Boosting Zero-Shot Learning via Contrastive Optimization of Attribute Representations

Yu Du, Miaojing Shi, Senior Member, IEEE, Fangyun Wei, and Guoqi Li

Abstract—Zero-shot learning (ZSL) aims to recognize classes that do not have samples in the training set. One representative solution is to directly learn an embedding function associating visual features with corresponding class semantics for recognizing new classes. Many methods extend upon this solution, and recent ones are especially keen on extracting rich features from images, e.g., attribute features. These attribute features are normally extracted within each individual image; however, the common traits for features across images yet belonging to the same attribute are not emphasized. In this article, we propose a new framework to boost ZSL by explicitly learning attribute prototypes beyond images and contrastively optimizing them with attribute-level features within images. Besides the novel architecture, two elements are highlighted for attribute representations: a new prototype generation module (PM) is designed to generate attribute prototypes from attribute semantics; a hard-example-based contrastive optimization scheme is introduced to reinforce attribute-level features in the embedding space. We explore two alternative backbones, CNN-based and transformer-based, to build our framework and conduct experiments on three standard benchmarks, Caltech-UCSD Birds-200-2011 (CUB), SUN attribute database (SUN), and animals with attributes 2 (AwA2). Results on these benchmarks demonstrate that our method improves the state of the art by a considerable margin. Our codes will be available at https://github.com/dyabel/CoAR-ZSL.git.

Index Terms—Attributes, contrastive learning, prototype generation, transformer, zero-shot learning (ZSL).

I. INTRODUCTION

VI SUAL recognition flourishes in the presence of deep neural networks (DNNs) [1], [2], [3]. Great success has been achieved with the availability of large amount of data and efficient machine learning techniques. The learned models are good experts to recognize visual classes that they were trained with, nonetheless, often fail to generalize on unseen classes with few or no training samples. To tackle this, several learning paradigms such as few-shot learning [4], [5], [6], zero-shot learning (ZSL) [7], [8], [9], and self-supervised learning [10], [11] are introduced given different data/label availability during training and testing. This article focuses on ZSL where models are learned with sufficient data of seen classes and deployed to recognize unseen classes. An extended setting to ZSL is referred to as the generalized ZSL (GZSL) where learned models are required to recognize both seen and unseen classes at the testing stage. GZSL is closer to the recognition problem we face in the real world. Below, unless specified, we use ZSL to refer to both the settings.

Despite that no training samples are provided for unseen classes in ZSL, the semantic information is available both the sides to relate seen and unseen classes. The essential idea is to transfer knowledge from seen classes to unseen classes which can be through either the visual-to-semantic mapping [12], [13], [14] or reversely the semantic-to-visual mapping [15], [16], [17]. Commonly used semantic information includes human-defined attributes of classes [18] and machine-generated word vectors of classes [19].

A number of representative methods in ZSL make use of generative models to synthesize visual features for unseen classes based on their class semantics [20], [21], [22]. Generative models can be hard to optimize as there exists a substantial gap between virtual features and features from real images. To bypass the use of generative models, other works tend to directly learn an embedding function associating visual features with the corresponding semantic features [12], [13], [23], [24], [25]. For instance, [12], [13] proposed to encode each image (object) into a number of visual attribute features and then align them with class semantics. Because of the enriched representations of visual objects, these methods have reported state-of-the-art results. Despite their success, they still face challenges: attribute features are extracted within each individual image; due to the intra-attribute variance, one attribute might appear in form of multiple scattered features in the embedding space (see Fig. 1). This weakens the feature representativeness, hence impairs the classification. To tackle it, we propose to boost ZSL via contrastive optimization between attribute prototypes across images and attribute-level features within images. As a result, the knowledge of attributes can be robustly learned and transferred from seen classes to unseen class in ZSL.

Our article follows embedding-based methods [12], [13], [23], [24], [25] in ZSL. The essential motivation is to empower the representations of visual objects such that those from
different classes or with different attributes can be easily discriminated, while those from the same class or with same attributes can be easily connected. To achieve this, we propose a new framework, CoAR-ZSL, to explicitly learn attribute prototypes and attribute-level features separately and boost ZSL via the contrastive optimization of these attribute representations. Specifically, a prototype generation module (PM) is first designed to take inputs of semantic features for attributes and generate the corresponding attribute prototypes. Meanwhile, an input image is passed through a DNN to extract attribute-level visual features via an attention-based attribute localization scheme inspired by [23] and [26]. Attribute-level features are optimized against the corresponding attribute prototypes using a contrastive triplet loss. To further reinforce these features, another hard-example-based contrastive loss is introduced to optimize them across images in the embedding space. Besides the optimization of attribute representations, we also follow [16], [27] to extract class-level features from images and optimize them with the corresponding class prototypes; the latter are obtained jointly with attribute prototypes through the proposed prototype generation module. We explore two alternative backbones for extracting class- and attribute-level features: the CNN-based and the transformer-based. Both the versions outperform the state of the art in the ZSL settings.

II. RELATED WORKS
A. Zero-Shot Learning
Modern ZSL methods can be broadly categorized as either generative-based or embedding-based [31]. Many generative-based methods [20], [21], [32], [33] synthesize visual features of unseen classes via the visual features of seen classes and the semantics of both seen and unseen classes. There are also generative-based methods that synthesize features for virtual classes using strategies such as MixUp [34], [35] such that prior unseen class semantics are no longer required to generate synthetic data during training. The generative-based methods perform well on datasets with sufficient training data so the feature generators (usually GAN [20], [32] or VAE [36]) can be fully optimized. On the other hand, embedding-based methods try to learn an embedding function connecting visual features with the corresponding semantic features without the use of feature generators. The embedding-based methods consist of many subcategories, including graph-based [37], meta-learning-based [38], attention-based [12], [13], [24], [25], [39], etc. Our work can be categorized as the attention-based. The attention-based methods focus on discriminative regions of images and encode them into local features. The attention mechanism is usually implemented explicitly [12], [13], [25], [39] but can also be simulated from the middle layers of the neural network [40].

Our work learns both class and attribute representations for ZSL. Before our work, class representations have been exploited in several works [16], [27], [41], where they learn class prototypes from class semantics and optimize them against visual features from images. We follow these works to use class representations. Our difference to them mainly lies in the network design where our new framework enables a joint optimization of class and attribute representations in ZSL. As for the attribute representation, [12], [13] make use of attribute features in images. However, their attribute features are extracted from each individual image and may not be representative enough as a consequence of the intra-attribute variance. We for the first time explicitly learn attribute prototypes beyond images and optimize them with attribute-level features within images. Moreover, [12], [13] learn a visual-to-semantic mapping while ours is the opposite. The former, according to [42], is more likely to face the hubness problem.

B. Contrastive Learning
The goal of contrastive learning is to learn an embedding space in which similar samples are pushed close and dissimilar ones are pulled away [43]. It can be applied to either labeled [44] or unlabeled data [10]. The latter is getting very popular in the self-supervised learning, where the essential idea is to enforce the embedding of the same sample of multiple views to be similar. Contrastive learning is recently
used in ZSL by [45]. It introduces class-level and instance-level contrastive losses into a generative-based model. The contrastive loss in our work is applied to attribute-level features in an embedding-based model. Unlike [45] using all the positive/negative samples for each anchor to construct the loss, we introduce a new hard-example-based contrastive loss which uses only hard attribute-level features based on their attention peaks and mutual distances. In addition, our loss forms are also different.

C. Transformers

Compared with the CNN-based attention architecture, the self-attention architecture in transformers has demonstrated to be superior in many natural language processing (NLP) tasks [46], [47], [48]. Because of its huge success in NLP, many computer vision researchers also start using it. One successful example is the vision transformer (ViT) [49]: its essential idea is to slice the image into a sequence of patches and treat their embeddings as patch tokens; an extra classification token is also added to the transformer to generate global features for image classification.

A recent work, ViT-ZSL [50], has tried to adapt the ViT into ZSL as a backbone for feature extraction. [50] replaces the classification head in ViT with an FC layer to project the global feature into the semantic space for ZSL. We instead use the global feature directly in the visual space. Also, the patch embedding in ViT is dropped in [50], while we use it in an attention-based attribute localization scheme for attribute-level feature embedding. Overall, we introduce a different way to adapt ViT as a backbone in our work.

III. METHOD

A. Problem Setting

ZSL aims to recognize classes that have no labeled data in the training set. We follow other embedding-based methods [23], [25], [26] to use the seen class semantics and their visual features to learn an embedding function during training.

We denote by \( \mathcal{X}^s \) and \( \mathcal{Y}^s \) the set of image visual features, and \( \mathcal{Y}^u \) and \( \mathcal{Y}^u \) the set of image class labels, for seen (\( s \)) and unseen (\( u \)) classes, respectively. The training set for seen classes is represented as \( \mathcal{D}^s = \{ x_i^s, y_i^s \}_{i=1}^{N^s} \) of \( N^s \) elements and has \( M^s \) classes in total. The test set for unseen classes is similarly represented as \( \mathcal{D}^u = \{ x_i^u, y_i^u \}_{i=1}^{N^u} \) for \( N^u \) elements and has \( M^u \) classes in total. There are in total \( K \) object attributes shared between seen and unseen classes while there is no overlap between \( \mathcal{D}^s \) and \( \mathcal{D}^u \). In the conventional ZSL, the task is to recognize only unseen classes. In the more realistic GZSL, the task is to recognize both the seen and unseen classes. Unless specified, below we omit superscript \( s \) or \( u \).

B. Overview

The overview of our proposed new framework CoAR-ZSL is shown in Fig. 2: two main data streams flow into it from the right and left sides for prototype generation and feature embedding, respectively. For feature embedding, two alternative backbones, i.e., CNN-based (default) and transformer-based, are presented.

1) Prototype Generation: Looking from the right, we design a class and attribute prototype generation module which takes the input of class and attribute semantics, \( \mathcal{C}^s = \{ c_s \}_{i=1}^{M^s} \) and \( \mathcal{A}^s = \{ a_s \}_{i=1}^{K} \), and outputs class and attribute prototypes, \( \mathcal{CP} = \{ cp_i \}_{i=1}^{M^s} \) and \( \mathcal{AP} = \{ ap_i \}_{i=1}^{K} \), respectively.

2) Class Representation Learning: Given an input image \( x \) from the left side, we can extract its class-level global feature \( cf \) directly from the backbone. Its cosine similarity is computed against \( \mathcal{CP} \) and optimized via the cross-entropy loss for classification (\( \mathcal{L}_{cls} \)).

3) Attribute Representation Learning: Given the input image \( x \), we can extract its attribute-level local features \( Af = \{ af_i \}_{i=1}^{K} \) via the attention-based attribute localization scheme, which produces an attention tensor AM incorporating a set of attention maps, which are used as soft masks to localize different attribute-related regions in \( x \) and extract \( Af \) from them. For each \( af \) in \( Af \), it is optimized against the corresponding \( ap \) using a contrastive triplet loss (\( \mathcal{L}_{attp} \)). For learning better \( af \) across images, another hard-example-based contrastive optimization loss (\( \mathcal{L}_{attp} \)) is also devised to reinforce the similarity of attribute-level features corresponding to the same attribute. Finally, to focus AM on the attribute-related regions, we max-pool it to obtain a semantic vector \( cs^e \) and minimize its L2 distance to the ground truth \( cs^s \) (\( \mathcal{L}_{sem} \)).

4) Inference: At testing, the input class semantics for seen class are replaced by the corresponding semantics of unseen classes (for ZSL) or of all classes (for GZSL) to embed new sets of class prototypes. We compute the cosine similarities from the class-level feature of a test image to embedded class prototypes to decide the class label of the image.

C. Class Representation Learning

Given the class semantics in \( \mathcal{C}^s \), each \( c_s \) is in the form of a \( K \)-dimensional vector indicating the presence/absence of the \( K \) attributes in this class. The digits in the vector can be binary or continuous (ours is continuous). They are stored as an \( M \times K \) matrix and fed into the prototype generation module (specified later) to obtain class prototypes \( \mathcal{CP} \). Each \( cp \) is of \( C \) dimensions. On the other hand, the class-level feature \( cf \) is easily obtained by applying global average pooling (GAP) on the backbone feature tensor \( F \) (Fig. 2).

Given \( \mathcal{CP} \) and \( cf \) for the input image \( x \), we can compute the cosine similarity \( \cos(cf, cp_i) \) from \( cf \) to any class prototype \( cp_i \). Assuming \( x \) belongs to the \( i \)th class, the predicted probability for this class is given by

\[
p_i = \frac{\exp(\alpha \cdot \cos(cf, cp_i))}{\sum_{j=1}^{M^s} \exp(\alpha \cdot \cos(cf, cp_j))}
\]

where \( \alpha \) is the scaling factor. The cross-entropy loss is used to optimize \( p_i \)

\[
\mathcal{L}_{cls} = -\log p_i.
\]

D. Attribute Representation Learning

1) Prototype Generation: We design a prototype generation module to perform the semantic-to-visual mapping from class
and attribute semantics to their corresponding visual features. It is a multilayer perceptron (MLP) with a few shared layers and two identical branches for class and attribute prototype generation, respectively. The shared layers include two FC layers with the first followed by Relu. Each separate branch contains one FC and two identical branches for class and attribute prototype generation, respectively. Inspired by the class normalization (CN) in [15], which was introduced to preserve the variance of semantic-to-visual mapping, we follow its design to have two CN before the FC in each branch. CN can be seen as BatchNorm without the affine transform.

For attribute prototype generation, the input attribute semantics $\mathbf{AS}$ are set as one-hot vectors such that $as_j$ only has its $j$th value being nonzero in the vector. $\mathbf{AS}$ can be seen as a basis to construct $\mathbf{CS}$. Other orthogonal bases may also work to distinguish attributes while the one-hot form works the best empirically. $\mathbf{AS}$ are stored as a $K \times K$ matrix and fed into the prototype generation module to obtain $\mathbf{AP}$. Each $ap$ is of $C$ dimensions.

Note class prototype generation also uses this module which takes the input of $\mathbf{CS}$ and outputs $\mathbf{CP}$.

2) Attribute-Level Feature Embedding: For the extraction of attribute-level features, we follow [23], [26] to adopt an attention-based mechanism. By integrating it into our backbone architecture, we present the attention-based attribute localization scheme.

We first add four convolution layers after all the four stages of the backbone (ResNet101, C1–C4), respectively to extract feature tensors of the same resolution $H \times W \times K$ over multiple scales. These attention tensors $\mathbf{AM}$ are pixelwisely added together to produce the AM, $\mathbf{AM} = \sum_{s=1}^{4} \mathbf{AM}^s$. We apply the Softmax function on each feature map of AM to move it in the range of 0–1. The feature map $am_j$ in AM hence serves as a soft mask indicating the potential localization of the $j$th attribute. To obtain attribute-level features, we apply the bilinear pooling between AM and the image feature tensor $\mathbf{F}$. It is basically to apply each attribute-level soft mask to feature maps in $\mathbf{F}$ to localize this attribute [51]. Specifically, each $am_j$ is pixelwisely multiplied to every feature map $f_i$ in $\mathbf{F}$ followed by average pooling. There are $C$ feature maps in $\mathbf{F}$, which ends up with a $C$-dimensional feature vector $af_j$ for the $j$th attribute. The process can be written as

$$af_j = \text{GAP} \left( \mathcal{R} \left( \text{Softmax}(am_j) \right) \odot F \right)$$

where Softmax($\cdot$) is applied to the 2-D ($H \times W$) input. We use $\mathcal{R}(\cdot)$ to expand $\text{Softmax}(am_j)$ (by replication) to $C$ channels so as to pixelwisely multiply with $F$; $\odot$ is the elementwise product. By iterating $am_j$ over $j$, we obtain the
final \( \mathcal{AF} = \{a_f j\}_{j=1}^K \). To guarantee that \( a_f j \) is related to the \( j \)th attribute, below we introduce the attribute representation optimization to optimize \( \mathcal{AF} \) in the visual and semantic space, respectively.

3) Attribute Representation Optimization: Attribute-level features \( \mathcal{AF} \) are extracted within each individual image, and the features for the same attribute can vary in forms across many images. Attribute prototypes, on the other hand, are generated beyond images from orthogonal attribute semantics. In order for the attribute representation optimization, we present two contrastive losses to let: 1) any attribute-level feature be close to its corresponding attribute prototype and 2) any attribute-level feature be close to other attribute-level features belonging to the same attribute.

a) Contrastive Optimization of Attribute-Level Features Against Attribute Prototypes: Given the set of attribute-level features \( \mathcal{AF} \) for the input image \( x \), we first filter out those attribute-level features \( a_f j \) whose corresponding attention maps \( am j \) have low peak values, i.e., \( \max_{a,b}(am j(a,b)) < T \) \( (T = 9) \); these attention maps normally fail to localize the corresponding attributes. It is possible that this filtering step may remove \( a_f j \) whose corresponding \( am j \) has insignificant peak response, while the object in the image contains the \( j \)th attribute. This, however, will not cause a problem for network training: according to [52], attributes have different importance for the discrimination of a certain object class, only a subset of attributes are necessary for the visual recognition in ZSL.

For a batch of images, we denote the eligible set of attribute-level features after filtering as \( \mathcal{AF} = \{a_f j\}_{j=1}^K \). Given any \( a_f j \), we want to optimize it to be close to its corresponding attribute prototype \( a_p j \) and be far from the other attribute prototypes \( a_p j \neq j \) in the embedding space. A triplet loss function suits this purpose

\[
\mathcal{L}_{\text{attp}} = \frac{1}{K} \sum_{j=1}^{K} \left| d(a_f j, a_p j) - \beta \min_{j' \neq j} d(a_f j, a_p j') \right|
\]  

(4)

where \( d(\cdot, \cdot) \) is the cosine distance (one minus cosine similarity), and \( \beta \) is a hyperparameter to control the extent of pulling \( d(a_f j, a_p j) \) away from \( \min d(a_f j, a_p j') \). \( \cdot \) denotes the ReLU function which enforces \( d(a_f j, a_p j) \) to be smaller than the minimal distance (times \( \beta \)) from \( a_f j \) to other \( a_p j' \).

b) Hard-Example-Based Contrastive Optimization of Attribute Features Across Images: Besides \( \mathcal{L}_{\text{attp}} \), we introduce a new hard-example-based contrastive optimization loss to reinforce the attribute representation learning: it pulls attribute-level features across images corresponding to the same attribute to be close; corresponding to different attributes to be away.

Similar to above, instead of using all the attribute-level features, we only keep those with high-peak values, i.e., \( \mathcal{AF} \). Furthermore, for every \( a_f j \) in \( \mathcal{AF} \), we have other features corresponding to the same attribute to \( a_f j \) as its positives; we keep only hard positives \( \{a_f j\}_{j=1}^U \) whose cosine similarities to \( a_f j \) are smaller than \( t \) \( (t = 0.8) \). Similarly, hard negatives \( \{a_f j\}_{j=1}^V \) are those features that correspond to different attributes to \( a_f j \) and whose cosine similarities to \( a_f j \) are larger than \( 1 - t \). The reason of using hard examples is to leave certain space for the intra-attribute variation. We use the SupCon loss [44] for the objective function

\[
\mathcal{L}_{\text{attp}} = -\frac{1}{K} \sum_{j=1}^{K} \log \tilde{p}_j
\]  

\[
\tilde{p}_j = \frac{\sum_{i=1}^{U} \exp \left( \frac{\cos(a_f j, a_f i_j) - t}{\tau} \right) + \sum_{i=1}^{V} \exp \left( \frac{\cos(a_f j, a_f i_j) + t}{\tau} \right)}{\sum_{i=1}^{U} \exp \left( \frac{\cos(a_f j, a_f i_j) - t}{\tau} \right) + \sum_{i=1}^{V} \exp \left( \frac{\cos(a_f j, a_f i_j) + t}{\tau} \right)}
\]  

(5)

where we iterate \( a_f j \) in \( \mathcal{AF} \) and average the loss values; \( \tau \) is a scalar temperature.

\( \mathcal{L}_{\text{attp}} \) and \( \mathcal{L}_{\text{attp}} \) are defined in the visual space as a result of the semantic-to-visual mapping (from \( as \) to \( ap \)). Apart from them, we follow [12], [13] to define another loss in the semantic space as a result of the visual-to-semantic mapping (from \( am \) to \( cs \)). This loss helps different channel maps in AM focus on different attribute-related regions in image \( x \); we apply Softmax and max-pooling to each feature map of AM and get a vector \( cs \) \( \epsilon \mathbb{R}^{1 \times 1 \times K} \) in the semantic space whose \( j \)th value indicates the maximum response of \( j \)th attribute in the image. Assuming \( x \) belongs to the class \( i \), we minimize the L2 distance between \( cs_i \) and the ground-truth class semantics \( cs_i^g \)

\[
L_{\text{sem}} = \| cs_i - cs_i^g \|^2.
\]  

(6)

E. Loss Function

The overall loss function for our framework is

\[
L = L_{\text{cls}} + \lambda_{\text{attp}} \mathcal{L}_{\text{attp}} + \lambda_{\text{attp}} \mathcal{L}_{\text{attp}} + \lambda_{\text{sem}} \tilde{L}_{\text{sem}}
\]  

(7)

where \( L_{\text{cls}} \) and \( \lambda_{\text{sem}} \) are defined for one image, and we use \( \tilde{L}_{\text{attp}} \) to denote the corresponding average loss in one batch so as to match with \( \mathcal{L}_{\text{attp}} \) and \( \mathcal{L}_{\text{attp}} \). \( \lambda_{\text{attp}} \), \( \lambda_{\text{attp}} \), and \( \lambda_{\text{sem}} \) are the corresponding loss coefficients.

F. Alternative: Transformer-Based Architecture

Inspired by the success of ViT [49], we provide an alternative backbone for our framework using the ViT. This change affects the left part of the framework for class- and attribute-level feature embeddings [see Fig. 2 (bottom)]: given the input image \( x \), it is sliced into \( P \) evenly squared patches of size \( \widetilde{Q} \times \widetilde{Q} \). They are embedded via the transformer encoder to obtain the feature tensor \( F \epsilon \mathbb{R}^{\widetilde{Q} \times \widetilde{Q} \times C} \). Positional embedding is added to patch embedding to keep the position information. Unlike the CNN-based architecture, the class-level feature \( cf \) is directly embedded by adding an extra learnable classification token (CLS). For attribute-level features, we adopt a similar attention-based attribute localization scheme to the CNN-based architecture: \( F \) is passed through a convolutional layer to produce the attended feature tensor \( AM \epsilon \mathbb{R}^{\widetilde{Q} \times \widetilde{Q} \times K} \), whose \( j \)th channel map \( am j \) serves as a soft mask for the \( j \)th attribute localization. AM is bilinearly pooled with \( F \) [see (3)] to obtain the attribute-level features \( \mathcal{AF} = \{a_f j\}_{j=1}^K \).
TABLE I
DATASETS STATISTICS IN TERMS OF GRANULARITY, ATTRIBUTES, CLASSES, AND DATA SPLIT

| Dataset | Granularity | #Attribute | #Class | Total | Train | Test |
|---------|-------------|------------|--------|-------|-------|------|
| CUB     | fine        | 312        | 200    | 50    | 11788 | 7057 | 4731|
| SUN     | fine        | 102        | 717    | 645   | 14340 | 10320| 4020|
| AWA2    | coarse      | 85         | 50     | 40    | 37332 | 23527| 13795|

IV. EXPERIMENTS

A. Dataset and Evaluation Metrics

We evaluate our method on three most widely used datasets CUB [28], SUN [29], and AwA2 [30] and follow the proposed train and test split in [30]. The statistics of the datasets is summarized in Table I: Caltech-UCSD Birds-200-2011 (CUB) [28] is the most fine-grained dataset with 312 attributes. It contains 11 788 images with 150 seen classes and 50 unseen classes. SUN [29] is a large scene dataset which has 14 340 images with 645 seen classes and 72 unseen classes. There are 102 attributes in total. Compared with the former two, AwA2 [30] is a relatively coarse dataset with only 85 attributes but it has 37 332 images which is the most of three datasets. It contains 40 seen classes and ten unseen classes.

We report results in both the ZSL and GZSL settings (Section III-A). In the ZSL setting, we only evaluate the performance on unseen classes and use the top-1 accuracy (T1) as the evaluation metric. In the GZSL setting, we evaluate the performance on both seen and unseen classes and follow [30] to use generalized seen accuracy (AccS), generalized unseen accuracy (AccU), and their generalized harmonic mean (AccH) as evaluate metrics. The former two are top-1 accuracy for seen and unseen classes, respectively, while the last one is obtained by

$$\text{Acc}_H = \frac{2 \cdot \text{Acc}_U \cdot \text{Acc}_S}{\text{Acc}_U + \text{Acc}_S}.$$  (8)

AccH measures the inherent bias toward seen classes [53], which is a more important metric in GZSL.

B. Implementation Details

As a default backbone, we choose the CNN-based architecture, ResNet101 [3], which is pretrained on ImageNet1K (1.28 million images, 1000 classes). The input image resolution is 448 × 448 and global feature cf is of 2048 dimensions. For the transformer-based architecture, the large variant of the ViT [49] is used, which is pretrained on ImageNet21k (14 million images, 21 843 classes). The input image resolution is 224 × 224 and global feature cf is of 1024 dimensions; the patch size is 16 × 16, such that there are 196 patch tokens in total. The hidden size of the prototype generation module is set to 1024. \(\tau\) in (5) is set to 0.4 for CUB, and 0.5 for SUN and AwA2. \(\beta\) in (4) is set to 0.5, and \(\alpha\) in (1) is 25 for all the datasets. \(\lambda_{\text{att}}\), \(\lambda_{\text{ant}}\), and \(\lambda_{\text{sem}}\) in (7) are set as 0.1, 1, and 1, respectively. We choose the SGD optimizer and set the momentum as 0.9, initial learning rate as 0.001, and weight decay as 0.0001. The learning rate is decayed every ten epochs, with the decay factor of 0.5. All the models are trained with synchronized SGD over four GPUs for 20 epochs with a mini-batch of 32. The mini-batch is organized as a 16-way two-shot episode following [13] meaning that we sample two images per class for 16 classes within this mini-batch.

We follow [8] to use the validation set for hyperparameter searching, which is a disjoint set of unseen classes left out from the training set of CUB, AwA2 and SUN, respectively. For each hyperparameter, we perform line search on a few candidate points to find one point that works well for all the datasets.

C. Comparisons to State of the Arts

In Table II, we compare our CoAR-ZSL to the recent state of the arts [12], [13], [15], [17], [20], [23], [25], [26], [33], [39], [41], [45], [50], [54], [55], [56], [57], [58], [59], [60] in both the ZSL and GZSL settings. We partition them into generative methods [17], [20], [33], [45], [54], [55], [56], [57], [58] and nongenerative methods [12], [13], [15], [23], [25], [26], [39], [41], [50], [59], [60] following [12], [13]. Our CoAR-ZSL is a nongenerative method.

1) CNN-Based Architecture: All the methods except for [50] use the same CNN-based ResNet101 backbone with our CoAR-ZSL. CoAR-ZSL significantly outperforms the state-of-the-art CNN-based methods on most of the indicators. In particular with T1 and AccU, which are two important indicators for ZSL and GZSL, it achieves 79.2, 66.7, 74.1 for T1 and 74.0, 43.4, 73.2 for AccH on CUB, SUN, and AwA2, respectively. Also, we would like to point out that our method produces very good results on all the three datasets while the previous best results are spread over different methods on the three datasets.

Unlike those generative-based methods [12], [13], [15], [23], [25], [26], [39], [41], [50], [59], [60], our CoAR-ZSL does not need to synthesize virtual features of unseen classes nor do we need the semantics or visual features of unseen classes during training. If we do a fairer comparison to other nongenerative methods [12], [13], [15], [23], [25], [26], [39], [41], [50], [59], [60], our improvements over the state of the art are even more!! Overall, CoAR-ZSL is apparently the most competitive method.

Besides the comparison on the accuracy, we also provide the analysis of computational efficiency between our method and the state of the art. Following [61], we use FLOPs as the measurement of the computation cost, which is the number of floating-point operations by forwarding a single sample in the network. We compare our method to other methods, i.e., latent feature guided attribute attention (LFGAA) [60], attentive region embedding network (AREN) [26], goal-oriented gaze estimation module (GEM) [13], and attribute prototype network (APN) [24], as they have released their codes and we are all embedding-based methods. For all the methods,
TABLE II
COMPARISON WITH STATE OF THE ART ON CUB, SUN, AND AWA2. WE REPORT TOP-1 ACCURACY (T1) FOR ZSL, ACCu, ACCs, AND ACCh FOR GZSL. METHODS USING CNN-BASED AND TRANSFORMER-BASED BACKBONES ARE COMPARED SEPARATELY. FOR THE FORMER, WE MARK THE BEST AND SECOND BEST RESULTS OF T1 AND ACCh IN RED AND BLUE, RESPECTIVELY. FOR THE LATTER, WE MARK THE BEST RESULTS WITH UNDERLINES IN RED. *INDICATES USING THE TRANSFORMER-BASED BACKBONE

| Method          | CUB |         |         |         |         |         |         |         |
|-----------------|-----|---------|---------|---------|---------|---------|---------|---------|
|                 | ZSL | T1      | Accu    | ACCs    | ACCh    | ZSL     | T1      | Accu    | ACCs    | ACCh    | ZSL     | T1      | Accu    | ACCs    | ACCh    |
| OCD-CVAB [54]   |     | 60.3    | 44.8    | 59.9    | 51.3    | 60.3    | 44.8    | 42.9    | 43.8    | 71.3    | 59.5    | 73.4    | 65.7    |         |
| LoGAN [20]      |     | 60.3    | 48.1    | 59.1    | 53.0    | 62.5    | 44.8    | 37.7    | 40.9    | -       | -       | -       | -       |         |
| CE-GZSL [47]    |     | 75.7    | 63.9    | 66.8    | 65.3    | 63.3    | 48.8    | 38.6    | 43.1    | 70.4    | 63.1    | 78.6    | 70.0    |         |
| FREE [55]       |     | -       | 55.7    | 59.9    | 57.7    | -       | 47.4    | 37.2    | 41.7    | -       | -       | 60.4    | 75.4    | 67.1    |
| E-PGN [56]      |     | 72.4    | 52.0    | 61.1    | 56.2    | -       | -       | -       | -       | 73.4    | 52.6    | 83.5    | 64.6    |         |
| GCM-CF [57]     |     | 72.4    | 61.0    | 59.7    | 60.3    | -       | 47.9    | 37.8    | 42.2    | -       | -       | 60.4    | 75.1    | 67.0    |         |
| TransferIP [58] |     | 72.4    | 52.1    | 53.3    | 52.7    | -       | 32.3    | 24.6    | 27.9    | -       | 76.8    | 66.9    | 71.5    |         |
| HSVN [17]       |     | 72.4    | 52.7    | 58.3    | 55.3    | -       | 46.8    | 39.0    | 43.3    | -       | 56.7    | 79.8    | 66.3    |         |
| IICCE [33]      |     | -       | 67.3    | 65.5    | 66.4    | -       | -       | -       | -       | -       | 65.3    | 82.3    | 72.8    |         |

Non-generator methods

| Method | ZSL | T1 | Accu | ACCs | ACCh |
|--------|-----|----|------|------|------|
| CN ZSL [15] | - | 49.9 | 50.7 | 50.3 | - |
| APZ [41]     | 53.2 | 58.2 | 37.8 | 45.9 | 61.5 |
| DVBE [59]     | - | 53.2 | 60.2 | 56.5 | 75.4 |
| AREN [26]    | 71.8 | 63.7 | 69.0 | 66.9 | 60.6 |
| LFGAA [60]   | 67.6 | 36.2 | 80.9 | 50.0 | 61.5 |
| RGBN [23]    | 76.1 | 60.0 | 73.5 | 66.1 | 63.8 |
| DAZLE [39]   | 66.0 | 56.7 | 59.6 | 58.1 | 59.4 |
| APN [12]     | 72.0 | 65.3 | 69.3 | 67.2 | 61.6 |
| GEM [13]     | 77.8 | 64.8 | 77.1 | 70.4 | 62.8 |
| MDSN [25]    | 76.1 | 68.7 | 67.5 | 64.1 | 65.8 |
| CoAR-ZSL (Ours) | 79.2 | 70.9 | 77.3 | 74.0 | 66.7 |

VIT-ZSL* [50] | - | 67.3 | 75.2 | 71.0 | - |

GEM-ZSL* [13] | 78.1 | 73.7 | 71.5 | 72.6 | 75.3 |

CoAR-ZSL* (Ours) | 79.9 | 72.5 | 76.3 | 74.4 | 79.4 |

Overall, the better accuracy obtained using the transformer-based architecture also comes with three shortcomings compared with that using the CNN-based architecture: 1) the occupied memory is bigger; 2) the inference time is longer; and 3) the pretraining cost is higher. In Table IV, we compare the occupied GPU memory during training, the inference time, and the pretrained dataset size between the CNN- and transformer-based models; specifically, they are ResNet101 and ViT-L [49] in our experiment. The statistics are collected by running both the models on the CUB dataset. One can clearly see that the transformer-based model is more expensive to be pretrained, learned, and deployed.

D. Ablation Study

Ablation study is on all the three datasets, and we report T1 and AccH for ZSL/GZSL, using the CNN backbone.

1) Impact of Attribute Representation Optimization: We show the impact of attribute representation learning by removing \( \tilde{L}_{sem} \), \( L_{attp} \), and \( L_{attf} \) for all the three losses. The results are shown in Table V where it clearly shows that \( \tilde{L}_{sem} \), \( L_{attp} \), and \( L_{attf} \) all help the performance and they are complementary. As a key contribution of our work, we for the first time distinguish the concepts between attribute prototypes and attribute-level features and introduce the contrastive optimization of attribute representations. The benefits of doing this are reflected in the two losses \( L_{attp} \) and \( L_{attf} \) where the former is to learn explicit and robust attribute prototypes while the latter is to reinforce attribute-level features. For instance, if we remove \( L_{attp} \), it means the attribute prototypes will no
TABLE IV
Comparison of ResNet101 and ViT-L in Terms of Occupied GPU Memory, Inference Time, and Pretrained Dataset Size

| Method     | Occupied GPU memory (GB) | Inference time (s) | Pretrained dataset size (million) |
|------------|--------------------------|--------------------|-----------------------------------|
| ResNet101  | 11.8                     | 0.046              | 1.28                              |
| ViT-L      | 13.6                     | 0.233              | 1.42                              |

TABLE V
Ablation Study of Loss Terms in (7)

| Method                          | CUB T1 Acc_H | SUN T1 Acc_H | AwA2 T1 Acc_H |
|---------------------------------|--------------|--------------|---------------|
| CoAR-ZSL                         | 79.2         | 74.0         | 66.7          | 43.4         | 74.1 | 73.2 |
| CoAR-ZSL w/o $\tilde{L}_{cls}$  | 77.0         | 71.0         | 66.4          | 43.1         | 72.4 | 72.9 |
| CoAR-ZSL w/o $L_{attr}$         | 76.6         | 71.1         | 65.6          | 43.4         | 73.1 | 71.1 |
| CoAR-ZSL w/ $L_{attr}$-no-HS    | 77.4         | 70.0         | 65.0          | 41.4         | 72.9 | 72.2 |
| CoAR-ZSL w/ $L_{attr}$-no-HS    | 75.8         | 69.8         | 64.0          | 40.7         | 69.7 | 70.0 |
| CoAR-ZSL w/ $L_{attr}$-no-HS    | 78.1         | 71.8         | 65.5          | 42.5         | 73.7 | 72.7 |

TABLE VI
Ablation Study on Prototype Module Structure

| Method                          | CUB T1 Acc_H | SUN T1 Acc_H | AwA2 T1 Acc_H |
|---------------------------------|--------------|--------------|---------------|
| PM-v1                           | 76.2         | 71.3         | 65.7          | 41.1         | 70.5 | 71.3 |
| PM-v2                           | 74.1         | 69.1         | 64.5          | 40.9         | 67.9 | 70.3 |
| PM w/o CN                       | 72.2         | 66.2         | 63.1          | 37.6         | 65.2 | 68.4 |
| PM                              | 79.2         | 74.0         | 66.7          | 43.4         | 74.1 | 73.2 |

TABLE VII
Ablation Study on Attribute Semantics

| Method                          | CUB T1 Acc_H | SUN T1 Acc_H | AwA2 T1 Acc_H |
|---------------------------------|--------------|--------------|---------------|
| Rnd                             | 74.6         | 69.7         | 63.9          | 40.3         | 67.8 | 69.8 |
| Rnd-ort                         | 76.4         | 72.4         | 64.2          | 41.3         | 68.8 | 70.8 |
| One-hot                         | 79.2         | 74.0         | 66.7          | 43.4         | 74.1 | 73.2 |

TABLE VIII
Ablation Study of Extracting Features From Different Positions of the CNN-Based Backbones

| Method                          | CUB T1 Acc_H | SUN T1 Acc_H | AwA2 T1 Acc_H |
|---------------------------------|--------------|--------------|---------------|
| ResNet101                       | 78.3         | 71.7         | 66.1          | 43.0         | 75.1 | 72.7 |
| 4th stage                       | 77.9         | 72.4         | 64.2          | 42.6         | 74.2 | 72.6 |
| 2nd-4th stages                  | 78.9         | 74.4         | 65.7          | 42.9         | 74.5 | 72.8 |
| 3rd-4th stages                  | 79.2         | 74.0         | 66.7          | 43.4         | 74.1 | 73.2 |
| 1st-4th stages                  | 65.1         | 63.3         | 59.9          | 39.2         | 68.5 | 67.6 |
| VGG19                           | 68.1         | 63.4         | 60.0          | 40.3         | 69.5 | 69.7 |

TABLE IX
Comparison Between Using GAPS and GMPs

| Method | CUB T1 Acc_H | SUN T1 Acc_H | AwA2 T1 Acc_H |
|--------|--------------|--------------|---------------|
| GMP    | 75.5         | 70.5         | 57.5          | 32.7         | 68.3 | 68.6 |
| GAP    | 79.2         | 74.0         | 66.7          | 43.4         | 74.1 | 73.2 |
Fig. 3. Parameter variation in hyperparameter τ in (5). (a) (T1, Acc_H)-τ curve of CUB. (b) (T1, Acc_H)-τ curve of SUN. (c) (T1, Acc_H)-τ curve of AwA2.

Fig. 4. Parameter variation in hyperparameter α in (1). (a) (T1, Acc_H)-α curve of CUB. (b) (T1, Acc_H)-α curve of SUN. (c) (T1, Acc_H)-α curve of AwA2.

Fig. 5. Parameter variation in hyperparameter β in (4). (a) (T1, Acc_H)-β curve of CUB. (b) (T1, Acc_H)-β curve of SUN. (c) (T1, Acc_H)-β curve of AwA2.

Fig. 6. Parameter variation in peak value threshold T. (a) (T1, Acc_H)-T curve of CUB. (b) (T1, Acc_H)-T curve of SUN. (c) (T1, Acc_H)-T curve of AwA2.

4) Attention Maps: We use features from all four stages of ResNet101 (C1–C4) to obtain attention maps AM. It is a common practice to use features from these stages for object recognition task [62], as they contain different levels of...
Fig. 7. Parameter variations for loss coefficients $\lambda_{\text{sem}}$, $\lambda_{\text{attp}}$, and $\lambda_{\text{attf}}$. (a) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{attp}}$ curve of CUB. (b) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{attf}}$ curve of CUB. (c) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{sem}}$ curve of CUB. (d) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{attp}}$ curve of SUN. (e) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{attf}}$ curve of SUN. (f) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{sem}}$ curve of SUN. (g) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{attp}}$ curve of AwA2. (h) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{attf}}$ curve of AwA2. (i) ($T_1$, $\text{Acc}_H$)-$\lambda_{\text{sem}}$ curve of AwA2.

E. Parameter Variation

1) Temperature $\tau$ in (5): In Fig. 3, we evaluate the effect of temperature $\tau$ in (5). We vary it from 0.05 to 1 on the three datasets and report $T_1$ and $\text{Acc}_H$ for ZSL and GZSL, respectively. It can be seen that the best performance occurs with $\tau$ equivalent to 0.4 and 0.6 for CUB and AwA2, respectively. The performance on SUN is rather stable by varying $\tau$. A low temperature will penalize more on hard
negatives but a too low temperature will make no tolerance for outliers which cannot be a good thing. In practice, we set $\tau$ as 0.4 for CUB and 0.6 for SUN and AwA2.

2) Scaling Factor $\alpha$ in (1): We show the impact of scaling factor $\alpha$ in (1) by varying it from 20 to 30 in Fig. 4. CoAR-ZSL achieves the best performance on CUB at $\alpha = 25$ for both $T$ and $\text{Acc}_H$. For SUN and AwA2, the performance variation with different $\alpha$ is rather small on both $T$ and $\text{Acc}_H$. In practice, we set $\alpha$ as 25 for all the datasets for simplicity.

3) Ratio $\beta$ in (4): We also evaluate the effect of ratio $\beta$ in (4). We plot the result in Fig. 5. CoAR-ZSL achieves the best performance on CUB at $\beta = 0.5$ for both $T$ and $\text{Acc}_H$. For SUN and AwA2, the performance variation with different $\beta$ is rather small on both $T$ and $\text{Acc}_H$. In practice, we set $\beta$ as 0.5 for all the datasets for simplicity.

4) Peak Value Threshold $T$ in Section III-D3: We sweep $T$ from 1 to 15 for each dataset. It can be seen in Fig. 6 that: 1) the best result for each dataset can indeed be slightly higher than that of using the default $T$ ($T = 9$) for all the three datasets and 2) with the change in $T$, the result on the CUB varies while the results on SUN and AwA2 are rather stable. This enables us to find one parameter that can work in general well for all the three datasets, which is also good for the model’s generalizability.

5) Loss Weights $\lambda_{\text{attf}}$, $\lambda_{\text{aff}}$ and $\lambda_{\text{cem}}$: For convenience, we perform line search on one loss weight while fixing the other two to 1. In Fig. 7, we show the results by varying each loss weight from 0.05 to 1 on the CUB, AwA2, and SUN datasets. In general, a smaller $\lambda_{\text{attf}}$ compared with $\lambda_{\text{aff}}$ and $\lambda_{\text{cem}}$ (by one order of magnitude approximately) is better for the overall performance. Our default setting is that $\lambda_{\text{attf}}$ is 0.1 while the other two are 1. This, despite not being the optimum, is a fine setting for all the three datasets. Theoretically, one could get better performance by optimizing loss weights via grid searching in each dataset, but this would be too costly and may not be beneficial for the model’s generalizability.

F. Qualitative Results

1) Attribute Localization: To visualize the attention-based attribute localization (Section III-D), we resize and normalize the attribute-related attention maps in AM into the range $[0, 1]$ and draw them onto the original image in Fig. 8. We show the examples using both the CNN-based and transformer-based architectures. In both the figures, CoAR-ZSL can accurately locate attribute-related regions in images.

2) Intermediate Feature Maps in (3): We illustrate the intermediate feature maps after fusion of intermediate layers, after softmax, and after fusion with the image feature tensor $F$. Referring to (3), they correspond to the feature map $am_{j, F}$, Softmax($am_{j, F}$), and $R($Softmax($am_{j, F}$)) $\odot F$, respectively. We draw the attribute “has bill shape dagger” in Fig. 9: one can clearly observe how this attribute is localized in $F$ via the corresponding normalized soft mask (Softmax($am_{j, F}$)). It aligns with the attribute on the bird’s beak.

3) t-SNE Visualization of Attribute Features: To validate the representativeness of the proposed attribute-level features, we draw the t-SNE of 4096 attribute-level features extracted over multiple images using methods of GEM [13] [Fig. 10(a)], APN [12] [Fig. 10(b)], and our CoAR-ZSL [Fig. 10(c)]. For better visualization, we only plot the first 25 attributes in different colors. We can see that the attribute-level features in our model are clearly clustered according to the attribute they belong to while the ones in GEM and APN are rather mixed. Since we only keep attribute-level features with high peak values in their corresponding attribute-related attention maps (AM), it can be seen that the kept features tend to form a number of big clusters for a number of attributes. This is indeed consistent to the attribute feature distribution shown in Fig. 10(d): only a number of attributes have high frequencies. The similar observation is made in [31] that many fine-grained classes contain discriminative information in a few regions.

4) Qualitative Evidence for Selecting High-Response Attention Maps for Contrastive Optimization: We plot three types of activation maps in Fig. 11 in three rows: (a) adding up all the attribute-related attention maps in AM; (b) adding up only high-response attribute-related attention maps in AM; and (c) adding up all the feature maps in $F$. We know that each attention map $am_{j, F}$ in AM signifies the feature response to one attribute $j$. Adding them together can be a reflection of these attributes on certain object class. On the other hand, since the class-level feature $cf$ is actually obtained from $F$, adding all the feature maps in $F$ is indeed a reflection of the object class in the image. Having a look at Fig. 11, activation maps in the second row are clearly cleaner and more object-focused than those in the first row and are visually closer to those in the third row. This suggests that there exist noises in those low-response attention maps in AM which could be brought
Fig. 9. Visualization of intermediate feature maps in (3). (a) input image, (b) after fusion of intermediate layers, (c) after softmax, and (d) after fusing with image tensor $F$.

Fig. 10. (a)–(c) t-SNE visualization of attribute features for GEM [13], APN [24], and our CoAR-ZSL. (d) Attribute feature distribution for the CUB dataset. We plot the frequency of the features for each attribute.

Fig. 11. Visualization of three types of activation maps: (a) adding up all the attribute-related attention maps in AM (first row), (b) adding up only high-response attribute-related attention maps in AM (second row), and (c) adding up all the feature maps in $F$ (third row).

into the activation map if adding them up. While those high-response attention maps in AM represent the real attributes contained in the object class, adding these maps can end up with a proper reflection of attributes on this object class.

V. CONCLUSION

This article proposes a novel embedding-based ZSL framework, CoAR-ZSL, for explicitly learning attribute representations. The representations include both image-specific features and image-agnostic prototypes for attributes. They are contrastively optimized in the network to learn a robust classifier. A hard-example-based contrastive learning scheme is also introduced to reinforce the learning of attribute representations. We use two backbones for CoAR-ZSL, CNN-based and transformer-based, where the latter performs better than the former yet requires more computational cost. Extensive experiments on standard benchmarks demonstrate the superiority of our CoAR-ZSL over state of the art.

One shortcoming of our method is that it relies on human-crafted attributes given in the dataset and cannot be applied to datasets without explicit attribute information. This limits the generalizability of our method. Possible solution may be directly learning shared prototypes across classes to simulate the attributes.

Besides above, there are also other places that can be improved in the future work: for instance, we now use hard samples for the contrastive learning of attribute features. This may be improved using the graph neural network where attribute relationships can be encoded into graph edges, so that attribute features can be optimized and refined via the message passing in the graph.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. NeurIPS, 2012, pp. 1097–1105.
[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.
[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2016, pp. 770–778.
[4] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in Proc. NeurIPS, 2017, pp. 4077–4087.
[5] Y. Yang, F. Wei, M. Shi, and G. Li, “Restoring negative information in few-shot object detection,” in Proc. NeurIPS, 2020, pp. 3521–3532.
[6] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1199–1208.
[59] S. Min, H. Yao, H. Xie, C. Wang, Z.-J. Zha, and Y. Zhang, “Domain-aware visual bias eliminating for generalized zero-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 12661–12670.

[60] Y. Liu, J. Guo, D. Cai, and X. He, “Attribute attention for semantic disambiguation in zero-shot learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 6697–6706.

[61] M. Alwani, Y. Wang, and V. Madhavan, “DECORE: Deep compression with reinforcement learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 12339–12349.

[62] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 936–944.

Yu Du received the B.S. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Precision Instrument. His current research interests include foundation model, multimodal learning, and reinforcement learning.

Miaojing Shi (Senior Member, IEEE) received the Ph.D. degree from Peking University, Beijing, China, in 2015. He held a Post-Doctoral position at the University of Edinburgh, Edinburgh, Scotland, and a Research Scientist at INRIA Rennes, Rennes, France. From 2020 to 2022, he was a Lecturer with the Department of Informatics, King’s College London, London, U.K. Since 2023, he has been a Full Professor at Tongji University, Shanghai, China, and a Visiting Senior Lecturer at King’s College London. He also engaged with a joint Ph.D. program with the University of Oxford, Oxford, U.K. and INRIA Rennes, for a year. He has authored or coauthored more than 50 articles. His current research focus is on visual learning and understanding, with multitask and limited supervision.

Fangyun Wei received the M.S. degree from Peking University, Beijing, China, in 2017. He is a Senior Research SDE at Microsoft Research Asia, Beijing. He has authored more than 30 papers in top conferences and journals including IEEE Conference on Computer Vision and Pattern Recognition (CVPR), International Conference on Computer Vision (ICCV), European Conference on Computer Vision (ECCV), Annual Conference on Neural Information Processing Systems (NeurIPS), and IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). He is the reviewer of CVPR, ICCV, ECCV, International Conference on Machine Learning (ICML), NeurIPS, AAAI Conference on Artificial Intelligence (AAAI), International Joint Conference on Artificial Intelligence (IJCAI), ACM International Conference on Multimedia (ACMMM), and International Journal of Computer Vision (IJCV). His current research interests include unsupervised learning, semi-supervised learning, multimodality learning and visual recognition.

Guoqi Li received the Ph.D. degree from Nanyang Technological University, Singapore, in 2011. From 2011 to 2014, he was a Scientist with the Data Storage Institute and the Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore. From 2014 to 2022, he was an Assistant Professor and an Associate Professor with Tsinghua University, Beijing, China. Since 2022, he has been with the Institute of Automation, Chinese Academy of Sciences, and the University of Chinese Academy of Sciences, where he is currently a Full Professor. He has authored or coauthored more than 180 papers in a number of prestigious journals including Nature, Nature Communications, Science Robotics, Proceedings of the IEEE, and top AI conference such as ICLR, Annual Conference on Neural Information Processing Systems (NeurIPS), International Conference on Machine Learning (ICML), and AAAI Conference on Artificial Intelligence (AAAI). His current research interests include brain-inspired intelligence, neuromorphic computing and spiking neural networks.

Dr. Li has been actively involved in professional services such as serving as a Tutorial Chair, an International Technical Program Committee Member, a PC Member, a Publication Chair, a Track Chair, and a Workshop Chair for several international conferences. He is an Editorial-Board Member for Control and Decision and an Associate Editor for Journal of Control and Decision, Frontiers in Neuroscience: Neuromorphic Engineering, and Neuromorphic Computing and Engineering. He is a reviewer for Mathematical Reviews published by the American Mathematical Society and for a number of prestigious international journals and top AI conferences.