Meta-ZSDETR: Zero-shot DETR with Meta-learning

Lu Zhang¹ Chenbo Zhang¹ Jiajia Zhao² Jihong Guan³ Shuigeng Zhou¹*
¹Shanghai Key Lab of Intelligent Information Processing, and School of Computer Science, Fudan University, China
²Science and Technology on Complex System Control and Intelligent Agent Cooperation Laboratory, China
³Department of Computer Science & Technology, Tongji University, China
{l_zhang19, sgzhou}@fudan.edu.cn, cbzhang21@m.fudan.edu.cn
zhaojiajia1982@gmail.com, jhguan@tongji.edu.cn

Abstract

Zero-shot object detection aims to localize and recognize objects of unseen classes. Most of existing works face two problems: the low recall of RPN in unseen classes and the confusion of unseen classes with background. In this paper, we present the first method that combines DETR and meta-learning to perform zero-shot object detection, named Meta-ZSDETR, where model training is formalized as an individual episode based meta-learning task. Different from Faster R-CNN based methods that firstly generate class-agnostic proposals, and then classify them with visual-semantic alignment module, Meta-ZSDETR directly predict class-specific boxes with class-specific queries and further filter them with the predicted accuracy from classification head. The model is optimized with meta-contrastive learning, which contains a regression head to generate the coordinates of class-specific boxes, a classification head to predict the accuracy of generated boxes, and a contrastive head that utilizes the proposed contrastive-reconstruction loss to further separate different classes in visual space. We conduct extensive experiments on two benchmark datasets MS COCO and PASCAL VOC. Experimental results show that our method outperforms the existing ZSD methods by a large margin.

1. Introduction

Object detection [34] is one of the most fundamental tasks in computer vision. Most existing object detection methods require huge amounts of annotated training data, which is expensive and time-consuming to acquire. Meanwhile, in reality novel categories constantly emerge, and there is seriously lack or even nonexistent of visual data of those novel categories for model training, such as endangered species in the wild. The above issues motivates the investigation of zero-shot object detection, which aims to localize and recognize objects of unseen classes.

A mainstream framework of the existing works that are based on Faster R-CNN, is illustrated in Fig. 1(a), where the RPN remains unchanged and the RoI classification head is replaced with different visual-semantic alignment modules, such as mapping to the same embedding space to calculate similarity between proposals and semantic vectors [2, 40, 41, 32, 44, 12, 25, 18, 7], synthesizing visual features from semantic vectors [17, 14, 42, 47, 27] etc.

However, we observe that the existing methods are sub-optimal, due to their obvious inherent shortcomings: i) The proposals from RPN are often not reliable enough to cover all unseen classes objects in an image because of lacking training data, which has also been identified by a recent study [19]. ii) The confusion between background and un-
seen classes is an intractable problem. Although many previous works have tried to tackle it [46, 14, 2], the results are still unsatisfactory.

Recently, object detection frameworks based on the Transformer have gained widespread popularity, such as DETR [5], Deformable DETR [48], etc. Such architectures are RPN-free and background-free, i.e., they do not involve RPN and background class, which are naturally conducive to building zero-shot object detection methods. However, how to build a ZSD method based on DETR detectors poses new challenges. An intuitive idea is to replace DETR’s RPN and background class, which are naturally conducive to building zero-shot object detection methods. However, such a method simply treats DETR as a large RPN for proposals generation, the overall framework is essentially the same as previous works.

In this paper, we present the first method that fully explores DETR detectors and meta-learning to perform zero-shot object detection, named Meta-ZSDETR, which can solve the two problems mentioned above that have plagued the field of ZSD for many years, and achieves the state-of-the-art performance. The comparison of Meta-ZSDETR with previous methods is shown in Fig. 1. Different from the previous works that firstly generate class-agnostic proposals and then classify them with visual-semantic alignment module, our method utilizes semantic vectors to guide both proposal generation and classification, which greatly improves the recall of unseen classes. Meanwhile, there is no background class in DETR detectors, which means the confusion between background and unseen classes is no more existent.

In order to detect unseen classes, we formalize the training process as an individual episode based meta-learning task. In each episode, we randomly sample an image $I$ and a set of classes $C_{π}$, which contains the positive classes that appear in $I$ and negative classes that do not appear. The meta-learning task is to make the model learn to detect all positive classes of $C_{π}$ on image $I$. Through the meta-learning task, the training and testing can be unified, i.e., in the model testing, we need only to employ the unseen classes as the set $C_{π}$. To enable the model to detect an arbitrary class set, we firstly fuse each object query with a projected semantic vector from the class set $C_{π}$, which transfers the query from class-agnostic to class-specific. Then, the decoder takes the class-specific query as input and predicts the locations of class-specific boxes, together with the probabilities that the boxes belong to the fused class. To achieve the above goal, we propose meta-contrastive learning, where all predictions are split into three different types and different combinations of them are chosen to optimize three different heads, i.e., the regression head to generate the locations of class-specific boxes, the classification head to predict the accuracy of generated boxes, and the contrastive head to separate different classes in visual space for performance improving with a contrastive-reconstruction loss. The bipartite matching and loss calculation are performed in a class-by-class manner, and the final loss is averaged over all classes in the sampled class set $C_{π}$.

In summary, our major contributions are as follows:

- We present the first method that explores DETR and meta-learning to perform zero-shot object detection, which formalizes the training as an individual episode based meta-learning task and ingeniously tackles the two problems that plague ZSD for years.
- We propose to train the decoder to directly predict class-specific boxes with class-specific queries as input, under the supervision of our meta-contrastive learning that contains three different heads.
- We conduct extensive experiments on two benchmark datasets MSCOCO and PASCAL VOC to evaluate the proposed method Meta-ZSDETR. Experimental results show that our method outperforms the existing ZSD methods.

2. Related work

2.1. Zero-shot learning

Zero-shot learning (ZSL) aims to classify images of unseen classes that do not appear during training. There are two main streams in ZSL: embedding based methods and generative based methods. The key idea of embedding based methods is to learn an embedding function that maps the semantic vectors and visual features into the same embedding space, where the visual features and semantic vectors can be compared directly [1, 4, 10, 21, 37]. Generative based methods aim to synthesize unseen visual features with variational autoencoder [20] and generative adversarial networks [39], which convert the ZSL into a fully supervised way [6, 13, 36, 38].

2.2. Zero-shot object detection

Zero-shot object detection (ZSD) has received a great deal of research interest in recent years. Most of ZSD methods are built on Faster R-CNN [11], YOLO [33] and RetinaNet [23]. The process of these methods can be summarized as: generating class-agnostic proposals and classifying proposals into seen/unseen and background classes. The main difference of these methods is that different visual-semantic alignment methods are used to complete the classification of proposals. These methods can be divided into two categories: mapping the semantic vectors and visual features to the same embedding space to calculate similarity [2, 40, 41, 32, 44, 12, 25, 18, 7] and synthesizing visual features from semantic vectors [17, 14, 42, 47, 27]. Although previous works have paid great efforts, there are
some problems that still have no satisfactory solution, such as the low recall of class-agnostic RPN for unseen classes and the confusion between background and unseen classes. These problems may be caused by the incompatibility of ZSD task and proposals-based architecture such as Faster R-CNN.

Different from previous works, Meta-ZSDETR is the first work built on Deformable DETR with meta-learning, where the semantic vectors are guided for class-specific boxes generation, instead of class-agnostic proposals in previous works, resulting in a higher unseen recall and precision. Meanwhile, since there is no background class in DETR detectors, the confusion between background and unseen classes is non-existent.

3. Method

3.1. Problem definition

Zero-shot object detection (ZSD) aims to detect objects of unseen classes with model trained on the seen classes. Formally, the class space \( \mathcal{C} \) in ZSD is divided into seen classes \( \mathcal{C}^s \) and unseen classes \( \mathcal{C}^u \), where \( \mathcal{C} = \mathcal{C}^s \cup \mathcal{C}^u \) and \( \mathcal{C}^s \cap \mathcal{C}^u = \emptyset \). The training set contains objects of seen classes, where each image \( I \) is provided with ground-truth class labels and bounding boxes coordinates. While the test set may contain only unseen objects (ZSD setting) or both seen and unseen classes (GZSD setting). During the training and testing, the semantic vectors \( \mathcal{W} = \{\mathcal{W}^s, \mathcal{W}^u\} \) is provided for both seen and unseen classes.

3.2. Revisit standard DETR in object detection

To begin with, we review the pipeline of a standard DETR in generic object detection, which contains the following steps: set prediction, optimal bipartite matching and loss calculation.

3.2.1 Set prediction

For an image \( I \), the global representation \( x_I \) is extracted by the backbone \( f_\phi \) and Transformer encoder \( g_\psi \), successively, which can be expressed as:

\[
x_I = g_\psi(f_\phi(I))
\]

Then, the decoder \( g_\theta \) infers \( N \) object predictions \( \hat{Y} \), where \( N \) is determined by the number of object queries \( Q \) that serve as learnable positional embedding:

\[
\hat{Y} = g_\theta(x_I, Q)
\]

where \( \hat{Y} = \{ (\hat{c}_i, \hat{b}_i) \}_{i=1}^N \) and \( Q = \{ q_i \}_{i=1}^N \). For each object query \( q_i \), the decoder \( g_\theta \) will output a prediction box, which contains two parts: the predicted class \( \hat{c}_i \) and predicted box location \( \hat{b}_i \).

3.2.2 Optimal bipartite matching

The optimal bipartite matching is to find the minimal-cost matching between the predictions \( \hat{Y} = \{ (\hat{c}_i, \hat{b}_i) \}_{i=1}^N \) and ground-truth boxes \( Y = \{ (c_i, b_i) \}_{i=1}^N \) (padded with no object \( \emptyset \)). Therefore, we search for a permutation of \( N \) elements \( \sigma \in \mathbb{S}_N \) with lowest cost:

\[
\sigma = \arg \min_{\sigma \in \mathbb{S}_N} \sum_{i=1}^N \left( L_{cls}(c_i, \hat{c}_{\sigma_i}) + L_{loc}(b_i, \hat{b}_{\sigma_i}) \right)
\]

where \( L_{cls}(c_i, \hat{c}_{\sigma_i}) \) and \( L_{loc}(b_i, \hat{b}_{\sigma_i}) \) are matching cost for class prediction and box location with index \( \sigma_i \), respectively. Bipartite matching produces one-to-one assignments, where each prediction \( (\hat{c}_i, \hat{b}_i) \) is assigned to either a ground-truth box \( (c_i, b_i) \) or \( \emptyset \) (no object). The permutation for lowest cost is calculated with Hungarian algorithm.

3.2.3 Hungarian loss

Hungarian loss is a widely used loss function in DETR, which takes the following form:

\[
\mathcal{L}_{Hung} = \sum_{i=1}^N \left[ L_{cls}(c_i, \hat{c}_{\sigma_i}) + \mathbb{I}_{\{c_i \neq \emptyset\}} L_{loc}(b_i, \hat{b}_{\sigma_i}) \right]
\]

where \( \sigma \) is the optimal assignment computed in Eq.(3). \( L_{cls} \) is the loss for classification, which usually takes the form of focal loss [23] or cross-entropy loss. \( L_{loc} \) is the location loss and usually contains \( l_1 \) loss and GIoU loss [35].

Challenge: Since the standard DETR can only locate the boxes and predict the classes of objects in training set, it is unable to detect unseen classes. In this paper, we utilize the meta-learning to make the model learn to detect objects according to the inputed semantic vectors, so that the model has the ability to detect objects of any category, as long as the semantic vector of the corresponding category is input.

3.3. Framework

We present the framework of Meta-ZSDETR in Fig. 2, which is based on Deformable DETR. Meta-ZSDETR follows the paradigm of meta-learning. The training is performed by episode based meta-learning task. In each episode, we randomly sample an image \( I \) and a class set \( \mathcal{C}_\pi \). The meta-learning task of each episode is to make the model learn to detect all appeared classes in \( \mathcal{C}_\pi \) on image \( I \). Specifically, the image feature is firstly extracted by backbone and Transformer encoder as in Eq.(1). In order for the decoder to detect categories in \( \mathcal{C}_\pi \), we add the projected semantic vectors of classes \( \mathcal{C}_\pi \) to the object queries, making the queries class-specific. Then, the decoder takes the queries as input and predicts the class-specific boxes directly. To achieve this, the model is optimized with meta-contrastive learning, which contains a regression head to generate the
coordinates of class-specific boxes, a classification to predict the accuracy of generated boxes and a contrastive head that utilize the proposed contrastive-reconstruction loss to further separate different classes in visual space.

### 3.4. Meta-ZSDETR with class-specific queries

To enable the model to detect any unseen class, we fuse the object queries with class semantic information, and make the model learn to predict the bounding boxes for the fused classes. Such a process is carried out in each meta-learning task.

Specifically, in each episode, we randomly sample a class set $C_\pi$ and an image $I$, where the $C_\pi$ satisfies $C_\pi \subseteq C^n$ and each element is unique. Meanwhile, $C_\pi = C^+_\pi \cup C^-_\pi$, where $C^+_\pi$ is the positive classes that appeared in image $I$ and $C^-_\pi$ is the randomly sampled negative classes that do not appear in $I$ for contrast. Meanwhile, we denote the size of $C_\pi$ as $L(C_\pi)$, where $L(\cdot)$ is the operation that calculate size. The positive rate $\frac{L(C^+_\pi)}{L(C_\pi)}$ is set to $\lambda_\pi$, which is a hyper-parameter.

Then, the corresponding semantic vectors $\mathcal{W}_\pi$ of class set $C_\pi$ are projected from semantic space to the visual space with a linear layer $h_\mathcal{W}$:

$$\bar{\mathcal{W}}_\pi = h_\mathcal{W}(\mathcal{W}_\pi)$$  \hspace{1cm} (5)

where $\bar{\mathcal{W}}_\pi$ is the projected semantic vectors of class set $C_\pi$. Since $L(\bar{\mathcal{W}}_\pi) \ll N$, i.e. the number of semantic vectors is smaller than the number of object queries $Q$, we expand $\bar{\mathcal{W}}_\pi$ by duplicating each element in $\bar{\mathcal{W}}_\pi$ for $T$ times, which satisfies $L(\bar{\mathcal{W}}_\pi) \cdot T \geq N$ and $L(\bar{\mathcal{W}}_\pi) \cdot (T - 1) < N$. For redundant elements more than $N$, we drop them.

Then, the projected semantic vectors $\bar{\mathcal{W}}_\pi$ is added to object queries $Q$ as follows:

$$Q_\pi = Q \oplus \bar{\mathcal{W}}_\pi$$  \hspace{1cm} (6)

where $Q_\pi = \{q^\pi_i\}_{i=1}^N$ is the class-specific queries that will be inputed into the Transformer decoder $g_\theta$ with image feature $x_I$ to generate predictions:

$$\hat{\mathcal{Y}} = g_\theta(x_I, Q_\pi)$$  \hspace{1cm} (7)

$\hat{\mathcal{Y}} = \{\delta_i, \hat{b}_i\}_{i=1}^N$ is the set of predictions, where $\hat{b}_i$ is predicted box location generated with query $q^\pi_i$ and $\delta_i$ is the probability of box $\hat{b}_i$ belonging to the fused class, i.e. the class of semantic vector that fused to query $q^\pi_i$. Meanwhile, different from the standard DETR that the classification head determines the class of predicted boxes, the class of predicted box $\hat{b}_i$ in Meta-ZSDETR is class-specific and is determined by the class of corresponding query and $\delta_i$ only has one dimension to represent the probability of $\hat{b}_i$ belongs to the fused class.

### 3.5. Meta-contrastive learning

In order for the regression head of decoder $g_\theta$ to generate more accurate class-specific box coordinate $\hat{b}_i$ and classification head to have a stronger discriminative ability of further judging the location accuracy of generated $\hat{b}_i$, we propose the meta-contrastive learning to train the heads of decoder $g_\theta$, i.e. a regression head to generate class-specific boxes, a classification head to filter inaccurate boxes, and moreover a contrastive head to further separate different classes in visual space, which will improve the performance of both seen classes and unseen classes.

Meta-contrastive learning performs the matching and optimization in a class-by-class manner. As shown in Fig. 3, for each class $c_j^\pi \in C_\pi$, the decoder takes queries of class $c_j^\pi$ as input, and the predictions are split into three types:

1) The positive predictions that are assigned to GT box of class $c_j^\pi$ by class-specific bipartite matching. 2) The negative predictions that are assigned to any other classes than $c_j^\pi$. 3) The negative predictions that belong to background.

Then, we takes different combinations of three types
of predictions to train three heads: a) For the classification head, we take all predictions for the training and give them corresponding positive/negative class-specific targets for class \( c_j \). b) For the regression head, since the regression head aims to generate class-specific boxes of class \( c_j \), we only utilize the positive predictions for optimization. c) For the contrastive head, since our intention is to separate different classes in visual space, we only use positive predictions of class \( c_j \) and negative predictions of other classes for contrastive-reconstruction loss.

### 3.5.1 Class-specific bipartite matching

Our meta-contrastive learning performs the matching in a class-by-class manner. For class \( c_j \in \mathcal{C}_m \), the predictions generated by queries of class \( c_j \) are selected for bipartite matching, which is denoted as \( \tilde{\mathcal{Y}}_{c_j} = \{(\tilde{\tau}_i, \tilde{\delta}_i)\}_{i=1}^{T_r} \), where \( \{\tau_i\}_{i=1}^{T_r} \) are the indexes for queries of class \( c_j \) in \( N \) queries. Since each query is duplicated for \( T \) or \( T-1 \) times, we denote the length uniformly as \( T_r \).

Then, the ground-truth matching targets are revised for each class \( c_j \in \mathcal{C}_m \). The class-specific matching labels are denoted as \( \Delta_{c_j} = \{\delta_{c_j}^e\}_{i=1}^{T_r} \) \( e \) (padded with no objects \( \emptyset \)), which satisfy:

\[
\delta_{c_i}^e = \begin{cases} 
1, & c_i = c_j \\
0, & c_i \neq c_j 
\end{cases} 
\]

where \( c_i \) is the origin class label for ground-truth box \( b_i \). The revised matching labels are generated according to whether \( c_i \) is equal to \( c_j \). Then, the matching targets \( \mathcal{Y}_{c_j} = \{(\tilde{\delta}_i, b_i)\}_{i=1}^{T_r} \) is utilized for bipartite matching of class \( c_j \), where a permutation of \( T_r \) elements \( \sigma \in \mathcal{S}_{T_r} \) with lowest cost is search as:

\[
\hat{\sigma} = \arg \min_{\sigma \in \mathcal{S}_{T_r}} \sum_{i=1}^{T_r} \left[ L_{cls}(\delta_i, \hat{\sigma}_i) + L_{loc}(b_i, \hat{\sigma}_i) \right] 
\]

where \( \hat{\sigma} \) is the optimal assignment between matching targets \( \mathcal{Y}_{c_j} \) and predictions \( \mathcal{Y}_{c_j} \). \( L_{cls} \) and \( L_{loc} \) are the same as that in Eq.(3).

### 3.5.2 Loss function

Based on \( \hat{\sigma} \), predictions in \( \tilde{\mathcal{Y}}_{c_j} \) are split into three types mentioned above, i.e. positive and two types of negative ones, and loss function of class \( c_j \) for three heads are calculated as follows:

\[
L_{c_j} = \sum_{i=1}^{T_r} \left[ L_{cls}(\delta_i^c, \hat{\sigma}_i) + \lambda_{c_i} L_{loc}(b_i, \hat{\sigma}_i) \right] + L_{cont} 
\]

where the loss \( L_{c_j} \) for class \( c_j \) is composed of classification loss \( L_{cls} \), regression loss \( L_{loc} \) and contrastive-reconstruction loss \( L_{cont} \).

**Classification loss.** \( L_{cls} \) takes all predictions as input and makes the model learn to distinguish whether a predicted box belongs to \( c_j \). The predicted box has the label \( \delta_i^c \) = 1 if and only if it is assigned to a ground-truth box with class label \( c_j \). We implement \( L_{cls} \) with focal loss [23].

**Regression loss.** Since our intention is to generate class-specific boxes, i.e. input decoder with query of class \( c_j \) and output box of class \( c_j \), we only select the ground-truth box with class label \( c_j \) for optimization, i.e. \( \delta_i^c = 1 \), making the predicted boxes to be closer to GT boxes with class label \( c_j \). The \( L_{loc} \) is implement with \( \ell_1 \) loss and GIoU loss [35].

**Contrastive-reconstruction loss.** In ZSD tasks, the original visual space is not well-structured due to the lack of discriminative information, many previous works solved it by introducing a reconstruction loss to guide the distribution of visual features. Here, we combine the reconstruction loss and contrastive loss to bring a higher intra-class compactness and inter-class separability of the visual structure.

In particular, we project the last hidden features of decoder to the semantic space, where the projected hidden features of positive predictions are constraint to be as close as possible to \( \omega_j^c \), which is the semantic vector of class \( c_j \), and the negative ones are constraint to be as far as possible to \( \omega_j^\emptyset \). Formally, we denote the last hidden feature of optimal box \( (\hat{\delta}_{\tau_s}, \hat{b}_{\tau_s}) \) that matched to GT box \( (c_i, b_i) \) as \( z_{\tau_s} \) and the \( L_{cont} \) is formulated as:

\[
L_{cont} = \frac{1}{N_{pos}} \sum_{i=1}^{T_r} \mathbb{1}_{(c_i = c_j)} L_{cont}(z_{\tau_s}) 
\]

\[
L_{cont}(z_{\tau_s}) = -\log \frac{\exp[h_p(z_{\tau_s}) \cdot \omega_j^c / \kappa]}{\sum_{k=1}^{T_r} \mathbb{1}_{(c_k \neq \emptyset)} \exp[h_p(z_{\tau_s}) \cdot \omega_j^\emptyset / \kappa]} 
\]

where \( h_p \) is a linear layer that project the hidden feature to the semantic space. \( N_{pos} \) is the number of positive predictions of class \( c_j \). \( \kappa \) is a temperature hyper-parameter.
as in InfoNCE [28]. The optimization of the above loss function increases the instance-level similarity between projected hidden features of positive predictions with semantic vector \( \omega_{\pi j} \) and space the negative ones. As a result, visual features of the same class will form a tighter cluster.

**Total loss function.** We compute the loss function with Eq. (10) for each class \( c_\pi j \in C_\pi \), separately. For negative classes \( C^- \), only classification loss is calculated. Finally, the loss of current episode is averaged over all classes in sampled class set \( C_\pi \):

\[
L = \frac{1}{L(C_\pi)} \sum_{j=1}^{L(C_\pi)} L_{c_\pi j}
\]

where \( L(C_\pi) \) is the number of classes in \( C_\pi \). We utilize \( L \) for model optimization.

### 4. Experiments

#### 4.1. Datasets and splits

Following previous works [40, 17], we take two benchmark datasets for evaluation: PASCAL VOC 2007+2012 [9] and MS COCO 2014 [24].

**Datasets:** PASCAL VOC contains 20 classes of objects for object detection. More specifically, the PASCAL VOC 2007 is composed of 2,501 training images, 2,510 validation images, and 5,011 test images. The PASCAL VOC 2012 dataset comprises 5,717 training images and 5,823 validation images, without test images released. MS COCO 2014 is a benchmark dataset designed for object detection and semantic segmentation tasks, which contains 82,783 training and 40,504 validation images from 80 categories. For PASCAL VOC and MS COCO, we adopt the FastText [26] to extract the semantic vectors following [14, 17].

**Seen/unseen splits:** All splits follow the previous setting [40, 17]. For PASCAL VOC, we use a 16/4 split proposed in [7]. For MS COCO, we follow the same procedures described in [2, 29, 17] to take 2 different splits, i.e. 48/17 and 65/15. For all datasets, the images that contains unseen classes in the training set are removed to guarantee that unseen objects will not be available during training.

#### 4.2. Evaluation protocols

We adopt the widely-used evaluation protocols proposed in [2, 7]. For PASCAL VOC, mAP with IoU threshold 0.5 is used to evaluate the performance. For MS COCO, mAP and recall@100 with three different IoU threshold (i.e. 0.4, 0.5 and 0.6) are utilized for evaluation. For GZSD setting that contains both seen and unseen classes, the performance is evaluated by Harmonic Mean (HM).

#### 4.3. Implementation details

We build Meta-ZSD for Deformable DETR [48] with ResNet-50 [16] as backbone. The number of queries \( N \) is set to 900. For the sampled class set \( C_\pi \), the positive classes consist of the classes that appear in the image and the negative classes is sampled from \( C^- \). The positive rate \( \lambda_\pi \) is set to 0.5. The number of Transformer encoder layers and decoder layers is set to 6. The temperature hyperparameter \( \kappa \) in Eq.(12) is set to 0.2. We train our model for total 500,000 iterations with batch size 16, i.e. each iteration contains 16 episodes in parallel. Following Deformable DETR, different coefficients are utilized to weight different loss functions, where 1.0 is used for classification loss, 5.0 is used for \( l_1 \) loss of regression head, 2.0 is used for GIoU loss of regression head and 1.0 is used for contrastive-reconstruction loss. More details can refer to our code.

#### 4.4. Comparison with existing methods

##### 4.4.1 PASCAL VOC

We present the results of PASCAL VOC in Tab. 1, where we can see that our method performs best among all existing methods under both ZSD and GZSD settings, and lift the mAP in PASCAL VOC to a higher level.

<table>
<thead>
<tr>
<th>Method</th>
<th>ZSD</th>
<th>GZSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAN [31]</td>
<td>59.1</td>
<td>48.0</td>
</tr>
<tr>
<td>HRE [8]</td>
<td>54.2</td>
<td>62.4</td>
</tr>
<tr>
<td>BLC [45]</td>
<td>55.2</td>
<td>22.9</td>
</tr>
<tr>
<td>SU [15]</td>
<td>64.9</td>
<td>47.1</td>
</tr>
<tr>
<td>Robust-Syn [17]</td>
<td>65.5</td>
<td>49.1</td>
</tr>
<tr>
<td>ContrastZSD [40]</td>
<td>65.7</td>
<td>46.5</td>
</tr>
<tr>
<td>Meta-ZSDTR</td>
<td><strong>70.3</strong></td>
<td><strong>56.3</strong></td>
</tr>
</tbody>
</table>

Table 1. The results of mAP in PASCAL VOC with IoU=0.5 under ZSD and GZSD settings. Here, “S” denotes seen classes, “U” denotes unseen classes and “HM” denotes harmonic mean.

<table>
<thead>
<tr>
<th>Method</th>
<th>car</th>
<th>dog</th>
<th>sofa</th>
<th>train</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAN [31]</td>
<td>56.2</td>
<td>85.3</td>
<td>62.6</td>
<td>26.4</td>
<td>57.6</td>
</tr>
<tr>
<td>HRE [8]</td>
<td>55.0</td>
<td>82.0</td>
<td>55.0</td>
<td>26.0</td>
<td>54.5</td>
</tr>
<tr>
<td>PL [30]</td>
<td>63.7</td>
<td>87.2</td>
<td>53.2</td>
<td>44.1</td>
<td>62.1</td>
</tr>
<tr>
<td>BLC [45]</td>
<td>43.7</td>
<td>86</td>
<td>60.8</td>
<td>30.1</td>
<td>55.2</td>
</tr>
<tr>
<td>SU [15]</td>
<td>59.6</td>
<td><strong>92.7</strong></td>
<td>62.3</td>
<td>45.2</td>
<td>64.9</td>
</tr>
<tr>
<td>Robust-Syn [17]</td>
<td>60.1</td>
<td>93.0</td>
<td>59.7</td>
<td>49.1</td>
<td>65.5</td>
</tr>
<tr>
<td>ContrastZSD [40]</td>
<td>65.5</td>
<td>86.4</td>
<td>63.1</td>
<td>47.9</td>
<td>65.7</td>
</tr>
<tr>
<td>Meta-ZSDTR</td>
<td><strong>69.0</strong></td>
<td><strong>92.4</strong></td>
<td><strong>65.7</strong></td>
<td><strong>54.1</strong></td>
<td><strong>70.3</strong></td>
</tr>
</tbody>
</table>

Table 2. Class-wise AP and mAP on unseen classes in PASCAL VOC under ZSD setting.
For GZSD setting, our method also achieves SOTAs in all three metrics, i.e. mAP on seen classes, unseen classes and harmonic mean, which brings about 4.4, 9.8 and 7.8 points improvement, respectively. It is worth noting that the improvement of our method on unseen classes in GZSD setting is extremely large, which proves that our method has a strong generalization on unseen classes, and can alleviate the problem that the unseen classes tend to be misclassified into seen classes to a certain extent. We also report class-wise mAP in ZSD setting in Tab. 2, where our method achieves the best performance on 3 classes.

### 4.4.2 MS COCO

We perform experiments on MS COCO, where the results of ZSD setting is shown in Tab. 3 and the results of GZSD setting is shown in Tab. 4. We can see that Meta-ZSDETR achieves the best results in all metrics under all settings.

For ZSD setting, we can see that mAP of our method in 48/17 and 65/15 splits outperforms the second-best by a margin of 1.7 and 2.7 mAP, respectively, which demonstrates that our method has a strong generalization on unseen classes. Meanwhile, we can see that Recall@100 decreases as the IoU increases in all methods. Compared with other methods, Meta-ZSDETR has a smaller drop, which is benefit from that our decoder can generate more accurate boxes with class semantic information as input.

For GZSD setting, our method achieves SOTAs in both seen and unseen classes. The harmonic means of mAP under 48/17 and 65/15 splits are improved from 20.4 to 22.5, and from 26.0 to 29.5, demonstrating the effectiveness and superiority of our method. Meanwhile, the Recall@100 also improves due to the powerful class-specific boxes generation capabilities. We also report the class-wise AP in ZSD setting in Tab. 2, where our method achieves the best performance on 3 classes.

### 4.5 Ablation study

We analyze the effects of various components in Meta-ZSDETR. Unless otherwise specified, the experiments are carried out on MS COCO with 65/15 split under GZSD setting and use mAP with IoU=0.5 as metric.

#### Effects of different loss functions

Here, we analyze the effects of three loss functions in meta-contrastive learning. We utilize different combinations of regression loss $L_{loc}$, classification loss $L_{cls}$ and contrastive-reconstruction loss $L_{cont}$ to optimize the model, and show the results in Tab. 5. Since the regression loss is necessary, we keep it for all combinations. For model without classification loss $L_{cls}$, we do not perform the boxes filter and directly use the class-specific boxes predicted from regression head, where the scores are generated randomly. As we can see, if the model is only trained with regression head to gen-
<table>
<thead>
<tr>
<th>Heads</th>
<th>$\hat{Y}_{pos}$</th>
<th>$\hat{Y}_{other}$</th>
<th>$\hat{Y}_{bg}$</th>
<th>Seen</th>
<th>Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Head</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>43.7</td>
<td>17.9</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>44.8</td>
<td>19.0</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>45.9</td>
<td>21.7</td>
</tr>
<tr>
<td>Regression Head</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>45.9</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>42.1</td>
<td>16.5</td>
</tr>
<tr>
<td>Contrastive Head</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>45.1</td>
<td>21.0</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>45.9</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>45.4</td>
<td>21.3</td>
</tr>
</tbody>
</table>

Table 6. Ablation study on using different combinations of predictions to train three heads.

For classification head, since it aims to filter all kinds of negative predictions, using all predictions to train it can achieve the best performance. For regression head, if we train it with GT boxes of all classes, i.e., using $\hat{Y}_{pos}$ and $\hat{Y}_{other}$, the regression head will degenerate into a class-agnostic RPN, which will greatly reduce the recall of unseen classes, thus lead to a lower mAP. For contrastive head, on one hand, if we only use $\hat{Y}_{pos}$ for training, it will degenerate into a reconstruction loss, which has been widely used in previous works and it will bring a 0.4 mAP improvement in unseen classes compared with the version without it. On the other hand, compared with using all predictions, removing background predictions will make the contrastive head focus on distinguishing $\hat{Y}_{other}$ and inputed semantic vectors of class $c_j^\pi$, thus brings more improvement.

**Study for training of different heads.** As describe, based on the class-specific bipartite matching for class $c_j^\pi$, all predictions are split into three different types: the positive predictions $\hat{Y}_{pos}$ assigned to GT boxes of $c_j^\pi$, the negative predictions $\hat{Y}_{other}$ assigned to other classes and the negative predictions $\hat{Y}_{bg}$ that belong to background. Here, we study different combinations of them to train three heads and the results are shown in Tab. 6. We can see that: 1) For classification head, since it aims to filter all kinds of negative predictions, using all predictions to train it can achieve the best performance. 2) For regression head, if we train it with GT boxes of all classes, i.e. using $\hat{Y}_{pos}$ and $\hat{Y}_{other}$, the regression head will degenerate into a class-agnostic RPN, which will greatly reduce the recall of unseen classes, thus lead to a lower mAP. 3) For contrastive head, on one hand, if we only use $\hat{Y}_{pos}$ for training, it will degenerate into a reconstruction loss, which has been widely used in previous works and it will bring a 0.4 mAP improvement in unseen classes compared with the version without it. On the other hand, compared with using all predictions, removing background predictions will make the contrastive head focus on distinguishing $\hat{Y}_{other}$ and inputed semantic vectors of class $c_j^\pi$, thus brings more improvement.

**Visualization for contrastive-reconstruction loss.** Here, we study the influence of contrastive-reconstruction loss on visual space by visualizing the distribution of hidden features with t-SNE. We visualize the last hidden features of decoder in unseen classes of PASCAL VOC. The result is shown in Fig. 4. As we can see, our contrastive-reconstruction loss can further separate different classes in visual space and bring a higher intra-class compactness and inter-class separability of the visual structure.

**Effect of number of queries and positive rate of $C_n$.** We study the effect of number of queries $N$ and positive rate $\lambda_n$ of sampled class set $C_n$. We found that $\lambda_n$ have different influence under different number of queries $N$. We change the positive rate $\lambda_n$ in different settings of $N$ and report the mAP of converged model in unseen classes. In each episode, we control $\lambda_n$ by sampling different number of negative classes. All models are trained for 500,000 episodes. The results are shown in Fig. 5. As we can see, a larger $N$ tends to have a better performance due to a higher recall, and of course a higher amount of calculation. Meanwhile, when $N$ is small (e.g. 100), a small positive rate will greatly reduce the amount of positive queries for training, thereby reducing the model performance. When $N$ is large (e.g. 900), the number of positive queries is guaranteed and more negative queries are need for the classification head to learn to distinguish among them. Therefore, we can see the best performance is achieved when $\lambda_n$ is 0.5 and $N$ is 900.

5. Conclusion

In this paper, we present the first work that combine DETR and meta-learning to perform zero-shot object detection, which formalize the training as individual episode
based meta-learning task. In each episode, we randomly sample an image and a class set. The meta-learning task is to make the model learn to detect all appeared classes of the sampled class set on the image. To achieve this, we train the decoder to directly predict class-specific boxes with class-specific queries as input, under the supervision of our meta-contrastive learning that contains three different heads. We conduct extensive experiments on the benchmark datasets MSCOCO and PASCAL VOC. Experimental results show that our method outperforms the existing ZSD methods. In the future, we will focus on further performance improvement.

Acknowledgement. Jihong Guan was supported by National Natural Science Foundation of China (NSFC) under grant No. U1936205.

References


6853


[45] Ye Zheng, Ruoran Huang, Chuanqi Han, Xi Huang, and Li Cui. Background learnable cascade for zero-shot object detection. In ACCV, 2020. 6, 7

