R(Det)^2: Randomized Decision Routing for Object Detection

Ya-Li Li Shengjin Wang*
Department of Electronic Engineering, Tsinghua University and BNRist, Beijing, China
liyali13,wgsgj@tsinghua.edu.cn

Abstract

In the paradigm of object detection, the decision head is an important part, which affects detection performance significantly. Yet how to design a high-performance decision head remains to be an open issue. In this paper, we propose a novel approach to combine decision trees and deep neural networks in an end-to-end learning manner for object detection. First, we disentangle the decision choices and prediction values by plugging soft decision trees into neural networks. To facilitate effective learning, we propose randomized decision routing with node selective and associative losses, which can boost the feature representative learning and network decision simultaneously. Second, we develop the decision head for object detection with narrow branches to generate the routing probabilities and masks, for the purpose of obtaining divergent decisions from different nodes. We name this approach as the randomized decision routing for object detection, abbreviated as R(Det)^2. Experiments on MS-COCO dataset demonstrate that R(Det)^2 is effective to improve the detection performance. Equipped with existing detectors, it achieves $1.4 \sim 3.6\%$ *AP improvement.*

1. Introduction

Object detection, which aims to recognize and localize the objects of interest in images, is a fundamental yet challenging task in computer vision. It is important for various applications, such as video surveillance, autonomous driving, and robotics vision. Due to its practical importance, object detection has attracted significant attention in the community. In recent decades, deep neural networks (DNNs) have brought significant progress into object detection. Typically, existing deep learning-based detection methods include one-stage detectors [22, 25, 31], two-stage detectors [1, 7, 16, 30, 33], end-to-end detectors [3, 39, 51].

Generally, current deep architectures constructed for object detection involve two components. One is the backbone for feature extraction, which can be pre-trained with large-scale visual recognition datasets such as ImageNet [35]. The other is the decision head, which produces the predictions for computing losses or inferring detection boxes. Collaborated with region sampling, object detection can be converted into a multitask learning issue, where the decision tasks include classification and bounding box (*bbox*) regression. For existing detection networks, the decision head is simply constructed by sequentially connecting several convolution or fully-connected layers. For one-stage detectors, the decision head is commonly constructed by stacking several convolutional layers. The decision head for region proposal in two-stage detectors is similar. For two-stage detectors, the region-wise decision in R-CNN stage is typically implemented with 2 fully-connected layers. Since the decision head is quite important for high-performance detectors, there are recently devoted researches [8, 12, 37, 43]. However, most of these works focus on task disentanglement and

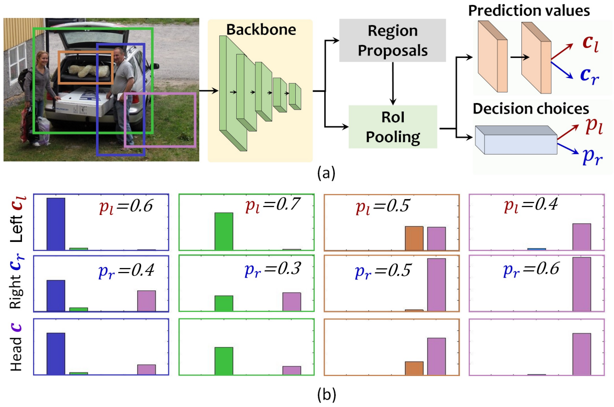

Figure 1. **Overview of the proposed approach.** (a) Inspired by decision trees, we disentangle the decision choices and predictive values by introducing tree structure for decision head in object detection. With multi-node prediction, we can explore more diverse cues. (b) We use the soft probability to denote decision choices for different routes of nodes. The overall decision is the weighted sum of prediction values from different nodes. Specially, we propose randomized decision routing to learn divergent decisions from different nodes for overall performance improvement.

[*] Corresponding author

task-aware learning, leaving the universal decision mechanism far from exploitation.

Considering that the features from DNNs show great potential for high-level vision tasks, the simple design of widely-adopted single-node decision might impede the performance of object detection. A natural question arises: *is single-node prediction good enough for feature exploration in object detection?* To answer this, we focus on novel decision mechanism and propose an approach to introduce soft decision trees into object detection. As in Figure 1, we integrate soft decision trees to disentangle the routing choices and prediction values. To jointly learn the soft decision trees and neural networks in an end-to-end manner, we propose the randomized decision routing with the combination of so-called *selective loss* and *associate loss*. Experiments validate the effectiveness of the proposed approach and address the necessity of introducing multi-node predictions. Since our work is mainly on **R**andomized **D**ecision rou**t**ing for object **Det**ection, we name it as R(Det)^2. From the perspective of machine learning, our R(Det)^2 is an attempt to bridge the neural networks and decision trees – two mainstream algorithms, which would bring insights into future research.

The contributions of this paper are three-fold.

- We propose to disentangle the route choices and prediction values for multi-node decision in object detection. In particular, we propose randomized decision routing for the end-to-end joint learning of the tree-based decision head.
- We construct a novel decision head for object detection, which introduces routing probabilities and masks to generate divergent decisions from multiple nodes for the overall decision boosting.
- Extensive experiments validate the effectiveness of our proposed R(Det)^2. In particular, R(Det)^2 achieves over 3.6% of AP improvement when equipped with Faster R-CNN. It improves the detection accuracy of large objects by a large margin as well.

2. Related work

One-stage detectors. Overfeat [36] predicts the decision values for classification and localization directly with convolutional feature maps. YOLO [31, 32] regresses the object bounds and category probabilities directly based on image gridding. SSD [25] improves the one-stage detection with various scales of multilayer features. Retina Net [22] proposes the focal loss to tackle the foreground-background imbalance issue. Besides, keypoints-based one-stage detectors [5, 11, 20, 49] have been extensively studied. CornerNet [20] generates the heatmaps of top-left and bottom-right corners for detection. CenterNet [11] uses a triplet of keypoints for representation with additional center points. Moreover, FCOS [40] and ATSS [47] introduce centerness branch for anchor-free detection. Other methods delve into sample assignment strategies [2, 14, 19, 28, 47, 50].

Two-stage detectors. R-CNN [16], Fast R-CNN [15], Faster R-CNN [33] predict object scores and bounds with pooled features of proposed regions. R-FCN [7] introduces position-sensitive score maps to share the per-ROI feature computation. Denet [41] predicts and searches sparse corner distribution for object bounding. CCNet [29] connects chained classifiers from multiple stages to reject background regions. Cascade R-CNN [1] uses sequential R-CNN stages to progressively refine the detected boxes. Libra R-CNN [30] mainly tackles the imbalance training. Grid R-CNN [27] introduces pixel-level grid points for predicting the object locations. TSD [37] decouples the predictions for classification and box bounding with the task-aware disentangled proposals and task-specific features. Dynamic R-CNN [46] adjusts the label-assigning IoU thresholds and regression hyper-parameters to improve the detection quality. Sparse R-CNN [38] learns a fixed set of sparse candidates for region proposal.

End-to-end detectors. DETR [3] models object detection as a set prediction issue and solve it with transformer encoder-decoder architecture. It inspires the researches on transformer-based detection frameworks [9, 10, 24, 39, 51]. Deformable DETR [51] proposes the sparse sampling for key elements. TSP [39] integrates FCOS and R-CNN head into set prediction issue for faster convergence.

Decision mechanism. The decision head in object detection frameworks usually involves multiple computational layers (*i.e.*, convolution layers, fully-connected layers and transformer modules). Typically, for one-stage detectors with dense priors [11, 22, 25, 31, 40], stacked convolutions are used to obtain features with larger receptive fields, with separate convolution for classification, localization and other prediction tasks. For the decision in R-CNN stages [1, 27, 30, 33, 46], stacked fully-connected layers are common. Double-head R-CNN [43] uses fully-connected layers for position-insensitive classification and fully-convolutional layers for position-sensitive localization. Dynamic head [8] unifies the scale-, spatial- and task-aware self-attention modules for multitask decisions.

3. Randomized decision trees

3.1. Soft decision trees

To disentangle the decision choices and prediction values, we first construct soft decision trees [13] for multiclass classification and *bbox* regression in object detection. We use the soft routing probability ranging from 0 to 1 to represent the decision choice and facilitate network optimization.

Soft decision tree for classification. For multiclass classification, the soft decision tree is formulated as:

$$\mathbf{c} = \sum_{j \in Nodes} p_j \mathbf{c}_j, \ \sum_{j \in Nodes} p_j = 1 \tag{1}$$

where $\mathbf{c}$ is the output of the whole classification tree and $\mathbf{c}_j$ is the prediction value from each node. p_j is the routing probability for decision choice. It indicates the probability of choosing j-th classification node. For all the nodes, $\sum_{j\in Nodes} p_j = 1$. Eqn. 1 shows that $\mathbf{c}$ is the weighted sum of the classification scores from all the nodes. Different from traditional decision tree, p_j is "soft" ranging from 0 to 1. p_j can be obtained in networks by a scalar score with activations such as *Softmax*, *Sigmoid*.

Soft decision tree for regression. For *bbox* regression, we formulate the soft decision tree in a similar way as:

$$\mathbf{b} = \sum_{j\in Nodes} q_j \mathbf{b}_j, \sum_{j\in Nodes} q_j = 1 \qquad (2)$$

where $\mathbf{b}_j$ is the regression value output from each node j. q_j is the routing probability for the j-th regression node. $\mathbf{b}$ is the output of the tree regressor. Similar to soft classification tree, the routing probability $q_j \in [0, 1]$ is "soft".

Noting that the routing probabilities p_j, q_j denote decision choices, which indicates the probability of routing the j-th node. It can be viewed as decision confidence in test phase. $\mathbf{c}_j$ and $\mathbf{b}_j$ are the prediction values for classification and regression tasks attached with the j-th node. Both the decision choices and prediction values can be easily obtained with neural layers. With soft decision trees, multiple discriminative and divergent decisions can be obtained with features from different aspects. To facilitate the discussion, we restrict the soft decision tree as binary and $j \in \{l, r\}$.

3.2. Randomized Decision Routing

To learn soft decision trees in neural networks, we propose randomized decision routing. The motivation is twofold. First, in order to obtain a high-performance decision head with tree structure, we need to avoid the high relevance of multiple predictions from different nodes. It means that we should differentiate the training to reduce the decision relevance of different nodes. Second, we also need to guarantee the decision performance of the whole tree. In a word, we need to achieve high-performance tree decision with low-relevant node decisions. To realize this, we propose the *selective loss* to supervise the per-node learning and *associative loss* to guide the whole-tree optimization. We then unify the *selective* and *associative loss* into a general training framework. Since we involve random factors to model the probability of routing different nodes, we name this training strategy as randomized decision routing.

To achieve node decisions with low relevance, we first perform node selection to identify the node with higher optimization priority. We then attach the selected node with a higher routing probability. Oppositely, a lower routing probability is attached with the remaining node. Divergent routing probabilities lead to different learning rates for different nodes. Therefore, to diversify the decision of different nodes, we construct the *selective loss* by setting different randomized weights for different node losses. As illustrated in Figure 2-left, the *selective losses* for classification and *bbox* regression are denoted as:

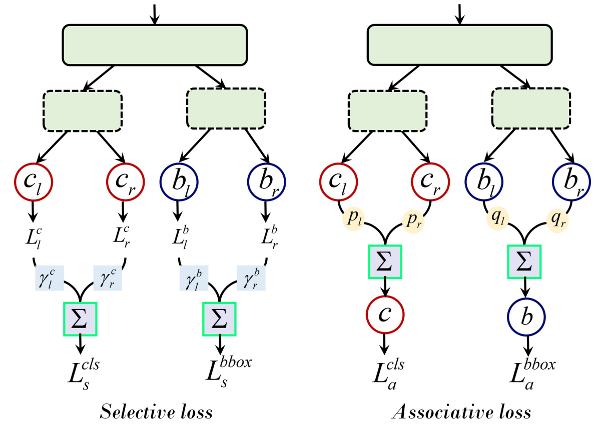

Figure 2. **Illustration on training deep networks with decision tree head.** We propose randomized decision routing which includes selective and associative losses. The selective loss identifies the dominant decisive prediction and weights the node loss accordingly in a randomized way. The associate loss learns the routing probability by measuring the difference between the fused output and the ground truth.

$$\begin{aligned} L_s^{cls}(\mathbf{c}_l, \mathbf{c}_r, y) &= \gamma_l^c L_l^c + \gamma_r^c L_r^c \\ &= \gamma_l^c L^{cls}(\mathbf{c}_l, y) + \gamma_r^c L^{cls}(\mathbf{c}_r, y) \end{aligned} \qquad (3)$$

$$\begin{aligned} L_s^{bbox}(\mathbf{b}_l, \mathbf{b}_r, B) &= \gamma_l^b L_l^b + \gamma_r^b L_r^b \\ &= \gamma_l^b L^{bbox}(\mathbf{b}_l, B) + \gamma_r^b L^{bbox}(\mathbf{b}_r, B) \end{aligned} \qquad (4)$$

where y is the ground truth label and B is the ground truth for *bbox* regression. γ_l^c, γ_r^c are the weights indicating the probability for selective routing of classification tree. γ_l^b, γ_r^b are the weights indicating the probability for selective decision routing of *bbox* regression tree.

We leverage random weights to differentiate the node learning. For classification, we set γ_l^c, γ_r^c based on the comparison of L_l^c, L_r^c. We set the nodes with lower loss values with higher random weights. For *bbox* regression, we set the weights γ_l^b, γ_r^b according to the relative comparison of q_l, q_r. For instance, if $q_l < q_r$, we restrict $\gamma_l^b < \gamma_r^b$. It is consistent with the intuition that we learn the selective node with higher priority in a fast way, meanwhile learning the remaining one in a slow way. Empirically, we sample the lower weight from $U(0.1, 0.3)$ and the higher weight from $U(0.9, 1.1)$. This slow-fast randomized manner would benefit the learning of the whole decision head.

Besides of differentiating node decisions, we also need to ensure the performance of the whole decision tree. That is, the predictive decision output from the whole tree should be good. To achieve this, we formulate *associative loss*

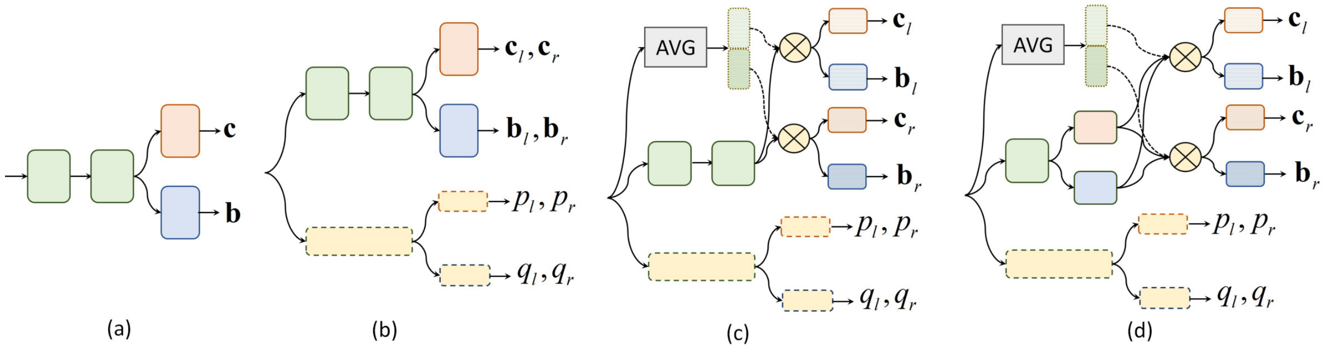

Figure 3. **Decision head for object detection.** (a) shows the common decision head. (b) shows *R(Det)*^{2}*-B* which disentangles the decision choice and values by soft decision trees. (c) shows *R(Det)*^{2}*-M* which leverages the routing masks to produce the divergent input features for decision. (d) shows *R(Det)*^{2}*-T* which unifies task disentanglement into R(Det)^2-based decision head.

based on the fused prediction $\mathbf{c}$, $\mathbf{b}$. The *associative loss* can be the same as the original classification or *bbox* regression loss in form, with the fused prediction as the input. As illustrated in Figure 2-right, the *associative loss* for classification and *bbox* regression is formulated as:

$$L_a^{cls}\left(\mathbf{c}, y\right) = L^{cls}\left(p_l \mathbf{c}_l + p_r \mathbf{c}_r, y\right) \tag{5}$$

$$L_a^{bbox}\left(\mathbf{b}, B\right) = L^{bbox}\left(q_l \mathbf{b}_l + q_r \mathbf{b}_r, B\right) \tag{6}$$

The routing probabilities and prediction values are simultaneously optimized with the *associative loss*. Specially, the routing probability which indicates the decision choice is only supervised by this *associative loss*, resulting in appropriate routing in inference.

The whole loss is formulated as follows:

$$L_{all} = \lambda\left(L_s^{cls} + L_s^{bbox}\right) + (1-\lambda)\left(L_a^{cls} + L_a^{bbox}\right) \tag{7}$$

where $\lambda \in [0, 1]$ is the coefficient to balance between *selective loss* and *associative loss*. It is noteworthy that the L^{cls}, L^{bbox} for computing the *selective* and *associative loss* can be commonly-used loss functions for classification (*e.g.,* cross-entropy loss, Focal loss [22]) and *bbox* regression (*e.g.*, Smooth-L$_1$ loss, IoU loss [34, 42, 45, 48]). With soft decision trees, we can generate multiple decisions with different visual cues. Moreover, the divergent learning helps enhance feature representations and suppress over-optimization, further promote object detection.

4. Decision head for detection

We construct the head with decision trees for object detection. The common-used head of R-CNN detectors [1, 17, 21, 33] is single-prediction type, as in Figure 3(a). Typically, two fully-connected (*fc*) layers are sequentially connected with region-pooled features, with one additional *fc* layer for classification and *bbox* regression, respectively. In order to obtain decision values for multiple nodes, we first generate predictions $\mathbf{c}_l, \mathbf{c}_r$ and $\mathbf{b}_l, \mathbf{b}_r$ with the features output from the same structure as the common head. We further add another narrow branch with 1~2 *fc* layers to produce the routing probabilities p_l, p_r and q_l, q_r, as illustrated in Figure 3(b). We record this as the **B**asic head for randomized decision routing, as *R(Det)*^{2}*-B*. The routing choices and predictions are disentangled with this basic head structure.

Moreover, we add the routing masks for features before prediction to increase the divergence of decisions from multiple nodes. The decision values $\mathbf{c}_l, \mathbf{c}_r$ and $\mathbf{b}_l, \mathbf{b}_r$ are generated with route-wise masked features. As in Figure 3(c), we average the batched region-wise features to obtain a single context-like vector. Another *fc* layer with *Sigmoid* is imposed on this vector to produce routing masks for different nodes. By multiplying the route-wise masks on the last features before decision, we further diversify the input for different nodes of decision. The dependence of node decisions can be further reduced. We record this as **M**asked head for randomized decision routing, as *R(Det)*^{2}*-M*.

Inspired by efforts on disentangling the classification and localization tasks for detection, we develop another *R(Det)*^{2}*-T*. We separate the last feature computation before the multitask prediction and unify the task-aware feature learning into our framework, as in Figure 3(d). Since it is not the main focus of this work, we have not involved more complicated task-aware head designs [37, 43, 46]. Yet it is noteworthy that the proposed R(Det)^2 can easily be plugged into these detectors for performance improvement.

5. Experiments

Datasets. We evaluate our proposed approach on the large-scale benchmark *MS COCO* 2017 [23]. Following common practice, we train detectors on *training* split with ~115k images and evaluate them on *val* split with 5k images. We also report the results and compare with the state-of-the-art on *COCO test-dev* split with 20k images. The

	B	*M*	*T*	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
2fc				37.4	58.1	40.4	21.2	41.0	48.1
2fc	✓			38.8	59.8	41.8	22.3	42.3	50.9
		✓		39.1	60.5	42.3	22.5	43.1	50.5
			✓	38.9	60.2	42.1	23.1	42.1	50.2
4conv	✓			38.7	59.0	41.9	22.4	42.0	50.4
1fc		✓		39.2	59.7	42.4	22.8	42.8	51.5
			✓	39.5	59.8	42.9	22.7	43.1	51.7
4conv	✓			39.3	60.2	42.7	22.5	42.8	51.6
(res)		✓		40.1	60.8	43.3	23.3	43.5	52.6
1fc			✓	40.4	61.2	44.1	23.8	43.7	53.0

Table 1. **Ablation study on different types with R(Det)^2.** The baseline is Faster R-CNN equipped with ResNet-50 backbone. *B*, *M* and *T* represents *R(Det)^2-B*, *R(Det)^2-M* and *R(Det)^2-T* for decision heads, respectively.

standard mean average precision (AP) across different IoU thresholds is used as the evaluation metric.

Training details. We implement the proposed R(Det)^2 as the plug-in head and integrate it into existing detectors. Our implementation is based on the popular mmdetection [4] platform. If not specially noted, the R(Det)^2 serves for the decision in R-CNN of two-stage detectors, as Faster R-CNN [33], Cascade R-CNN [1]. We train the models with ResNet-50/ResNet-101 [18] backbones with 8 Nvidia TitanX GPUs. The learning rate is set to 0.02 and the weight decay is 1e-4, with momentum 0.9. The models for ablation studies are trained with the standard 1× configuration. No data augmentation is used except for standard horizontal image flipping. We only conduct multiscale training augmentation for evaluation on COCO *test-dev* to compare with the state-of-the-art.

Inference details. It is noteworthy that the randomized decision routing is only performed in training phase. In inference, we perform on the single image scale without specific noticing. Following standard practice, we evaluate the models with test time augmentation (TTA) as multiscale testing to compare with the state-of-the-art.

5.1. Ablation study

Effects of components. We first conduct the ablative experiment to evaluate the effects of different components for R(Det)^2 (Table. 1). We integrate the proposed decision head structure into the R-CNN stage and apply randomized decision routing for training. We first follow the common setting with 2×1024 fully-connected layers (referred as *2fc*) to generate region-wise features, with decision values for multiclass classification and *bbox* regression predicted based on them. By converting *2fc* to R(Det)^2-B, we increase the detection AP to 38.8%, yielding 1.4% of improvement. By adding routing masks for region-wise features, R(Det)^2-M achieves 39.1% detection AP, 1.7% of improvement. It is reasonable since the mask multiplying would promote the decision differences between nodes, leading to the improvement of joint decision. We further replace *2fc* with 4×256 convolutional layers with 1 fully-connected layer (referred as *4conv1fc*). The achieved AP increases to 38.7%, 39.2% and 39.5% with R(Det)^2-B, R(Det)^2-M, R(Det)^2-T, respectively. We further add residual connections between neighboring convolutions for feature enhancement, referred to as *4conv(res)1fc*. By integrating *4conv(res)1fc* with R(Det)^2-B, we achieve AP of 39.3% and AP_{75} of 42.7%. By integrating R(Det)^2-M, the achieved AP is 40.1% and AP_{75} is 43.3%. With task disentanglement as R(Det)^2-T, we achieve AP, AP_{50}, AP_{75} of 40.4%, 61.2% and 44.1%, respectively. Compared to the baseline, the AP, AP_{50}, AP_{75} is increased by 3.0%, 3.1% and 3.7%, respectively. In particular, the R(Det)^2 significantly improves the detection accuracy on large objects, leading to the AP_L improvement by a large margin. Compared with the baseline, we achieve 4.9% of AP_L improvement ultimately. It verifies that the features contain much more information to be exploited, especially for larger objects with high-resolution visual cues. Our proposed R(Det)^2 which produces decisions with multiple nodes can focus on the evidence from diverse aspects, leading to significant performance improvement.

L^{cls}	L^{bbox}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline		37.4	58.1	40.4	21.2	41.0	48.1
CE	S-L1	40.4	61.2	44.1	23.8	43.7	53.0
Focal	S-L1	40.5	61.2	44.4	24.2	43.6	52.6
CE	IoU	40.9	61.2	44.5	23.9	44.2	53.7
Focal	IoU	41.0	61.1	44.5	24.3	44.3	53.7

Table 2. **Comparison with different loss functions.** The baseline model is Faster R-CNN with ResNet-50 as the backbone. CE indicates the cross-entropy loss. Focal indicates the original focal loss [22]. S-L1 indicates the Smooth-L$_1$ loss. IoU indicates the loss computed by the negative-log of intersection-over-union [45].

Effectiveness with different loss functions. The proposed randomized decision routing can be combined with any existing classification and localization losses. We conduct experiments to evaluate the effectiveness of R(Det)^2 with different loss functions(Table 2). When we apply the *Softmax* cross-entropy loss for classification and Smooth-L$_1$ loss for *bbox* regression, we achieve 40.4% AP, 61.2% AP_{50}, 44.1% AP_{75}. Compared to baseline Faster R-CNN with the same losses, we increase the AP, AP_{50}, AP_{75} by 3.0%, 3.1%, 3.7%, respectively. The AP is slightly higher with focal loss [22] applying for classification. The detection AP is further increased with IoU loss [45] applied for *bbox* regression. The detection AP reaches 41.0%. Com-

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
R50	37.4	58.1	40.4	21.2	41.0	48.1
+R(Det)^2	**41.0**	**61.2**	**44.8**	**24.6**	**44.1**	**53.7**
	(+3.6)	**(+3.1)**	**(+4.4)**	**(+3.4)**	**(+3.1)**	**(+5.6)**
R50-DCN	41.3	62.4	45.0	24.6	44.9	54.4
+R(Det)^2	**44.2**	**64.5**	**48.3**	**26.6**	**47.7**	**58.6**
	(+2.9)	**(+2.1)**	**(+3.3)**	**(+2.0)**	**(+2.8)**	**(+4.2)**
R101	39.4	60.1	43.1	22.4	43.7	51.1
+R(Det)^2	**42.5**	**62.8**	**46.3**	**25.1**	**46.4**	**55.7**
	(+3.1)	**(+2.7)**	**(+3.2)**	**(+2.7)**	**(+3.7)**	**(+4.8)**
R101-DCN	42.7	63.7	46.8	24.9	46.7	56.8
+R(Det)^2	**45.0**	**65.4**	**49.2**	**27.2**	**48.8**	**59.6**
	(+2.3)	**(+1.7)**	**(+2.4)**	**(+2.3)**	**(+2.1)**	**(+2.8)**

Table 3. **Comparison with different backbone networks.** R-50 and R-101 indicates ResNet-50 and ResNet-101, respectively. R(Det)^2 is plugged in Faster R-CNN with various backbones and achieves consistent performance gains.

pared with the baseline, the AP is increased by 3.6% and AP_L is increased by 5.6%. It indicates that the proposed R(Det)^2 performs well with different combinations of loss functions, which further demonstrates its effectiveness.

Effectiveness on different backbone networks. With Faster R-CNN as the baseline detector, we conduct the ablative experiment to evaluate the effectiveness of R(Det)^2 on various backbones(Table 3). With ResNet-50 as the backbone, the achieved AP, AP_{50} and AP_{75} of R(Det)^2 is improved by 3.6%, 3.0%, and 4.1%, respectively. With ResNet-50-DCN (ResNet-50 with deformable convolution) as the backbone, we achieve the detection AP of 44.2%, 2.9% improvement. The performance gain of R(Det)^2 with ResNet-101 is also significant. By equipping with R(Det)^2, the detection AP of ResNet-101 reaches 42.5% and AP_{75} reaches 46.3%, 3.1% and 3.2% higher than the baseline. With ResNet-101-DCN as the backbone, the AP reaches 45.0% and AP_{75} is 49.2%. In particular, the detection accuracy over large objects is improved significantly. The AP_L over the different backbones is increased by 5.6%, 4.2%, 4.8% and 2.8%, respectively. Experiments show that the proposed R(Det)^2 is effective among object detectors with various backbones.

Generalization on different detectors. We plug R(Det)^2 into existing detectors to evaluate the generalization capability (Table 4). Other than Faster R-CNN, we integrate R(Det)^2 with libra R-CNN [30], dynamic R-CNN [46], cascade R-CNN [1]. The backbone is ResNet-50. Upon libra R-CNN, R(Det)^2 improves the detection AP by 3.1% and AP_{75} by 3.6%, yielding 41.4% AP and 45.5% AP_{75}. On cascade R-CNN, the powerful detector with cascade structure, R(Det)^2 also shows consistent improvement. It improves the detection AP by 2.2% and AP_{50} by 2.4%, respectively. Since the dynamic R-CNN [46] adaptively changes the hyperparameters of Smooth-L$_1$ loss for *bbox* regression, we present the detection accuracy by randomized routing upon Smooth-L$_1$ loss, instead of IoU loss with better performance. By equipping R(Det)^2, the AP and AP_{75} is increased by 2.1%. Besides, R(Det)^2 is quite effective to improve the detection performance of large objects. The AP_L of libra R-CNN and cascade R-CNN is increased by a large margin with R(Det)^2, leading to 5.2% and 4.1% improvement, respectively. For DoubleHead R-CNN [43] and one-stage RetinaNet [22] with designed head, we fix the head for task-aware decision. Only randomized routing based training leads to 1.4% of AP improvement with DoubleHead R-CNN and 1.8% of AP improvement with RetinaNet [22]. The experiment validates that the proposed R(Det)^2 performs well on existing detectors.

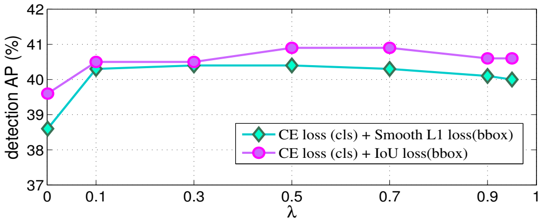

Figure 4. **Effects on hyperparameter** λ to balance the *selective loss* and *associative loss* for decision routing.

Effects of hyperparameter λ. We leverage the hyperparameter λ to balance the *selective* and *associative loss* in randomized decision routing. We further evaluate the effects of λ with ResNet-50-based Faster R-CNN. The curves of detection AP changing along with λ are plotted in Figure 4. The detection accuracy is the highest when $\lambda = 0.5$. That means we assign the weights for the *selective* and *associative loss* nearly equal. The detection AP remains stable when λ is between 0.1 to 0.9. If we further reduce λ to 0.001 and reduce the impact of *selective loss*, the detection AP with Smooth-L$_1$ loss for *bbox* regression decreases to 38.6%, by 1.8% points. It indicates that the *selective loss* which aims to differentiate node decisions is essential for performance improvement. Since only associative loss guides the optimization of routing probabilities, increasing λ to nearly 1 would lead to unstable models (the parameters to generate routing probabilities p_l, p_r, q_l, q_r is nearly the same as random initialized ones), we restrict $\lambda \leq 0.95$. The detection AP at $\lambda = 0.95$ is decreased by 0.3$\sim$0.4%.

Model complexity and computational efficiency. The model complexity of R(Det)^2 is mainly caused by the additional branches for routing probability, routing mask, and task-aware features. From Table 5 we can see that the complexity is mainly caused by task-aware feature computation. Considering this, we develop *R(Det)^2-Lite* with nar-

Detector	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Libra R-CNN [30]	38.3	59.5	41.9	22.1	42.0	48.5
+R(Det)^2	**41.4(+3.1)**	**61.4(+1.9)**	**45.5(+3.6)**	**24.7(+2.5)**	**45.0(+3.0)**	**53.7(+5.2)**
Cascade R-CNN [1]	40.3	58.6	44.0	22.5	43.8	52.9
+R(Det)^2	**42.5(+2.2)**	**61.0(+2.4)**	**45.8(+1.8)**	**24.6(+2.1)**	**45.5(+1.7)**	**57.0(+4.1)**
Dynamic R-CNN [46]	38.9	57.6	42.7	22.1	41.9	51.7
+R(Det)^2	**41.0(+2.1)**	**59.7(+2.1)**	**44.8(+2.1)**	**23.3(+1.2)**	**44.2(+2.3)**	**54.8(+3.1)**
DoubleHead R-CNN [43]	40.1	59.4	43.5	22.9	43.6	52.9
+R(Det)^2	**41.5(+1.4)**	**60.8(+1.4)**	**44.5(+1.0)**	**24.2(+1.3)**	**45.0(+1.4)**	**53.9(+1.0)**
RetinaNet [22]	36.5	55.4	39.1	20.4	40.3	48.1
+R(Det)^2	**38.3(+1.8)**	**57.4(+2.0)**	**40.8(+1.7)**	**22.6(+2.2)**	**42.0(+1.7)**	**50.5(+2.4)**

Table 4. **Generalization with different detectors.** R(Det)^2 shows AP improvement on various detectors.

Type	*#FLOPs*	*#params*	*AP(%)*
4conv1fc	129.0G	15.62M	37.6
R(Det)^2-B	132.6G	19.31M	39.8
R(Det)^2-M	132.6G	25.88M	40.5
R(Det)^2-T	146.3G	45.97M	40.9
R(Det)^2-Lite	130.2G	18.48M	40.2

Table 5. Model complexity comparison of R(Det)^2 head.

row computation for routing probabilities and masks, leading to 40.2% AP and nearly ignorable model complexity.

Visualization. We present the comparative visualization in Figure 5. The detected results by ResNet-101 based Faster R-CNN are shown in Figure 5(a) and those from the R(Det)^2 are shown in Figure 5(b). It can be seen that the proposed R(Det)^2 is effective to improve both the detection and localization performance. Specially, the R(Det)^2 is quite effective in reducing the repeated detections and avoiding over-confident ones.

5.2. Comparison with the state-of-the-art

We integrate the proposed R(Det)^2 into Cascade R-CNN to compare with the state-of-the-art methods on *COCO test-dev* dataset. The backbone is ResNeXt-101 (64×4d) [44] with deformable convolution and swin transformer [26]. The comparative study is presented in Table 6. We first compare the single-model single-scale model performance. With 12 epochs (1×) of training, the R(Det)^2 achieves AP of 50.0%, outperforming Faster R-CNN [33], Libra R-CNN [30], Cascade R-CNN [1] by a large margin. Compared with the recent Sparse R-CNN [38] with the same backbone, we achieve 1.1% AP improvement with 1/3 training iterations. It is also comparable with deformable DETR [51] with transformer architecture and much more epochs of training (50 epochs). The detection accuracy is further improved with more epochs of training and test-time augmentation as multi-scale testing and horizontal image flipping. With 24 epochs of training and TTA, the R(Det)^2 achieves AP of 54.1% and AP_{50} of 72.4%. Compared with DyHead with stacked self-attention modules [8], the AP_{50}, AP_L is improved by 0.3% and 1.0%, respectively. Besides, we adapt the backbone of ViT as swin transformer [26]. With 12 epochs of training, the achieved AP of single-scale testing is 55.1% and that of multi-scale testing is 57.4%. It validates the R(Det)^2 performs well with various backbones and is effective for high-performance object detection.

6. Conclusion

The decision head is important for high-performance object detection. In this paper, we propose a novel approach as the randomized decision routing for object detection. First, we plug soft decision trees into neural networks. We further propose the randomized routing to produce accurate yet divergent decisions. By randomized routing for soft decision trees, we can obtain multi-node decisions with diverse feature exploration for object detection. Second, we develop the decision head for detection with a narrow branch to generate routing probabilities and a wide branch to produce routing masks. By reducing the relevance of node decisions, we develop a novel tree-like decision head for deep learning-based object detection. Experiments validate the performance of our proposed R(Det)^2.

Acknowledgement

This work was supported by the state key development program in 14th Five-Year under Grant Nos.2021QY1702, 2021YFF0602103, 2021YFF0602102. We also thank for the research fund under Grant No. 2019GQG0001 from the Institute for Guo Qiang, Tsinghua University.

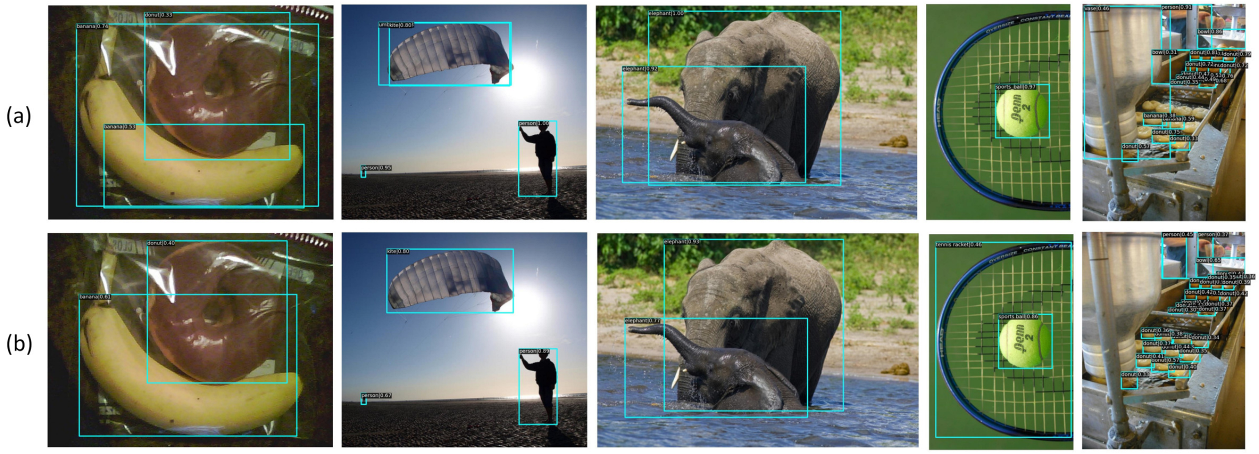

Figure 5. **Comparison of detection results for the baseline Faster R-CNN and R(Det)^2 equipped one.** The models are with ResNet-101 as the backbone and trained with *COCO* 115k-*train*. The example test images are from *COCO* 5k-*val*. The rectangles mark the detected bounding boxes with attached category labels and confidences. The detection results of baseline model are presented in (a) (39.3% AP) and those of R(Det)^2 are presented in (b) (42.5% AP).

Methods	Backbone	ME	TTA	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Retina-Net [22]	ResNeXt-101	18e		40.8	61.1	44.1	24.1	44.2	51.2
FCOS [40]	ResNeXt-101	24e		43.2	62.8	46.6	26.5	46.2	53.3
ATSS [47]	ResNeXt-101-DCN	24e		47.7	66.5	51.9	29.7	50.8	59.4
OTA [14]	ResNeXt-101-DCN	24e		49.2	67.6	53.5	30.0	52.5	62.3
IQDet [28]	ResNeXt-101-DCN	24e		49.0	67.5	53.1	30.0	52.3	62.0
Faster R-CNN [33]	ResNet-101	12e		36.7	54.8	39.8	19.2	40.9	51.6
Libra R-CNN [30]	ResNeXt-101	12e		43.0	64.0	47.0	25.3	45.6	54.6
Cascade R-CNN [1]	ResNet-101	18e		42.8	62.1	46.3	23.7	45.5	55.2
TSP-RCNN [39]	ResNet-101-DCN	96e		47.4	66.7	51.9	29.0	49.7	59.1
Sparse R-CNN [38]	ResNeXt-101-DCN	36e		48.9	68.3	53.4	29.9	50.9	62.4
Deformable DETR [51]	ResNeXt-101-DCN	50e		50.1	69.7	54.6	30.6	52.8	64.7
Ours - R(Det)^2	ResNeXt-101-DCN	12e		**50.0**	**69.2**	**54.3**	**30.9**	**53.0**	**63.9**
Ours - R(Det)^2	Swin-L [26]	12e		**55.1**	**74.1**	**60.4**	**36.0**	**58.6**	**70.0**
Centernet [11]	Hourglass-104	100e	✓	47.0	64.5	50.7	28.9	49.9	58.9
ATSS [47]	ResNeXt-101-DCN	24e	✓	50.7	68.9	56.3	33.2	52.9	62.4
IQDet [28]	ResNeXt-101-DCN	24e	✓	51.6	68.7	57.0	34.5	53.6	64.5
OTA [14]	ResNeXt-101-DCN	24e	✓	51.5	68.6	57.1	34.1	53.7	64.1
Dynamic R-CNN [46]	ResNet-101-DCN	36e	✓	50.1	68.3	55.6	32.8	53.0	61.2
TSD [37]	SENet154-DCN	36e	✓	51.2	71.9	56.0	33.8	54.8	64.2
Sparse R-CNN [38]	ResNeXt-101-DCN	36e	✓	51.5	71.1	57.1	34.2	53.4	64.1
RepPoints v2 [5]	ResNeXt-101-DCN	24e	✓	52.1	70.1	57.5	34.5	54.6	63.6
Deformable DETR [51]	ResNeXt-101-DCN	50e	✓	52.3	71.9	58.1	34.4	54.4	65.6
RelationNet++ [6]	ResNeXt-101-DCN	24e	✓	52.7	70.4	58.3	35.8	55.3	64.7
DyHead [8]	ResNeXt-101-DCN	24e	✓	54.0	72.1	59.3	37.1	57.2	66.3
Ours - R(Det)^2	ResNeXt-101-DCN	24e	✓	**54.1**	**72.4**	**59.4**	**35.5**	**57.0**	**67.3**
Ours - R(Det)^2	Swin-L [26]	12e	✓	**57.4**	**76.1**	**63.0**	**39.4**	**60.5**	**71.5**

Table 6. **Comparison of R(Det)^2 with the state-of-the-art object detection methods on *COCO test-dev* dataset.** DCN indicates that using the deformable convolution to enhance the feature representations of backbone. TTA indicates test-time augmentation such as multi-scale testing and horizontal image flipping. ME indicates more epochs of training.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. *CVPR*, pages 6154–6162, 2018. 1, 2, 4, 5, 6, 7, 8

[2] Yuhang Cao, Kai Chen, Chen Change Loy, and Dahua Lin. Prime sample attention in object detection. *CVPR*, pages 11583–11591, 2020. 2

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ECCV*, pages 213–229, 2020. 1, 2

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5

[5] Yihong Chen, Zheng Zhang, Yue Cao, Liwei Wang, Stephen Lin, and Han Hu. Reppoints v2: Verification meets regression for object detection. *NIPS*, pages 5621–5631, 2020. 2, 8

[6] Cheng Chi, Fangyun Wei, and Han Hu. Relationnet++: Bridging visual representations for object detection via transformer decoder. *NIPS*, pages 13564–13574, 2020. 8

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *NIPS*, pages 379–387, 2016. 1, 2

[8] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. *CVPR*, pages 7373–7382, 2021. 1, 2, 7, 8

[9] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. *ICCV*, pages 2988–2997, 2021. 2

[10] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. *CVPR*, pages 1601–1610, 2021. 2

[11] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. *ICCV*, pages 6569–6578, 2019. 2, 8

[12] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R. Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. *ICCV*, pages 3490–3499, 2021. 1

[13] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017. 2

[14] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. *CVPR*, pages 303–312, 2021. 2, 8

[15] Ross Girshick. Fast r-cnn. *ICCV*, pages 1440–1448, 2015. 2

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, pages 580–587, 2014. 1, 2

[17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. *ICCV*, pages 2961–2969, 2017. 4

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, pages 770–778, 2016. 5

[19] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, Lei Li, and Jianbo Shi. Foveabox: Beyound anchor-based object detection. *IEEE Trans. Image Proc.*, 29:7389–7398, 2020. 2

[20] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. *ECCV*, pages 734–750, 2018. 2

[21] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv:1711.07264*, 2017. 4

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *ICCV*, pages 2980–2988, 2017. 1, 2, 4, 5, 6, 7, 8

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar. Microsoft coco: Common objects in context. *ECCV*, pages 740–755, 2014. 4

[24] Fanfan Liu, Haoran Wei, Wenzhe Zhao, Guozhen Li, Jingquan Peng, and Zihao Li. Wb-detr: Transformer-based detector without backbone. *ICCV*, pages 2979–2987, 2021. 2

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *ECCV*, pages 21–37, 2016. 1, 2

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, pages 10012–10022, 2021. 7, 8

[27] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid r-cnn. In *CVPR*, pages 7363–7372, 2019. 2

[28] Yuchen Ma, Songtao Liu, Zeming Li, and Jian Sun. Iqdet: Instance-wise quality distribution sampling for object detection. *CVPR*, pages 1717–1725, 2021. 2, 8

[29] Wanli Ouyang, Kun Wang, Xin Zhu, and Xiaogang Wang. Chained cascade network for object detection. *ICCV*, pages 1938–1946, 2017. 2

[30] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. *CVPR*, pages 821–830, 2019. 1, 2, 6, 7, 8

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CVPR*, pages 779–788, 2016. 1, 2

[32] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018. 2

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. on Pat. Anal. and Mach. Intell.*, 39(6):1137–1149, 2017. 1, 2, 4, 5, 7, 8

[34] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. *CVPR*, pages 658–666, 2019. 4

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *arXiv:1409.0575*, 2014. 1

[36] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013. 2

[37] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. *CVPR*, pages 11563–11572, 2020. 1, 2, 4, 8

[38] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse r-cnn: End-to-end object detection with learnable proposals. *CVPR*, pages 14454–14463, 2021. 2, 7, 8

[39] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris Kitani. Rethinking transformer-based set prediction for object detection. *ICCV*, pages 3611–3620, 2021. 1, 2, 8

[40] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. *ICCV*, pages 9627–9636, 2019. 2, 8

[41] Lachlan Tychsen-Smith and Lars Petersson. Denet: Scalable real-time object detection with directed sparse sampling. *ICCV*, pages 428–436, 2017. 2

[42] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and bounded iou loss. *CVPR*, pages 6877–6885, 2018. 4

[43] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization for object detection. *CVPR*, pages 10186–10195, 2020. 1, 2, 4, 6, 7

[44] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 7

[45] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520, 2016. 4, 5

[46] Hongkai Zhang, Hong Chang, Bingpeng Ma, Naiyan Wang, and Xilin Chen. Dynamic r-cnn: Towards high quality object detection via dynamic training. *ECCV*, pages 260–275, 2020. 2, 4, 6, 7, 8

[47] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. *CVPR*, pages 9759–9768, 2020. 2, 8

[48] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. *AAAI*, 34(07):12993–13000, 2020. 4

[49] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019. 2

[50] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. *CVPR*, pages 840–849, 2019. 2

[51] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *ICLR*, 2021. 1, 2, 7, 8