdots, H_1(a_0), a_0) \end{aligned} \quad (3)$$

The model of the convolutional neural network can be abstract into F, which is the mapping function from x0 to y. As for H_k, it is the k-th layer of CNN. Generally, a set of convolutional layers and a nonlinear activate function consistutes the Hk. As for ResNet and DenseNet, we can also show their models by the following equations (4) and (5):

$$\begin{aligned} a_k &= R_k(a_{k-1}) + a_{k-1} \\ &= R_k(a_{k-1}) + R_{k-1}(a_{k-2}) + \cdots + R_1(a_0) + a_0 \end{aligned} \quad (4)$$

$$\begin{aligned} x_k &= [C_k(a_{k-1}), a_{k-1}] \\ &= [C_k(a_{k-1}), C_{k-1}(a_{k-2}), \cdots, C_1(a_0), a_0] \end{aligned} \quad (5)$$

Where the R and C represent the operational computation of the residual layers and convolutional layers, generally they consists of two or three convolutional layers.

From the equations above, all the inputs of each convolutional layers are originally generated from the previous convolutional outputs. Under this circumstance, the gradient flow could be propagated more efficiently due to the reduction of the gradient path length. However, their methods would result in the propagation into all layers from k-1 to 1 layers, which is redundant for repeated training process.

Therefore, we build the multipath flow network, from equation (1) and (2) and Figure 4, the architecture and feature extraction illustrates the RAWNet reuses the initial features, and in the meantime prevents iteratively propagating gradient information by cutting down the gradient flow. The insightful vision of the design of the network is to separate the gradient flow and refine the feature, and finally make the fusion in the last convolutional layers which could enhance the feature extraction efficiency.

The specific multipath refinement flow network backbone is illustrated in Table 1. It reference the advantage of multiscale feature extraction as RefineNet and DarkNet53, and intensively modified by introducing light weight and residual

strategy, Pass wise connection and attention modules to enhance the efficiency and accuracy.

TABLE I: Specific parameters of the MRFNet backbone.

Network	Kernel	Stride	Padding	Network	Kernel	Stride	Padding
Conv1	5×5×32	1	2	conv_r2_m2	1×1×128	1	1
Conv2	1×1×64	1	1	conv_r2_m3	3×3×128	1	1
Conv3	5×5×128	2	2	conv_r2_m4	1×1×128	1	1
Conv4	1×1×128	1	1	deconv2	4×4×64	1	2
Conv5	3×3×256	1	2	conv_r3_1	3×3×64	1	1
Conv6	1×1×256	1	1	conv_r3_m1	3×3×64	1	1
Conv7	3×3×256	1	2	conv_r3_m2	1×1×64	1	1
Conv8	1×1×256	1	1	conv_r3_m3	3×3×64	1	1
conv_r1_1	3×3×256	1	1	conv_r3_m4	1×1×64	1	1
conv_r1_m1	3×3×256	1	1	deconv3	4×4×32	1	2
conv_r1_m2	1×1×256	1	1	conv_r4_1	3×3×32	1	1
conv_r1_m3	3×3×256	1	1	conv_r4_m1	1×1×32	1	1
conv_r1_m4	3×1×256	1	1	conv_r4_m2	3×3×32	1	1
deconv1	4×4×128	1	2	conv_r4_m3	1×1×32	1	1
conv_r2_1	3×3×128	1	1	conv_r4_m4	3×3×32	1	1

1.3. Pass Wise Connection

Compared to the above MRFNet

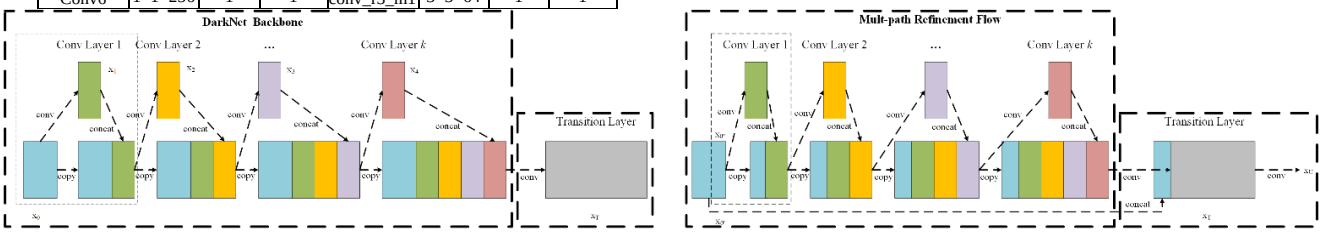


Fig. 3 *The difference of data flow between FPN and MRF*

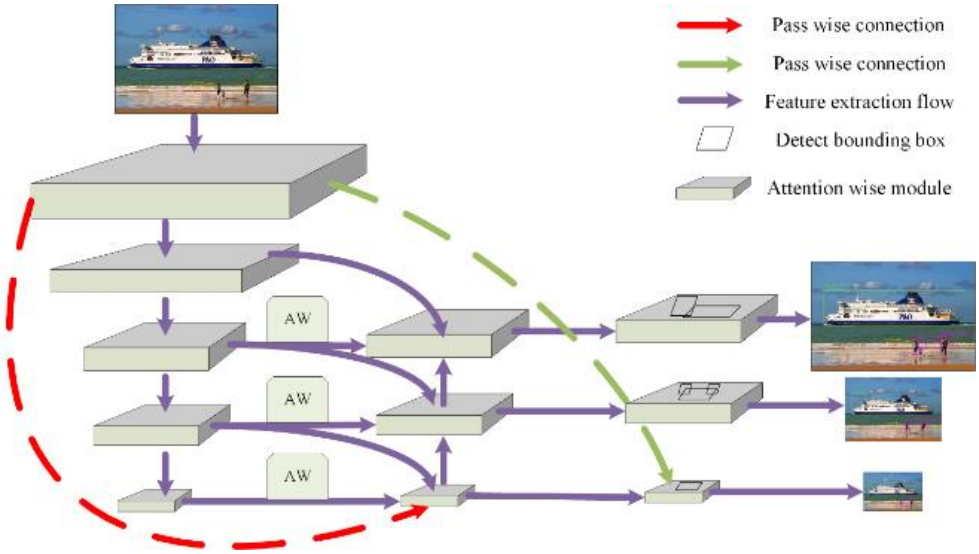


Fig. 4 *The overall network architecture of RAWNet. Our method shows excellent performance in balancing average precision accuracy and frames per second speed.*

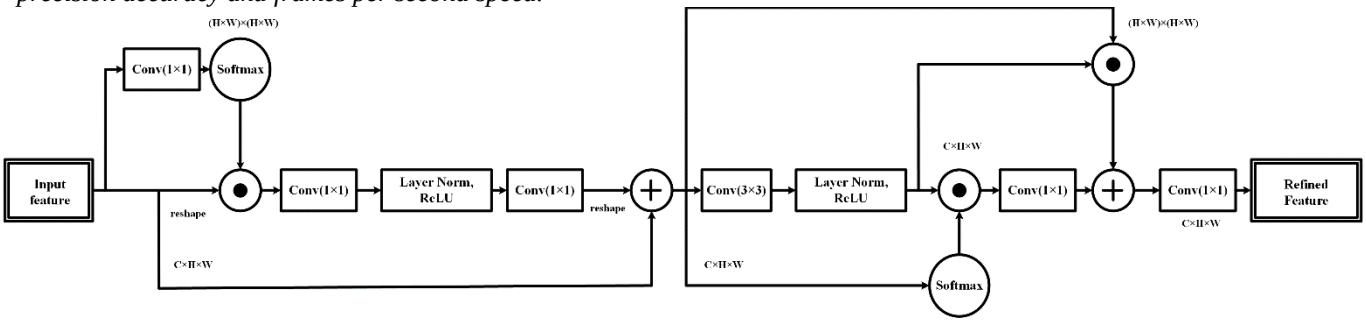


Fig. 5 *A graphical representation of the refinement concern layer. Firstly, the matching value of foreground block and background block is calculated by convolution (as a convolution filter). Then, we applied softmax to compare and get the attention value for each pixel. Finally, foreground patches and background patches are reconstructed by deconvolution of attention value. The context refinement layer is distinguishable and fully convoluted.*

, this section we just make a little trick to pass features to the next extraction stage. The main purpose of MRF, PWC and ResNet aims to learn robust features and train deeper networks, which can address vanishing gradient problems.

There is no doubt that convolutional unit in high level strongly correspond to entire objects while others are more likely to be activated by local texture and patterns manifests the necessity of augmenting a top-down path to propagate semantically strong features and enhance all features with reasonable classification capability in multiscale networks. The connection wise path improves the capabilities of locating positions and propagating strong responses of low-level patterns.

The data augmentation layer and the neck layer extract features in a parallel way, sacrificing the occupation of memory to enhance the accuracy and benefit feature extraction.

high response to edges or instance parts is a strong indicator to accurately localize instances. To this end, regardless of the complicated multipath refinement data flow,

we additionally add two direct connections to pass feature maps. As Fig.2 is depicted, one link in red directly pass the first layer patches of backbone to the last layer of the neck. Another link in green directly pass the first convolutional results to the last layer of data augmentation.

1.4. Attention Wise Module

It is well known that attention plays an important role in image and video perception. On one hand, one-stage-detector aims to handle images in a light weight manner, making instance recognition fast and easy. On the other hand, one important property of visual CNN system is that one does not attempt to process a whole scene at once. we solve the contradictory problem by adding attention module between backbone and neck, the whole network MRF belongs to one-stage method, while also exploits the attention wise modules (AW) to pre-process the feature patches, and aware the contextual and position information several times in a multiscale manner. The design of contextual attention wise unit and pixel refinement wise unit is depicted in Fig.5.

Specifically, the connection between p_i' and p_i can be described as follows:

$$C_{max_i} = \frac{\langle p_i, p_i' \rangle}{\|p_i\| \cdot \|p_i'\|} \tag{6}$$

As adjacent foregrounds are filled, and intuitively the most similar patch is next to element, so we also need to calculate coherent value of adjacent coherent value as follows:

$$C_{ad_i} = \frac{\langle p_i, p_{i-1} \rangle}{\|p_i\| \cdot \| p_{i-1}\|} \tag{7}$$

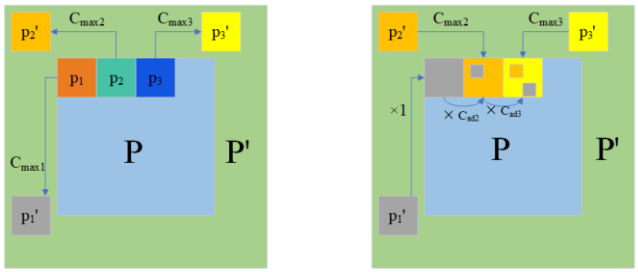


Fig. 6 *We define p' as background and p as foreground. Firstly, each neural patch in the blank area P searches for the most coherent patch on the boundary p'. Then, previous generated patch and most similar contextual patch are combined to generate current one. This process continues four rounds from small scale to large scale, which is called multipath contextual attention algorithm.*

As the Fig.6 shows, patch search for background information with equation (7) and search for adjacent information with equation (8), multipath makes iterative results as followings:

$$p_1 = p_1', C_{ad_i} = 0$$
$$p_i = \frac{C_{ad_i}}{C_{ad_i}+C_{max_i}} \times p_{i-1} + \frac{C_{max_i}}{C_{ad_i}+C_{max_i}} \times p_i' \quad i \in (2,n) \tag{8}$$

Local features generated by traditional convolutional layers would result in misrecognition of specific stuff. It could also lead to a high computation cost to search for specific objects from background. In order to model rich contextual relationships over local features, we continue to analyse the refinement features from the context attention wise unit, and pass the patches to the pixel refinement wise unit to figure out the coherency transformation to the latent position. The pixel refinement wise unit encodes a wider range of contextual information into local features, thus enhancing their representation capability.

What class the target is associated with the background scenarios could be figured out by the contextual attention module. In the meantime, where the latent class of object pixels are likely to attend is located by pixel refinement wise module.

Specifically, the attention process can be seen in Fig.5. First, the multipath refinement patches flow into the context attention wise module, which aims to calculate the coherency between the bounding box and the background. This unit simplify the $C \times H \times W$ patches into the softmax function and then combines the copies with matrix multiplication. We apply a softmax layer to obtain the context attention map m_{ij}:

$$m_{ij} = \frac{exp\,(P_i \cdot P_j)}{\sum_{i=1}^{C} exp\,(P_i \cdot P_j)} \tag{9}$$

The original patches P_i and reshaped branches P_j aggregation into one middle results. In addition, we multiply the result by a scale parameter α and perform a sum operation with M_j to obtain the output result R_{ij}:

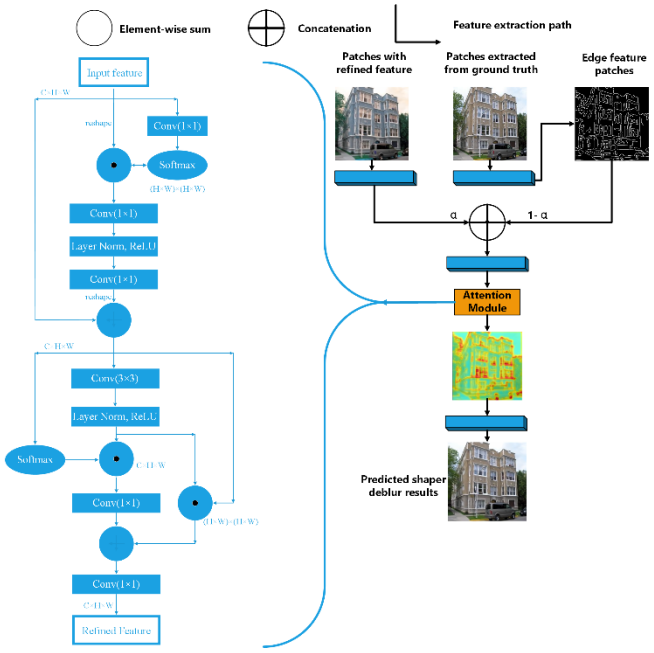
$$R_j = \alpha \sum_{i=1}^{C} (x_{ij} P_i) + M_j \tag{10}$$

Then the output result R is separated into W_i, X_j, Y_i, the W_i and X_j are reshaped and send to the softmax function, then combines with the multiplication as R_{ij}:

$$R_{ij} = \frac{exp\,(W_i \cdot X_j)}{\sum_{i=1}^{N} exp\,(W_i \cdot X_j)} \tag{11}$$

Meanwhile, the Y_i also combines with the reshaped S_{ij} using sum function. we multiply it by a scale parameter α and perform an element-wise sum operation with the features Y_i to obtain the final output $S_j \epsilon R_{ij}^{C \times H \times W}$ as follows:

$$S_j = \alpha \sum_{i=1}^{N} (r_{ij}\, Y_i) + A_j \tag{12}$$



Due to the recent studies of the single object detectors, there are three branches to obtain the things which we are concerned for. First, we adopt the softmax function to weight the importance of the latent meaningful objects which are obtained from the background, then use the location and class coherency to get the attention scores. This is not a promising method due to the weak supervised methods could not achieves the higher average recognition precision. Second,

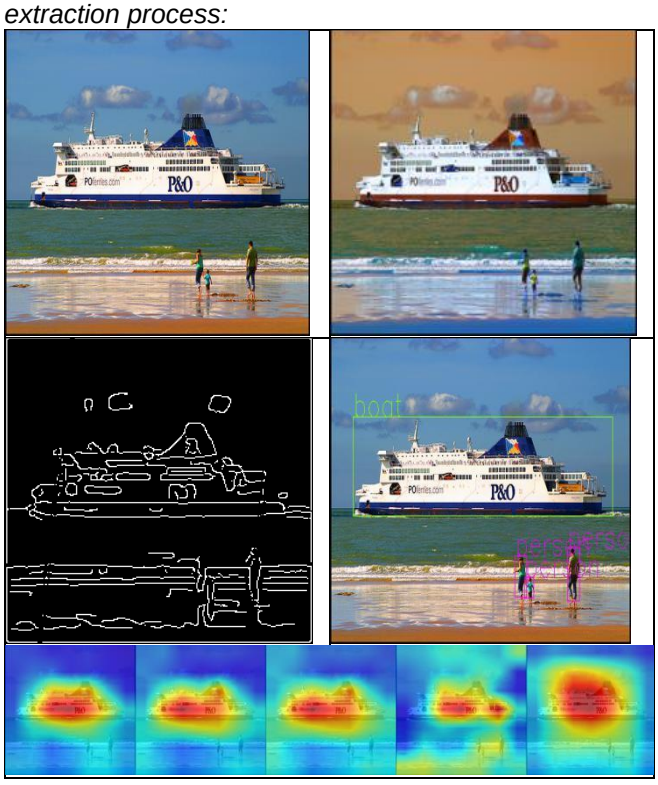
combine the object detection and instance segmentation modals into consideration, but the abundant feature extraction and training computation cost is too high. Thirdly, we combine the two branches and make a trade-off strategy: we adopt the light weight spatial and class feature extraction channels to recognize the latent object classes, and then the attention features are refined by the edge information to reinforce the boundary features for further inference. In this way, we use the edge information instead of instance segmentation to avoid iterative training process and computation cost. Thus, the attention wise modules also attain the location and coherency information in a light weight and refinement way.

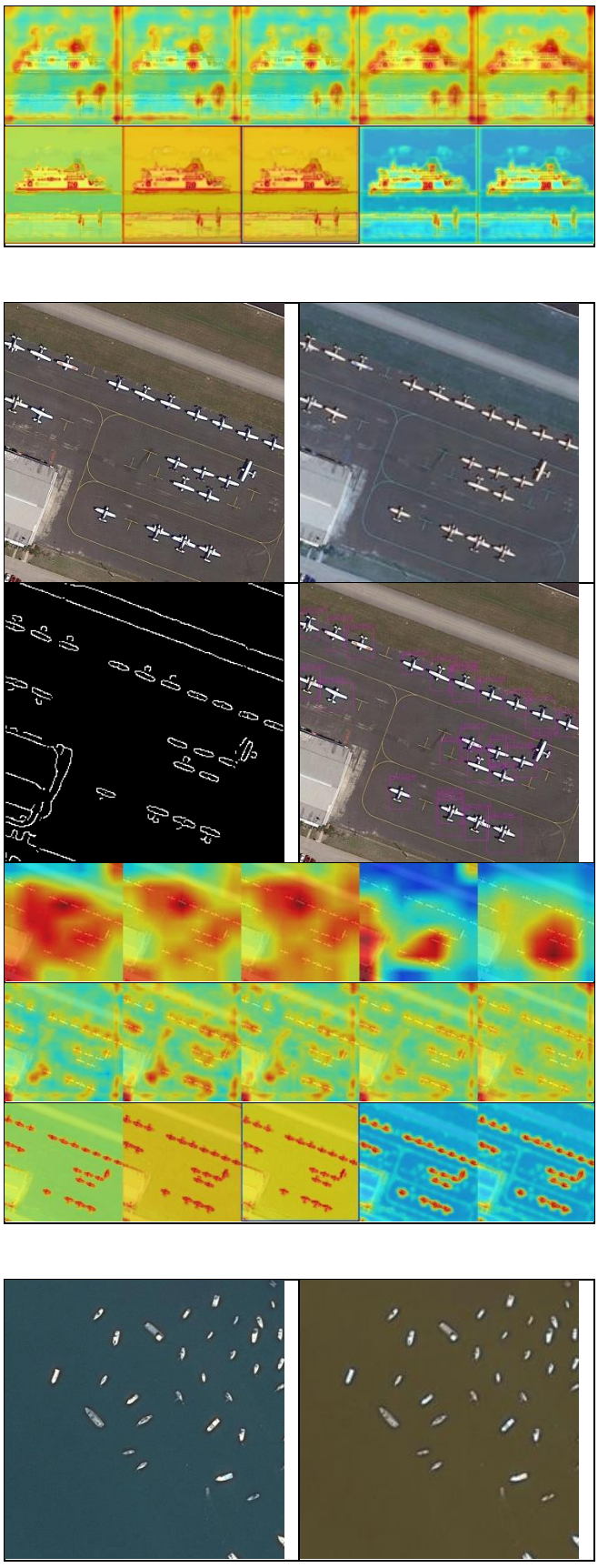
We can consider training process in the feature extraction view instead of the network dataflow. Patches with convolutional refined features are combined with the edge feature patches which are extracted from the ground truth counterparts. The parameter α determines the proportion of the edge information and the back ground recognition feature extraction.

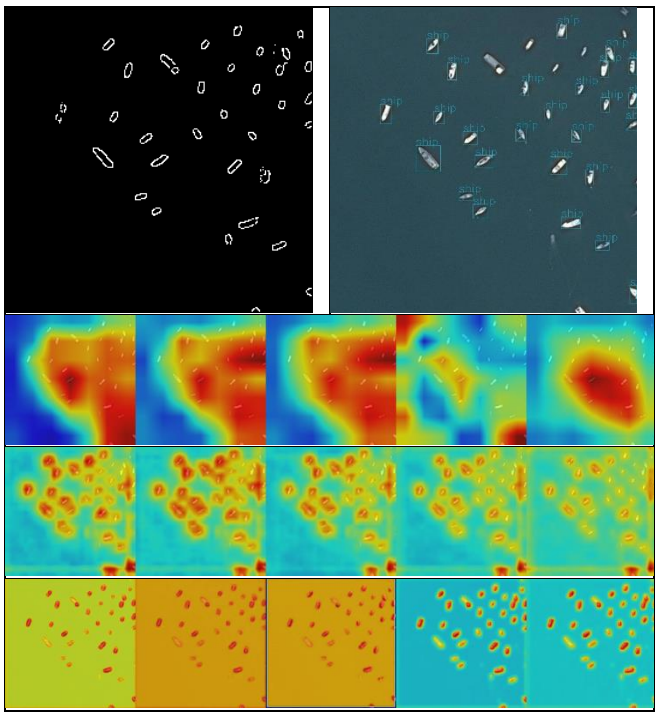
Then the fusion middle results are sent to the attention wise modules and finally make the inferences for object detection average precision.

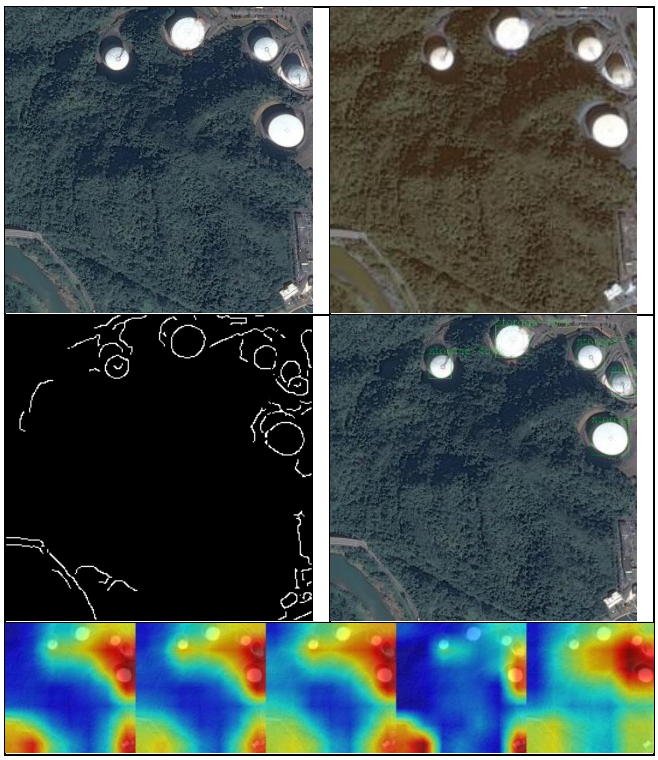
The Figure shows the input images and the edge feature maps and the attention heatmaps as middle outputs which could benefit the RAWNet for efficient recognition. When we change the parameter a and the perceptive field radius for different object scales, heatmaps visualization shows that the dense small objects. Due to the smart design of network feature extraction and efficient searching latent attention strategy, this rotated attention wise network RAWNet achieves fast real-time recognition speed and significant precision enhancement among the state-of-the-art methods.

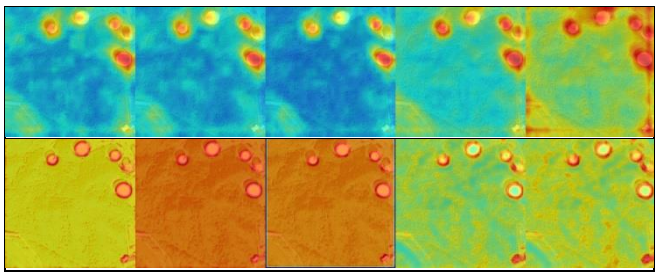
The followings are attention and edge feature extraction process:











1.5. Rotated bounding box

Recent state-of-the- art object recognition methods usually adopt rectangular shaped, horizontal/vertical bounding boxes drawn over an object to accurately localization. To overcome these limitations, this research presents object detection improvements that aid tighter and more precise detections.

The tight rotated bounding training loss process and vision effect are shown in Fig.7, Fig.8 and Fig.9.

We use the five parameters x,y,w,h, θ.(x,y) to represent the rotated bounding boxes. θ.(x,y) is the center of the latent object, and w,h is the width and height of the object. The last parameter is the angle of rotation which is original from the polar coordinate as follows:

$$t_x = (x - x_a)/\omega_a \, , t_y = (y - y_a)/h_a$$
$$t_\omega = \log(\omega/\omega_a) \, , t_h = \log(h/h_a) \, , t_\theta = \theta - \theta_a \quad (13)$$
$$t'_x = (x' - x_a)/\omega_a \, , t'_y = (y' - y_a)/h_a$$
$$t'_\omega = \log(\omega'/\omega_a) \, , t' = \log(h'/h_a) \, , t'_\theta = \theta' - \theta_a \quad (14)$$

X is the ground truth boxes, xa is the anchor boxes, and x' is the prediction of bounding boxes. So the loss function is denoted in Equation 15:

$$L = \frac{\lambda_1}{N} L_{attention} + \frac{\lambda_2}{N} L_2 + \frac{\lambda_3}{N} L_{AW} + \frac{\lambda_4}{N} L_{CLS} + \frac{\lambda_5}{N} L_{Obj} + \frac{\lambda_6}{N} L_{X,Y,W,H} \quad (15)$$

Where N denotes the number of anchors, the hyperparameters λ_k control the trade-off and are set to 1 by default. The classification loss L_{cls} is implemented using focal loss and smooth L_2 loss. We also add the xy loss and wh loss for the bounding box position precision, and add the object loss to analyse how many objects are missing.

The specific training process loss is shown in Fig.8. Tighter and specific recognition bounding boxes benefits the training process, with the comparison results displayed in Fig. 9.

2. Experiments

We compare RAWNet with LRF, RFBNet, YOLOv3, CenterMask and EfficientDet on FPS, AP and visual effects.

2.1. Experimental Setup

Fig. 7 *Different training loss function strategies are adopted in the training process which is beneficial for convergence speed.*

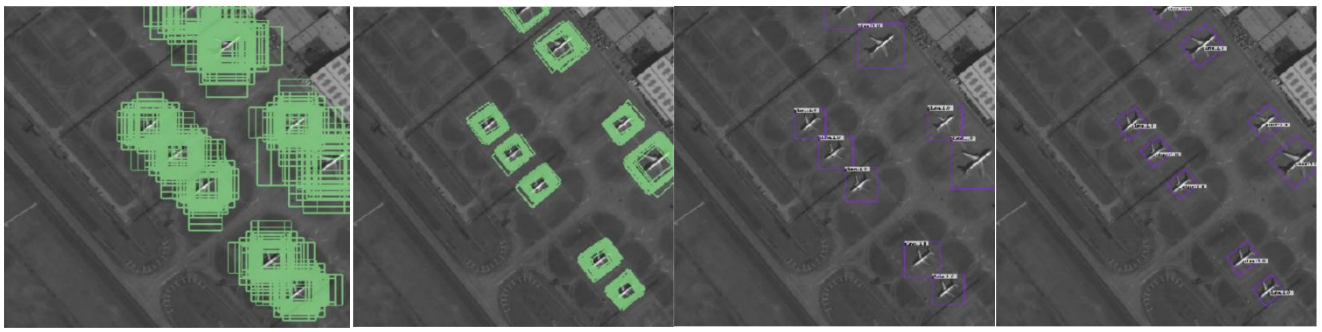


Fig. 8 *The visualization of comparing general and rotated traning process.*

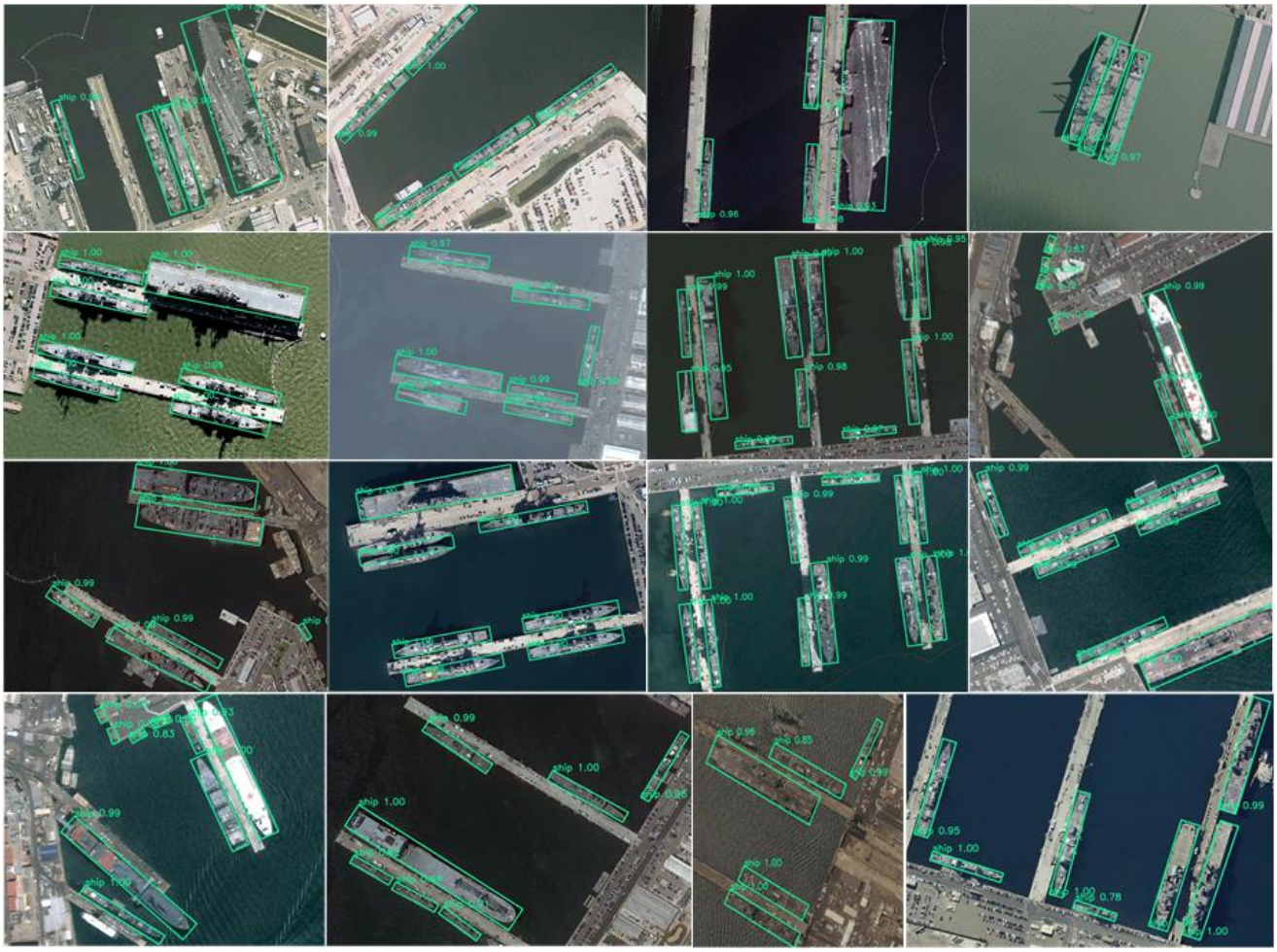


Fig. 9 *The visual results of rotated bounding boxes..*

We implement our model by Pytorch. The model is trained with Adam ($\beta1 = 0.9$, $\beta2 = 0.999$). Batch size of 16 is adopted for training in four NVIDIA RTX2080Ti GPUs with 11 GB. At the beginning of each epoch, learning rate is initialized as $10-4$ and subsequently decayed by half every 10 epochs. We trained 100 epochs on COCO and 150 epochs on DOTA.

2.2. Dataset and Augmentation

We evaluate our method on two well-known benchmarks: COCO and DOTA. The comparative experiments are under same circumstances (training on same GPU and dataset)

The COCO benchmark is a large-scale object detection, segmentation, and captioning dataset. It has 330K images, with more than 200K labelled-images and 80 object categories which is beneficial for object detection training.

The DOTA benchmark is the largest and most challenging dataset with oriented bounding box annotations for aerial image object detection. This dataset consists of 2806 remote sensing images collected by multi-resolution sensors and platforms, with a total of 188,282 in- stances. The image size ranges from approximately 800×800 to 4000×4000 pixels and contains objects with a wide variety of scales, orientations, and shapes. These images have been annotated by experts using 16 common object categories and Table 4 shows our approach shows promising performance on category balance and accuracy. The object categories include helicopter (HC), large vehicle (LV), small vehicle (SV), tennis court (TC), ground track field (GTF), basketball court (BC), soccer field (SBF), baseball diamond (BD), storage tank (ST), swimming pool (SP), and roundabout (RA). DOTA uses random selections of the original image: half as the training set, 1/6th as the validation set, and 1/3rd as test set. DOTA performs two detection tasks to obtain a horizontal bounding box (HBB) and an orientated bounding box (OBB). In our experiment, we split images of different

sizes into 512×512 image patches with an overlap pixels of 150. Therefore, a training set of 27,000 picture patches is generated.

In terms of data augmentation, images are flipped horizontally, vertically, and rotated at random angles. For color, RGB channels are replaced randomly. For image color degradation, saturation in HSV color space is multiplied by a random number in [0,5]. In addition, Gaussian blur is added to blurred image. Finally, to improve the robustness to noise of different intensities, standard deviation of noise is also randomly sampled from Gaussian distribution N (0-1).

2.3. Experiment Analysis

Accuracy. RAWNet outperforms YOLOv4 in accuracy as is shown on Table 1 under the same training process and dataset without data augmentation, therefore, we simply use the advanced MRFNet backbone as the benchmark, and then add attention wise module, the speed is slightly higher than

Table 1 Comparison of the speed and accuracy of different object detectors on the MS COCO dataset (testdev 2017). (Real-time detectors with FPS 24 or higher are highlighted here. We compare the results with batch=1 without using tensorRT.)

Method	Backbone	Size	FPS ()	AP (%)	AP_{50} (%)	AP_{75} (%)	AP_S (%)	AP_M (%)	AP_L (%)
RAWNet：A Learnable multiflow refinement network with rotated attention wise object detectors									
RAWNet	RAWNet	416	31	36.1	59.3	39.1	18.5	40.0	53.3
RAWNet	RAWNet	512	24	38.8	60.8	42.7	20.8	42.4	50.7
RAWNet	RAWNet	608	17	39.7	62.2	43.3	23.1	42.8	48.6
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	30	35.6	57.8	38.2	17.3	39.2	52.1
YOLOv4	CSPDarknet-53	512	22	37.9	60.0	41.9	19.8	41.5	49.8
YOLOv4	CSPDarknet-53	608	16	38.2	60.9	42.5	21.7	42.1	47.4
Learning rich features at high-speed for single-shot object detection									
LRF	VGG-16	300	39.0	26.8	46.7	29.4	8.3	30.3	42.6
LRF	RseNet-101	300	36.2	29.2	50.0	32.2	8.6	33.2	45.9
LRF	VGG-16	512	27.9	31.9	51.7	33.8	14.7	35.4	44.3
LRF	RseNet-101	512	19.7	32.6	53.2	35.1	15.2	38.0	45.4
Receptive Field Block Net for Accurate and Fast Object Detection									
RFBNet	VGG-16	300	32.0	25.3	44.5	27.1	6.9	27.2	41.3
RFBNet	VGG-16	512	22.5	29.2	50.1	31.2	11.7	32.4	42.5
RFBNet-E	VGG-16	512	17.3	29.6	50.7	31.5	12.9	31.8	42.7
YOLOv3: An incremental improvement									
YOLOv3	Darknet-53	320	27	23.3	46.8	25.1	7.2	25.8	38.4
YOLOv3	Darknet-53	416	23	26.4	50.6	27.8	10.3	28.2	38.2
YOLOv3	Darknet-53	608	14	28.0	53.0	29.6	13.7	30.7	36.9
YOLOv3-SPP	Darknet-53	608	16	31.2	56.2	33.5	15.6	33.4	41.4
CenterMask: Real-Time Anchor-Free Instance Segmentation									
CenterMask-Lite	MobileNetV2-FPN	600×	27	25.5	-	-	9.2	27.1	36.3
CenterMask-Lite	VoVNetV-19-FPN	600×	20	31.2	-	-	14.9	33.0	41.2
CenterMask-Lite	VoVNetV-39-FPN	600×	12	35.7	-	-	17.8	38.6	48.5
EfficientDet: Scalable and Efficient Object Detection									
EfficientDet-D0	Efficient-B0	512	26	29.0	47.2	31.2	7.3	33.3	46.2
EfficientDet-D1	Efficient-B1	640	23	34.6	53.8	37.5	13.1	39.8	51.0
EfficientDet-D2	Efficient-B2	768	16	38.2	57.3	41.2	17.9	42.4	53.6
EfficientDet-D3	Efficient-B3	896	13	41.3	60.6	44.6	21.6	45.1	55.1

YOLOv4 except the additional module. However, considering the considerable accuracy improvement and fast training convergence speed, it is worthwhile to modify the model into an attention-wise multipath refinement flow counterpart.

Speed. As for FPS, the speed is in the high level in comparative experiments. Compare to the LRF, YOLOv3, and YOLOv4, ours is slower but shows a significant progress in accuracy. Our method outperforms RFBNet, CenterMask, EfficientDet in both speed and accuracy. It proves that it is a trade-off method regarding accuracy and cost compared with YOLOv4 and it outperforms most of the recent state-of-the-art works.

We can conclude that RAWNet outperforms most of existing methods in accuracy and speed.

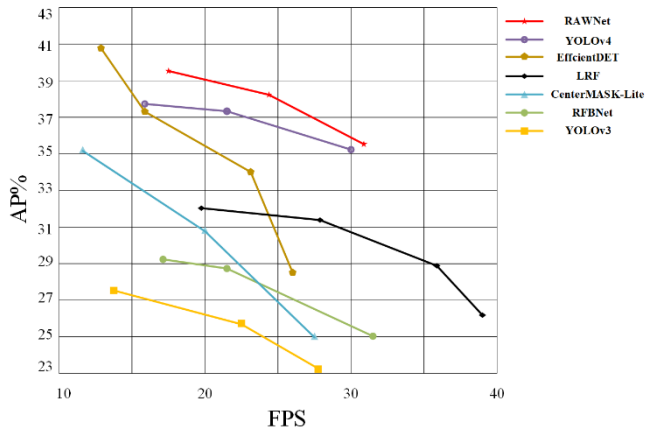


Fig. 10 *Our method shows excellent performance in balancing average precision accuracy and frames per second speed.*

Comparison of the results obtained with other state-of-the-art object detectors are shown in Fig.10 and Fig.11. Our RAWNet are located on the Pareto optimality curve and superior to the fastest and most accurate detectors in terms of both speed and accuracy.

Table 2 Ablation Studies of Network Architecture.(Size 512×512).

Model	AP(%)	AP_{50}(%)	AP_{75}(%)
MRFNet	37.1	58.2	38.2
MRFNet +PWC	37.3	58.1	39.8
MRFNet +RC	37.6	58.6	41.5
MRFNet+PWC+RC	36.9	59.1	44.7
RAWNet	**37.9**	**59.7**	**45.2**

We also make an ablation experiments on COCO by using different attention modules as Table 2 illustrates. MRFNet, PWC, RC and RAWNet represents the benchmark, the pass wise connection, the benchmark adopts the residual strategy, and adopt the above strategies and modules to enhance the average precision (AP), respectively.

Table 3 Using different attention modules for detector training (all other training parameters are similar in all models). (Size 512×512).

Model (with optimal	AP(%)	AP_{50}(%)	AP_{75}(%)
MRFNet	37.6	59.8	41.3
MRFNet +CAW	37.5	59.6	41.0
MRFNet +PRW	37.5	59.3	41.2
MRFNet+CAW+PRW	37.6	60.2	41.5
RAWNet	**38. 2**	**60.4**	**41.8**

In addition, we using different feature extraction methods and network structure as Table 3 illustrates. CAW represents the contextual attention wise modules, PRW represents the position refinement wise modules. RAWNet here adopts the above approaches.

Fig. 10 *Our validation visual results on DOTA using RAWNet backbones.*

Table 4 Detection accuracy **(%)** on different objects and overall performance using current methods on DOTA.

Methods	**PL**	**BD**	**BR**	**GTF**	**SV**	**LV**	**SH**	**TC**	**BC**	**ST**	**SBF**	**RA**	**HA**
RFBNet	40.57	10.21	1.68	14.12	1.32	1.43	2.19	17.22	28.57	10.34	28.26	10.11	4.12

LRF	40.59	21.29	37.74	24.20	9.93	2.19	5.86	45.44	39.45	35.72	17.22	38.73	48.34
CenterMask	90.60	81.97	6.57	67.08	71.12	79.66	79.16	91.81	86.26	85.42	62.91	**64.77**	69.12
EfficientDet	90.02	82.31	47.1 1	72.86	72.96	78.34	80.54	**91.96**	85.14	85.62	57.69	62.13	65.25
YOLOv4	**91.13**	82.13	50.28	72.64	72.78	80.43	80.47	91.89	85.76	85.73	60.12	62.64	68.09
RAWNet	90.08	**86.56**	**54.01**	**74.94**	**76.75**	**82.52**	**81.32**	91.83	**87.96**	**86.34**	**65.14**	61.85	**70.17**

3. Conclusions

In this work we propose a novel detector called RAWNet with attention wise modules. The patches flow in the multipath refinement flow network and features are extracted by pass-wise connection which contributes to a considerable accuracy improvement and training efficiency. The model is evaluated on two public datasets comparing to state-of-the-art approaches and achieves best performance in both accuracy and speed under the same conditions. Our model shows a more significant improvement in learning strategy (adding attention module and rotated tight bounding boxes makes a significant progress on reconstructed deep learning models) and achieves comparable results to similar algorithms such as Region-based Fully Convolutional Network LRF, RFBNet, CenterMask, EfficientDet, YOLOv3.

In the future, we will revise the attention module to further reduce the cost of computation. Also, continuous improvements on object detection detectors could be conducted by applying different data augmentation skills, various feature extraction methods and network enhancement approaches as well. we will explore whether rotated boundaries could be replaced by the instance segmentation and achieves better results on specific object detection such as dense and small object detection or adversarial training process or even video real-time recognition.