2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Zero-shot Object Detection Based on Dynamic Semantic Vectors

Haoyu Li[1,2], Jilin Mei[1,3], Jiancong Zhou[1,2] and Yu Hu[1,3,4,*]

Abstract— **Zero-shot object detection has shown its ability to overcome the problems of data scarcity and novel classes. Existing methods generally utilize static semantic vectors to classify objects and guide the network to map visual features to semantic vectors. However, the distribution of semantic vectors cannot adequately represent visual features, which makes migration from seen to unseen classes difficult. This work explores the dynamic semantic vector method to align the distributions of semantic vectors and visual features. The main challenge is to get a more reasonable distribution of semantic vectors. To address this issue, we proposed a two-way classification branch network and introduce N-pair loss into the dynamic semantic vector optimization process. Experiments on the MS-COCO dataset and SiTi (a real-world autonomous driving dataset collected by us) demonstrate the effectiveness and generalization of our method. Our code is available at https://github.com/HaoyuLizju/ZSD_tcb**

I. INTRODUCTION

Object detection methods based on deep learning have practical applications in autonomous driving [1]. In spite of their surprising performance, these methods require a large amount of data for training. Thus, data scarcity and novel classes (i.e., those not seen in the training phase) still hinder their application in real-world scenarios [2]. In order to overcome the problems listed above, zero-shot object detection has been proposed and has attracted extensive research attention [3], [4], [5].

In zero-shot object detection, a model is trained using only seen classes, then it needs to locate and classify unseen classes during inference. Unseen classes are not included in model training but appear during model inference [4]. Currently, most zero-shot object detection methods are based on Faster-RCNN [4], [6], [7], [8], [9], [10], [11], [12]. In these methods, Faster-RCNN's bounding box regression network is assumed to be independent of category, which means it can be applied directly to unseen classes without requiring parameter adjustment. Thus, the primary issue with these methods is how to make Faster-RCNN's classification network capable of identifying unseen classes.

In previous methods [4], [6], [7], [8], [9], semantic information is used to construct semantic embedding space for all categories and complete the detection of unseen classes by mapping visual features to the semantic space. Semantic vectors are used as constant cluster centers in the classification network.

From the perspective of information sources, there are differences between semantic vectors and visual features. Semantic vectors are usually learned from a large amount of corpus through natural language processing models such as FastText [13], word2vec [14], Glove [15], etc., while visual features are directly extracted from object images using convolutional neural networks. Therefore, semantic vectors cannot adequately represent visual features. Due to the inconsistent distribution between semantic and visual spaces, directly using semantic vectors for classification will decrease the accuracy of both seen and unseen classes.

In order to solve this problem, we propose a two-way classification branch network (TCB). In our method, semantic vectors are used as model parameters and adjusted according to the visual features in the training process, so that the distributions of semantic vectors and visual features in their respective space tend to be consistent. The original semantic vectors contain more semantic knowledge, whereas the optimized semantic vectors contain more visual knowledge, so we adopt the idea of model ensemble and design the TCB.

The two branches generate classification results respectively, then a max function is used to integrate the classification results to obtain the final one. More details are shown in Fig. 1. The main contributions of our work are: 1) we consider semantic vectors as parameters and introduce N-pair loss for semantic vector training and improve classification accuracy; 2) experiments on MS-COCO dataset demonstrate the effectiveness of our method; and 3) experiments on our in-house autonomous driving dataset SiTi demonstrate the generalization of our method.

II. RELATED WORK

According to the different model architectures, zero-shot object detection methods can be divided into generative methods and projection methods.

A. Generative methods

Generative methods [16], [17], [18], [19], [20], [21] synthesize visual features of unseen classes using generative adversarial neural networks (GANs) [22], [23], and the classifier in the object detection network is trained using the simulated visual features. These methods are trained in four steps: 1) train a detection model with the seen classes data, then extract the visual features and the corresponding

†This work was supported by Key Research Project of Zhejiang Lab (No. 2022PC0AC01) and National Natural Science Foundation of China under Grant No. 62203424 and No.62176250.

[1]Research Center for Advanced Computing Systems, Zhejiang Laboratory, Hangzhou 311100, China.
[2]Hangzhou Institute for Advanced Study, UCAS, Hangzhou 310024, China.
[3]Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China.
[4]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, 100049, China.
*Correspondence: Yu Hu, huyu@ict.ac.cn

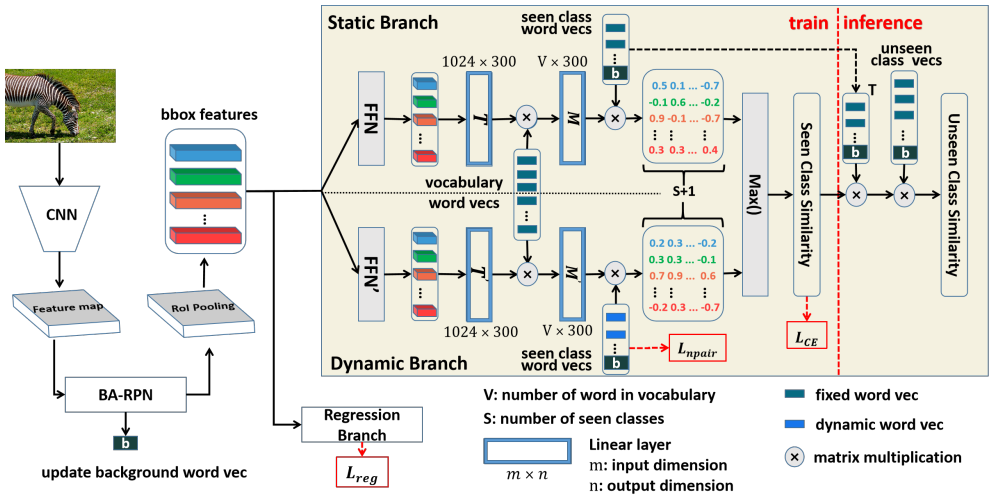

Fig. 1. An overview of our proposed approach. The decoder used to enhance the network mapping capability is omitted from the figure.

categories of the candidate boxes; 2) utilize the visual features and the corresponding category semantics to train the adversarial neural network; 3) generate unseen classes visual features using the generator of GAN; 4) train the classifier of detection model using the simulated unseen classes visual features. After these steps, the detection model has the ability to detect unseen class objects. The quality of generated visual features can be improved by adding additional loss items. Hayat et al. [17] added model seeking losses to diversify the generated visual features to avoid model collapse. Huang et al. [20] propose intra-class semantic diverging loss and inter-class structure preserving loss to make visual features diverse and easy to distinguish between different categories. Generative methods require additional GANs, which improve accuracy but make the training process cumbersome and unstable [24], [25].

B. Projection methods

The projection methods [4], [6], [7], [8], [9], [10], [11], [12], [26] map the visual features and semantic vectors of objects into the same space for classification. Based on object detection models such as YOLO [27] and Faster-RCNN [28], these methods map the extracted visual features of objects to the semantic space through a neural network and classify objects according to the similarity between the mapped visual features and the semantic vectors. As a result of the above process, it is possible to make the model more capable of recognizing unseen classes by designing additional loss items or network structures. Zheng et al. [6] propose the BA-RPN structure, which generates the background class semantic vector according to the background visual features and replaces the static background class vector in the original semantic vector. Li et al. [9] replaces the entropy loss with SoftPlus Margin focal loss in the classification branch of the Faster RCNN, and incorporates semantic information into the object bounding box regression branch. Yan et al. [12] introduce the contrast learning method, which uses seen class data to optimize parameters related to unseen classes. Projection methods are easy to implement since they simply add semantic structure to object detection models to handle semantic information. Our method is also in this branch, and it focuses on how to enhance the semantic vectors.

The above two branches of zero-shot object detection take the semantic vectors of categories as constants, which are not changing during training. These methods ignore the discrepancy of information between semantic vectors and visual features. We propose a method that updates the semantic vectors of seen classes according to the visual features of seen classes during training. As a result, it is able to obtain more reasonable semantic vector distributions, thus improving its accuracy in classifying unseen classes.

III. Methodology

A. Problem Definition

Let X denote the whole dataset and $Y = Y_s \cup Y_u$ denote the label of object in X. The label of seen class is represented by Y_s, with $Y_s = \{1, 2..., S\}$, S is the number of visible classes; Unseen class labels are denoted by Y_u, with $Y_u = \{S+1, S+2, ..., S+U\}$, U is the number of unseen classes. The train set X_s is a subset of X that contains only Y_s category objects, and the test set X_u is a subset of X that contains Y_u category objects. In the train process, the train set X_s, the label Y_s and the corresponding ground truth bounding box $B \in R^4$ are provided to the model. In addition, the semantic vectors W about Y are also provided to the model, where $W = \{W_s, W_u\}$, W_s are the word vectors corresponding to the seen classes, W_u are the word vectors corresponding to the unseen classes. $W_s \in R^{d \times S}$,$W_u \in R^{d \times U}$, and d is the semantic vectors' dimension. In the inference process, we have different problem settings. In the Zero-Shot Object Detection (ZSD), the model only needs to identify the unseen class objects in the X_u, while in Generalized Zero-Shot Object Detection (GZSD), the model needs to identify not only the unseen class class objects in the X_u, but also the seen class objects in the X_s.

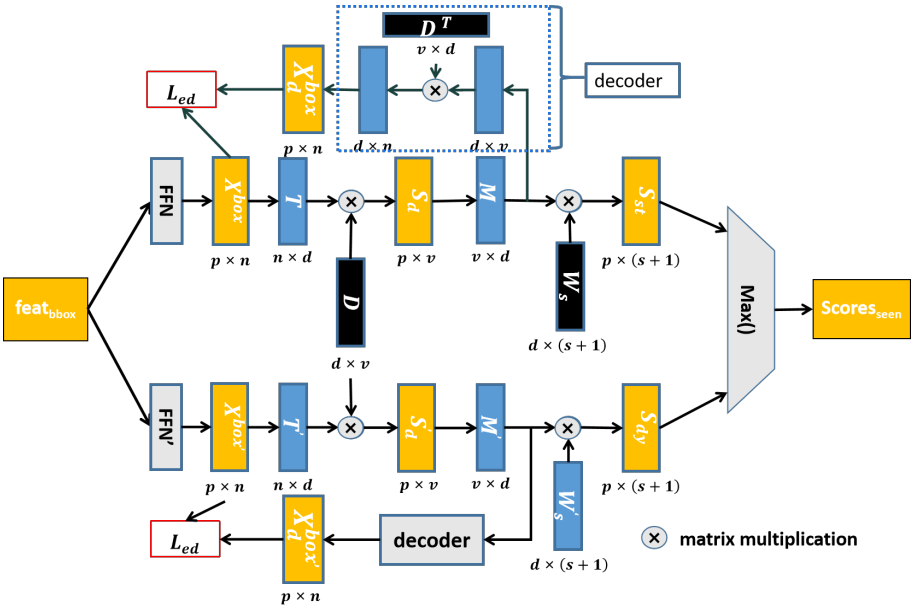

Fig. 2. The details of TCB. The upper is static semantic vector branch, and the lower is dynamic semantic vector branch. W_s and $W_s^{'}$: (static and dynamic) seen class word vectors; D: the additional vocabulary word vectors; p: the number of proposals; n: the dimension of compressed bbox feature; d: the dimension of word vector; v: the number of the word in D; s: the number of seen classes. Yellow blocks represent data; blue blocks for parameters; black blocks for constants. The decoder is the logical inverse function of the mapping network T and M, and it projects the feature from semantic space back to visual space to improve the mapping ability of T and M.

B. Model Architecture

We propose a two-way classification branch network based on the object detection part of [11]. The structure of the model and the detection process is shown in Fig. 1. The model is based on Faster-RCNN architecture. The backbone of the model is a Resnet101 [29] pre-trained on Imagenet. For RPN, we use Background Aware Region Proposal Network (BA-RPN) [11] to generate the semantic vector of the background while generating the candidate box. For the image input $I \in X_s$ in the training process, the feature is extracted by ResNet while the candidate box $B_s \in R^4$ and the semantic vector $W_{bg} \in R^d$ of the background category are generated by BA-RPN. The candidate box features f_S with fixed size is obtained by the RoI layer. The candidate box feature f_S is fed into the regression network and the classification network respectively to get the coordinates and classification results of the candidate box. In the inference process, the model uses the parameters learned on the seen class data to detect the unseen class objects.

C. Two-way Classification Branch Network

Our proposed two-way classification branch network (TCB) consists of a static semantic vector branch and a dynamic semantic vector branch. The semantic vectors in static branch are constants and don't change during training, while the semantic vectors in dynamic branch are variables and changed during training on the seen classes. The final classification result for the seen object is obtained by filtering the prediction results of the two classification branches using a max function. The structure of TCB is shown in Fig. 2. The basic structure of each branch network comes from [11], and the specific structure will be elaborated below.

1) Static Semantic Vector Branch: The static semantic vector branch consists of five parts: candidate bounding box feature compression network FFN, feature semantic alignment linear layer (we use T to represent the parameters in this layer) which projects visual features into semantic space, additional semantic word vocabulary D, an attention linear layer (we use M to represent the parameters in this layer) which constructs the connection between seen classes and their related words in D, seen class category semantic vector matrix W_s (including the background class semantic vector generated by BA-RPN). The FFN consists of two linear layers, using RELU as the activation function. D and W_s are constants, and FFN, T and M are modified during training. All word vectors in W_s are normalized to unit vectors.

In the static semantic vector branch, FFN compress the feature $feat_{bbox}$ in proposals into X^{box}. X^{box} is mapped to the semantic space through a linear layer with parameters T, and the cosine similarity S_d of the mapped feature with each semantic vector in D is calculated in the semantic space by matrix multiplication. S_d is again mapped to the semantic space through the linear layer M, and the category probability S_{st} in static semantic vector branch is calculated through W_s. The static semantic vector branch can be expressed as the formula:

$$X^{box} = FFN(feat_{bbox}) \tag{1}$$

$$S_{st} = W_s M D T X^{box} \tag{2}$$

2) Dynamic Semantic Vector Branch: The structure of dynamic semantic vector branch is similar to that of static semantic vector branch, which also includes five parts: candidate bounding box feature compression network $FFN^{'}$, feature semantic alignment layer (we use $T^{'}$ to represents the parameters in this layer), additional semantic word vocabulary D, an attention layer (we use $M^{'}$ to represent the parameters in this layer) that construct the connection between seen classes and their related words in D, seen class category semantic vector matrix $W_s^{'}$. $FFN^{'}$, $T^{'}$ and $M^{'}$ have the same structure as FFN, T and M, but they don't share parameters. In the dynamic semantic vector branch, only D is constant, $FFN^{'}$, $T^{'}$, $M^{'}$, $W_s^{'}$ (except the background semantic vector) are modified during training. Here $W_s^{'}$ is initialized identically to W_s and will be updated during training. All word vectors in $W_s^{'}$ are normalized to unit vectors.

In the dynamic semantic vector branch, $FFN^{'}$ compress the feature $feat_{bbox}$ in proposals into $X^{box'}$. $X^{box'}$ is mapped to the semantic space through layer $T^{'}$, and the *cosinesimilarity* $S_d^{'}$ of the mapped feature with each semantic vector in D is calculated in the semantic space by matrix multiplication. $S_d^{'}$ is mapped to the semantic space again through the linear layer $M^{'}$, and the category probability S_{dy} in dynamic semantic vector branch is calculated through $W_s^{'}$. The dynamic semantic vector branch can be expressed as the formula:

$$X^{box'} = FFN'(feat_{bbox}) \tag{3}$$

$$S_{dy} = W_s^{'} M^{'} D T^{'} X^{box'} \tag{4}$$

Finally, the seen class classification result of TCB can be expressed as:

$$Scores_{seen} = \sigma(\max(S_{st}, S_{dy})) \tag{5}$$

Where σ represents the softmax function, and the max function compares the similarity of two branches class-by-class.

Following the methods of previous studies [30], [6], the unseen class classification result of TCB can be expressed as:

$$Scores_{unseen} = W_u W_s^T Scores_{seen} \tag{6}$$

Where W_u stands for the unseen class semantic vector, and W_s^T stands for the transpose of the static semantic branch seen class semantic vector matrix. Here we use static but not dynamic semantic vectors for the knowledge from NLP models are included in original semantic distribution.

3) N-pair loss for $W_s^{'}$: In the cosine similarity classifier for each branch's determination of object categories, the semantic vectors, as the category centers, need to be as far away as possible to achieve better results. Those categories which are too close to other categories are easy to be confused. Inspired by N-pair loss [31], we use a similar loss term:

$$l_i = \log\left(\sum_{k=1}^{S} e^{(\cos(w_i, w_k)-1)}\right) \tag{7}$$

where w_i, w_k stands for any seen class semantic vector, $\cos(w_i, w_k)$ means compute the cosine similarity of w_i and w_k.

$$L_{npair} = \sum_{i=1}^{S} l_i \tag{8}$$

The overall N-pair loss term for $W_s^{'}$ is shown in Equation(8), which is the cumulative result of calculating Equation(7) for each seen class category. The loss term makes the semantic vectors separate from each other while improving the ability to represent visual features, which improves the classification effect.

D. Loss Function

As shown in the following formula, the loss function of our proposed method contains four parts.

$$L_{TCB} = L_{CE} + L_{reg} + \lambda_1 L_{npair} + \lambda_2 L_{ed} \tag{9}$$

Where L_{CE} is the cross-entropy loss for the classification. L_{reg} is the bounding box regression loss of the regression network, and the loss function is SmoothL1. λ_1 is the weight hyperparameter corresponding to the N-pair loss term; λ_2 is the weight hyperparameter corresponding to the Encoder-Decoder loss term [11], which regards the mapping network T and M as encoders, and adds additional network layers as the logical inverse function of the mapping network, that is, the decoder. L_{ed} uses the mean square error to measure the difference between the features before encoding and the features after decoding, which can improve the mapping ability of T and M.

IV. Experiments

A. Datasets

We conduct experiments on the MS-COCO [32] dataset and an autonomous driving dataset SiTi consisting of the Sichuan-Tibet Highway (a part of the G318 National Highway) scenes.

MS-COCO (2014) dataset includes 80 classes, 82783 images in the training set, and 40504 images in the validation set. According to previous works [3], [30], we conduct experiments on 65/15 split and 48/17 split, where 65/15 means we split MS-COCO dataset into 65 seen classes with 15 unseen classes and 48/17 means 48 seen classes with 17 unseen classes. The train set includes only seen class objects, and the test set includes both seen and unseen class objects.

SiTi, an autonomous driving dataset collected by us, includes 5370 images and 10 categories for now. For ZSD setting, we select 8 seen classes and 2 unseen classes.

B. Evaluation Protocol

We report the experimental results of our proposed method on ZSD and GZSD problem settings. For ZSD setting, only seen class objects need to be detected during inference. For GZSD setting, both seen class and unseen class objects need to be detected. mAP and Recall metrics are used to compare methods. For ZSD setting, we refer to previous works [3], [30], [11], using mAP with IoU threshold 0.5

TABLE I

AVERAGE PRECISION (%) OF EACH UNSEEN CLASS ON MS-COCO DATASET WITH 65/15 DATA SPLIT.

AP	airplane	cat	frisbee	sandwich	mouse	train	bear	snowboard	hot dog	toaster	parking meter	suitcase	fork	toilet	hair drier
ZSI [11]	4.9	57.9	**1.1**	15.3	**0.2**	**45.3**	22.2	**31.2**	**5.6**	0.5	0.5	4.9	11.4	**1.1**	**0.0**
dyn&npair	**7.1**	**60.5**	0.4	**15.8**	0.1	44.5	**54.6**	29.3	5.2	**0.7**	**0.8**	**5.8**	**12.0**	1.0	0.0

TABLE II

COMPARISON OF DYNAMIC SEMANTIC VECTORS AND STATIC SEMANTIC VECTORS METHOD WITH GZSD SETTING ON MS-COCO 65/15 SPLIT.

GZSD	Seen/Unseen	Seen		Unseen		HM	
		mAP	Recall	mAP	Recall	mAP	Recall
ZSI [11]	65/15	38.68	67.11	13.60	**58.93**	20.13	**62.76**
dyn&npair	65/15	**38.81**	**68.97**	**15.12**	51.80	**21.77**	59.16

and Recall@100 with IoU thresholds (0.4, 0.5, 0.6) for fair comparisons. For GZSD setting, we use mAP with IoU threshold 0.5 and Recall@100 with IoU threshold 0.5 for both seen and unseen classes.

C. Implementation Details

The semantic vectors are generated by Word2vec [14], and extra vocabulary word vectors D are consistent with [30]. The dimensions of word vectors are all 300 (100 for autonomous driving scenario data). We maintain the mask head in [11] for model training because it can increase the model's detection ability. As for weight hyperparameters λ_1 is set to 0.01 for 65/15, 0.075 for 48/17 and 0.025 for SiTi. λ_2 is set to 0.5 for all settings. We adopt Resnet101 pre-trained on ImageNet [33] as the backbone.

D. Dynamic Semantic Vectors

At first, we just implement our idea of dynamic semantic vector on a single-path classification branch. Specifically, we set seen class semantic vectors as model parameters on the basis of ZSI [11]. The AP of each unseen class is shown in Table I. "dyn&npair" means we use dynamic semantic vectors with N-pair loss on ZSI. The results show an improvement in AP in some unseen classes. However, the results in Table II show that the Recall@100 corresponding to dynamic semantic vector method decreases by 7.1%. Such a huge recall drop is unacceptable for safety-demanding automated systems such as autonomous driving. The above results indicate that the change of semantic distribution is positive for some unseen classes, while negative for others. How to make dynamic semantic vector method take advantage of its strengths and avoid its weaknesses? A simple idea is to use dynamic semantic vectors for positively optimized classes and original semantic vectors for negatively optimized classes. Therefore, we design a two-way classification branch network, one branch using dynamic semantic vector, the other using static semantic vector. We choose the Max function to screen the classification results in two ways.

E. Comparisons on MS-COCO ZSD/GZSD

We compare our method with PL [30], BLC [6], ZSI [11]. TCB is similar to the above three methods in the process of feature extraction and projection. Comparison with them can reflect the advantages of TCB's dynamic semantic vector and ensemble model design.

For ZSD setting, the result on two split benchmarks is shown in Table III. On 65/15 split, our method surpasses all of them. Compared with ZSI [11], our method brings 1.0% gain of Recall@100. On 48/17 split, our method is not as well as ZSI [11]. The result of ZSD shows that our method can increase the detection effect on ZSD setting by updating the distribution of semantic word vectors. The detection on 48/17 split is not as good as that on 65/15 split. We believe that this is because there are fewer seen class vectors available for optimization on 48/17 split, so it is more difficult to find a more reasonable semantic vector distribution.

For GZSD setting, the result on two split benchmarks is shown in Table IV. On 65/15 split, our method surpasses all of them. Compared with ZSI [11], our method brings 2.27% and 1.24% gain in terms of Recall@100 and mAP for seen class and 0.92% and 0.21% gain in terms of Recall@100 and mAP for unseen class. On 48/17 split, compared with ZSI [11], our method brings 1.21% and 0.86% gain in terms of Recall@100 and mAP for seen class and 0.07% gain in mAP for unseen class. The result on GZSD setting indicates that the updated seen class semantic vectors are more consistent with visual features and the prediction of seen class can be better transferred to the prediction of unseen class.

TABLE III

COMPARISON OF OUR METHOD WITH THE PREVIOUS STATE-OF-THE-ART ZSD WORKS ON MS-COCO. OUR METHOD SIGNIFICANTLY SURPASSES ALL OTHER WORKS ON 65/15 SPLIT.

ZSD	Seen/Unseen	Recall@100			mAP
		0.4	0.5	0.6	0.5
PL [30]	48/17	-	43.6	-	10.1
BLC [6]	48/17	49.6	46.4	41.9	9.9
ZSI [11]	48/17	**57.4**	**53.9**	**48.3**	**11.4**
TCB (ours)	48/17	55.5	52.4	48.1	**11.4**
PL [30]	65/15	-	37.7	-	12.4
BLC [6]	65/15	54.2	51.7	47.9	13.1
ZSI [11]	65/15	61.9	58.9	54.4	13.6
TCB (ours)	65/15	**62.5**	**59.9**	**55.1**	**13.8**

TABLE IV

COMPARISON OF OUR METHOD WITH THE PREVIOUS STATE-OF-THE-ART GZSD WORKS ON MS-COCO.

GZSD	Seen/Unseen	Seen		Unseen		HM	
		mAP	Recall	mAP	Recall	mAP	Recall
PL [30]	48/17	35.92	38.24	4.12	26.32	7.39	31.18
BLC [6]	48/17	42.10	57.56	4.50	46.39	8.20	51.37
ZSI [11]	48/17	46.51	70.76	4.83	**53.85**	8.75	**61.16**
TCB (ours)	48/17	**47.37**	**71.97**	**4.90**	52.41	**8.89**	60.65
PL [30]	65/15	34.07	36.38	12.40	37.16	18.18	36.76
BLC [6]	65/15	36.00	56.39	13.10	51.65	19.20	53.92
ZSI [11]	65/15	38.68	67.11	13.60	58.93	20.13	62.76
TCB (ours)	65/15	**39.92**	**69.38**	**13.81**	**59.85**	**20.53**	**64.26**

TABLE V

AVERAGE PRECISION (%) OF EACH UNSEEN CLASS ON MS-COCO DATASET WITH 65/15 DATA SPLIT

AP	airplane	cat	frisbee	sandwich	mouse	train	bear	snowboard	hot dog	toaster	parkingmeter	suitcase	fork	toilet	hair drier
ZSI	**4.9**	57.9	1.1	**15.3**	**0.2**	45.3	**22.2**	31.2	**5.6**	0.5	0.5	4.9	11.4	1.1	**0.0**
TCB	4.6	**60.6**	**1.6**	15.3	0.1	**46.1**	19.9	**32.7**	5.5	**0.6**	**0.6**	**6.3**	**12.2**	**1.3**	0.0

TABLE VI

COMPARISON OF OUR METHOD WITH ZSI ON AUTONOMOUS DRIVING SCENARIO DATA. OUR METHOD SIGNIFICANTLY SURPASSES ZSI.

ZSD	Seen/Unseen	Recall@100			mAP
		0.4	0.5	0.6	0.5
ZSI [11]	8/2	**69.4**	**61.7**	49.8	15.7
TCB (ours)	8/2	66.2	61.2	**50.9**	**18.1**

For MS-COCO dataset, we report the results on 65/15 split with mAP metric in Table V. From the experimental results, we find that our method achieves AP improvements on 9/15 unseen classes. From the experiment result, we can say that our method outperforms ZSI [11] on MS-COCO 65/15 split.

F. Comparisons on autonomous driving scenario data ZSD

We compare our method with baseline ZSI [11]. The result on autonomous driving scenario data is shown in Table VI. Compared with ZSI [11], our method brings 2.4% gain in mAP for unseen class. The result indicates the generalization of our method.

G. Ablation Studies

The idea of our method is that better classification of seen classes can be better migrating to unseen classes, so we perform ablation experiments on GZSD settings and MS-COCO 65/15 split. We build a baseline network with only one classification branch and dynamic seen class word vectors. The results are shown in Table VII. "Two-way" means we implement a two-way classification branch on the baseline network. "N-pair" means N-pair loss for $W_s^{'}$ is implemented during training. "Summary Function" means the function we use to summarize the results of two classification branch, where "Linear" means a linear layer and "Max" means the max function. From these results, we can learn that: 1) for networks using dynamic semantic vectors, both N-pair loss and two-way classification branch can improve mAP and Recall@100 for both seen and unseen classes. So these two parts that we proposed are effective; 2) compared with one way classification branch with N-pair loss, adding two-way structure can reduce the unseen class mAP by 1.26%, but improve the unseen class Recall@100 by 8.05%. We believe that this tradeoff between mAP and Recall@100 is worthwhile. And it is in line with our original intention of designing the two-way structure–to improve the recall of unseen classes; 3) as for the summary function, the Max function is better than linear layer. Compared with the linear layer, the Max function can guarantee the independence between the two branch's semantic vector spaces to the greatest extent, and only one branch's parameters are optimized in each backpropagation.

TABLE VII

EFFECTIVENESS FOR TWO-WAY CLASSIFICATION BRANCH AND N-PAIR LOSS FOR $W_s^{'}$. THE MAP (IOU=0.5) AND RECALL@100 (IOU=0.5) RESULTS FOR GZSD ARE REPORTED ON 65/15 SPLIT OF MS-COCO RESPECTIVELY.

		Summary Function		Seen		Unseen		HM	
Two-way	N-pair	Linear	Max	mAP	Recall	mAP	Recall	mAP	Recall
×	×			33.98	60.20	11.15	30.98	16.80	40.90
×	✓			**38.81**	**68.97**	**15.12**	**51.80**	**21.77**	**59.16**
✓	×	✓		39.63	69.01	13.60	58.24	20.25	63.17
✓	✓	✓		39.48	68.81	13.73	59.14	20.37	63.60
✓	×		✓	39.12	68.50	**13.86**	59.10	20.46	63.45
✓	✓		✓	**39.92**	**69.38**	13.81	**59.85**	**20.53**	**64.26**

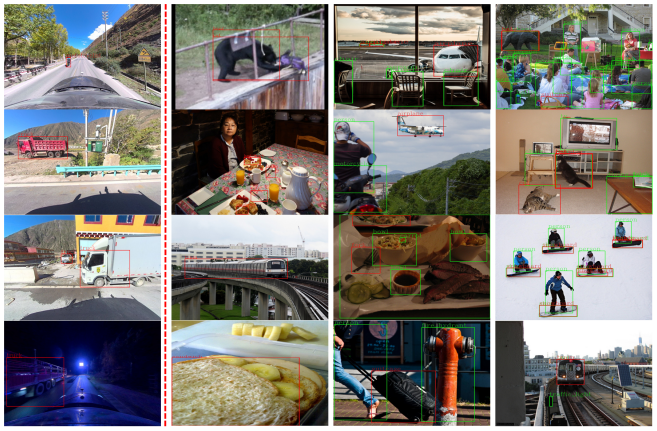

Fig. 3. Qualitative results on SiTi for ZSD (first column), MS-COCO for ZSD (second column) and GZSD (third and fourth columns). Seen classes are shown with green and unseen with red.

H. Qualitative Results

We give some object detection results of unseen objects in Fig. 3 as the qualitative results for our method. We find that our method can detect different unseen class objects and is able to detect even in the presence of multiple unseen objects while retaining the ability to detect seen classes.

V. CONCLUSION

This paper propose a zero-shot object detection method using dynamic semantic vectors as parameters for the first time. In this paper, we take semantic vectors as model parameters, propose a two-way classification branch network based on dynamic semantic vectors, and introduce N-pair loss into the optimization process of dynamic semantic vectors. Our method make the distribution of semantic vectors and visual features tend to be consistent, which is helpful in classification. Experiments on the MS-COCO dataset demonstrate the effectiveness of our method, and on the 65/15 data split, our method outperforms other previous state-of-the-art methods. We also demonstrate the generalization of our method on an in-house autonomous driving dataset SiTi.

REFERENCES

[1] M. Hnewa and H. Radha, "Object detection under rainy conditions for autonomous vehicles: A review of state-of-the-art and emerging techniques," *IEEE Signal Processing Magazine*, vol. 38, no. 1, pp. 53–67, 2020.

[2] X. Zhu, D. Anguelov, and D. Ramanan, "Capturing long-tail distributions of object subcategories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 915–922.

[3] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran, "Zero-shot object detection," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 384–400.

[4] S. Rahman, S. Khan, and F. Porikli, "Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts," in *Proceedings of the Asian Conference on Computer Vision*, 2018, pp. 547–563.

[5] B. Demirel, R. G. Cinbis, and N. Ikizler-Cinbis, "Zero-shot object detection by hybrid region embedding," *arXiv preprint arXiv:1805.06157*, 2018.

[6] Y. Zheng, R. Huang, C. Han, X. Huang, and L. Cui, "Background learnable cascade for zero-shot object detection," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[7] S. Rahman, S. H. Khan, and F. Porikli, "Zero-shot object detection: Joint recognition and localization of novel concepts," *International Journal of Computer Vision*, vol. 128, no. 12, pp. 2979–2999, 2020.

[8] D. Baek, Y. Oh, and B. Ham, "Exploiting a joint embedding space for generalized zero-shot semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9536–9545.

[9] Q. Li, Y. Zhang, S. Sun, X. Zhao, K. Li, and M. Tan, "Rethinking semantic-visual alignment in zero-shot object detection via a softplus margin focal loss," *Neurocomputing*, vol. 449, pp. 117–135, 2021.

[10] Y. Li, P. Li, H. Cui, and D. Wang, "Inference fusion with associative semantics for unseen object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 1993–2001.

[11] Y. Zheng, J. Wu, Y. Qin, F. Zhang, and L. Cui, "Zero-shot instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2593–2602.

[12] C. Yan, X. Chang, M. Luo, H. Liu, X. Zhang, and Q. Zheng, "Semantics-guided contrastive network for zero-shot object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[13] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[15] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[16] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5542–5551.

[17] N. Hayat, M. Hayat, S. Rahman, S. Khan, S. W. Zamir, and F. S. Khan, "Synthesizing the unseen for zero-shot object detection," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[18] D. Das and C. G. Lee, "A constrained generative approach to generalized zero-shot object recognition," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2021, pp. 1849–1854.

[19] Y. Ye, Y. He, T. Pan, J. Li, and H. T. Shen, "Alleviating domain shift via discriminative learning for generalized zero-shot learning," *IEEE Transactions on Multimedia*, vol. 24, pp. 1325–1337, 2021.

[20] P. Huang, J. Han, D. Cheng, and D. Zhang, "Robust region feature synthesizer for zero-shot object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7622–7631.

[21] C. Yang, W. Wu, Y. Wang, and H. Zhou, "A novel feature-based model for zero-shot object detection with simulated attributes," *Applied Intelligence*, vol. 52, no. 6, pp. 6905–6914, 2022.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[23] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 214–223.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[25] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1429–1437.

[26] P. Zhu, H. Wang, and V. Saligrama, "Zero shot detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 4, pp. 998–1010, 2019.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[30] S. Rahman, S. Khan, and N. Barnes, "Improved visual-semantic alignment for zero-shot object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 932–11 939.

[31] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.