algorithms

MDPI

Article

Knowledge-Driven Network for Object Detection

Yundong Wu, Jiajia Liao, Yujun Liu, Kaiming Ding, Shimin Li, Zhilin Zhang, Guorong Cai and Jinhe Su *

Computer Engineering College, Jimei University, Xiamen 361021, China; yundongwu@jmu.edu.cn (Y.W.); jiajialiao@jmu.edu.cn (J.L.); yujunliu@jmu.edu.cn (Y.L.); kmding@jmu.edu.cn (K.D.); a1097736845@gmail.com (S.L.); zhangzhilin_forid@163.com (Z.Z.); guorongcai.jmu@gmail.com (G.C.)
* Correspondence: sujh@jmu.edu.cn

Abstract: Object detection is a challenging computer vision task with numerous real-world applications. In recent years, the concept of the object relationship model has become helpful for object detection and has been verified and realized in deep learning. Nonetheless, most approaches to modeling object relations are limited to using the anchor-based algorithms; they cannot be directly migrated to the anchor-free frameworks. The reason is that the anchor-free algorithms are used to eliminate the complex design of anchors and predict heatmaps to represent the locations of keypoints of different object categories, without considering the relationship between keypoints. Therefore, to better fuse the information between the heatmap channels, it is important to model the visual relationship between keypoints. In this paper, we present a knowledge-driven network (KDNet)—a new architecture that can aggregate and model keypoint relations to augment object features for detection. Specifically, it processes a set of keypoints simultaneously through interactions between their local and geometric features, thereby allowing the modeling of their relationship. Finally, the updated heatmaps were used to obtain the corners of the objects and determine their positions. The experimental results conducted on the RIDER dataset confirm the effectiveness of the proposed KDNet, which significantly outperformed other state-of-the-art object detection methods.

Keywords: convolutional neural network; object detection; relation module; deep learning

check for updates

Citation: Wu, Y.; Liao, J.; Liu, Y.; Ding, K.; Li, S.; Zhang, Z.; Cai, G.; Su, J. Knowledge-Driven Network for Object Detection. *Algorithms* **2021**, *14*, 195. https://doi.org/10.3390/a14070195

Academic Editors: Mounim A. El Yacoubi, Mehdi Ammi and Hui Yu

Received: 31 May 2021
Accepted: 25 June 2021
Published: 28 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid development and remarkable achievements of convolutional neural networks in the field of computer vision, CNNs [1–4] have penetrated many fields of computer vision, such as image classification [5–7], object detection [8–11], and pose estimation [12,13]. As one of the most fundamental computer vision tasks, object detection has resulted in a surge of research interests as it can be easily applied to real-world applications in intelligent cities, intelligent transportation, and other fields. The task of object detection is essentially the extraction of relevant object features from an input image, finding out the objects' regions of interest from these object features, and classifying these while returning to the location of the objects from the region of interest. However, there are various kinds of objects in aerial images. These objects have different appearances, shapes, postures, and there are some interference factors (e.g., illumination and occlusion), which make the detection more difficult. Therefore, object detection is a challenging and non-trivial task.

At present, most of the mainstream high-performance object detectors follow the anchor-based frameworks [8,9]. Firstly, they generate object proposals and then classify and regress each proposal independently. This is a general method based on the two-stage framework, such as Fast R-CNN [9] and Faster R-CNN [8]. It is worth mentioning that anchor-free object detectors (e.g., CornerNet [14], CenterNet [15,16], and ExtremeNet [17]) have appeared in recent years. These change the original inherent object detector that follows the anchor-based paradigm, eliminating the need for designing anchor boxes. These issues of detecting objects are turned into a problem of detecting a pair of keypoints.

In addition, the methods adopt corner pooling to help the CNN locate keypoints with more accuracy. A single convolution network is used to predict the heatmaps, and then the embedding and the offset are performed independently for each class of channels. The heatmaps contain the category and location information of objects. Each channel corresponds to a category of heatmaps. The results of anchor-free methods have shown that detecting corners is the most challenging task. Therefore, improving the accuracy of the heatmap becomes a significant issue for an anchor-free framework. Currently, most of the anchor-free object detection algorithms independently detect objects, but if the model can learn the relationship between objects, this will clearly help to improve the detection effect. Therefore, the use of the relative positional relationship concerning the object in the anchor-free algorithm is a noteworthy direction.

For many years, researchers in the computer vision field have believed that contextual information or relationships between objects contribute to object recognition [18–22]. The basic idea is to take the relational features of an object as the weights of the apparent features of other objects in the image and measure them. This method validates the advantages of modeling objects and creates a general module, which can improve the accuracy of the anchor-based algorithm on the original basis. However, as the anchor-free algorithm does not have the design of an anchor, it is unable to generate a lot of proposals representing the appearance feature of the object and cannot directly measure the relationship between heatmaps. Therefore, it is a very challenging problem to explore the target relationship in heatmaps. In the literature [22], there is strong evidence that object relations can be used to support various visual tasks (e.g., image recognition and object detection). Inspired by the relation network, KDNet proposes a new knowledge-driven network to solve the relationship between objects, shown in Figure 1. The two objects are classified as the same object because of the use of an anchor-free network. The reason for false detection is that in the heatmaps, the keypoints of each object are an independent prediction, and there is no correlation. In the task of object detection, different objects may have some kind of connection. Therefore, we used a knowledge-driven network to establish the relationship between the keypoints of each object and fused the information between heatmaps channels to generate the corner maps that can enhance the characteristics of the detected object.

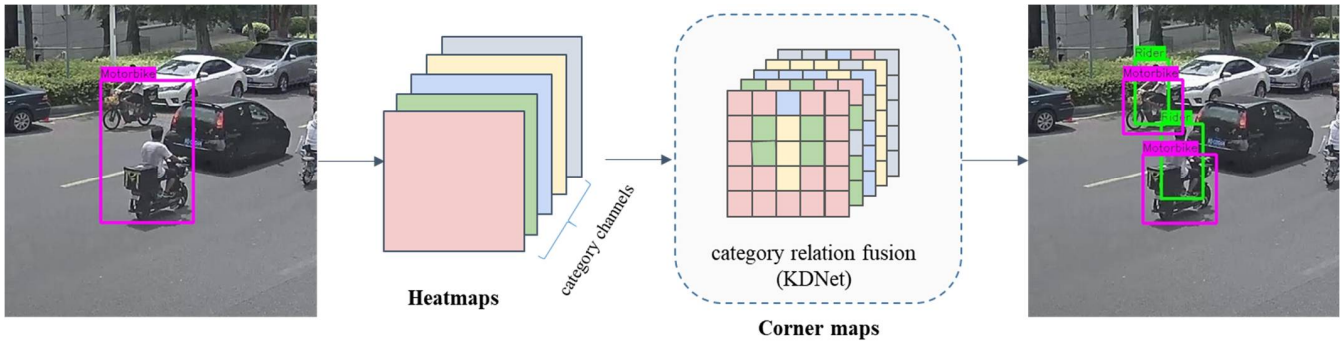


Figure 1. A deficiency of a traditional anchor-free object detection framework. Shown in the first image, the anchor-free network detects two riders as the same object. There are two reasons for this: (1) The process of detecting objects is independent, and it is not possible to assign corresponding weight to the current detection when detecting an object; thus, the network does not learn the relationship between objects. (2) In the algorithm based on keypoint detection, the heatmaps are the keypoints of each object independently. Therefore, in order to better integrate the information between each category channel of heatmaps, we propose a knowledge-driven network (KDNet) to aggregate and model the relationship between keypoints and apply the information of each category channel of heatmaps to establish relationships between all categories. Finally, KDNet has fused category relation information to obtain the final detection results.

In our paper, we propose a modeling corner feature and geometry feature scheme, called a knowledge-driven network (KDNet), which is an effective plug-and-play network. More specifically, the feature map aggregates key information of each category through the corner pooling network and then intercepts the local information around the corner maps

based on the keypoints of the response value top-K of each channel. In this way, the global knowledge is constructed, the geometric features and semantic features of the keypoints are integrated, and links between the keypoints are strengthened so that the categories can be constrained and correct each other. Finally, the correction of the response value on the heatmaps is implemented. Based on the prior knowledge between the categories, the detection of the network is driven to improve the accuracy of corner detection. The code is available at: https://github.com/jmuyjl/KDNet (accessed date 27 June 2021).

2. Related Work

Two-stage approaches. The two-stage method roughly divides the object detection task into two steps. Firstly, a large number of proposal boxes are generated. Secondly, the object categories are classified and these proposals are regressed. In the first step, there are many ways to generate proposals, such as the earliest two-stage algorithm R-CNN [23], which uses a selective search [24] to extract a large number of regions of interest from images. This method is inefficient and computationally expensive. The second way that Faster R-CNN [8] generates proposals is by using a region proposal network (RPN), which sets a series of anchor sizes, then uses a sliding window to generate a large number of proposals on the feature map. In addition to the above methods, Mask R-CNN [25] adds a mask prediction branch on the Faster R-CNN. This can not only detect objects but also segment the object semantically. Cascade R-CNN [26] trains multiple cascaded detectors by using different IOU thresholds; it is an efficient object detector. Moreover, in the remote sensing scene field, traditional detectors for oriented objects are commonly used to rotate anchors based on the RCNN architecture, which multiplies the number of anchors with a variety of angles and adds rotating NMS so that the computational complexities of these models are greatly increased. The typical network is O^2D-Net [27]. In visual object tracking, Siam RCNN [28] proposed a two-stage re-detection architecture, which combines a novel tracklet-based dynamic programming algorithm and object detection method to model the full history of both the object to be tracked and potential distractor objects, thereby enhancing the accuracy of current object detection and object tracking processes. In the instance segmentation field, the hybrid task cascade (HTC) method [29] proposes a new framework and effectively integrates the cascade into instance segmentation by weaving detection and segmentation features together for joint multi-stage processing. In addition, Double-Head R-CNN [30] proposes disentangling the sibling head into two specific branches for classification and regression, respectively. In contrast, the task-aware spatial disentanglement (TSD) method [31] decouples the classification and regression from the spatial dimension by generating two disentangled proposals for them, which are estimated by the shared proposal.

One-stage approaches. The one-stage object detectors classify and directly regress the position of the objects in a single network. Typical algorithms include YOLO [10,32,33], SSD [11], and RetinaNet [34]. YOLOv3 [33] directly extracts features from images through a deep neural network and produces multi-scale feature maps after fusion. Finally, it generates the bounding box of the proposals with different resolutions from the feature maps. The advantage of this scheme is that it can obtain more scale features and reduce the loss of information. In contrast to YOLOv3 [20], SSD [11] has no feature fusion and directly places the anchors densely on the input image, using features of different convolutional layers to conduct regression and classification of the anchors. RetinaNet [34] is a novel algorithm, which focuses on the design of loss function. The designed focal loss solves the problem of uneven positive and negative samples in a one-stage framework and achieves the result of the state-of-the-art MS-COCO dataset [35]. In the remote sensing field, the vehicle detection network (AVDNet) [36] proposes a new framework to robustly detect small-sized vehicles in aerial scenes. In the AVDNet work, these authors introduced ConvRes residual blocks at multi-scales to alleviate the problem of vanishing features for smaller objects caused because of the inclusion of deeper convolutional layers. In addition, AVDNet [36] proposes a recurrent-feature aware visualization (RFAV) technique

to analyze the network behavior. In R3Det [37], the authors proposed an end-to-end refined single-stage rotation detector for fast and accurate positioning of objects and designed a feature-refinement module to improve detection performance by obtaining more accurate features. The key idea of the feature refinement module is to re-encode the position information of the current refined bounding box to the corresponding feature points through feature interpolation to realize feature reconstruction and alignment.

Anchor-free approaches. The anchor-free detection method is a relatively new paradigm in object detection [38–41], eliminating the need for anchor boxes and providing a simplified detection framework. CornerNet [14] transforms the detection of an object into the problem of detecting a pair of keypoints; this eliminates the design trouble of the anchor. It is worth noting that this method achieves high performance. CornerNet-squeeze [42] focuses on detection efficiency, using lightweight backbone networks to improve efficiency without sacrificing accuracy. CenterNet [15] notably introduces the center keypoint based on CornerNet to reduce misjudgment and improve detection accuracy. Additionally, ExtremeNet [17] detects four extreme points of the object (corresponding to the top, bottom, left, and right, respectively) to determine the location and category of the object. The anchor-free algorithm transforms object detection into a pure appearance-based keypoint estimation problem without regional classification and implicit feature learning. Referring to the idea of instance segmentation, FCOS proposes a fully convolutional one-stage object detection structure to solve object detection in per-pixel prediction fashion. Compared to YOLO, FCOS takes advantages of all points in a ground-truth bounding box to predict the bounding boxes, and the low-quality detected bounding boxes are suppressed by the proposed "center-ness" branch. In IENet, the authors proposed an interacting embranchment one-stage anchor-free detector for orientational objects in aerial images. In addition, they used the self-attention mechanisms to develop an IE module to force the orientation prediction task to interact with the features in the classification and localization branches to further improve the accuracy of orientation detection.

Relation for object detection. Many object detection methods based on deep neural networks only use internal features to classify the proposals. However, it is also very important for object detection to model the relationship between objects [43–45]. The success of the object relation module (ORM) [22] verifies the opinion that modeling between objects contributes to object detection. ORM is applied in the anchor-based object detectors. The detection effect is optimized by the relationship between objects in the image and by modeling the relationship between each ROI using geometric features and appearance features so as to obtain global auxiliary information. Based on ORM, there are corresponding extensions for different application scenarios, such as RDN [46], which can promote object detection in the video by capturing the interaction between objects in the context of time and space. Simultaneously, some networks focus on data relations. They make the relationship when labeling data. In this way, supervised training makes the relationship between objects inseparable. Another way is to use the graph convolution network to build the knowledge map and to model the relationship among the objects [47,48]. These methods all attempt to extend the characteristics of the object by using the relationship information between the objects, which shows that it is meaningful and important to use prior knowledge to drive the object detection. The above methods achieve the preset effect for different application scenarios, but most of them are anchor-based algorithms. We believe that the relation between corner points can also be modeled in the algorithm based on an anchor-free detector. In this way, the relationship between all categories was constructed to improve the accuracy of the heatmap and the accuracy of object detection.

3. Proposed Approach

By consolidating the idea of modeling object relations with appearance features and geometry, we first propose a novel knowledge-driven network (KDNet), which is an anchor-free object detector. In this section, we introduce the overall architecture of our proposed method: KDNet; the detailed structure is shown in Figure 2.

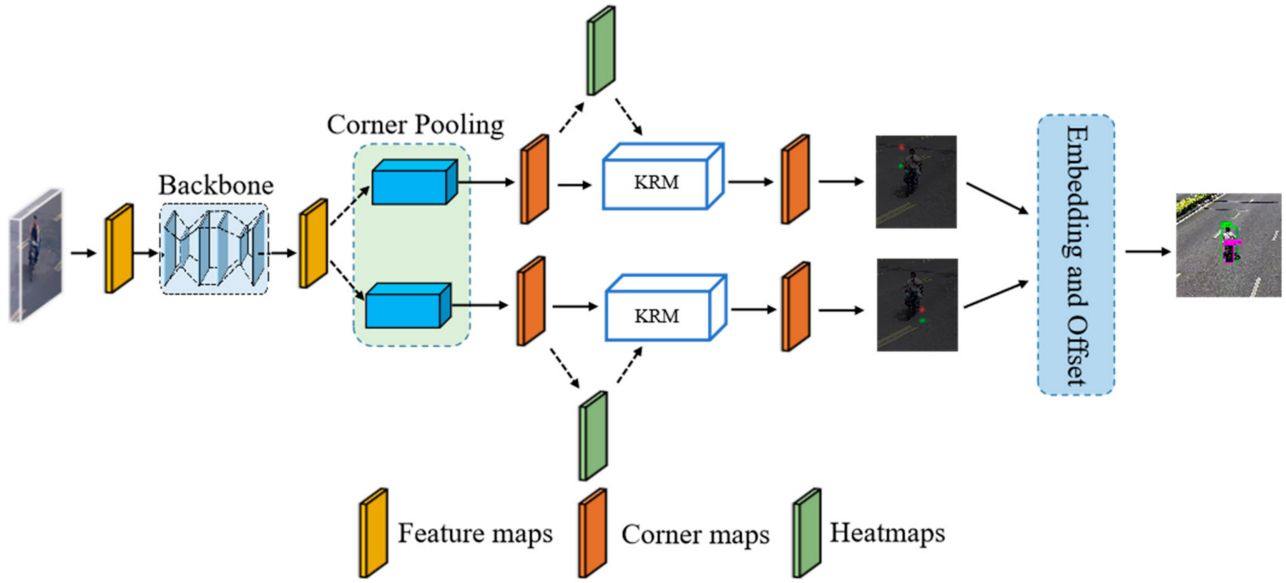


Figure 2. An overview of knowledge-driven networks (KDNet) for object detection. In KDNet, we used a 512 × 512-pixel image as input. Firstly, the input image was downsampled to 64 × 64, and then the features were extracted through the backbone network to generate a feature map with the same dimension as the input. Secondly, the feature map generated a corner map through the independent corner pooling network and generated a heatmap using the corner map as its characteristic angle. Finally, the keypoint relation module (KRM) was used to model the relationship of each keypoint, and then, the keypoints with knowledge fusion were used to reposition the geometric position of the keypoints; thus, we obtained a more accurate heatmap.

3.1. Backbone Parts

Input resolution. As mentioned before, the input of the network was 512 × 512 × 3 RGB images. Before the backbone network, we reduced the image resolution by 8 times using a 7 × 7 convolution kernel module with a stride of 2 and a channel of 128, followed by a residual block with a stride of 2 and a channel of 256. Therefore, the resolution of the image after data preprocessing was reduced to 64 × 64, which was in order to reduce the resolution of the original image to facilitate extraction features.

Backbone network. The backbone network can integrate the features of different levels of an input image and then carry out end-to-end classification. The layers of features can be enriched by training deeper models. Detectors based on the deep neural network typically use a classification network transmitted from the ImageNet classifications as a backbone. Thus, using a good backbone network has a great impact on the accuracy and efficiency of the model. Our method uses the same backbone network as SqueezeNet [49]. It was modified based on the standard SqueezeNet and adopted different convolution methods from the traditional ones; we adopted a 3 × 3 depth-wise separable convolution to replace the 3 × 3 standard convolution. The advantage of using this convolution is that it can reduce the parameter and calculations when the loss accuracy is not high. The entire feature extraction process was sampled using pooling, reducing the parameters while retaining the main object features. It then upsampled the features back to the original resolution by a series of upsampling layers and convolution layers. These novel modules can extract interesting features in the image for further classification and regression.

Corner pooling layers. Most of the algorithms based on the anchor-free framework use the method of corner pooling to aggregate the corners of the object. It is generally believed that the corners of the object tend to fall on the outside of the object and lack local appearance features. Corner pooling can solve this problem well. In the horizontal direction, there are changes from right to left to the maximum value of the currently traversed row elements. In the vertical direction, there are sequential changes from bottom to top to the maximum value of the column elements that have been traversed. After this, the

corresponding positions of the feature maps were added to obtain the final result. Taking the coordinates of the top left corner as an example, the top edge associated with the top left corner contains feature information at the top of the object; the left edge contains the feature information on the left side of the object. By gradually spreading the maximum value of the eigenvalues on both sides to the upper left corner area, and finally superposing, the response of the coordinates of this point will be larger than the sum of the other neighboring regions. The corner maps generated by this scheme can effectively aggregate the corner information of objects. Specifically, to determine whether a pixel is a top-left corner, we need to look horizontally towards the right for the topmost boundary of an object and look vertically towards the bottom for the leftmost boundary. We define the bottom-right corner pooling layer in a similar way. For example, for the top-left corner, we scanned from right to left for the horizontal max-pooling and from bottom to top for the vertical max-pooling. For the bottom-right corner, we scanned from left to right for the horizontal max-pooling and from top to bottom for the vertical max-pooling. We then directly added two max-pooled feature maps. We thus adopted corner pooling to better aggregate the top-left corner and the bottom-right corner by encoding explicit prior knowledge.

Loss function. In the task of object detection, the principle of loss function should be designed to optimize object localization and recognition. Our loss function for an image is defined as Equation (1):

$$Loss = L_{heatmap} + \alpha L_{pull} + \beta L_{push} + \gamma L_{offset}, \tag{1}$$

where $L_{heatmap}$ designates a variant of focal loss, which is used to train the network to detect corners. L_{pull} is a pull loss for corners, which determines whether a pair comprising the top left corner and bottom-right corner is from the same bounding box or not. L_{push} is a push loss for corners, which aims at separating the corners from different bounding boxes. L_{offset} is used to train the network to predict the offsets of corners and applies the smooth L1 loss. α, β, and γ denote the weights for the corresponding losses, and we set them as 0.1, 0.1, and 1, respectively, as we have found that if the values of α and β are set as 1 or larger, this leads to poor detection performance. $L_{heatmap}$, L_{pull}, L_{push}, and L_{offset} are all defined in the following:

$$L_{heatmap} = \frac{-1}{\mathrm{N}} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} \begin{cases} (1-p_{chw})^{\lambda} \log(p_{chw}) & if\ y_{chw} = 1 \\ (1-y_{chw})^{\nu} (p_{chw})^{\lambda} \log(1-p_{chw}) & otherwise \end{cases}, \tag{2}$$

$$L_{pull} = \frac{1}{N} \Sigma_{k=1}^{N} \left[(e_{tk} - e_k)^2 + (e_{bk} - e_k)^2 \right], \tag{3}$$

$$L_{push} = \frac{1}{N(N-1)} \Sigma_{k=1}^{N} \Sigma_{j=1, j \neq k}^{N} \max\left(0, \Delta - \left|e_k - e_j\right|\right), \tag{4}$$

$$L_{offset} = \frac{1}{N} \Sigma_{k=1}^{N} SmoothL1(_k, \hat{k}), \tag{5}$$

where p_{chw} is the score at location (h, w) for class c in the predicted heatmaps, and y_{chw} is the ground-truth heatmap augmented with the unnormalized Gaussians. N denotes the number of objects in an image, and λ and ν are the hyperparameters that control the contribution of each point. In Equation (3), e_{tk} is used to denote the embedding for the top-left corner of object k, e_{bk} stands for the bottom-right corner, and $e_k = (e_{tk} + e_{bk})/2$. In Equation (4), the maximum distance is 1 for two corners from different objects, and we set Δ to be 1 in all our experiments. In Equation (5), $_k$ is the ground-truth offset, $\hat{k}$ denotes the predicted offset, and $_k = \left(\frac{x_k}{n} - \left|\ \frac{x_k}{n}\ \right|, \frac{y_k}{n} - \left|\ \frac{y_k}{n}\ \right|\right)$, where x_k and y_k are the x and y coordinates for corner k.

3.2. Keypoint Relation Module

In our analysis, the existing anchor-free algorithms relied on independent detection between instances instead of trying to exploit the relationship between them while learning.

However, we assume that if the model can learn the relationship between different objects, it will be very helpful for object detection. Information about other objects around it is likely to help with the classification and positioning of an object. In summary, we have proposed a new module, named KRM, which is embedded in the corner maps, and the input and output of the module are unchanged. This KRM aims to model the corner feature and geometry feature of the relationship between corners, which is relevant to the object detection, enhancing the corner features on corner maps to produce a more accurate heatmap. As shown in Figure 3, firstly, for each channel of the heatmap (each channel represents a category), we selected the top-K corners as samples to construct the region. Then, according to the coordinate information of the top-K corners, the $r \times r$ area around each corner of the corner maps was intercepted. Therefore, not only can the region of interest around the corner be obtained, but also the object area. Local features can also make full use of the depth features in corner maps. Each extraction region of corners (RoC) is fully connected to the dc dimension, which is sent to the relation module for modeling the corner relationship and the geometric position relationship, which then outputs the dc dimension vector. To be able to embed back to the original corner maps, the dc dimension vector is restored to the original dimension through a full connection, thus updating the corner maps. The entire KRM uses corner features and geometric features to model the angular relationship between the categories. The independent corner features are weighted with coordinate information, and the resulting corners fuse the corner features of all categories. Based on this relationship, the generated heatmap has semantic information between the channels, making the detection process independent.

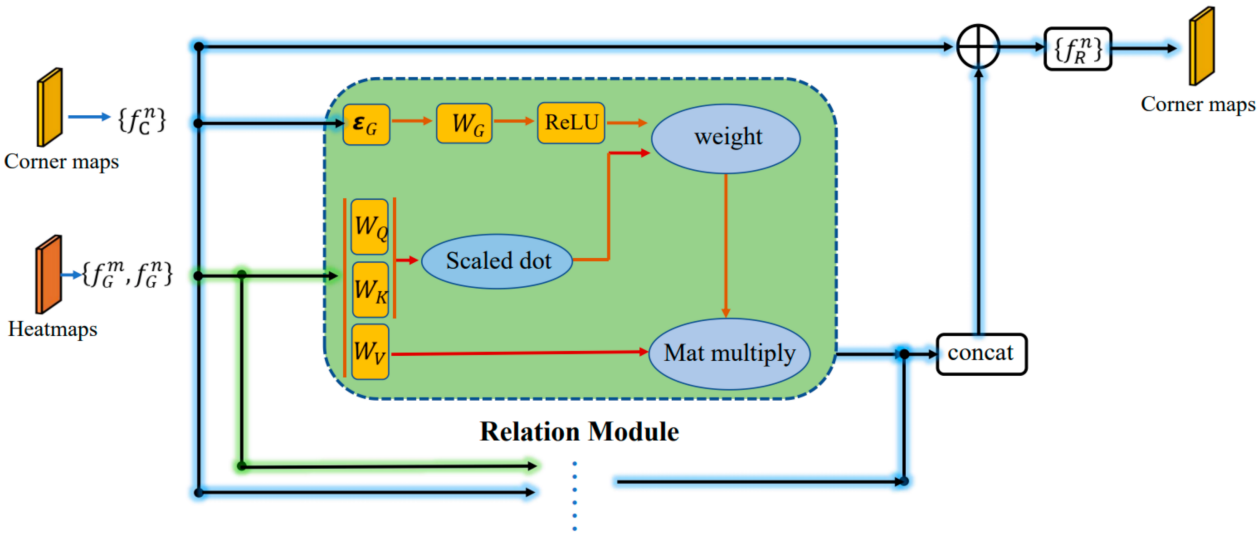


Figure 3. The architecture of the keypoint relation module (KRM).

Geometry features. The geometric features come from the top-K corners selected by each channel of the heatmap. The corner maps are extracted by the corner pooling, and the corner information of the object has been enhanced so that each corner can provide corresponding geometric coordinate information. Note that the four-dimensional features representing $f_G = \{x, y, c, r\}$ contain three parts. Specifically, (x, y) is the center position of the corner; c is the category information of each channel; and r is a hyperparameter, which is the size of the intercepted RoC, as shown in the following:

$$\left(\log\left(\frac{|x_m - x_n|}{r_m}\right), \log\left(\frac{|y_m - y_n|}{r_m}\right), \log\left(\frac{c_n}{c_m}\right), \log\left(\frac{r_n}{r_m}\right)\right)^T,$$

where x_m, x_n, y_m, y_n, r_m, and r_n are mentioned in the heatmaps of the keypoint relation module (KRM) in Figure 3. n and m represent the n-th corner point and m-th corner

point, respectively, because each corner point needs to calculate the relationship with other corner points. This 4D feature is embedded in a high-dimensional representation, which computes the cosine and sine functions of different wavelengths. The feature dimension after embedding is dg ($dg = 64$ in this paper).

Corner features. The corner features use the above-mentioned extracted corner position, centering on the corner point and intercepting the area of size $r \times r$ on the corner maps to obtain the depth feature in the corner maps. The advantage of doing this is that the corner maps have more semantic information and can obtain global knowledge. Each corner feature region (RoC) that is intercepted is fully connected to the dc (dc = 1024 in this paper) dimension, and this corner feature is represented by f_C,

$$f_C = \{top - K, dc\}, \tag{6}$$

where $top - K$ denotes the first K representative points, which are used to calculate the local and geometric features. By modeling the relationship between f_C and f_G, the f_C and f_G weights of other corners are fused into their own f_C, and a variety of local knowledge can be obtained to drive the recognition and localization of the object.

Relation module. The relation module borrows the idea of its object relation module (ORM) to model the corner features and geometric features of keypoints. Firstly, the respective weights W_v and W_g are calculated according to two characteristics, and the parameters indicate the weights of the local features and geometric features of the corner points, respectively. The total weight ω^{mn} (Equation (9)) is obtained from the weights of the two features ω_G^{mn} and ω_C^{mn} (Equations (11) and (10)), and then each relation module is weighted according to the total weight of the m-th keypoint to the current keypoint.

$$f_R(n) = \sum_m \omega^{mn} \cdot (W_V \cdot f_C^m), \tag{7}$$

$$f_C^n = f_C^n + Concat\left[f_R^1(n), \ldots, f_R^{N_r}(n)\right], \ for\ all\ n, \tag{8}$$

$$\omega^{mn} = \frac{\omega_G^{mn} \cdot \exp\left(\omega_C^{mn}\right)}{\sum_k \omega_G^{kn} \cdot \exp\left(\omega_C^{kn}\right)}, \tag{9}$$

$$\omega_C^{mn} = \frac{\mathrm{dot}\left(W_K \mathbf{F}_C^m, W_Q \mathbf{F}_C^n\right)}{\sqrt{d_c}}, \tag{10}$$

$$\omega_G^{mn} = \max\{0, W_G \cdot \mathcal{E}_G(\mathbf{F}_G^m, \mathbf{F}_G^n)\}, \tag{11}$$

where m and n represent the n-th corner point and m-th corner point, respectively. $\mathcal{E}_G$ is to map the geometric feature $\mathbf{F}_G$ between the m-th corner point and the n-th corner point to a high-dimensional space. $\mathbf{F}_G^m$ represents the geometric feature of the m-th corner point, and $\mathbf{F}_G^n$ denotes the geometry of the n-th corner point feature.

4. Experiments

4.1. Datasets

To better demonstrate the correctness of our approach, we constructed a new dataset called RIDER to evaluate our method. The dataset came from real video surveillance scenes, covering many scenes, such as night, sunny, cloudy, and foggy scenes. Moreover, there were many objects to be detected with different scales. Most importantly, the relationship between categories was obvious, which was very suitable for the test difficulty of this work. To guarantee that the distribution of training, validation, and test data were similar, we randomly selected 46,449 images as the training set and 5162 images as the test set. RIDER contained five categories: *Rider*, *Motorbike*, *Bike*, *Helmet*, and *Pedestrian*, and the number of images per subclass is given in Table 1. The labeled *Rider* is characterized by a person driving a two-wheeled vehicle, and the entire body is visible (green rectangles); *Motorbike* is characterized by a two-wheeled vehicle, and the tire part is similar to a bike

(pink rectangles); *Bike* is characterized by a simple structure and a narrow width (blue rectangles); *Helmet* is a safety helmet, which is elliptical (purple rectangles); *Pedestrian* is characterized by a person walking on the road (red rectangles). Example images are shown in Figure 4.

Table 1. The number of per subclass on RIDER dataset.

Rider	*Motorbike*	*Bike*	*Pedestrian*	*Helmet*	*Total*
31,996	32,956	6510	13,434	25,011	51,611

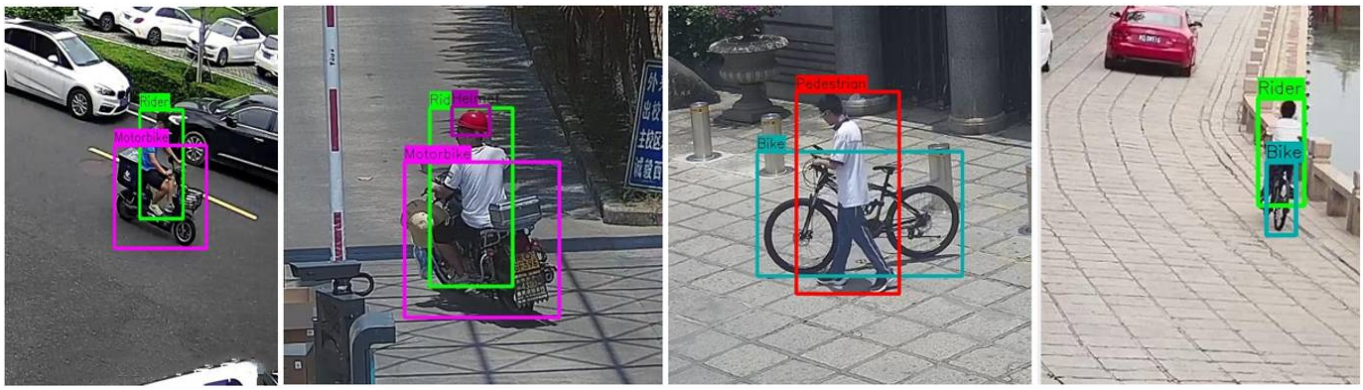


Figure 4. Categories in RIDER.

4.2. Implementation Details

We demonstrated the effectiveness of our method on the RIDER dataset. In addition, we chose several mainstream algorithms, including YOLOv3 [33], RetinaNet [34], and Faster R-CNN [8], to test on the RIDER dataset as baselines. In YOLOv3 [33], we used Darknet-53 as the backbone network. The initial learning rate was 10×10^{-3}, the dimension of the input image was $608 \times 608 \times 3$, the training batch was set to 4, and the gradient descent momentum was 0.9. RetinaNet [34] used the Keras framework to implement the Resnet-50 model pre-trained by Keras on ImageNet as the backbone. The initial learning rate was 10^{-5}, the dimension of the input image was $600 \times 1000 \times 3$, and the training batch was set to 1. The gradient descent momentum was 0.9. In Faster R-CNN, we used the pre-trained VGG16 model on ImageNet as the backbone network. The learning rate was 10e-3, the dimension of the input image was $600 \times 1000 \times 3$, and the training batch was set to be 1. We trained the KDNet with a batch size of 27 on two 1080Ti GPUs and implemented our proposed in the PyTorch. For all the algorithms, we used Adam for iterations with the initial learning rate of 10×10^{-3}, a weight decay of 10×10^{-4}, and a momentum of 0.9. To train a robust model and avoid over-fitting, we adopted the data augmentation strategy, including random horizontal flipping, random cropping, random scaling, and random color jittering, as well as adjusting the brightness, saturation, and contrast of a training image. Finally, we used principal component analysis (PCA) on the input image.

4.3. Overall Accuracy Evaluation

The results are summarized in Table 2, which shows the mAPs and the APs from different categories. It can be seen that the mAP of KDNet was higher than state-of-the-art algorithms, including YOLOv3 [33], RetinaNet [34], and Faster-RCNN [8]. The results show that the knowledge-driven object detection strategy adopted in this paper, combined with the anchor-free framework, can provide more reliable detection results. The results show that the knowledge-driven strategy was adopted to fuse the adjacent spatial coordinate information of keypoints to the process of keypoints generation. The extracted features can fuse the characteristics of other objects, thus greatly improving the detection effect.

Table 2. The mAP and AP of the tested algorithms.

Method	mAP	*Rider*	*Motorbike*	*Bike*	*Pedestrian*	*Helmet*
YOLOv3	0.424	0.392	0.510	0.414	0.449	0.353
RetinaNet	0.683	0.665	0.733	**0.747**	0.625	**0.643**
Faster R-CNN	0.560	0.510	0.652	0.652	0.560	0.427
KDNet (Ours)	**0.709**	**0.718**	**0.783**	0.737	**0.703**	0.603

4.4. The Evaluation on Different Categories

First, we analyzed the results on *Pedestrian*, *Rider*, and *Motorbike*. Figure 5 shows the results of a typical scene that were obtained using four methods. As shown in Figure 5, the object in the image is ambiguous and there is occlusion; it is very challenging to extract the object region. Figure 5a–c shows the detection results of YOLOv3 [33], RetinaNet [34], and Faster-RCNN [8], respectively. Note that YOLOv3 [33] and RetinaNet [34] failed to distinguish between the pedestrian and the rider. As for the RPN-based method, the number of proposals was relatively large; thus, Faster R-CNN [8] can effectively detect objects, while the byproduct is a large number of false-positive detections. In contrast, our proposed method in this paper combined the geometric features and corner features between objects and successfully detected objects while using knowledge-driven methods to effectively avoid false detection.

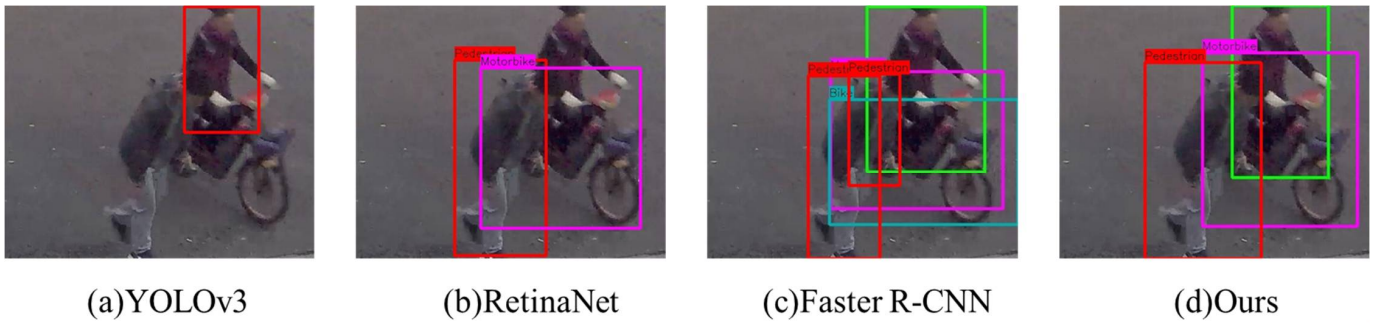


Figure 5. The detection results of the position of *Pedestrian*, *Rider*, and *Motorbike*.

The second set of experiments was *Bike* and *Rider*, shown in Figure 6. It can be seen that this task was challenging, because the objects to be detected were very close. In this complicated background, YOLOv3 [33] missed the objects, and the Faster-RCNN [8] and RetinaNet [34] algorithms not only failed to detect the right objects, but also generated a large number of false bounding boxes. However, KDNet (Figure 6d) combined the information from features that can help the detect the object more accurately. This shows that adding the category information in the geometric feature can increase the semantic information between the objects. After the corner feature and the geometric feature were merged, it effectively avoided false detection of missing parts.



Figure 6. The detection results of the position of *Rider* and *Bike*.

4.5. Typical Fail Cases

Although the proposed KDNet achieved the best performance, considering the value of mAP, our method did not work in some cases. Figure 7 shows the detection results in which the proposed KDNet failed. For example, the *Helmet* in Figure 7 had a small object shape and was almost integrated with the boundary background, which posed a great challenge for detection. However, for this small object, YOLOv3 has multi-scale fusion, and it can be well detected; Faster R-CNN can generate a large number of proposals by using an RPN strategy, and therefore the objects were also detected. The reason for our proposed KDNet failing to extract the helmet was that the top-left corner of the hat was close to the top-left corner of the rider. Therefore, after the relation module was completed, the keypoint near the area may have been the NMS algorithm, thus causing a missed detection.



Figure 7. Typical detection results in which the proposed KDNet failed.

4.6. Inference Time

The inference time between the proposed KDNet and other algorithms is compared in Table 3. The average inference time of KDNet on a 1080Ti GPU was 450 ms per image. We found that the inference time of our proposed model was slower than other models. The reason may be that we added a network (KRM) to build object relations, which contained 58.5 M parameters and a large number of matrix operations, which caused the inference time to slow down.

Table 3. Comparison of inference time of different algorithms.

Method	YOLOv3	RetinaNet	Faster R-CNN	KDNet (Ours)
Inference Time (ms)	51	150	298	450

5. Conclusions

In this paper, a novel modeling corner feature and geometry feature scheme, called a knowledge-driven network (KDNet), is presented. It simultaneously processes a set of keypoints through their local features and coordinates and the interaction between semantic information, allowing it to model their relationships. Based on this relationship, the generation of an updated heatmap caused semantic information to spread between channels; thus, the detection process was not independent. The experimental results conducted on the RIDER dataset showed that the proposed KDNet had a better mAP than mainstream algorithms such as YOLOv3, RetinaNet, and Faster R-CNN. In particular, the APs of all categories exceeded the state-of-the-art object detection methods. We believe that these promising results pave the way to numerous further applications and developments.

Author Contributions: Conceptualization, G.C. and Y.W.; methodology, J.S.; software, J.L.; validation, Y.L., J.L. and K.D.; formal analysis, S.L.; investigation, Y.L.; resources, J.L.; data curation, J.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.W.; visualization, Z.Z.; supervision, Y.L.; funding acquisition, J.S., G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant no. 41971424 and no. 61701191; the Natural Science Foundation of Fujian Province, China under Grant 2020J01701, and in part by the Fujian Provincial Science and Technology Program Project under Grants JAT190318.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
2. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
3. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
4. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
5. Haralick, R.M.; Shanmugam, K.; Dinstein, I. Textural Features for Image Classification. *IEEE Trans. Syst. Man Cybern.* **1973**, *6*, 610–621. [CrossRef]
6. Wei, Y.; Xia, W.; Lin, M.; Huang, J.; Ni, B.; Dong, J.; Zhao, Y.; Yan, S. HCP: A Flexible CNN Framework for Multi-Label Image Classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 1901–1907. [CrossRef] [PubMed]
7. Hershey, S.; Chaudhuri, S.; Ellis, D.P.; Gemmeke, J.F.; Jansen, A.; Moore, R.C.; Plakal, M.; Platt, D.; Saurous, R.A.; Seybold, B. CNN architectures for large-scale audio classification. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 131–135.
8. Ren, S.; He, K.; Girshick, R.; Sun, J.J. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv* **2015**, arXiv:1506.01497.
9. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
12. Güler, R.A.; Neverova, N.; Kokkinos, I. Densepose: Dense human pose estimation in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7297–7306.
13. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.-E.; Sheikh, Y. OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 172–186. [CrossRef] [PubMed]
14. Law, H.; Deng, J. Cornernet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 734–750.
15. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. Centernet: Keypoint triplets for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 6569–6578.
16. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as points. *arXiv* **2019**, arXiv:1904.07850.
17. Zhou, X.; Zhuo, J.; Krahenbuhl, P. Bottom-up object detection by grouping extreme and center points. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 850–859.
18. Galleguillos, C.; Rabinovich, A.; Belongie, S. Object categorization using co-occurrence, location and appearance. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
19. Torralba, A.; Murphy, K.P.; Freeman, W.T.; Rubin, M.A. Context-based vision system for place and object recognition. In Proceedings of the Computer Vision, IEEE International Conference on IEEE Computer Society, Madison, WI, USA, 18–20 June 2003; p. 273.
20. Tu, Z. Auto-context and its application to high-level vision tasks. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
21. Mottaghi, R.; Chen, X.; Liu, X.; Cho, N.-G.; Lee, S.-W.; Fidler, S.; Urtasun, R.; Yuille, A. The role of context for object detection and semantic segmentation in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 891–898.

22. Hu, H.; Gu, J.; Zhang, Z.; Dai, J.; Wei, Y. Relation networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 3588–3597.
23. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
24. Uijlings, J.R.; Van De Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective Search for Object Recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
25. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
26. Cai, Z.; Vasconcelos, N. Cascade r-cnn: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6154–6162.
27. Wei, H.; Zhang, Y.; Chang, Z.; Li, H.; Wang, H.; Sun, X. Oriented objects as pairs of middle lines. *ISPRS J. Photogramm. Remote Sens.* **2020**, *169*, 268–279. [CrossRef]
28. Voigtlaender, P.; Luiten, J.; Torr, P.H.; Leibe, B. Siam r-cnn: Visual tracking by re-detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 6578–6588.
29. Chen, K.; Pang, J.; Wang, J.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Shi, J.; Ouyang, W. Hybrid task cascade for instance segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4974–4983.
30. Wu, Y.; Chen, Y.; Yuan, L.; Liu, Z.; Wang, L.; Li, H.; Fu, Y. Rethinking classification and localization for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 10186–10195.
31. Song, G.; Liu, Y.; Wang, X. Revisiting the sibling head in object detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 11563–11572.
32. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Venice, Italy, 22–29 October 2017; pp. 7263–7271.
33. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
34. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
35. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Berlin/Heidelberg, Germany; pp. 740–755.
36. Mandal, M.; Shah, M.; Meena, P.; Devi, S.; Vipparthi, S.K. AVDNet: A small-sized vehicle detection network for aerial visual data. *IEEE Geosci. Remote. Sens. Lett.* **2019**, *17*, 494–498. [CrossRef]
37. Yang, X.; Liu, Q.; Yan, J.; Li, A.; Zhang, Z.; Yu, G. R3det: Refined single-stage detector with feature refinement for rotating object. *arXiv* **2019**, arXiv:1908.05612.
38. Kong, T.; Sun, F.; Liu, H.; Jiang, Y.; Li, L.; Shi, J. Foveabox: Beyound anchor-based object detection. *IEEE Trans. Image Process.* **2020**, *29*, 7389–7398. [CrossRef]
39. Mylavarapu, S.K.; Choudhuri, S.; Shrivastava, A.; Lee, J.; Givargis, T. FSAF: File system aware flash translation layer for NAND flash memories. In Proceedings of the 2009 Design, Automation & Test in Europe Conference & Exhibition, Grenoble, France, 9–13 March 2020; pp. 399–404.
40. Tian, Z.; Shen, C.; Chen, H.; He, T. Fcos: Fully convolutional one-stage object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, CA, USA, 15–20 June 2019; pp. 9627–9636.
41. Yu, J.; Jiang, Y.; Wang, Z.; Cao, Z.; Huang, T. Unitbox: An advanced object detection network. In Proceedings of the 24th ACM International Conference on Multimedia, Palo Alto, CA, USA, 7–10 November 2016; pp. 516–520.
42. Law, H.; Teng, Y.; Russakovsky, O.; Deng, J. Cornernet-lite: Efficient keypoint based object detection. *arXiv* **2019**, arXiv:1904.08900.
43. Pang, Y.; Xie, J.; Khan, M.H.; Anwer, R.M.; Khan, F.S.; Shao, L. Mask-guided attention network for occluded pedestrian detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, CA, USA, 15–20 June 2019; pp. 4967–4975.
44. Du, X.; Shi, X.; Huang, R. Repgn: Object detection with relational proposal graph network. *arXiv* **2019**, arXiv:1904.08959.
45. Zhou, P.; Chi, M. Relation parsing neural network for human-object interaction detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, CA, USA, 15–20 June 2019; pp. 843–851.
46. Deng, J.; Pan, Y.; Yao, T.; Zhou, W.; Li, H.; Mei, T. Relation distillation networks for video object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, CA, USA, 15–20 June 2019; pp. 7023–7032.
47. Yang, J.; Lu, J.; Lee, S.; Batra, D.; Parikh, D. Graph r-cnn for scene graph generation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 670–685.
48. Xu, H.; Jiang, C.; Liang, X.; Li, Z. Spatial-aware graph relation network for large-scale object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9298–9307.
49. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.