Learning Attentive Pairwise Interaction for Fine-Grained Classification

Peiqin Zhuang,[*1,2] **Yali Wang,**[*1,2] **Yu Qiao**[†1,2]
[1]ShenZhen Key Lab of Computer Vision and Pattern Recognition, SIAT-SenseTime Joint Lab,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2]SIAT Branch, Shenzhen Institute of Artificial Intelligence and Robotics for Society
{pq.zhuang, yl.wang, yu.qiao}@siat.ac.cn

Abstract

Fine-grained classification is a challenging problem, due to subtle differences among highly-confused categories. Most approaches address this difficulty by learning discriminative representation of individual input image. On the other hand, humans can effectively identify contrastive clues by comparing image pairs. Inspired by this fact, this paper proposes a simple but effective Attentive Pairwise Interaction Network (API-Net), which can progressively recognize a pair of fine-grained images by interaction. Specifically, API-Net first learns a mutual feature vector to capture semantic differences in the input pair. It then compares this mutual vector with individual vectors to generate gates for each input image. These distinct gate vectors inherit mutual context on semantic differences, which allow API-Net to attentively capture contrastive clues by pairwise interaction between two images. Additionally, we train API-Net in an end-to-end manner with a score ranking regularization, which can further generalize API-Net by taking feature priorities into account. We conduct extensive experiments on five popular benchmarks in fine-grained classification. API-Net outperforms the recent SOTA methods, i.e., CUB-200-2011 (90.0%), Aircraft (93.9%), Stanford Cars (95.3%), Stanford Dogs (90.3%), and NABirds (88.1%).

1 Introduction

Over the past years, CNNs have achieved remarkable successes for visual recognition (He et al. 2016; Huang et al. 2017). However, these classical models are often limited to distinguish fine-grained categories, due to highly-confused appearances. Subsequently, a number of fine-grained frameworks have been proposed by finding key part regions (Fu, Zheng, and Mei 2017; Zheng et al. 2017; Yang et al. 2018), learning patch relations (Lin, RoyChowdhury, and Maji 2015; Cai, Zuo, and Zhang 2017; Yu et al. 2018), etc. But most of them take individual image as input, which may limit their ability to identify contrastive clues from different images for fine-grained classification.

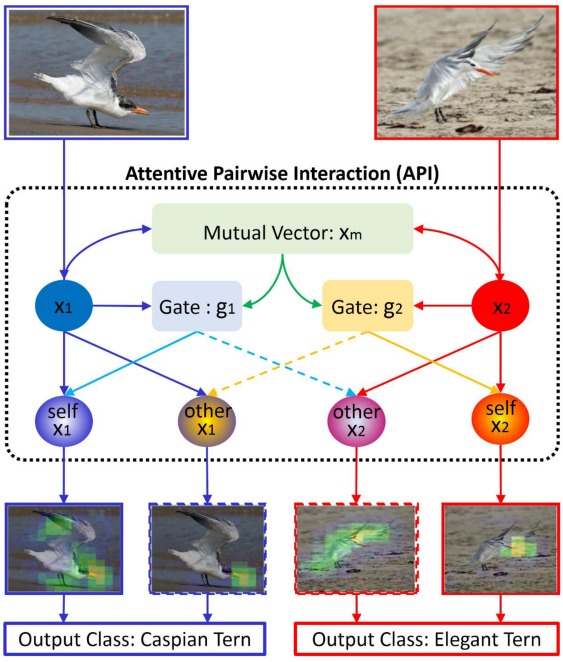

Figure 1: Motivation (Best view in color). *Caspian Tern* and *Elegant Tern* are two highly-confused bird species. Humans often distinguish them by pairwise comparison, instead of checking each individual image alone. First, humans would exploit contrastive clues (e.g. body and mouth) from the image pair, and then check each image with these mutual contexts to further discover distinct attentions on each image. Finally, humans recognize both images via comparing subtle differences jointly. To mimic this capacity, we propose a novel API-Net. More details can be found in Section 1.
通过图像对对比两种海鸥

On the contrary, humans often recognize fine-grained objects by comparing image pairs (Bruner 2017), instead of checking single images alone. For example, *Caspian Tern* and *Elegant Tern* are two highly-confused bird species, as shown in Fig. 1. If we only check individual image alone, it is difficult to recognize which categories it belongs to, especially when the bird is self-occluded with a noisy back-

[*]Equally-contributed first authors
[†]Corresponding author

ground. Alternatively, we often take a pair of images together, and summarize contrastive visual appearances as context, e.g., body, mouth and etc. Then, we check individual images with these mutual contexts, so that we can further understand distinct aspects of each image, e.g.,

the body of the bird is an important part for the left image, while the mouth of the bird is a key characteristic for the right image. With such discriminative guidance, we pay different attentions to the body and the mouth of two birds. Note that, for each bird in the pair, we not only check its prominent part but also have a glance at the distinct part found from the other bird. This comparative interaction can effectively tell us that, *Caspian Tern* has a fatter body, while *Elegant Tern* has a more curved mouth. Consequently, we recognize both images jointly.

To mimic this capacity of human beings, we introduce a novel Attentive Pairwise Interaction Network (API-Net) for fine-grained classification. It can adaptively discover contrastive clues from a pair of fine-grained images, and attentively distinguish them via pair interaction.

More specifically, API can effectively recognize two fine-grained images, by a progressive comparison procedure like human beings. To achieve this goal, API-Net consists of three submodules, i.e., mutual vector learning, gate vector generation, and pairwise interaction. By taking a pair of fine-grained images as input,

API-Net first learns a mutual vector to summarize contrastive clues of input pair as context. Then, it compares mutual vector with individual vectors. This allows API-Net to generate distinct gates, which can effectively highlight semantic differences respectively from the view of each individual image.

Consequently, API-Net uses these gates as discriminative attentions to perform pairwise interaction. In this case, each image can generate two enhanced feature vectors, which are activated respectively from its own gate vector and the gate vector of the other image in the pair. Via an end-to-end training manner with a score-ranking regularization, API-Net can promote the discriminative ability of all these features jointly with different priorities. Additionally, it is worth mentioning that, one can easily embed API into any CNN backbones for fine-grained classification, and flexibly unload it for single-input test images without loss of generalization capacity. Such plug-and-play property makes API as a preferable choice in practice. Finally, we evaluate API-Net on five popular benchmarks in fine-grained recognition, namely CUB-200-2011, Aircraft, Stanford Cars, Stanford Dogs and NABirds. The extensive results show that, API-Net achieves the state-of-the-art performance on all these datasets.

2 Related Works

A number of research works have been recently proposed for fine-grained classification. In the following, we mainly summarize and discuss those related works.

Object Parts Localization. These approaches mainly utilize the pre-defined bounding boxes or part annotations to capture visual details in the local regions (Zhang et al. 2014; Lin et al. 2015). However, collecting such annotations is often labor-intensive or infeasible in practice. Hence, several weakly-supervised localization approaches have been recently proposed by designing complex spatial attention mechanisms (e.g., RA-CNN (Fu, Zheng, and Mei 2017), MA-CNN (Zheng et al. 2017)), learning a bank of discriminative filters (Wang, Morariu, and Davis 2018), guiding region detection with multi-agent cooperation (Yang et al. 2018), etc. But these approaches mainly focus on mining local characteristics of fine-grained images. Consequently, they may lack the capacity of discriminative feature learning.

Discriminative Feature Learning. To learn the representative features, many approaches have been exploited by patch interactions (Lin, RoyChowdhury, and Maji 2015; Cai, Zuo, and Zhang 2017; Yu et al. 2018). A well-known approach is B-CNN (Lin, RoyChowdhury, and Maji 2015), which performs bilinear pooling on the representations of two local patches in an image. Following this direction, several high-order approaches have been proposed via polynomial kernel formulation (Cai, Zuo, and Zhang 2017), cross-layer bilinear representation (Yu et al. 2018), etc. However, these approaches take a single image alone as input, while neglecting comparisons between different images, i.e., an important clue to distinguish highly-confused objects.

Metric Learning. Metric learning refers to the method that uses similarity measurements to model relations between image pairs (Kulis and others 2013). It has been widely used in Face Verification (Schroff, Kalenichenko, and Philbin 2015), Person ReID (Hermans, Beyer, and Leibe 2017) and so on. Recently, it has been introduced for fine-grained classification, e.g., triplet loss design (Zhang et al. 2016a), pairwise confusion regularization (Dubey et al. 2018a), multi-attention multi-class constraint (Sun et al. 2018), etc. However, these approaches mainly leverage metric learning to improve sample distributions in the feature space. Hence, they often lack the adaptation capacity, w.r.t., how to discover visual differences between a pair of images.

Different from previous approaches, our API-Net can adaptively summarize contrastive clues, by learning a mutual vector from an image pair. As a result, we can leverage it as guidance, and attentively distinguish two fine-grained images via pairwise interaction.

3 Attentive Pairwise Interaction

In this section, we describe Attentive Pairwise Interaction Network (API-Net) for fine-grained classification. Our design is partially inspired by the observation that, instead of learning visual concepts alone with individual image, humans often compare a pair of images jointly to distinguish subtle differences between similar objects (Bruner 2017).

To imitate this capacity, we simultaneously take a pair of images as input to API-Net, and progressively distinguish them by three elaborative submodules, i.e., mutual vector learning, gate vector generation, and pairwise interaction. The whole framework is demonstrated in Fig. 2.

Mutual Vector Learning. First, we feed two fine-grained images into a CNN backbone, and extract their D-dimension

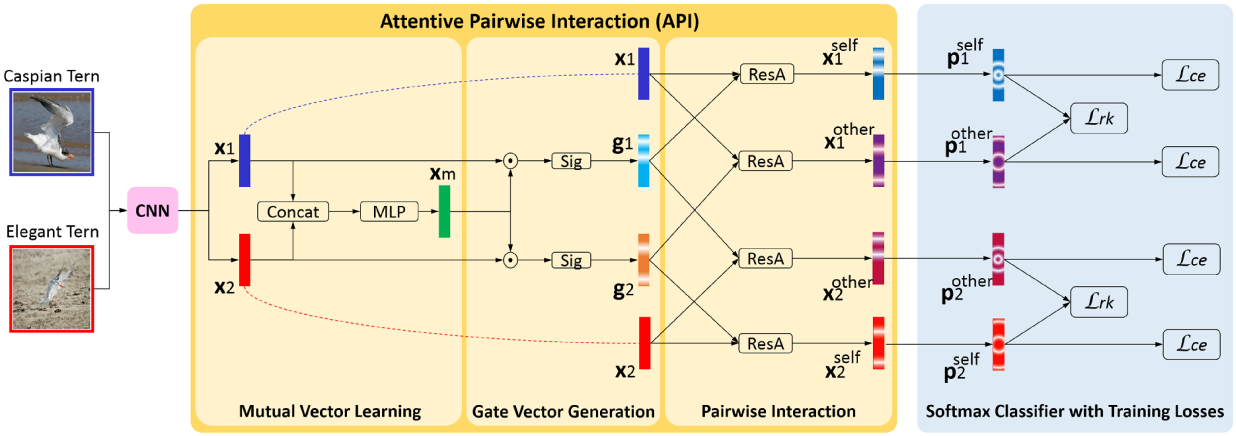

Figure 2: The framework of API-Net (Best view in color). API can progressively recognize a pair of fine-grained images, based on a novel human-like learning procedure. It consists of three submodules. **1. Mutual Vector Learning**. API learns a mutual vector $\mathbf{x}_m$ from individual $\mathbf{x}_1$ and $\mathbf{x}_2$ (Eq. 1). In this case, it can summarize contrastive cues in the pair. **2. Gate Vector Generation**. API further compares $\mathbf{x}_m$ with $\mathbf{x}_1$ and $\mathbf{x}_2$, and generates two distinct gate vectors $\mathbf{g}_i$ (Eq. 2). These gates allow API to discover distinct clues respectively from the view of each individual image. **3. Pairwise Interaction**. API performs pairwise interaction with guidance of gate vectors (Eq. 3-6). Via training API-Net with a score-ranking regularization, we can distinguish all these features jointly with the consideration of feature priorities (Eq. 8-10). Additionally, API is a practical plug-and-play module, i.e., one can combine API with CNNs during training, and flexibly unload it for single-input test images.

feature vectors respectively, i.e., $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^D$. Then, we learn a mutual vector $\mathbf{x}_m \in \mathbb{R}^D$ from individual $\mathbf{x}_1$ and $\mathbf{x}_2$,

$$\mathbf{x}_m = f_m([\mathbf{x}_1, \mathbf{x}_2]), \tag{1}$$

where $f_m(\cdot)$ is a mapping function of $[\mathbf{x}_1, \mathbf{x}_2]$, e.g., a simple MLP works well in our experiments. Since $\mathbf{x}_m$ is adaptively summarized from both $\mathbf{x}_1$ and $\mathbf{x}_2$, it often contains feature channels which indicate high-level contrastive clues (e.g. body and mouth of two birds in Fig.1) in the pair.

Gate Vector Generation. After learning the mutual vector $\mathbf{x}_m$, we propose to compare it with $\mathbf{x}_1$ and $\mathbf{x}_2$. The main reason is that, we should further generate distinct clues respectively from the view of each individual image, in order to distinguish this pair afterwards.

In particular, we perform channel-wise product between $\mathbf{x}_m$ and $\mathbf{x}_i$, so that we can leverage $\mathbf{x}_m$ as guidance to find which channels of individual $\mathbf{x}_i$ may contain contrastive clues. Then, we add a sigmoid function to generate the gate vector $\mathbf{g}_i \in \mathbb{R}^D$,

$$\mathbf{g}_i = sigmoid(\mathbf{x}_m \odot \mathbf{x}_i), \quad i \in \{1, 2\}. \tag{2}$$

As a result, $\mathbf{g}_i$ becomes a discriminative attention which highlights semantic differences with a distinct view of each individual $\mathbf{x}_i$, e.g., the body of the bird is important in one image of Fig.1, while the mouth of the bird is a key in the other image. In our experiments, we evaluate different gating strategies and show effectiveness of our design.

Pairwise Interaction. Next, we perform pairwise interaction by gate vectors. Our design is partially motivated by the fact that, to capture subtle differences in a pair of fine-grained images, human check each image not only with its prominent parts but also with distinct parts from the other image. For this reason, we introduce an interaction mechanism via residual attention,

$$\mathbf{x}_1^{self} = \mathbf{x}_1 + \mathbf{x}_1 \odot \mathbf{g}_1, \tag{3}$$

$$\mathbf{x}_2^{self} = \mathbf{x}_2 + \mathbf{x}_2 \odot \mathbf{g}_2, \tag{4}$$

$$\mathbf{x}_1^{other} = \mathbf{x}_1 + \mathbf{x}_1 \odot \mathbf{g}_2, \tag{5}$$

$$\mathbf{x}_2^{other} = \mathbf{x}_2 + \mathbf{x}_2 \odot \mathbf{g}_1. \tag{6}$$

As we can see, each individual feature $\mathbf{x}_i$ in the pair produces two attentive feature vectors, i.e., $\mathbf{x}_i^{self} \in \mathbb{R}^D$ is highlighted by its own gate vector, and $\mathbf{x}_i^{other} \in \mathbb{R}^D$ is activated by the gate vector of the other image in the pair. In this case, we enhance $\mathbf{x}_i$ with discriminative clues that come from both images. Via distinguishing all these features jointly, we can reduce confusion in this fine-grained pair.

Training. After obtaining four attentive features $\mathbf{x}_i^j$ in the pair (where $i \in \{1, 2\}$, $j \in \{self, other\}$), we feed them into a $softmax$ classifier,

$$\mathbf{p}_i^j = softmax(\mathbf{W}\mathbf{x}_i^j + \mathbf{b}), \tag{7}$$

where $\mathbf{p}_i^j \in \mathbb{R}^C$ is the prediction score vector, C is the number of object categories, and $\{\mathbf{W}, \mathbf{b}\}$ is the parameter set of classifier. To train the whole API-Net effectively, we design the following loss $\mathcal{L}$ for a pair,

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda \mathcal{L}_{rk}, \tag{8}$$

where $\mathcal{L}_{ce}$ is a cross entropy loss, and $\mathcal{L}_{rk}$ is a score ranking regularization with coefficient λ.

1) Cross Entropy Loss. The main loss is the cross entropy loss $\mathcal{L}_{ce}$,

$$\mathcal{L}_{ce} = -\sum\nolimits_{i \in \{1,2\}} \sum\nolimits_{j \in \{self, other\}} \mathbf{y}_i^\top \log(\mathbf{p}_i^j), \tag{9}$$

where $\mathbf{y}_i$ is the one-hot label vector for image i in the pair, $\top$ denotes the vector transposition. By using this loss, API-Net can gradually recognize all the attentive features $\mathbf{x}_i^j$, with supervision of label $\mathbf{y}_i$.

2) Score Ranking Regularization. Additionally, we introduce a hinge loss as the score ranking regularization $\mathcal{L}_{rk}$,

$$\mathcal{L}_{rk} = \sum\nolimits_{i\in\{1,2\}} \max(0, \mathbf{p}_i^{other}(c_i) - \mathbf{p}_i^{self}(c_i) + \epsilon), \quad (10)$$

where $\mathbf{p}_i^j(c_i) \in \mathbb{R}$ is the score obtained from the prediction vector $\mathbf{p}_i^j$, and c_i is the corresponding index associated with the ground truth label of image i. Our motivation of this design is that, x_i^{self} is activated by its own gate vector. Hence, it should be more discriminative to recognize the corresponding image, compared with x_i^{other}.

To take this knowledge into account, we use $\mathcal{L}_{rk}$ to promote the priority of $\mathbf{x}_i^{self}$, i.e., the score difference $\mathbf{p}_i^{self}(c_i) - \mathbf{p}_i^{other}(c_i)$ should be larger than a margin ϵ. With such a regularization, API-Net learns to recognize each image in the pair by adaptively taking feature priorities into account.

3) Pair Construction. Eq. (8) defines the loss for a training pair. Next, we explain how to construct multiple pairs in a batch for end-to-end training. Specifically, we randomly sample N_{cl} classes in a batch. For each class, we randomly sample N_{im} training images. Then, we feed all these images into CNN backbone to generate their feature vectors. For each image, we compare its feature with others in the batch, according to Euclidean distance. As a result, we can construct two pairs for each image, i.e., the intra / inter pair consists of its feature and its most similar feature from intra / inter classes in the batch. This design allows our API-Net for learning to distinguish which are highly-confused or truly-similar pairs. We also investigate different construction strategies in our experiments. Consequently, there are $2 \times N_{cl} \times N_{im}$ pairs in each batch. We pass them into our API module, and summarize the loss $\mathcal{L}$ over all these pairs for end-to-end training.

Testing. We would like to emphasize that, API is a practical plug-and-play module for fine-grained classification. In the training phase, this module can summarize contrastive clues from a pair, which can gradually generalize the discriminative power of CNN representations for each individual image. Hence, in the testing phase, one can simply unload API for single-input test images, without much loss of generalization. Specifically, we feed a test image into the CNN backbone to extract its feature $\mathbf{x}_\star \in \mathbb{R}^D$. Then, we directly put $\mathbf{x}_\star$ into softmax classifier (Eq. 7). The resulting score vector $\mathbf{p}_\star \in \mathbb{R}^C$ is used for label prediction. By doing so, our testing manner is just the same as that of a plain CNN, which largely boosts the value of API-Net for fine-grained classification.

4 Experiments

Data Sets. We evaluate API-Net on five popular fine-grained benchmarks, i.e., CUB-200-2011(Wah et al. 2011), Aircraft(Maji et al. 2013), Stanford Cars(Krause et al. 2013), Stanford Dogs(Khosla et al. 2011) and NABirds (Van Horn et al. 2015). Specifically, CUB-200-2011 / Aircraft / Stanford Cars / Stanford Dogs / NABirds consists of 11,788 / 10,000 / 16,185 / 20,580 / 48,562 images, from 200 bird / 100 airplane / 196 car / 120 dog / 555 bird classes. We use the official train & test splits for evaluation.

Implementation Details. Unless stated otherwise, we implement API-Net as follows. First, we resize each image as 512×512, and crop a 448×448 patch as input to API-Net (train: random cropping, test: center cropping). Furthermore, we use ResNet-101 (pretrained on ImageNet) as CNN backbone, and extract the feature vector $\mathbf{x}_i \in \mathbb{R}^{2048}$ after global pooling average operation. Second, for all the datasets, we randomly sample 30 categories in each batch. For each category, we randomly sample 4 images. For each image, we find its most similar image from its own class and the rest classes, according to Euclidean distance between features. As a result, we obtain an intra pair and an inter pair for each image in the batch. For each pair, we concatenate $\mathbf{x}_1$ and $\mathbf{x}_2$ as input to a two-layer MLP, i.e., FC(4096→512)-FC(512→2048). Consequently, this operation generates the mutual vector $\mathbf{x}_m \in \mathbb{R}^{2048}$. Finally, we implement our network by PyTorch. For all the datasets, the coefficient λ in Eq. (8) is 1.0, and the margin ϵ in the score-ranking regularization is 0.05. We use the standard SGD with momentum of 0.9, weight decay of 0.0005. Furthermore, the initial learning rate is 0.01 (0.001 for Stanford Dogs), and adopt cosine annealing strategy to adjust it. The total number of training epochs is 100 (50 for Stanford Dogs). Besides, during training phase, we freeze the conv layers and only train the newly-added fully-connected layers in the first 8 epochs(12 epochs for Standord Dogs).

4.1 Ablation Studies

To investigate the properties of our API-Net, we evaluate its key designs on CUB-200-2011. For fairness, when we explore different strategies of one design, others are set as the basic strategy stated in the implementation details.

Basic Methods. First of all, we compare our API-Net with Baseline, i.e., the standard ResNet-101 without our API design. As expected, API-Net largely outperforms it in Table 1, showing the effectiveness of API module.

Mutual Vector. To demonstrate the essentiality of $\mathbf{x}_m$ in Eq. (1), we investigate different operations to generate it. **(I)** *Individual* Operation. The key of $\mathbf{x}_m$ is to learn mutual information from both images in the pair. For comparison, we introduce a baseline without it. Specifically, we replace mutual learning in Eq. (1) by individual learning $\tilde{\mathbf{x}}_i = f_m(\mathbf{x}_i)$, and use $\tilde{\mathbf{x}}_i$ to generate the gate vector $\mathbf{g}_i = sigmoid(\tilde{\mathbf{x}}_i)$ where $i \in \{1, 2\}$. **(II)** *Compact Bilinear Pooling* Operation. We generate $\mathbf{x}_m$ by compact bilinear pooling (Gao et al. 2016) between $\mathbf{x}_1$ and $\mathbf{x}_2$. **(III)** *Elementwise* Operations. We perform a number of widely-used elementwise operations to generate $\mathbf{x}_m$, including *Subtract Square* $\mathbf{x}_m = (\mathbf{x}_1 - \mathbf{x}_2)^2$, *Sum* $\mathbf{x}_m = \mathbf{x}_1 + \mathbf{x}_2$, and *Product* $\mathbf{x}_m = \mathbf{x}_1 \odot \mathbf{x}_2$. **(IV)** *Weight Attention* Operation. We use a two-layer MLP to generate the weight of $\mathbf{x}_1$ and $\mathbf{x}_2$, i.e., $[w_1, w_2] = softmax(f_w([\mathbf{x}_1, \mathbf{x}_2]))$, where the output dimension of the 1st FC layer is 512. Then, we use the normalized weight vector as attention to generate the mutual

Method	Baseline	API-Net
Accuracy	85.4	**88.6**

Table 1: Comparison with basic methods. Baseline is the standard ResNet-101 without our API design.

Mutual Vector	Accuracy
Individual	87.9
Compact Bilinear Pooling	88.2
Subtract Square	88.3
Sum	88.5
Product	88.4
Weight Attention	88.4
MLP	**88.6**

Table 2: Different operations of mutual vector (CUB-200-2011). More details can be found in Section 4.1.

Gate Vector	Accuracy	Interaction	Accuracy
Single	87.7	$\mathcal{L}_{ce}$	88.1
Pair	**88.6**	$\mathcal{L}_{ce} + \mathcal{L}_{rk}$	**88.6**

Table 3: Different strategies of gate vector and interaction (CUB-200-2011). More details can be found in Section 4.1.

vector, i.e., $\mathbf{x}_m = \sum w_i \mathbf{x}_i$. **(V)** *MLP* Operation. It is the mapping function described in the implementation details. As shown in Table 2, the *Individual* operation (i.e., the setting without $\mathbf{x}_m$) performs worst. Hence, it is necessary to learn mutual context by $\mathbf{x}_m$, which often plays an important role in finding distinct clues for individual image. Moreover, the performance of $\mathbf{x}_m$ operations is competitive. We choose the simple but effective *MLP* to generate the mutual vector in our experiments.

Gate Vector. We discuss different approaches to generate the gate vector. **(I)** *Single*. Since $\mathbf{x}_m$ inherits mutual characteristics from both $\mathbf{x}_1$ and $\mathbf{x}_2$, a straightforward choice is to use $\mathbf{g}_m = sigmoid(\mathbf{x}_m)$ as a single gate vector. Subsequently, one can train API-Net with attentive features $\mathbf{x}_i^{self} = \mathbf{x}_i + \mathbf{x}_i \odot \mathbf{g}_m$, where $i \in \{1, 2\}$. **(II)** *Pair*. This is the proposed setting in Eq. (2). As shown in Table 3, the *Pair* setting outperforms the *Single* setting. It illustrates that, we have to discover discriminative clues from the view of each image, by comparing $\mathbf{x}_i$ with $\mathbf{x}_m$.

Interaction. We explore different interaction strategies. **(I)** $\mathcal{L}_{ce}$. We train API-Net only with cross entropy loss $\mathcal{L}_{ce}$. In this case, score-ranking regularization $\mathcal{L}_{rk}$ is not used for training. **(II)** $\mathcal{L}_{ce} + \mathcal{L}_{rk}$. We train API-Net with the proposed loss $\mathcal{L}$. In Table 3, our proposed loss performs better. It illustrates that, $\mathcal{L}_{rk}$ can further generalize pairwise interaction with different feature priorities.

Pair Construction. We investigate different strategies to construct input image pairs. **(I)** *Random*. We randomly sample 240 image pairs in a batch. **(II)** *Class-Image*. We sample 240 image pairs, according to class (i.e., Intra / Inter) and Euclidean distance between features (i.e., Similar(S) / Dissimilar(D)). This can generate 8 *Class-Image* settings in Table 4. For example, we randomly sample 30 classes in a batch. For each class, we randomly sample 4 images. For each image, we construct 2 pairs, i.e., the intra / inter pair consists of this image and its most similar image from intra / inter classes. This is denoted as Intra(S) & Inter(S). The results of different settings are shown in Table 4. First, most *Class-Image* settings outperform the *Random* setting. It illustrates that, one should take the prior knowledge of class and similarity into account, when constructing image pairs. Second, Inter(S) outperforms Inter(D), no matter which the intra setting is. The reason is that, each pair in Inter(S) / Inter(D) consists of two most similar / dissimilar images from different classes, i.e., the pairs in Inter(S) increases the difficulty of both being recognized correctly at the same time. By checking such pairs in Inter(S), API-Net can be trained to distinguish subtle semantic differences. Third, the performance is competitive between Intra(S) and Intra(D), no matter which the inter setting is. This is mainly because intra pairs are with the same label. API-Net does not need to put much effort to recognize why these pairs are similar. Finally, the settings with both intra and inter pairs outperform those with only intra or inter pairs. It is credited to the fact that, API-Net can gradually distinguish which are highly-confused or truly-similar pairs, by leveraging both intra and

Pair Construction	Intra	Inter	Accuracy
Random	-	-	86.4
Class-Image	-	D	85.4
	-	S	87.2
	D	-	87.1
	S	-	87.6
	D	D	87.0
	S	D	87.4
	D	S	88.3
	S	S	**88.6**

Table 4: Different strategies of pair construction (CUB-200-2011). *Random*: We randomly sample 240 image pairs in a batch. *Class-Image*: We sample 240 image pairs in a batch, according to class (i.e., Intra / Inter) and Euclidean distance (i.e., Similar(S) / Dissimilar(D)). This can generate 8 *Class-Image* settings. For example, we randomly sample 30 classes in a batch. For each class, we randomly sample 4 images. For each image, we construct 2 pairs, i.e., the intra / inter pair consists of this image and its most similar image from intra / inter classes. This is denoted as Intra(S) & Inter(S). More explanations can be found in Section 4.1.

Class Size (N_{cl})	N_{cl}=10	N_{cl}=20	N_{cl}=30
Accuracy	83.5	87.0	**88.6**
Image Size (N_{im})	N_{im}=2	N_{im}=3	N_{im}=4
Accuracy	88.1	88.2	**88.6**
(N_{cl}, N_{im})	(24, 5)	(30, 4)	(40, 3)
Accuracy	87.7	**88.6**	88.2

Table 5: Class Size & Image Size (CUB-200-2011). After choosing Intra(S) & Inter(S) in the *Class-Image* rule, we further explore the number of sampled classes and images in each batch. When varying the class/image size, we fix the image/class size as 4/30.

Method	Backbone	Extra S.	CUB
DeepLAC (Lin et al. 2015)	AlexNet	Yes	80.3
Part-RCNN (Zhang et al. 2014)		Yes	81.6
PA-CNN (Krause et al. 2015)	VGGNet-19	Yes	82.8
MG-CNN (Wang et al. 2015)		Yes	83.0
FCAN (Liu et al. 2016)	ResNet-50	Yes	84.7
TA-FGVC (Li et al. 2018a)		Yes	88.1
HSE (Chen et al. 2018)		Yes	88.1
PDFR (Zhang et al. 2016b)	VGGNet-16	No	84.5
Grassmann Pool(Wei et al. 2018)		No	85.8
KP (Cui et al. 2017)		No	86.2
HBP (Yu et al. 2018)		No	87.1
G^2DeNet (Wang, Li, and Zhang 2017)		No	87.1
MG-CNN (Wang et al. 2015)	VGGNet-19	No	81.7
B-CNN (Lin, RoyChowdhury, and Maji 2015)		No	84.1
RACNN (Fu, Zheng, and Mei 2017)		No	85.3
MACNN (Zheng et al. 2017)		No	86.5
Deep KSPD (Engin et al. 2018)		No	86.5
ST-CNN (Jaderberg et al. 2015)	GoogleNet	No	84.1
FCAN (Liu et al. 2016)	ResNet-50	No	84.3
DFL-CNN (Wang, Morariu, and Davis 2018)		No	87.4
NTS-Net (Yang et al. 2018)		No	87.5
MAMC (Sun et al. 2018)	ResNet-101	No	86.5
TripletNet (Hoffer and Ailon 2015)		No	86.6
iSQRT-COV (Li et al. 2018b)		No	88.7
MaxEnt (Dubey et al. 2018b)	DenseNet-161	No	86.5
PC (Dubey et al. 2018a)		No	86.9
Our API-Net	ResNet-50	No	87.7
Our API-Net	ResNet-101	No	88.6
Our API-Net	DenseNet-161	No	**90.0**

Table 6: Comparison with The-State-of-The-Art (CUB-200-2011). Extra S.: Extra Supervision.

Method	Backbone	Extra S.	Aircraft
BoT (Wang et al. 2016)	VGGNet-16	Yes	88.4
MG-CNN (Wang et al. 2015)	VGGNet-19	Yes	86.6
KP (Cui et al. 2017)	VGGNet-16	No	86.9
LRBP (Kong and Fowlkes 2017)		No	87.3
G^2DeNet (Wang, Li, and Zhang 2017)		No	89.0
Grassmann Pool(Wei et al. 2018)		No	89.8
HBP (Yu et al. 2018)		No	90.3
DFL-CNN (Wang, Morariu, and Davis 2018)		No	92.0
B-CNN (Lin, RoyChowdhury, and Maji 2015)	VGGNet-19	No	84.1
RACNN (Fu, Zheng, and Mei 2017)		No	88.4
MACNN (Zheng et al. 2017)		No	89.9
Deep KSPD (Engin et al. 2018)		No	91.5
NTS-Net (Yang et al. 2018)	ResNet-50	No	91.4
iSQRT-COV (Li et al. 2018b)	ResNet-101	No	91.4
PC (Dubey et al. 2018a)	DenseNet-161	No	89.2
MaxEnt (Dubey et al. 2018b)		No	89.8
Our API-Net	ResNet-50	No	93.0
Our API-Net	ResNet-101	No	93.4
Our API-Net	DenseNet-161	No	**93.9**

Table 7: Comparison with The-State-of-The-Art (Aircraft). Extra S.: Extra Supervision.

inter pairs. We choose the setting with the best performance, i.e., Intra(S) & Inter(S) in our experiment.

Class Size & Image Size. After choosing Intra(S) & Inter(S) in the *Class-Image* setting, we further explore the number of sampled classes and images in each batch. When varying the class/image size, we fix the image/class size as 4/30. The results are shown in Table 5. One can see that, API-Net is more sensitive to class size than image size. The main reason is that, more classes often produce richer diversity of images pairs. We choose the best setting in our experiment, i.e., class size=30 and image size=4.

4.2 Comparison with The-State-of-The-Art

We compare API-Net with a number of recent works on five widely-used benchmarks in Table 6-10. First, API-Net outperforms the object-part-localization approaches such as RACNN (Fu, Zheng, and Mei 2017) and MACNN (Zheng et al. 2017), showing the importance of API-Net for discriminative feature learning. Second, API-Net outperforms the patch-interaction approaches such as B-CNN (Lin, RoyChowdhury, and Maji 2015), HBP (Yu et al. 2018). It demonstrates that, we need to put more efforts to distinguish two different images jointly, instead of learning with individual image alone. Third, API-Net outperforms the metric-learning approaches such as PC (Dubey et al. 2018a). Particularly, we also re-implement TripletNet (Hoffer and Ailon 2015) in Table 6, a classical approach optimized by extra triplet loss, and compare it with API-Net. They all illustrate that, we should further exploit how to discover differential visual clues in the image pair, instead of straightforwardly regularizing the sample distribution in the feature space. Fourth, API-Net even outperforms the approaches with extra supervision such as localization annotations (Zhang et al. 2014), super-category labels (Chen et al. 2018), or text (Li et al. 2018a). It shows the effectiveness of API mod-

Method	Backbone	Extra S.	Cars
BoT (Wang et al. 2016)	VGGNet-16	Yes	92.5
PA-CNN (Krause et al. 2015)	VGGNet-19	Yes	92.8
FCAN (Liu et al. 2016)	ResNet-50	Yes	91.3
KP (Cui et al. 2017)	VGGNet-16	No	92.4
G^2DeNet (Wang, Li, and Zhang 2017)		No	92.5
Grassmann Pool(Wei et al. 2018)		No	92.8
HBP (Yu et al. 2018)		No	93.7
DFL-CNN (Wang, Morariu, and Davis 2018)		No	93.8
B-CNN (Lin, RoyChowdhury, and Maji 2015)	VGGNet-19	No	91.3
RACNN (Fu, Zheng, and Mei 2017)		No	92.5
MACNN (Zheng et al. 2017)		No	92.8
Deep KSPD (Engin et al. 2018)		No	93.2
FCAN (Liu et al. 2016)	ResNet-50	No	89.1
NTS-Net (Yang et al. 2018)		No	93.9
MAMC (Sun et al. 2018)	ResNet-101	No	93.0
iSQRT-COV (Li et al. 2018b)		No	93.3
PC(Dubey et al. 2018a)	DenseNet-161	No	92.9
MaxEnt (Dubey et al. 2018b)		No	93.0
Our API-Net	ResNet-50	No	94.8
Our API-Net	ResNet-101	No	94.9
Our API-Net	DenseNet-161	No	**95.3**

Table 8: Comparison with The-State-of-The-Art (Stanford Cars). Extra S.: Extra Supervision.

Method	Backbone	Extra S.	Dogs
TA-FGVC (Li et al. 2018a)	ResNet-50	Yes	88.9
PDFR (Zhang et al. 2016b)	AlexNet	No	72.0
DVAN (Zhao et al. 2017)	VGGNet-16	No	81.5
B-CNN (Lin, RoyChowdhury, and Maji 2015)	VGGNet-19	No	82.1
RACNN (Fu, Zheng, and Mei 2017)		No	87.3
FCAN (Liu et al. 2016)	ResNet-50	No	84.2
MAMC (Sun et al. 2018)	ResNet-101	No	85.2
MaxEnt (Dubey et al. 2018b)	DenseNet-161	No	83.6
PC (Dubey et al. 2018a)		No	83.8
Our API-Net	ResNet-50	No	88.3
Our API-Net	ResNet-101	No	**90.3**
Our API-Net	DenseNet-161	No	89.4

Table 9: Comparison with The-State-of-The-Art (Stanford Dogs). Extra S.: Extra Supervision.

Method	Backbone	Extra S.	NABirds
Van et al. (Van Horn et al. 2015)	AlexNet	Yes	75.0
Branson et al. (Branson et al. 2014)	AlexNet	No	35.7
PC (Dubey et al. 2018a)	DenseNet-161	No	82.8
MaxEnt (Dubey et al. 2018b)		No	83.0
Our API-Net	ResNet-50	No	86.2
Our API-Net	ResNet-101	No	86.6
Our API-Net	DenseNet-161	No	**88.1**

Table 10: Comparison with The-State-of-The-Art (NABirds). Extra S.: Extra Supervision.

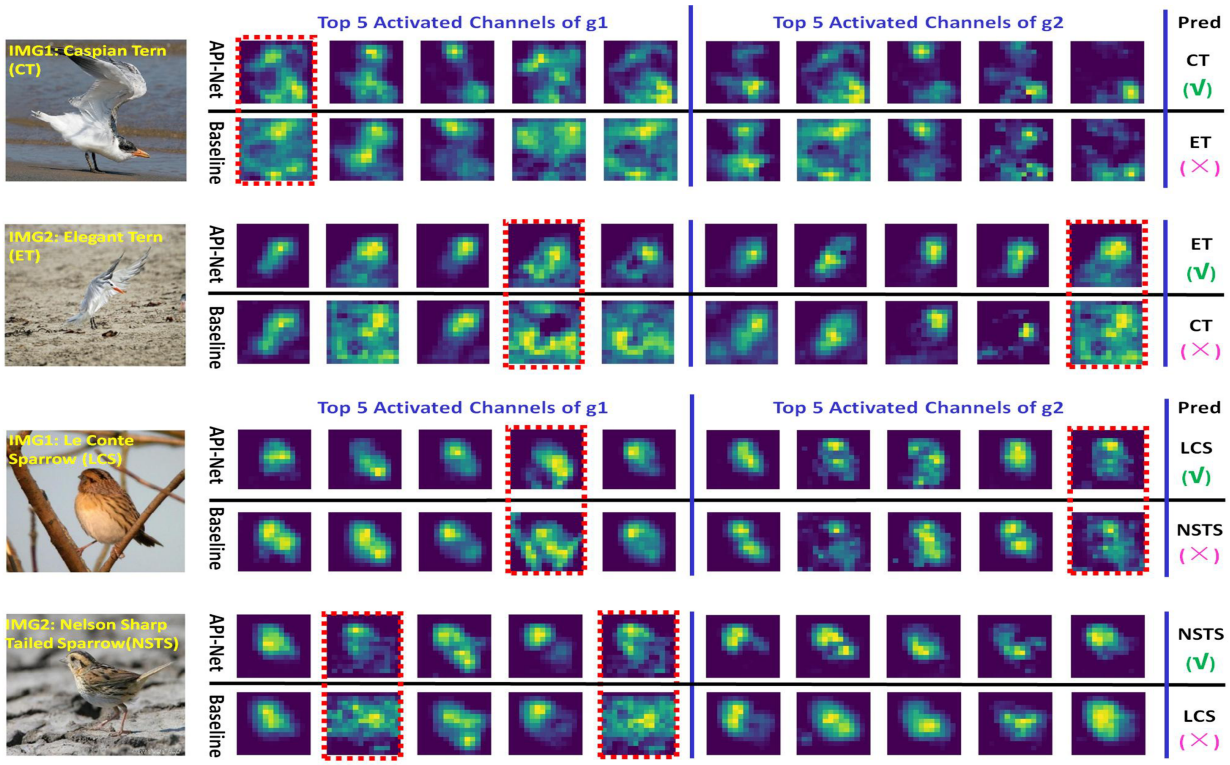

Figure 3: Visualization. As shown in red-dashed boxes, many feature maps of Baseline are confused or noisy, e.g., the object regions are blurring, or certain background regions are activated. On the contrary, our API-Net can effectively discover and then distinguish discriminative clues via attentive pairwise interaction. More explanations can be found in Section 4.3.

ule. Finally, without complex architectures and multi-stage learning in the previous works, our API module can be easily integrated into the standard CNN training procedure and serve as a plug-and-play unit for discovering subtle visual differences. Hence, API-Net is a concise and practical deep learning approach for fine-grained classification.

4.3 Visualization

We further visualize API-Net in Fig. 3. First, we choose two pairs from the highly-confused categories in CUB-200-2011, i.e., *Caspian Tern* (CT) vs. *Elegant Tern* (ET), and *Le Conte Sparrow* (LCS) vs. *Nelson Sharp Tailed Sparrow* (NSTS). Second, we feed each pair into API-Net, and find top-5 activated channels of gate vectors $\mathbf{g}_1$ and $\mathbf{g}_2$. Subsequently, we show the corresponding feature maps (14×14) before global pooling. Additionally, we show the corresponding feature maps in Baseline, i.e., ResNet-101 without attentive pairwise interaction.

As shown in red-dashed boxes of Fig. 3, many feature maps of Baseline are confused or noisy, e.g., the object regions are blurring, or certain background regions are activated. On the contrary, our API-Net can effectively discover distinct features, e.g., $\mathbf{g}_1$ mainly focuses on the body of *Caspian Tern*, while $\mathbf{g}_2$ mainly focuses on the mouth of *Elegant Tern*. These contrastive clues allow API to correctly distinguish such two birds.

Additionally, it is interesting to mention that, API-Net can automatically pay attention to discriminative object parts in the feature maps, even though it mainly works on high-level feature vectors. This observation indicates that, CNN can be well generalized via attentive pairwise interaction. As expected, our API-Net successfully recognizes all these pairs, while Baseline makes wrong predictions.

5 Conclusion

In this paper, we propose a novel Attentive Pairwise Interaction Network (API-Net) for fine-grained classification. It can adaptively discover contrastive cues from a pair of images, and attentively distinguish them via pairwise interaction. The results on five popular fine-grained benchmarks show that, API-Net achieves the state-of-the-art performance.

6 Acknowledgments

This work is partially supported by the National Key Research and Development Program of China (No. 2016YFC1400704), and National Natural Science Foundation of China (61876176, U1713208), Shenzhen Basic Research Program (JCYJ20170818164704758, CXB201104220032A), the Joint Lab of CAS-HK, Shenzhen Institute of Artificial Intelligence and Robotics for Society.

References

[Branson et al. 2014] Branson, S.; Van Horn, G.; Belongie, S.; and Perona, P. 2014. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*.

[Bruner 2017] Bruner, J. 2017. *A study of thinking*. Routledge.

[Cai, Zuo, and Zhang 2017] Cai, S.; Zuo, W.; and Zhang, L. 2017. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *ICCV*, 511–520.

[Chen et al. 2018] Chen, T.; Wu, W.; Gao, Y.; Dong, L.; Luo, X.; and Lin, L. 2018. Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding. *arXiv preprint arXiv:1808.04505*.

[Cui et al. 2017] Cui, Y.; Zhou, F.; Wang, J.; Liu, X.; Lin, Y.; and Belongie, S. 2017. Kernel pooling for convolutional neural networks. In *CVPR*, 2921–2930.

[Dubey et al. 2018a] Dubey, A.; Gupta, O.; Guo, P.; Raskar, R.; Farrell, R.; and Naik, N. 2018a. Pairwise confusion for fine-grained visual classification. In *ECCV*, 70–86.

[Dubey et al. 2018b] Dubey, A.; Gupta, O.; Raskar, R.; and Naik, N. 2018b. Maximum-entropy fine grained classification. In *NIPS*.

[Engin et al. 2018] Engin, M.; Wang, L.; Zhou, L.; and Liu, X. 2018. Deepkspd: learning kernel-matrix-based spd representation for fine-grained image recognition. In *ECCV*, 612–627.

[Fu, Zheng, and Mei 2017] Fu, J.; Zheng, H.; and Mei, T. 2017. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 4438–4446.

[Gao et al. 2016] Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T. 2016. Compact bilinear pooling. In *CVPR*, 317–326.

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

[Hermans, Beyer, and Leibe 2017] Hermans, A.; Beyer, L.; and Leibe, B. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

[Hoffer and Ailon 2015] Hoffer, E., and Ailon, N. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, 84–92. Springer.

[Huang et al. 2017] Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*, 4700–4708.

[Jaderberg et al. 2015] Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *NIPS*, 2017–2025.

[Khosla et al. 2011] Khosla, A.; Jayadevaprakash, N.; Yao, B.; and Li, F.-F. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshops*.

[Kong and Fowlkes 2017] Kong, S., and Fowlkes, C. 2017. Low-rank bilinear pooling for fine-grained classification. In *CVPR*, 365–374.

[Krause et al. 2013] Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *ICCV Workshops*, 554–561.

[Krause et al. 2015] Krause, J.; Jin, H.; Yang, J.; and Fei-Fei, L. 2015. Fine-grained recognition without part annotations. In *CVPR*, 5546–5555.

[Kulis and others 2013] Kulis, B., et al. 2013. Metric learning: A survey. *Foundations and Trends® in Machine Learning* 5(4):287–364.

[Li et al. 2018a] Li, J.; Zhu, L.; Huang, Z.; Lu, K.; and Zhao, J. 2018a. I read, i saw, i tell: texts assisted fine-grained visual classification. In *ACMMM*, 663–671.

[Li et al. 2018b] Li, P.; Xie, J.; Wang, Q.; and Gao, Z. 2018b. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *CVPR*, 947–955.

[Lin et al. 2015] Lin, D.; Shen, X.; Lu, C.; and Jia, J. 2015. Deep lac: Deep localization, alignment and classification for fine-grained recognition. In *CVPR*, 1666–1674.

[Lin, RoyChowdhury, and Maji 2015] Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 1449–1457.

[Liu et al. 2016] Liu, X.; Xia, T.; Wang, J.; Yang, Y.; Zhou, F.; and Lin, Y. 2016. Fully convolutional attention networks for fine-grained recognition. *arXiv preprint arXiv:1603.06765*.

[Maji et al. 2013] Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

[Schroff, Kalenichenko, and Philbin 2015] Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 815–823.

[Sun et al. 2018] Sun, M.; Yuan, Y.; Zhou, F.; and Ding, E. 2018. Multi-attention multi-class constraint for fine-grained image recognition. In *ECCV*, 805–821.

[Van Horn et al. 2015] Van Horn, G.; Branson, S.; Farrell, R.; Haber, S.; Barry, J.; Ipeirotis, P.; Perona, P.; and Belongie, S. 2015. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *CVPR*, 595–604.

[Wah et al. 2011] Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*.

[Wang et al. 2015] Wang, D.; Shen, Z.; Shao, J.; Zhang, W.; Xue, X.; and Zhang, Z. 2015. Multiple granularity descriptors for fine-grained categorization. In *ICCV*, 2399–2406.

[Wang et al. 2016] Wang, Y.; Choi, J.; Morariu, V.; and Davis, L. S. 2016. Mining discriminative triplets of patches for fine-grained classification. In *CVPR*, 1163–1172.

[Wang, Li, and Zhang 2017] Wang, Q.; Li, P.; and Zhang, L. 2017. G2denet: Global gaussian distribution embedding network and its application to visual recognition. In *CVPR*, 2730–2739.

[Wang, Morariu, and Davis 2018] Wang, Y.; Morariu, V. I.; and Davis, L. S. 2018. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *CVPR*, 4148–4157.

[Wei et al. 2018] Wei, X.; Zhang, Y.; Gong, Y.; Zhang, J.; and Zheng, N. 2018. Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In *ECCV*, 355–370.

[Yang et al. 2018] Yang, Z.; Luo, T.; Wang, D.; Hu, Z.; Gao, J.; and Wang, L. 2018. Learning to navigate for fine-grained classification. In *ECCV*, 420–435.

[Yu et al. 2018] Yu, C.; Zhao, X.; Zheng, Q.; Zhang, P.; and You, X. 2018. Hierarchical bilinear pooling for fine-grained visual recognition. In *ECCV*, 574–589.

[Zhang et al. 2014] Zhang, N.; Donahue, J.; Girshick, R.; and Darrell, T. 2014. Part-based r-cnns for fine-grained category detection. In *ECCV*, 834–849. Springer.

[Zhang et al. 2016a] Zhang, X.; Zhou, F.; Lin, Y.; and Zhang, S. 2016a. Embedding label structures for fine-grained feature representation. In *CVPR*, 1114–1123.

[Zhang et al. 2016b] Zhang, X.; Xiong, H.; Zhou, W.; Lin, W.; and

Tian, Q. 2016b. Picking deep filter responses for fine-grained image recognition. In *CVPR*, 1134–1142.

[Zhao et al. 2017] Zhao, B.; Wu, X.; Feng, J.; Peng, Q.; and Yan, S. 2017. Diversified visual attention networks for fine-grained object classification. *TMM* 19(6):1245–1256.

[Zheng et al. 2017] Zheng, H.; Fu, J.; Mei, T.; and Luo, J. 2017. Learning multi-attention convolutional neural network for fine-grained image recognition. In *ICCV*, 5209–5217.