arXiv:2007.08508v1 [cs.CV] 16 Jul 2020

RepPoints v2: Verification Meets Regression for Object Detection

Yihong Chen[*][12], **Zheng Zhang**[1], **Yue Cao**[1], **Liwei Wang**[2], **Stephen Lin**[1], **Han Hu**[1]
[1]Microsoft Research Asia
[2]Peking University
{v-yich,zhez,yuecao,stevelin,hanhu}@microsoft.com
wanglw@cis.pku.edu.cn

Abstract

Verification and regression are two general methodologies for prediction in neural networks. Each has its own strengths: verification can be easier to infer accurately, and regression is more efficient and applicable to continuous target variables. Hence, it is often beneficial to carefully combine them to take advantage of their benefits. In this paper, we take this philosophy to improve state-of-the-art object detection, specifically by RepPoints. Though RepPoints provides high performance, we find that its heavy reliance on regression for object localization leaves room for improvement. We introduce verification tasks into the localization prediction of RepPoints, producing RepPoints v2, which provides consistent improvements of about 2.0 mAP over the original RepPoints on the COCO object detection benchmark using different backbones and training methods. RepPoints v2 also achieves 52.1 mAP on COCO `test-dev` by a single model. Moreover, we show that the proposed approach can more generally elevate other object detection frameworks as well as applications such as instance segmentation. The code is available at https://github.com/Scalsol/RepPointsV2.

1 Introduction

Two common methodologies for neural network prediction are verification and regression. While either can drive network features to fit the final task targets, they each have different strengths. For the object localization problem, verification can be easier to infer because each feature is spatially aligned with the target to be verified. On the other hand, regression is often more efficient and it can also predict continuous target variables that enable subtle localization refinement.

To take advantage of all these benefits, earlier object localization methods [7, 18, 16] combined verification and regression by first performing coarse localization through verifying several anchor box hypotheses, and then refining the localization by regressing box offsets. This combination approach was shown to be effective and led to state-of-the-art performance at the time. However, recent methods based purely on regression, which directly regress the object extent from each feature map position [30, 27, 32], could perform competitively or even better, when comparing a representative regression method, RepPoints, to RetinaNet [16].

In this work, we examine whether pure regression based methods can be enhanced by the inclusion of verification methodology. We observe that verification has proven to be advantageous when used in certain ways. In CornerNet [13], feature map points are verified as a bounding box corner or not, in contrast to verifying anchor boxes for coarse hypothesis localization in RetinaNet [16]. This use of verification leads to significantly better localization performance as shown in Table 1. The

[*]This work is done when Yihong Chen is an intern at Microsoft Research Asia.

Preprint. Under review.

difference may be attributed to corner points representing the exact spatial extent of a ground-truth object box, while an anchor box gives only a coarse hypothesis. In addition, each feature in corner point verification is well aligned to the corresponding point, while in anchor verification, the center feature used for verification lies away from the boundary area.

To elevate the performance of regression-based methods, specifically RepPoints [30], we thus seek to incorporate effective and compatible forms of verification. However, the different granularity of object representations processed by the two methods, i.e., whole objects in RepPoints and object parts (corners) in corner verification, presents an obstacle. To address this issue, we propose to model verification tasks by *auxiliary side-branches* that are added to the major regression branch at only the feature level and result level, without affecting intermediate representations. Through these auxiliary side-branches, verification can be fused with regression to provide the following benefits: better features by multi-task learning, feature enhancement through inclusion of verification cues, and joint inference by both methodologies. The fusion is simple, intuitive, general enough to utilize any kind of verification cue, and does not disrupt the flow of the RepPoints algorithm.

Through different techniques for harnessing verification, the localization and classification ability of RepPoints is substantially improved. The resulting detector, called RepPoints v2, shows consistent improvements of about 2.0 mAP over the original RepPoints on the COCO benchmark with different backbones. It also achieves 52.1 mAP on the COCO object detection `test-dev` set with a single ResNeXt-101-DCN model.

The proposed approach of choosing proper verification tasks and introducing them into a regression framework as *auxiliary* branches is flexible and general. It can be applied to object detection frameworks other than RepPoints, such as FCOS [27]. The additional corner and within-box verification tasks are shown to improve a vanilla FCOS detector by 1.3 mAP on COCO `test-dev` using a ResNet-50 model. This approach can be also applied beyond object detection, such as to instance segmentation by Dense RepPoints [31], where additional contour and mask verification tasks improve performance by 1.3 mAP using a ResNet-50 model on the COCO instance segmentation `test-dev` set, reaching 38.9 mask mAP.

Table 1: Analysis of the performance on COCO `val` set among different methods. "RepPoints*" indicates our improved re-implementation of RepPoints.

Method	methodology	backbone	AP	AP$_{50}$	AP$_{60}$	AP$_{70}$	AP$_{80}$	AP$_{90}$
RetinaNet [16]	ver.+reg.	ResNeXt-101	40.0	60.9	56.4	48.7	35.8	14.6
CornerNet [13]	verification	HG-104	40.6	56.1	52.0	46.8	38.8	23.4
RepPoints* [30]	regression	ResNet-50	39.1	58.8	54.8	48.0	35.5	14.4
RepPoints v2	ver.+reg.	ResNet-50	41.0	59.9	55.9	49.1	37.2	18.5

2 Related Works

Verification based object detection Early deep learning based object detection approaches [26, 24] adopt a multi-scale sliding window mechanism to verify whether each window is an object or not. Corner/extreme point based verification is also proposed [28, 13, 35, 4] where the verification of a 4-d hypothesis is factorized into sub-problems of verifying 2-d corners, such that the hypothesis space is more completely covered. A sub-pixel offset branch is typically included in these methods to predict continuous corner coordinates through regression. However, since this mainly deals with quantization error due to the lower resolution of the feature map compared to the input image, we treat these methods as purely verification based in our taxonomy.

Regression based object detection Achieving object detection by pure regression dates back to YOLO [20] and DenseBox [10], where four box borders are regressed at each feature map position. Though attractive for their simplicity, their accuracy is often limited due to the large displacements of regression targets, the issue of multiple objects located within a feature map bin, and extremely imbalanced positive and negative samples. Recently, after alleviating these issues by a feature pyramid network (FPN) [15] structure along with a focal loss [16], regression-based object detection has regained attention [27, 12, 34, 30], with performance on par or even better than other verification or hybrid methods. Our work advances in this direction, by leveraging verification methodology into regression based detectors without disrupting its flow and largely maintaining the convenience of the original detectors. We mainly base our study on the RepPoints detector, but the method can be generally applied to other regression based detectors.

Hybrid approaches Most detectors are hybrid, for example, those built on anchors or proposals [6, 5, 21, 16, 1, 19]. The verification and regression methodologies are employed in succession, where the anchors and proposals which provide coarse box localization are verified first, and then are refined by regression to produce the detection output. The regression target is usually at a relatively small displacement that can be easily inferred. Our work demonstrates a different hybrid approach, where the verification and regression steps are not run in serial but instead mostly in *parallel* to better combine their strengths. Moreover, this paper utilizes the more accurate corner verification tasks to complement regression based approaches.

3 Verification Meets Regression for Object Detection

3.1 A Brief Review of a Pure Regression Method: RepPoints

RepPoints [30] adopts pure regression to achieve object localization. Starting from a feature map position $\mathbf{p} = (x, y)$, it directly regresses a set of points $\mathcal{R}' = \{\mathbf{p}'_i = (x'_i, y'_i)\}_{i=1}^{n}$ to represent the spatial extent of an object using two successive steps:

$$\mathbf{p}_i = \mathbf{p} + \Delta\mathbf{p}_i = \mathbf{p} + \mathbf{g}_i\left(F_{\mathbf{p}}\right), \quad \mathbf{p}'_i = \mathbf{p}_i + \Delta\mathbf{p}'_i = \mathbf{p}_i + \mathbf{g}'_i(\text{concat}(\{F_{\mathbf{p}_i}\}_{i=1}^{n}))\,, \qquad (1)$$

where $\mathcal{R} = \{\mathbf{p}_i = (x_i, y_i)\}_{i=1}^{n}$ is the intermediate point set representation; $F_{\mathbf{p}}$ denotes the feature vector at position $\mathbf{p}$; $\mathbf{g}_i$ and $\mathbf{g}'_i$ are 2-d regression functions implemented by a linear layer. The bounding boxes of an object are obtained by applying a conversion function $\mathcal{T}$ on the point sets $\mathcal{R}$ and $\mathcal{R}'$, where $\mathcal{T}$ is modeled as the *min-max*, *partial min-max* or *moment* function.

The direct regression in RepPoints [30] makes it a simple framework without anchoring. Though no anchor verification step is employed, it performs no worse than anchor-based detectors, i.e. RetinaNet [16], in localization accuracy as shown in Table 1. Nevertheless, we are motivated by the potential synergy between regression and verification to consider the following questions: What kind of verification tasks can benefit the regression-based RepPoints [30]? Can various verification tasks be conveniently fused into the RepPoints framework without impairing the original detector?

3.2 Verification Tasks

We first discuss a pair of verification tasks that may help regression-based localization methods.

3.2.1 Corner Point Verification

Two corner points, e.g. the top-left corner and bottom-right corner, can determine the spatial extent of a bounding box, providing an alternative to the usual 4-d descriptor consisting of the box's center point and size. This has been used in several bottom-up object detection methods [13, 35, 28], which in general perform worse than other kinds of detectors in classification, but is significantly better in object localization, as seen in Table 1. In later sections, we show that this verification task can complement regression based methods to obtain more accurate object localization.

Corner point verification operates by associating a score to each point in the feature map, indicating its probability of being a corner point. An additional offset is predicted to produce continuous coordinates for corner points, which are initially quantized due to the lower resolution of the feature map compared to the input image, e.g. $8\times$ downsampling. Following the original implementation [13], corner pooling is computed in the head, with a focal loss [16] to train the corner score prediction and a smooth L1 loss for the sub-pixel offset prediction. In label assignment, each feature map point is labeled positive if a ground truth corner point is located within its feature bin, and others are labeled negative. In computing the loss, the negative samples around each ground truth corner are assigned lower weights by an inverse Gaussian function with respect to its distance to the ground-truth corner point. A more detailed description is given in Appendix A.

Different from CornerNet [13], which employs a special backbone architecture with an Hourglass structure and a single-level high resolution feature map ($4\times$ downsampled from the original image), most other recent object detectors adopt an FPN backbone with multi-level feature maps. We adapt the corner verification to utilize multi-level feature maps, e.g. the C3-C7 settings in RepPoints [30].

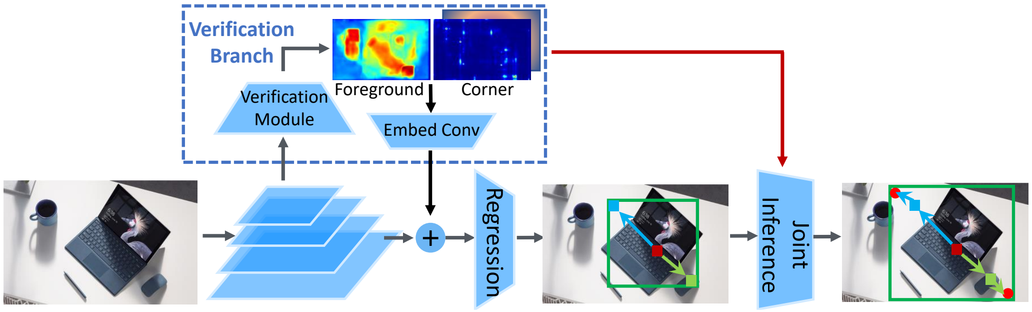

Figure 1: Overview of the general fusion method. The outputs of verification modules (corner and foreground) are incorporated with the input feature to elevate the performance of regression-based object localization, and then joint inference is further employed.

Specifically, all ground truth corner points are assigned to every feature map level, contrary to the usual practice in FPN-based object detection of assignment according to object size. We find that assignment in this manner performs slightly better although it disregards the scale normalization issue, probably due to more positive samples at each level in training. It also performs substantially better than training on a single feature map level of highest resolution, i.e. C3, and then copying/resizing the predicted score/offset map to other levels.

3.2.2 Within-box foreground verification

Another verification task with potential to benefit regression based object detectors is to verify whether a feature map point is located within an object box or not. This within-box foreground verification task provides localization information evenly inside an object box, in contrast to corner points which focus on the box extremes. It is thus not as precise as corner points in describing object bounds, but may benefit object detectors given a coarse localization criterion.

We also differentiate among different object categories by using a non-binary category-aware foreground heatmap. Concretely, for C object categories, there is a C-channel output, with each channel indicating the probability of a feature point being in the corresponding object category. The same as for corner point verification, each ground truth object is assigned to every level of an FPN backbone.

Normalized focal loss. In training, a vanilla focal loss lets larger objects contribute significantly more than smaller objects, resulting in poorly learnt foreground scores for small objects. To address this issue, a normalized focal loss is proposed, which normalizes every positive feature map point by the total number of positive points within the same object box in the feature map. For negative points, the normalized loss uses the number of positive points as the denominator. A more detailed description is given in Appendix A.

3.3 A General Fusion Method

In this section, we incorporate these forms of verification to elevate the performance of regression-based methods. In general, regression-based methods detect objects in a top-down manner where all intermediate representations model the whole object. Since the two verification tasks process object parts, such as a corner or a foreground point, the different granularity of their object representations complicates fusion of the two methodologies.

To address this issue, we propose to model verification tasks by auxiliary side-branches that are fused with the major regression branch in a manner that does not affect its intermediate representations, as illustrated in Figure 1. Fusion occurs only at the feature level and result level. With these auxiliary side-branches, the detector can gain several benefits:

Better features by multi-task learning The auxiliary verification tasks provide richer supervision in learning, yielding stronger features that increase detection accuracy, as shown in Table 4. Note that this multi-task learning is different from that of Mask R-CNN [8]. In Mask R-CNN [8], the bounding box object detection benefits from the object mask prediction task, but it requires extra annotation of the object mask. In contrast, our additional auxiliary tasks are automatically generated from only the object bounding box annotation, allowing them to be applied in scenarios where just bounding box annotations are available.

Feature enhancement for better regression The verification output includes strong cues regarding corner locations and the foreground area, which should benefit the regression task. Since the prediction output of these verification tasks has the same resolution as the feature map used for regression at each FPN level, we directly fuse them by applying a *plus* operator on the original feature map and an embedded feature map produced from the verification output by one 1×1 conv layer. The embedding aims to project any verification output to the same dimension as the original feature map, and is shared across feature map levels. Note that for the verification output, a copy detached from back-propagation is fed into the embedding convolution layer to avoid affecting the learning of that verification task.

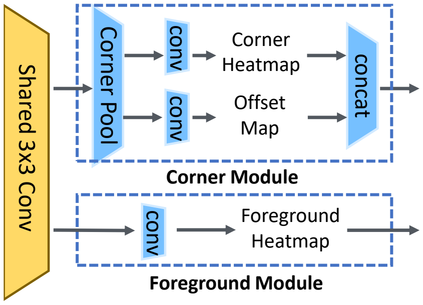

Figure 2: Illustration of the corner module and foreground module.

Joint inference by both methodologies Feature-level fusion implicitly aids object localization. We also explicitly utilize the verification output from corner prediction together with regression-based localization in a joint inference approach that makes use of both of their strengths. Specifically, by corner verification, the sub-pixel corner localization in a small neighborhood is usually more accurate than that by the main regression branch, but is worse at judging whether it is a real corner point since it lacks a whole picture of the object. On the contrary, the main regression branch is better for the latter while worse in accurate sub-pixel localization. To combine their strengths, we refine a corner point $\mathbf{p}_t$ of the bounding box predicted from the main regression branch by

$$\text{refine}(\mathbf{p}^t) = \underset{\left\{\mathbf{q}^t \middle| \|\mathbf{q}^t - \mathbf{p}^t\| \le r\right\}}{\arg\max} \; s(\mathbf{q}^t), \tag{2}$$

where t indicates the corner type (top-left or bottom-right); $\mathbf{q}^t$ is a sub-pixel corner point produced by a corner prediction branch at a feature map position; $s(\mathbf{q}^t)$ is the verification score; r is the neighborhood threshold, set to 1 by default. Note that this result-level fusion is designed for the corner verification task only.

This fusion method is flexible and general, utilizing any kind of verification cue, as it avoids interaction with the intermediate representations in the main branch, and thus has few requirements on the types of verification target. It also does not disrupt the overall flow of the main branch and largely maintains the convenience of the original detector built on the main branch.

3.4 RepPoints v2: Fusing Verification into RepPoints

RepPoints is a pure regression based object detector. We now complement it with verification tasks of different forms, specifically for corners and within-box foreground. To increase the compatibility of RepPoints with the auxiliary verification tasks, we first make a small modification to it, such that the first two points explicitly represent the top-left and bottom-right corner points. We refer to this as the *explicit-corners* variant. These corner points replace the conversion function used in the original RepPoints, so that the bounding box is defined by these corner points instead of by a *min-max* or *momentum* operation on the point set. With the corner points, the conversion function becomes

$$\mathcal{T}(\mathcal{R}) = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, x_2 - x_1, y_2 - y_1\right). \tag{3}$$

where the four numbers denote x-center, y-center, width and height, respectively. To this *explicit-corners* variant of RepPoints, we add the auxiliary side-branches for verification. Specifically, we take the feature map right after the 3rd conv layer of the localization head as input, to reuse the existing head for computational savings. As illustrated in Figure 2, a 3×3 convolutional layer is applied on this feature map, followed by two small sub-networks for the two verification tasks. The corner sub-network consists of a corner pooling layer [13] followed by a 1×1 conv layer to predict heatmap scores and sub-pixel offsets. The foreground sub-network is a single 1×1 conv layer to predict the foreground score heatmap. In training, we adopt a multi-task loss:

$$L = L_{\text{RepPoints}} + \lambda_1 L_{\text{corner}} + \lambda_2 L_{\text{foreground}}, \tag{4}$$

with loss weights $\lambda_1 = 0.25$ and $\lambda_2 = 0.1$. More details are given in Appendix A.

Customizing the general fusion method of Section 3.3 to RepPoints, we use corner verification for multi-task learning, feature enhancement and joint inference. Foreground verification is instead used only for multi-task learning and feature enhancement. The resulting detector is named RepPoints v2.

3.5 Extension to Other Detectors and Problems

The fusion method used for RepPoints can also improve other detectors such as FCOS [27]. As FCOS shares similar classification and localization heads as in RepPoints, the fusion of RepPoints v2 can directly be applied to it. Concretely, the corner and foreground verification heads are applied on the feature map after the 3$^{\text{rd}}$ conv layer. The verification output maps are fused into the main branch, and the final regression results are obtained by the joint inference described in Section 3.3.

The fusion method can also be extended to other tasks such as instance segmentation by Dense RepPoints [31], a regression-based method. Since there is an additional object mask annotation, more fine-grained verification formats can be used, such as object contour verification and category-aware semantic segmentation. As shown in Table 8, the additional verification methodology brings 1.3 mask AP gains to Dense RepPoints on the COCO `test-dev` set. More details are presented in Appendix B.

4 Experiments

We conduct experiments on the challenging MS COCO 2017 benchmark [14], which is split into `train`, `val` and `test-dev` sets with 115K, 5K and 20K images, respectively. We train all the models using the `train` set and conduct an ablation study on the `val` set. A system-level comparison to other methods is reported on the `test-dev` set.

4.1 Implementation Details

We use the mmdetection codebase [2] for experiments. All experiments perform training with an SGD optimizer on 8 GPUs with 2 images per GPU, using an initial learning rate of 0.01, a weight decay of 0.0001 and momentum of 0.9. In ablations, most experiments follow the $1\times$ settings where 12 epochs with single-scale training of $[800, 1333]$ are used, with learning rate decayed by $10\times$ after epoch 8 and 11. Most of the ablations use a ResNet-50 [9] backbone pretrained on ImageNet [23]. We also test our approach using multi-scale ($[480, 960]$) and longer training ($2\times$ settings with 24 epochs in total and the learning rate decayed at epoch 16 and 22) on stronger backbones to see whether the gains by the proposed approaches hold on these stronger baselines.

In inference, unless otherwise specified, we adopt a single-scale test approach with the image size the same as in single-scale training. We also conduct multi-scale testing on the strongest backbone for comparison with the previous state-of-the-art approaches. An IoU threshold of 0.6 is applied for Non-Maximum Suppression (NMS) to remove duplicate boxes.

For RepPoints [30], we use an improved implementation by replacing the IoU assigner with an ATSS assigner [32], yielding 39.1 mAP on COCO `val` using a ResNet-50 model and the $1\times$ settings, 0.9 mAP higher than that reported in the original paper.

4.2 Ablation Study

Explicit-corners variant. We first validate the effectiveness of the *explicit-corners* variant of RepPoints described in Section 3.4, as shown in Table 2. This variant performs on par with the three variants used in the original RepPoints, but performs 0.2-0.3 mAP better than other variants when the verification module is added. This could be contributed to more effective interaction between verification and regression tasks in this *explicit-corners* variant.

Table 2: Performance of the *explicit-corners* variant of RepPoints.

variant	+verification	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
min-max		39.1	58.8	42.4	22.4	42.8	50.5
	✓	40.7	59.8	43.7	23.3	44.4	54.0
partial min-max		39.0	58.7	42.4	21.8	42.5	50.7
	✓	40.7	59.7	43.6	23.1	44.4	54.0
momentum		39.1	58.9	42.2	22.3	42.6	50.8
	✓	40.8	59.7	43.7	23.5	44.7	53.9
explicit-corners		39.1	58.8	42.5	22.4	42.6	50.6
	✓	41.0	59.9	43.9	23.8	44.8	54.0

Table 3: Ablations on two forms of verification.

corner	foreground	AP	AP_{50}	AP_{75}	AP_{90}	AP_S	AP_M	AP_L
		39.1	58.8	42.5	14.4	22.4	42.6	50.6
✓		40.5	59.0	43.5	18.4	23.4	44.1	53.5
✓	✓	41.0	59.9	43.9	18.5	23.8	44.8	54.0

Forms of verification Table 3 ablates the two forms of verification. The corner verification task alone brings 1.4 mAP gains over the RepPoints baseline. The benefits are mainly for higher IoU criteria, e.g. AP_{90} is improved by 4.0 mAP while AP_{50} increases by only 0.2 mAP. The additional foreground verification task brings another 0.5 mAP in gains, but mainly on lower IoU criteria, for example, AP_{50} is improved by 0.9 AP while AP_{90} remains about the same.

Table 4: Ablations on three types of fusion.

multi-task	enhance feature	joint inference	AP	AP_{50}	AP_{75}	AP_{90}	AP_S	AP_M	AP_L
			39.1	58.8	42.5	14.4	22.4	42.6	50.6
✓			39.5	58.9	42.7	14.6	22.5	43.1	51.0
✓	✓		40.2	60.0	43.5	15.7	24.1	43.8	52.5
✓	✓	✓	41.0	59.9	43.9	18.5	23.8	44.8	54.0

Types of fusion Table 4 ablates the types of fusion, specifically multi-task learning, feature enhancement for regression, and joint inference. Multi-task learning brings a 0.4 mAP gain over the RepPoints baseline. Note that this multi-task learning does not rely on annotations beyond bounding boxes, in contrast to that in Mask R-CNN [8]. The additional feature enhancement operation brings another 0.7 gain. The explicit fusion by joint inference brings increases mAP by 0.8, such that the full approach surpasses its counterpart without verification modules by 1.9 mAP.

Complexity analysis. Our approach involves slightly more parameters (38.3M vs 37.0M) and marginally more computation (244.2G vs 211.0G) than the original RepPoints. This overhead mainly occurs at the additional heads to produce verification score/offset maps. We also conduct RepPoints with heavier computation, by adding one more convolutional layers on the heads, resulting in a baseline with similar parameters and computations as our approach (38M/235.8G v.s. 38.3M/244.2G). The enhanced baseline model performs 0.2 mAP better than the vanilla RepPoints, indicating that the improvements by our approach are mostly not due to more parameters and computation.

Stronger baselines. We further validate our method on stronger RepPoints baselines, using longer/multi-scale training (2× settings) and stronger backbones, as shown in Table 5. It can be seen that the gains are well maintained on these stronger RepPoints baselines, at about 2.0 mAP. This indicates that the proposed approach is largely complementary to improved baseline architecture, in contrast to many techniques that have exhibited decreasing gains with respect to stronger baselines.

Visualization. The visualization results are given in Appendix C.

4.3 Comparison to State-of-the-art Methods

We compare the proposed method to other state-of-the-art object detectors on the COCO2017 `test-dev` set, as shown in Table 6. We use GIoU [22] loss instead of smooth-l1 loss in the regression branch here. With ResNet-101 as the backbone, our method achieves 46.0 mAP without bells and whistles. By using stronger ResNeXt-101 [29] and DCN [3] models, the accuracy rises to 49.4 mAP. With additional multi-scale tests as in [32], the proposed method achieves 52.1 mAP.

Table 5: Experiments on RepPoints baselines with stronger backbones using 2× settings (24 epochs) and multi-scale training ($[480, 960]$) on COCO val set.

backbone	+verification	AP	AP_{50}	AP_{60}	AP_{70}	AP_{80}	AP_{90}
ResNet-50		41.8	61.8	58.1	51.1	38.6	15.9
	✓	43.9	63.1	59.3	52.5	40.1	20.6
ResNet-101		43.4	63.3	59.4	53.0	40.4	18.0
	✓	45.5	64.5	60.6	54.1	42.3	22.2
ResNeXt-101		45.5	65.9	62.1	55.2	42.4	19.7
	✓	47.3	66.9	62.9	56.1	44.0	23.7

Table 6: Comparison of RepPoints v2 to state-of-the-art detectors on COCO `test-dev`. * denotes that the number is obtained by multi-scale testing.

method	backbone	epoch	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet [16]	ResNet-101	18	39.1	59.1	42.3	21.8	42.7	50.2
FCOS [27]	ResNeXt-101	24	43.2	62.8	46.6	26.5	46.2	53.3
DCN V2* [36]	ResNet-101+DCN	18	46.0	67.9	50.8	27.8	49.1	59.5
RepPoints* [30]	ResNet-101+DCN	24	46.5	67.4	50.9	30.3	49.7	57.1
MAL* [11]	ResNeXt-101	24	47.0	66.1	51.2	30.2	50.1	58.9
FreeAnchor* [33]	ResNeXt-101	24	47.3	66.3	51.5	30.6	50.4	59.0
ATSS* [32]	ResNeXt-101+DCN	24	50.7	68.9	56.3	33.2	52.9	62.4
TSD* [25]	SENet154+DCN	24	51.2	71.9	56.0	33.8	54.8	64.2
CornerNet [13]	HG-104	100	40.5	56.5	43.1	19.4	42.7	53.9
ExtremeNet [35]	HG-104	100	40.2	55.5	43.2	20.4	43.2	53.1
CenterNet [4]	HG-104	100	44.9	62.4	48.1	25.6	47.4	57.4
RepPoints v2	ResNet-50	24	44.4	63.5	47.7	26.6	47	54.6
RepPoints v2	ResNet-101	24	46.0	65.3	49.5	27.4	48.9	57.3
RepPoints v2	ResNeXt-101	24	47.8	67.3	51.7	29.3	50.7	59.5
RepPoints v2	ResNet-101+DCN	24	48.1	67.5	51.8	28.7	50.9	60.8
RepPoints v2	ResNeXt-101+DCN	24	49.4	68.9	53.4	30.3	52.1	62.3
RepPoints v2*	ResNeXt-101+DCN	24	**52.1**	70.1	57.5	34.5	54.6	63.6

4.4 Extension to Other Detectors and Applications

Direct application to FCOS FCOS [27] is another popular regression based object detector. We directly apply our approach without modification to this detector, and 1.3 mAP improvements are obtained as shown in Table 7, indicating generality of the proposed approach.

Extension to instance segmentation Table 8 shows the effect of additional verification modules in the instance segmentation method of Dense RepPoints [31]. The additional contour and foreground modules improve accuracy by 1.3 mAP, demonstrating the broad applicability of the fusion method.

5 Conclusion

In this paper, we propose *RepPoints v2*, which enhances the original regression-based *RepPoints* by fusing verification tasks in various ways. A new variant of RepPoints is proposed to increase the compatibility with the auxiliary verification tasks. The resulting object detector shows consistent improvements over the original RepPoints under different backbones and training approaches. It also achieves 52.1 mAP on the COCO `test-dev`. Moreover, this approach could be easily transferred to other detectors and the instance segmentation domain, boosting the performance of the base detector/segmenter by a considerable margin.

Table 7: Applying the verification module to FCOS, which is implemented in mmdetection.

	backbone	AP	AP_{50}	AP_{75}	AP_{90}	AP_S	AP_M	AP_L
FCOS	ResNet-50	38.2	57.1	41.2	15.3	22.2	42.3	49.5
+verification	ResNet-50	39.5	57.7	41.9	18.4	22.3	43.2	52.7

Table 8: Adding the verification module to the instance segmentation algorithm Dense RepPoints on COCO `test-dev`.

	backbone	AP_{mask}	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Dense RepPoints	ResNet-50	37.6	60.4	40.2	20.9	40.5	48.6
+verification	ResNet-50	38.9	61.5	41.9	21.2	42.0	51.1

Broader Impact

Since this work is about designing better object detectors, researchers and engineers engaged in object detection and instance segmentation on natural images, medical images and even video data may benefit from this paper. If there is any failure in this system, the model may not detect objects correctly. Similar to most object detectors, the detection results may not be interpretable, thus it is hard to predict failure scenarios. This object detector also leverages biases in the dataset used for training, and may incur a performance drop on datasets which have a domain gap with the training dataset.

References

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

[2] 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.

[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017.

[4] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *ICCV*, 2019.

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

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

[7] Ross B. Girshick. Fast R-CNN. In *ICCV*, pages 1440–1448, 2015.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017.

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

[10] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[11] Wei Ke, Tianliang Zhang, Zeyi Huang, Qixiang Ye, Jianzhuang Liu, and Dong Huang. Multiple anchor learning for visual object detection. In *CVPR*, pages 7363–7372, 2020.

[12] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, and Jianbo Shi. Foveabox: Beyond anchor-based object detector. *arxiv*, abs/1904.03797, 2019.

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

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, pages 740–755, 2014.

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *ICCV*, pages 2117–2125, 2017.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

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

[19] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid R-CNN. In *CVPR*, pages 7363–7372, 2019.

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

[21] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015.

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

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[24] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[25] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *CVPR*, 2020.

[26] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *NeurIPS*, pages 2553–2561, 2013.

[27] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *ICCV*, 2019.

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

[29] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995, 2017.

[30] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *ICCV*, pages 9656–9665, 2019.

[31] Ze Yang, Yinghao Xu, Han Xue, Zheng Zhang, Raquel Urtasun, Liwei Wang, Stephen Lin, and Han Hu. Dense reppoints: Representing visual objects with dense point sets. *arxiv*, abs/1912.11473, 2019.

[32] 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. *arXiv preprint arXiv:1912.02424*, 2019.

[33] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. Freeanchor: Learning to match anchors for visual object detection. In *NeurIPS*, pages 147–155, 2019.

[34] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arxiv*, abs/1904.07850, 2019.

[35] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019.

[36] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets V2: more deformable, better results. In *CVPR*, pages 9308–9316, 2019.

A Details of Verification Tasks

A.1 Corner Point Verification

Ground-truth assignment. We follow CornerNet [13] to assign ground-truth corners. For each corner, only the corner itself is positive location, and all other locations are negative. Moreover, the penalty given to negative locations within a radius of the positive location is reduced. Specifically, for a given corner point $p = (p_x,\ p_y)$ on original image, the size of the ground-truth heatmap Y with $s\times$ downsampled rate is $\frac{H}{s} \times \frac{W}{s}$, and the corresponding location of p on Y is $\hat{p} = \left\lfloor \frac{p}{s} \right\rfloor$. The penalty weight of negative locations is defined as a inverse Gaussian function:

$$Y_{xy} = \exp\left(-\frac{(x-\hat{p_x})^2 + (y-\hat{p_y})^2}{2\sigma_p^2}\right) \tag{5}$$

where σ_p is an object size-adaptive standard deviation, x and y indicate the location of a negative point. Note that for different positive points, the penalty weight of a negative point may be different. Therefore, the largest one as the penalty weight of the negative point.

For additional offset prediction, we follow [13] that only supervises the positive locations. For a given corner point p and its corresponding downsampled location $\hat{p}$, the training target is:

$$o(\hat{p}) = \left(\frac{p_x}{s} - \left\lfloor \frac{p_x}{s} \right\rfloor,\ \frac{p_y}{s} - \left\lfloor \frac{p_y}{s} \right\rfloor\right) \tag{6}$$

Loss. Follow CornerNet [13], we use a modified focal loss [16] to learn the corner heatmap. The loss is defined as

$$L_{\text{heatmap}} = \frac{-1}{N}\sum_{i=1}^{H}\sum_{j=1}^{W}\begin{cases} (1-p_{ij})^{\alpha}\log(p_{ij}) & \text{if } y_{ij} = 1 \\ (1-y_{ij})^{\beta}(p_{ij})^{\alpha}\log(1-p_{ij}) & \text{otherwise} \end{cases} \tag{7}$$

where N is the number of objects in an image, p_{ij} and y_{ij} are the score and label at location (i, j) in the predicted heatmap. We set $\alpha = 2$ and $\beta = 4$, following [13].

In addition, the loss to learn offset are defined as:

$$L_{\text{offset}} = \frac{1}{N}\sum_{k=1}^{N}\text{SmoothL1Loss}\left(o(\hat{p}_k), \hat{o}(\hat{p}_k)\right) \tag{8}$$

where o is the groundtruth offset, $\hat{o}$ is the predicted offset, $\hat{p}_k$ is the k-th corner point. Finally, the overall loss L_{corner} of corner branch is simply defined as the summation of L_{heatmap} and L_{offset}.

A.2 Within-box Foreground Verification

Normalized focal loss. The normalized focal loss is defined as:

$$\mathcal{L}_{\text{fg}} = \sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}\begin{cases} \frac{-1}{N_W} w_{cij}\cdot\alpha\,(1-p_{cij})^{\gamma}\log(p_{cij}) & \text{if } y_{cij} = 1 \\ \frac{-1}{N}(1-\alpha)\,(p_{cij})^{\gamma}\log(1-p_{cij}) & \text{otherwise} \end{cases} \tag{9}$$

where y_{cij} is the value on the ground-truth foreground heatmap, p_{cij} is the c-th category score at location (i, j) of the predicted heatmap, w_{cij} is the normalizing factor, which is defined as:

$$w_{cij} = \begin{cases} \frac{1}{S_{cij}} & \text{if } y_{cij} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where S_{cij} is the area of the object that (i, j) lies in. If multiple objects of the same category collide at the same location, we would take the smallest size. N_W is defined as $\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W} w_{cij}$, the sum of normalizing factor at all locations. N is the number of positive points. α and γ is set as 0.25, 2, respectively.

Table 9: Adding the verification module to the instance segmentation algorithm Dense RepPoints on COCO `test-dev`.

	backbone	AP_{mask}	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Dense RepPoints	ResNet-50	37.6	60.4	40.2	20.9	40.5	48.6
+contour&fg	ResNet-50	38.6	61.4	41.7	21.3	41.8	50.8
+joint inference	ResNet-50	38.9	61.5	41.9	21.2	42.0	51.1

A.3 Overall Loss

The overall loss is defined as:

$$L = L_{\text{RepPoints}} + \lambda_1 L_{\text{corner}} + \lambda_2 L_{\text{fg}}, \tag{11}$$

and $\lambda_1 = 0.25$ and $\lambda_2 = 0.1$. L_{corner} and L_{fg} are defined above.

B Extension to Instance Segmentation

Training settings. We based on Dense RepPoints [31] to validate the effectiveness of our method, due to the Dense RepPoints is the state-of-the-arts regression-based instance segmentation approach. Because the contour points has no type, only one heatmap is used for predicting all contour points. Other parameters, network architectures and training details are same as object detection.

Joint inference. With only a few modifications, joint inference can also be used for instance segmentation. For a predicted representative point, if it is close to the contour point, then we refine the predicted representative point set by adding the adjacent contour point into the set. More specifically, if the score of representative point in the contour heatmap is greater than 0.5, then the point with the highest contour score among all the points with a distance less than 1 are added to the set.

Experimental results. The results is given in Table 9. ResNet-50 backbone and 3x scheduler are adopted. By adding verification module, the performance are elevated by 1.0 mAP, further applying the joint inference, additional 0.3 mAP is improved. This demonstrates the flexibility of our proposed method.

C Visualization

Figure 3 shows some object detection results comparison on COCO 2017 [17] between *RepPoints v1* [30] and *RepPoints v2*. Both methods adopt ResNet-50 backbone and 1x scheduler. As can be seen, compared to *RepPoints v1*, *RepPoints v2* could provide us more precise localization results.

Figure 4 gives the visualization of main component of *RepPoints v2*. From left to right are set of representative points predicted, foreground prediction, top-left corner prediction and bottom-right corner prediction. As can be seen, all components could provide informative cues, leading to better performance.

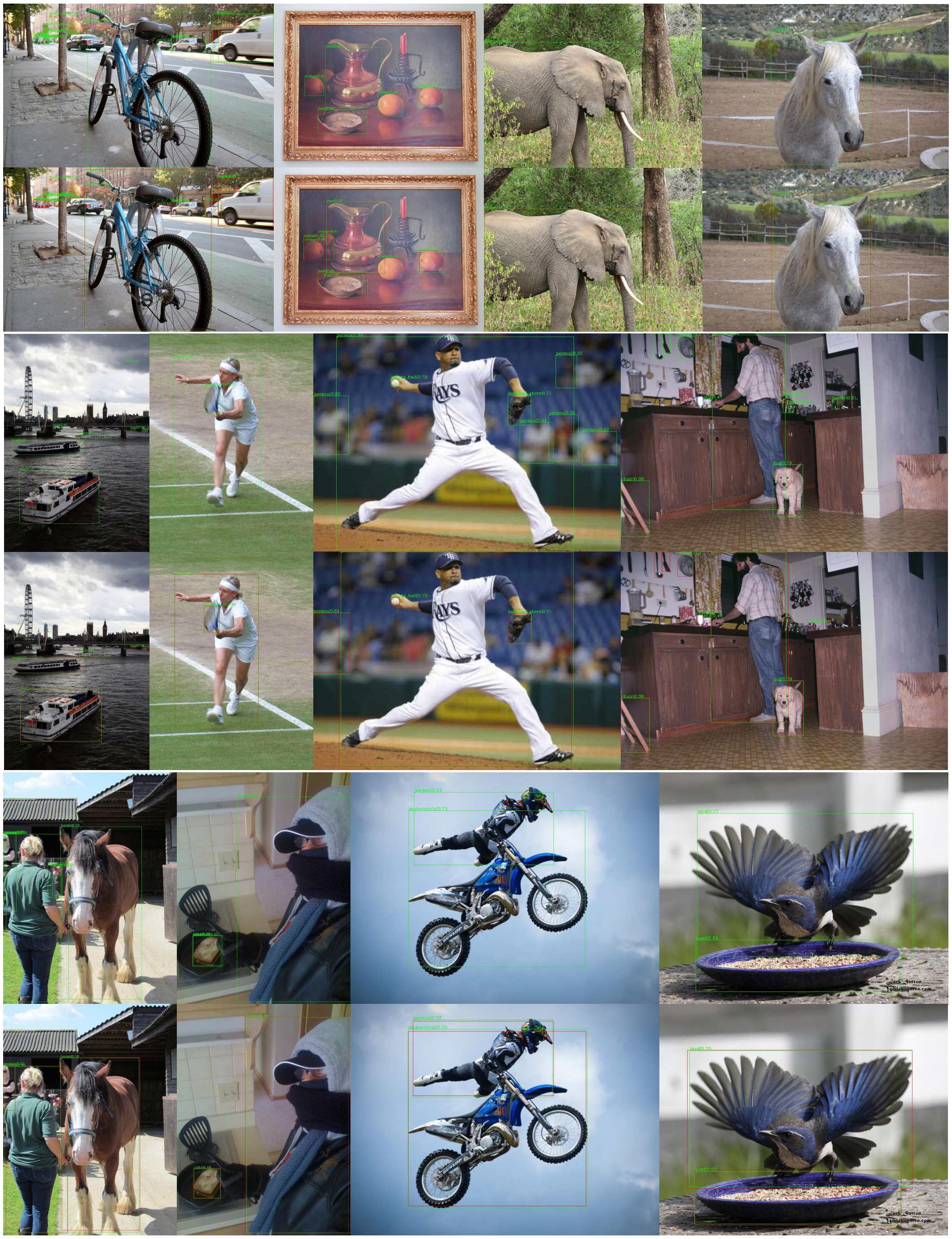

Figure 3: Visualization results of *RepPoints v1* and *RepPoints v2*. image on the top row is the detection of *RepPoints v1* and the bottom row is for *RepPoints v2*. The red boxes are generated without joint inference while green boxes adopts joint inference. As can be seen, our full version of *RepPoints v2* could achieve better localization results.

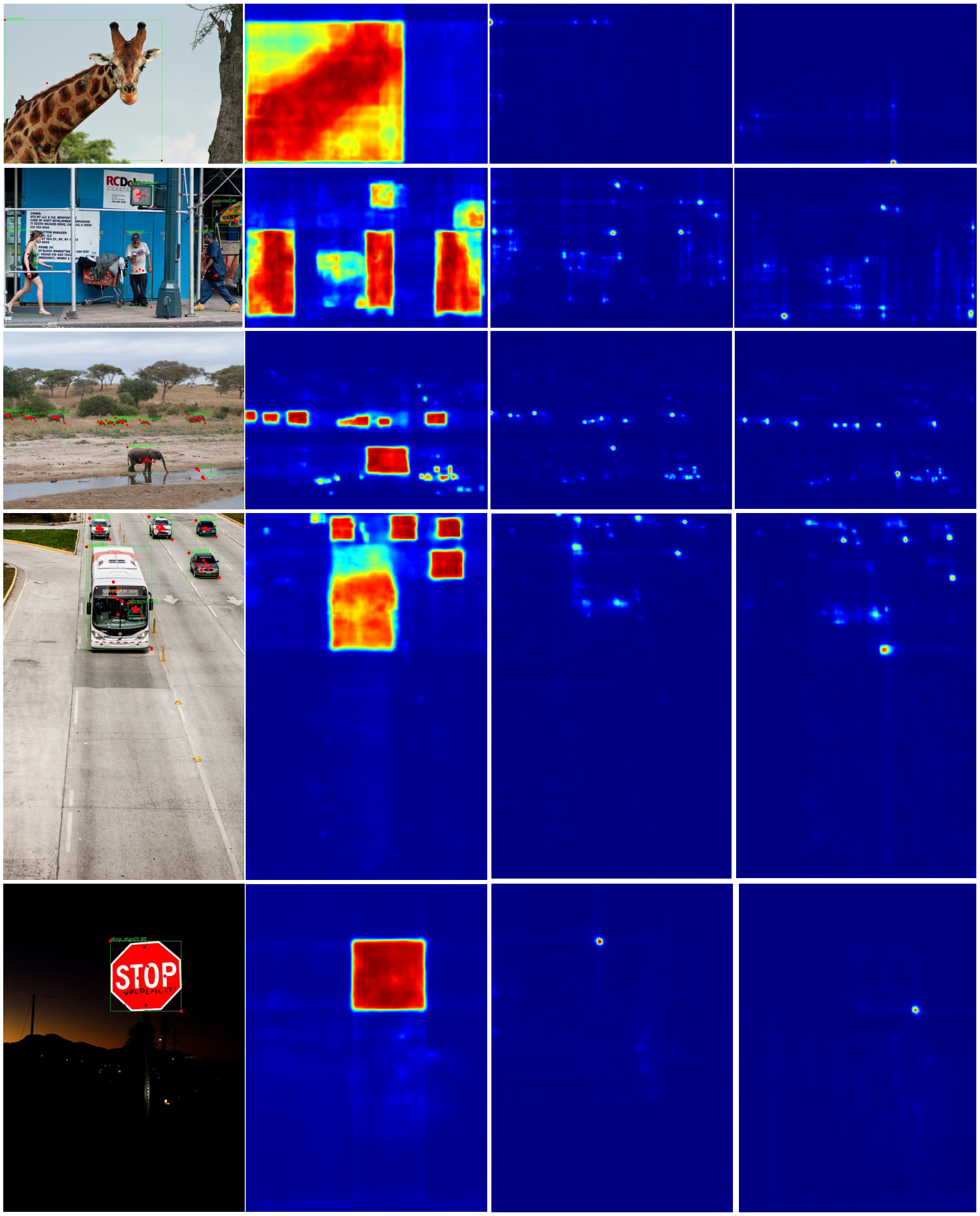

Figure 4: Visualization of main component of *RepPoints v2*. From left to right are set of representative points predicted, foreground prediction, top-left corner prediction and bottom-right corner prediction.