Fine-Grained Visual Classification via Progressive Multi-Granularity Training of Jigsaw Patches

Ruoyi Du[1][0000−0001−8372−5637], Dongliang Chang[1][0000−0002−4081−3001], Ayan Kumar Bhunia[2][0000−0001−7928−7646], Jiyang Xie[1][0000−0003−3659−9476], Zhanyu Ma[1⋆][0000−0003−2950−2488], Yi-Zhe Song[2][0000−0001−5908−3275], and Jun Guo[1]

[1] Pattern Recognition and Intelligent System Laboratory, School of Artificial Intelligence, Beijing University of Posts and Telecommunications
{beiyoudry,changdongliang,xiejiyang2013,mazhanyu,guojun}@bupt.edu.cn
[2] SketchX, CVSSP, University of Surrey
{a.bhunia,y.song}@surrey.ac.uk

Abstract. Fine-grained visual classification (FGVC) is much more challenging than traditional classification tasks due to the inherently subtle intra-class object variations. Recent works are mainly part-driven (either explicitly or implicitly), with the assumption that fine-grained information naturally rests within the parts. In this paper, we take a different stance, and show that part operations are not strictly necessary – the key lies with encouraging the network to learn at different granularities and progressively fusing multi-granularity features together. In particular, we propose: (i) a progressive training strategy that effectively fuses features from different granularities, and (ii) a random jigsaw patch generator that encourages the network to learn features at specific granularities. We evaluate on several standard FGVC benchmark datasets, and show the proposed method consistently outperforms existing alternatives or delivers competitive results. The code is available at *https://github.com/PRIS-CV/PMG-Progressive-Multi-Granularity-Training*.

1 Introduction

Fine-grained visual classification (FGVC) aims at identifying sub-classes of a given object category, *e.g.*, different species of birds, and models of cars and aircrafts. It is a much more challenging problem than traditional classification due to the inherently subtle intra-class object variations amongst sub-categories. Most effective solutions to date rely on extracting fine-grained feature representations at local discriminative regions, either by explicitly detecting semantic parts [11, 41, 38, 12, 39] or implicitly via saliency localization [33, 10, 4, 25]. It follows that such locally discriminative features are collectively fused to perform final classification.

⋆ Corresponding author

Early work mostly finds discriminative regions with the assistance of manual annotations [2, 21, 37, 40, 16]. However, human annotations are difficult to obtain, and can often be error-prone resulting in performance degradations [41]. Research focus has consequently shifted to training models in a weakly-supervised manner given only category labels [41, 38, 33, 26, 4]. Success behind these models can be largely attributed to being able to locate more discriminative local regions for downstream classification. However little or no effort has been made towards (i) at which granularities are these local regions most discriminative, e.g., head or beak of a bird, and (ii) how can information across different granularities be fused together to classification accuracy, e.g., can do head and beak work together.

Information cross various granularities is however helpful for avoiding the effect of large intra-class variations. For example, experts sometimes need to identify a bird using *both* the overall structure of a bird's head, and finer details such as the shape of its beak. That is, it is often not sufficient to identify discriminative parts, but also how these parts interact amongst each other in a complementary manner. Very recent research has focused on the "zooming-in" factor [11, 39], i.e., not just identifying parts, but also focusing on the truly discriminative regions within each part (e.g., the beak, more than the head). Yet these methods mostly focuses on a few parts and ignores others as zooming in beyond simple fusion. More importantly, they do not consider how features from different zoomed-in parts can be fused together in a synergistic manner. Different to these approaches, we further argue that, one not only needs to identify parts and their most discriminative granularities, but meanwhile how parts at different granularities can be effectively merged.

In this paper, we take an alternative stance towards fine-grained classification. We do not explicitly, nor implicitly attempt to mine fine-grained feature representations from parts (or their zoomed-in versions). Instead, we approach the problem with the hypothesis that the fine-grained discriminative information lies *naturally* within different visual granularities – it is all about encouraging the network to learn at different granularities and simultaneously fusing multi-granularity features together. This can be better explained by Figure 1.

More specifically, we propose a consolidated framework that accommodates part granularity learning and cross-granularity feature fusion simultaneously. This is achieved through two components that work synergistically with each other: (i) a progressive training strategy that effectively fuses features from different granularities, and (ii) a random jigsaw patch generator that encourages the network to learn features at specific granularities. Note that we refrain from using "scale" since we do not apply Gaussian blur filters on image patches, rather we evenly divide and shuffle image patches to form different granularity levels.

As the first contribution, we propose a multi-granularity progressive training framework to learn the complementary information across different granularities. This differs significantly to prior art where parts are first detected, and later fused in an ad-hoc manner. Our progressive framework works in steps during training, where at each step the training focuses on cultivating granularity-specific

Fig. 1. Illustration of features learned by general methods (a and b) and our proposed method (c and d). (a) Traditional convolution neural networks trained with cross entropy (CE) loss tend to find the most discriminative parts. (b) Other state-of-the-art methods focus on how to find more discriminative parts. (c) Our proposed progressive training (Here we use last three stages for explanation.) gradually locates discriminative information from low stages to deeper stages. And features extracted from all trained stages are concatenated together to ensure complementary relationships are fully explored, which is represented by "Stage Concat." (d) With assistance of jigsaw puzzle generator the granularity of parts learned at each step are restricted inside patches.

information with a corresponding stage of the network. We start with finer granularities which are more stable, gradually move onto coarser ones, which avoids the confusion made by large intra-class variations that appear in large regions. On its own, this is akin to a "zooming out" operation, where the network would focus on a local region, then zoom out a larger patch surrounding this local region, and finish when we reach the whole image. More specifically, when each training step ends, the parameters trained at the current step will pass onto the next training step as its parameter initialization. This passing operation essentially enables the network to mine information of larger granularity based on the region learned in its previous training step. Features extracted from all stages are concatenated only at the last step to further ensure complementary relationships are fully explored.

However, applying progressive training naively would not benefit fine-grained feature learning. This is because the mulit-granularity information learned via progressive training may tend to focus on the similar region. As the second contribution, we tackle this problem by introducing a jigsaw puzzle generator to form different granularity levels at each training step, and only the last step is still trained with original images. This effectively encourage the model to operate on patch-level, where patch sizes are specific to a particular granularity. It essentially forces each stage of the network to focus on local patches other than holistically across the entire image, therefore learning information specific to a given granularity level. This effect is demonstrated in Figure 1 and the Figure 2 illustrates the learning process of progressive training with the jigsaw

puzzle generator. Note that, the very recent work of [4] first adopted a jigsaw solver to solve for fine-grained classification. We differ significantly in that we do not employ jigsaw solver as part of feature learning. Instead, we simply generate jigsaw patches randomly as means of introducing different object parts levels to assist progressive training.

Main contributions of this paper can be summarized as follows:

1. We propose a novel progressive training strategy to solve for fine-grained visual classification (FGVC). It operates in different training steps, and fuses information from previous levels of granularity at each step, ultimately cultivating the inherent complementary properties across different granularities for fine-grained feature learning.
2. We adapt a simple yet effective jigsaw puzzle generator to form images with different levels of granularity. This allows the network to focus on different "scales" of features as per prior work.
3. The proposed Progressive Multi-Granularity (PMG) Training framework obtains state-of-the-art or competitive performances on three standard FGVC benchmark datasets.

2 Related Work

2.1 Fine-Grained Classification

Recent studies on FGVC have moved from strongly-supervised scenario with additional annotations *e.g.*, bounding box [2, 21, 37, 40, 16], to weakly-supervised conditions with only category labels [11, 41, 38, 36, 12, 22, 39, 42].

In the weakly-supervised configuration, recent studies mainly focus on locating the most discriminative parts, more complementary parts, and parts of various granularities. However, few considered how to fuse information from these discriminative parts together. Current fusion techniques can be roughly divided into two categories. The first category conducts predictions based on different parts and then directly combines their probabilities together. For example, Zhang *et al.* [39] trained several networks focusing on features of different granularities to produce diverse prediction distributions, and then weighted their results before combining them together. The other group concatenate features extracted from different parts together for next prediction [41, 11, 12, 38]. Fu *et al.* [11] found region detection and fine-grained feature learning can reinforce each other, and built a series of networks which located discriminative regions for the next network while conducting predictions. With similar motivation, Zheng *et al.* [41] jointly learned part proposals and the feature representations on each part, and located various discriminative parts before prediction. Both of them train a fully-connected fusion layer to fuse features extracted from different parts. Ge *et al.* [12] went one step further by fusing features from complementary object parts with two LSTMs stacked together.

Fusing features from different parts is still a challenging problem with limited efforts. In this work, we tackle it based on the intrinsic characteristics of

fine-grained objects: although with large intra-class variation, the subtle details exhibit stability at local regions. Hence, instead of locating discriminative parts first, we guide the network to learn features from small granularity to large granularity progressively.

2.2 Image Splitting Operations

Splitting an image into pieces with the same size has been utilized for various task in prior art. Amongst them, one typical solution is to solve the jigsaw puzzle [6, 31]. It can also go one step further by adopting the jigsaw puzzle solution as the initialization to a weakly-supervised network, which leads to better transformation performance [35]. This helps the network to exploit the spatial relationship of local image regions. In one-shot learning, image splitting operation was used for data augmentation [5], which split two images and exchanged patches across to generate new training ones. In more recent research, DCL [4] first adopted image splitting operation for FGVC, who destructed images to emphasize local details and then reconstructed them to learn semantic correlation among local regions. However, it split images with the same size during the whole training process, which made it difficult to exploit multi-granularity regions. In this work, we apply a jigsaw puzzle generator to restrict the granularity of learned regions at each training step.

2.3 Progressive Training

Progressive training methodology was originally proposed for generative adversarial networks [18], where it started with low-resolution images, and then progressively increased the resolution by adding new layers to the network. Instead of learning information from all scales, this strategy allows the network to discover large-scale structure of the image distribution and then shift attention to increasingly finer scale details. Recently, progressive training strategy has been widely utilized for generation tasks [19, 29, 34, 1], since it can simplify the information propagation within the network by intermediate supervision.

For FGVC, the fusion of multi-granularity information is critical to the model performance. In this work, we adopt the idea of progressive training to design a single network that can learn these information with a series of training stages. The input images are firstly split into small patches to train low-level layers of the model. Then the number of patches are progressively increased and the corresponding high-level layers are added and trained. Most of the existing work with progressive training are focusing on the task of sample generation. To the best of our knowledge, this has not been attempted before for the task of FGVC.

3 Approach

In this section, we present our proposed Progressive Multi-Granularity (PMG) training framework. We encourage the model to learn stable fine-grained information in the shallower layers, and gradually focus on learning more abstracted

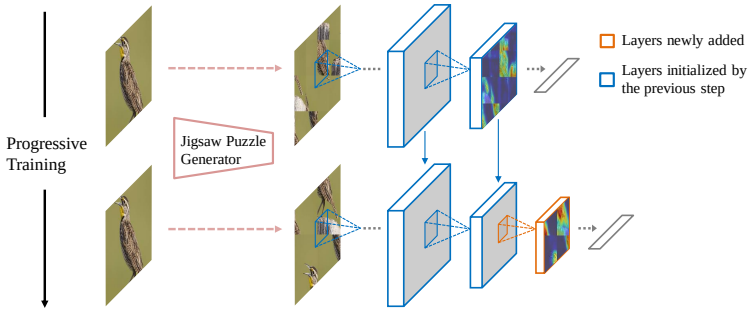

Fig. 2. The illustration of the progressive training process. The network is trained from shallow stages with smaller patches to deeper stages with larger patches. At the end of each training step, the parameter from current step will initialize the parameter of following step. This enables the network to further mine information of larger granularity based on the detail knowledge learned in the previous training step.

information of larger granularity in the deeper layers as the training progresses. Please refer to Figure 2.

3.1 Progressive Training

Network Architecture Our network design for progressive training is generic and could be implemented on the top of any state-of-the-art backbone feature extractors, like Resnet [14]. Let F be our backbone feature extractor, which has L stages. The output feature-map from any intermediate stage is represented as $F^l \; 2 \; \mathbb{R}^{H_l \times W_l \times C_l}$, where H_l, W_l, C_l are the height, width and number of channels of the feature map at l-th stage, and $l = f1; 2; :::; \; Lg$. Here, our objective is to impose classification loss on the feature-map extracted at different intermediate stages. Hence, in addition to F, we introduce convolution block H^l_{conv} that takes l-th intermediate stage output F^l as input and reduces it to a vector representation $V^l = H^l_{conv}(F^l)$. Thereafter, a classification module H^l_{class} consisting of two fully-connected layers with Batchnorm [17] and Elu[7] non-linearity, corresponding to l-th stage, predicts the probability distribution over the classes as $y^l = H^l_{class}(V^l)$. Here, we consider last S stages: $l = L; L \quad 1; :::; L \quad S + 1$. Finally, we concatenate the output from the last S stages as

$$V^{concat} = \text{concat}[V^{L-S+1}; :::; V^{L-1}; V^L] \tag{1}$$

This is followed by an additional classification module $y^{concat} = H^{concat}_{class}(V^{concat})$

Training Process During training, each iteration contains $S + 1$ steps where low-level stages of the model are trained first and new stages are progressively added. Since the receptive field and representation ability of low-level stages are limited, the network will be forced to first exploit discriminative information

Algorithm 1 Progressive Training

Training data set D, Training data for a batch d, Training label for a batch y.
for $epoch \in [0; epoch\_num)$ **do**
  for $b \in [0; batch\_num)$ **do**
    $d; y \Leftarrow$ batch b of D
    for $l \in [L - S + 1; L]$ **do**
      $n \Leftarrow 2^{L \quad l+1}$
      $V^l \Leftarrow H^l_{conv}(F^l(P(d; n)))$
      $y^l \Leftarrow H^l_{class}(V^l)$
      $\mathscr{L}_l \Leftarrow \quad \times \mathscr{L}_{CE}(y^l; y)$
      Backprop($\mathscr{L}_l$)
    end for
    $V^{concat} = \text{concat}[V^{L \quad S+1}; : : : ; V^{L \quad 1}; V^L]$
    $y^{concat} = H^{concat}_{class}(V^{concat})$
    $\mathscr{L}_{concat} \Leftarrow \quad \times \mathscr{L}_{CE}(y^{concat}; y)$
    Backprop($\mathscr{L}_{concat}$)
  end for
end for

from local details (*i.e.* object textures). Directly training the whole network intends to learn all the granularities simultaneously. In contrast to that, step-wise incremental training naturally allows the model to mine discriminative information from local details to global structures when the features are gradually sent into higher stages.

For training, we compute cross entropy (CE) loss $\mathscr{L}_{CE}$ between the ground truth label y and the predicted output from every stage.

At each iteration, a batch of data d will be used for $S + 1$ steps, and we only train one stage's output at each step in sequence. It needs to be clear that all parameters are used in the current prediction will be optimized, even they may have been updated in the previous steps, and this can help all stages in the model working together.

3.2 Jigsaw Puzzle Generator

Jigsaw Puzzle solving [35] has been found to be suitable for self-supervised task in representation learning. On the contrary, we borrow the notion of Jigsaw Puzzle to generate input images for different steps of progressive training. The objective is to devise different granularity regions and force the model to learn information specific to the corresponding granularity level at each training step. Given an input image $d \; 2 \; R^{3\times W\times H}$, we equally split it into $n \quad n$ patches which have $3 \quad \frac{W}{n} \quad \frac{H}{n}$ dimensions. Then, the patches are shuffled randomly and merged together into a new image $P(d; n)$. Here, the granularities of patches are controlled by the hyper-parameter n.

Regarding the choice of hyper-parameter n for each stage, two conditions needs to be satisfied: (i) the size of the patches should be smaller than the receptive field of the corresponding stage, otherwise, the performance of the jigsaw

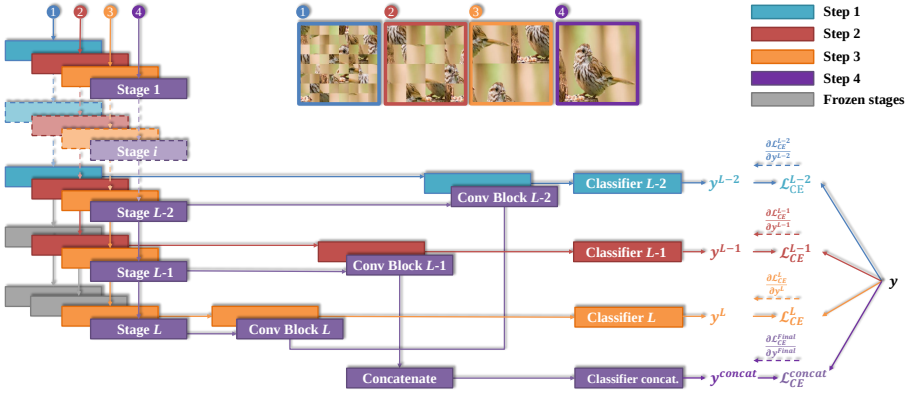

Fig. 3. The training procedure of the progressive training strategy which consists of $S+1$ steps at each iteration (Here $S=3$ for explanation). The *Conv Block* represents the combination of two convolution layers and a max pooling layer, and *Classifier* represent two fully connected layers with a softmax layer at the end. At each iteration, the training data are augmented by the jigsaw generator and sequentially fed into into the network by $S+1$ steps. In our training process, the hyper-parameter n is 2^{L-l+1} for the l^{th} stage. At each step, the output from the corresponding classifier will be used for loss computation and parameter updating.

puzzle generator will be reduced; (ii) the patch size should increase proportionately with the increase of the receptive fields of the stages. Usually, the receptive field of each stage is approximately double than that of the last stage. Hence, we set n as 2^{L-l+1} for the l^{th} stage's output.

During training, a batch of training data d will first be augmented to several jigsaw puzzle generator-processed batches, obtaining $P(d;n)$. All the jigsaw puzzle generator-processed batches share the same label y. Then, for the l^{th} stage's output y^l, we input the batch $P(d;n); n=2^{L-l+1}$, and optimize all the parameters used in this propagation. Figure 3 illustrates the whole progressive training process with the jigsaw puzzle generator step by step.

It should be clarified that the jigsaw puzzle generator cannot always guarantee the completeness of all the parts which are smaller than the size of the patch, because they still have chances of getting split. However, it should not be a bad news for model training, since we adopt random cropping which is a standard data augmentation strategy before the jigsaw puzzle generator and leads to the result that parts with appropriate granularities, which are split at this iteration due to the jigsaw puzzle generator, will not be always split in other iterations. Hence, it brings an additional advantage of forcing our model to find more discriminative parts at the specific granularity level.

3.3 Inference

At the inference phase, we merely input the original images into the trained model and the jigsaw puzzle generator is unnecessary. If we only use y^{concat} for

prediction, the FC layers for the other three stages can be removed which leads to less computational budget. In this case, the final result C_1 can be expressed as

$$C_1 = argmax(y^{concat}): \tag{2}$$

However, the prediction from a single stage with information of a specific granularity is unique and complementary, which leads to a better performance when we simply combine all outputs together with equal weights. The multi-output combined prediction C_2 can be written as

$$C_2 = argmax(\sum_{l=L-S+1}^{L} y^l + y^{concat}): \tag{3}$$

4 Experimental Results and Discussion

In this section, we evaluate the performance of the proposed method on three fine-grained image classification datasets: CUB-200-2011 (CUB) [32], Stanford Cars (CAR) [20], and FGVC-Aircraft (AIR) [27]. Firstly, the implementation details are introduced in Section 4.1. Subsequently, the classification accuracy comparisons with other state-of-the-art methods are provided in Section 4.2. In order to illustrate the advantages of different components and design choices in our method, a comprehensive ablation study and a visualization are provided in Section 4.3 and 4.5. Besides, the discussion about the hyper-parameter selection and the fusion techniques are provided in Section 4.4.

4.1 Implementation Details

We perform all experiments using PyTorch [28] with version higher than 1.3 over a cluster of GTX 2080 GPUs. The proposed method is evaluated on the widely used backbone networks: VGG16 [30] and ResNet50 [14], which means the total number of stages $L = 5$. For the best performance, we set $S = 3$, $= 1$, and $= 2$. The category labels of the images are the only annotations used for training. The input images are resized to a fixed size of 550 550 and randomly cropped into 448 448, and random horizontal flip is applied for data augmentation when we train the model. During testing, the input images are resized to a fixed size of 550 550 and cropped from center into 448 448. All the above settings are standard in the literatures.

We use stochastic gradient descent (SGD) optimizer and batch normalization as the regularizer. Meanwhile, the learning rates of the convolution layers and the FC layers newly added by us are initialized as 0.002 and reduced by following the cosine annealing schedule [24]. The learning rates of the pre-trained convolution layers are maintained as 1/10 of those of the newly added layers. For all the aforementioned models, we train them for up to 200 epochs with batch size as 16 and used a weight decay as 0.0005 and a momentum as 0.9.

Table 1. Comparison with other state-of-the-art methods.

Method	Base Model	CUB (%)	CAR (%)	AIR (%)
FT VGG (CVPR18) [33]	VGG16	77.8	84.9	84.8
FT ResNet (CVPR18) [33]	ResNet50	84.1	91.7	88.5
B-CNN (ICCV15) [23]	VGG16	84.1	91.3	84.1
KP (CVPR17) [8]	VGG16	86.2	92.4	86.9
RA-CNN (ICCV17) [11]	VGG19	85.3	92.5	-
MA-CNN (ICCV17) [41]	VGG19	86.5	92.8	89.9
PC (ECCV18) [10]	DenseNet161	86.9	92.9	89.2
DFL (CVPR18) [33]	ResNet50	87.4	93.1	91.7
NTS-Net (ECCV18) [38]	ResNet50	87.5	93.9	91.4
MC-Loss (TIP20) [3]	ResNet50	87.3	93.7	92.6
DCL (CVPR19) [4]	ResNet50	87.8	94.5	<u>93.0</u>
MGE-CNN (ICCV19) [39]	ResNet50	88.5	93.9	-
S3N (ICCV19) [9]	ResNet50	88.5	94.7	92.8
Stacked LSTM (CVPR19) [12]	ResNet50	**90.4**	-	-
PMG	VGG16	88.2	94.2	92.4
PMG (Combined Accuracy)	VGG16	88.8	94.3	92.7
PMG	ResNet50	88.9	<u>95.0</u>	92.8
PMG (Combined Accuracy)	ResNet50	<u>89.6</u>	**95.1**	**93.4**

Table 2. The p-value of one-sample Student's t-tests between combined accuracies of our method and methods with close performances on three datasets. The proposed method has statistically significant difference from a referred technique if the corresponding p-value is smaller than 0.05

Method	CUB	CAR	AIR
DCL (CVPR19) [4]	6.4e-07	2.5e-06	2.2e-05
MGE-CNN (ICCV19) [39]	8.7e-06	1.5e-07	-
S3N (ICCV19) [9]	4.3e-06	1.4e-05	4.5e-06

4.2 Comparisons with State-of-the-Art Methods

The comparisons of our method with other state-of-the-art methods on CUB-200-2011, Stanford Cars, and FGVC-Aircraft are presented in Table 1. Both the accuracy of the single output C_1 and the combined output C_2 are listed. In addition, we run our method 5 times with random initialization and conduct a one-sample Student's t-test to confirm the significance of our results in Table 2. Results show that our improvement is statistically significant with significance level 0.05.

CUB-200-2011 We achieve a competitive result on this dataset in a much easier experimental procedure, since only single feed-forward propagation through

one network is needed during testing. Our method outperforms RA-CNN [11] and MGE-CNN [39] by 4.3% and 1.1%, even though they build several different networks to learn information of various granularities. They train the classification of each network separately and then combine their information for testing, which proofs our advantage of exploiting multi-granularity information gradually in one network. Besides, even Stacked LSTM [12] obtains better performance than our method, it is a two phase algorithm that requires Mask-RCNN [13] and CPF to offer complementary object parts and then uses bi-directional LSTM [15] for classification, which leads to longer inference time and more computation budget.

Stanford Cars Our method achieves state-of-the-art performance with Resnet50 as the base model. Since the performance of y^{concat} is good enough, the improvement of combining multi-stage outputs is not obvious. The result of our method surpasses PC [10] even it acquires great performance gains by adopting more advanced backbone network *i.e.* DenseNet161. For MA-CNN [41] and NTS-Net [38] which first locate several different discriminative parts to combine feature extracted from each of them for final classification. We outperform them by a large margin of 2.3% and 1.2%, respectively.

FGVC-Aircraft On this task, the multi-output combined result of our method also achieves the state-of-the-art performance. Although S3N [9] finds both discriminative parts and complementary parts for feature extraction, and applies additional inhomogeneous transform to highlight these parts, we still outperform it by 0.6% with the same backbone network ResNet50, and show competitive result even when we adopt VGG16 as the base model.

4.3 Ablation Study

We conduct ablation studies to understand the effectiveness of the progressive training strategy and the jigsaw puzzle generator. We choose CUB-200-2011 dataset for experiments and ResNet50 as the backbone network, which means the total number of stages L is 5. We first design different runs with the number of stages used for output S increasing from 1 to 5 and no jigsaw puzzle generator, as shown in Table 3. The y^{concat} is kept for all runs and number of steps is $S+1$. It is clear that the increasing of S boosts the model performance significantly when $S < 4$. However, we also notice the accuracy starts to decrease when $S = 4$. The possible reason is that the low stage layers are mainly focus on the class-irrelevant features, but the additional supervision will force it to distill class-relevant information and then affect the overall performance.

In Table 3, we also report the results of our method with the jigsaw puzzle generator. The hyper-parameter n of the jigsaw puzzle generator for l^{th} stage follows the pattern that $n = 2^{L-l+1}$. It is obvious that the jigsaw puzzle generator improves the model performance on the basis of progressive training when $S < 4$. When $S = 4$, the model with the jigsaw puzzle generator does not show

Table 3. The performances of the proposed method by using different hyper-parameter s with/without the jigsaw puzzle generator.

S,n	Accuracy (%)	Combined Accuracy (%)
1,{1,1}	86.3	86.5
2,{1,1,1}	87.6	88.0
3,{1,1,1,1}	**88.3**	**88.7**
4,{1,1,1,1,1}	87.8	88.5
5,{1,1,1,1,1,1}	87.7	88.3
1,{2,1}	86.9	86.9
2,{4,2,1}	88.5	88.7
3,{8,4,2,1}	**88.9**	**89.6**
4,{16,8,4,2,1}	88.0	88.5
5,{32,16,8,4,2,1}	87.2	87.7

Table 4. The combined accuracies of our method with different and

;	CUB(%)	CAR(%)	AIR(%)
1; $\frac{1}{3}$	88.6	94.5	92.8
. 1; $\frac{1}{2}$	88.9	94.7	92.8
1,1	89.2	95.0	93.1
1,2	**89.6**	**95.1**	**93.4**
1,3	89.1	**95.1**	93.2

any advantages, and when $S = 5$ the jigsaw puzzle generator lowers the model performance. This is because when $n > 8$ the split patches are too small to keep meaningful information, which instigates confusion in the model training.

According to the above analysis, progressive training is beneficial for fine-grained classification task when we choose appropriate S. In such a case, the jigsaw puzzle generator can further improve the performance.

4.4 Discussions

The choice of hyper-parameter (α and β) In our training procedure, the first S steps and the last step are trained for different goals: learning features with increasing granularity as the network going deeper, and learning correlations between multi-granularity features. Hence, we introduce two hyper-parameter and to adjust their training loss. The model performances with different choice of and are listed in Table 4. When we keep = 1 , it can be observed that the accuracy increases and then decreases as changes. And the model achieves the best performance on both three datasets when = 2.

Fusion of multi-granularity information. In the experiments, we generate images contain multi-granularity information via jigsaw puzzle generator with

Table 5. The comparison between our fusion technique and the other manners.

Fusion Technique	Combined Accuracy(%)
Four Networks Separately	87.7
One network non-progressively	89.1
One network progressively	**89.6**

$n = \{8, 4, 2, 1\}$, and fused these information with one network in a progressive manner. In order to demonstrate the advantage of the fusion strategy under the same configuration, we conduct two experiment on (i) training four different networks with generated images where $n = \{8, 4, 2, 1\}$ separately and concatenating their features for final classification with a fully connected fusion layer, which is similar to the fusion technique used in RA-CNN [11], and (ii) training a model with same architecture as ours but back-propagating the losses of four outputs in one step. We choose CUB-200-2011 dataset for experiments with ResNet50 as the base model and the results are listed in Table 5. The performance of four networks trained separately is higher than a lot of state-of-the-art methods but our method still outperforms it by a large margin, which indicates the effectiveness of our fusion technique. When we back-propagate losses of four outputs in one step, which means multi-granularity information is learnt simultaneously, the performance clearly drops even the other configurations are unchanged. Hence, the unique advantage of progressively learning multi-granularity information is significant.

4.5 Visualization

In order to demonstrate the achievement of our motivation, we apply the Grad-CAM to visualize the last three stages' convolution layers of both our method and the baseline model. Columns (a)-(c) in Figure 4 are visualization of the convolution layers from the third to the fifth stage of our model's backbone network, which are supervised by the jigsaw puzzle generator-processed images with $n = \{8, 4, 2\}$ sequentially. It is shown in column (a) that the model concentrates on discriminative parts of small granularity at the third stage like bird eyes and small pattern or texture of birds' feathers. And when it comes to column (c), the fifth stage of the model pays attention to parts of larger granularity. The visualization result demonstrates that our model truly gives predictions based on discriminative parts from small granularity to large granularity gradually.

When compared with the activation map of the baseline model, our model shows more meaningful concentration on the target object, while the baseline model only shows the correct attention at the last stage. This difference indicates that the intermediate supervision of progressive training can help the model distill useful information at low-level stages. Besides, we find the baseline model usually only concentrates on one or two parts of the object at the last stage. However, the attention regions of our method nearly cover the whole object at

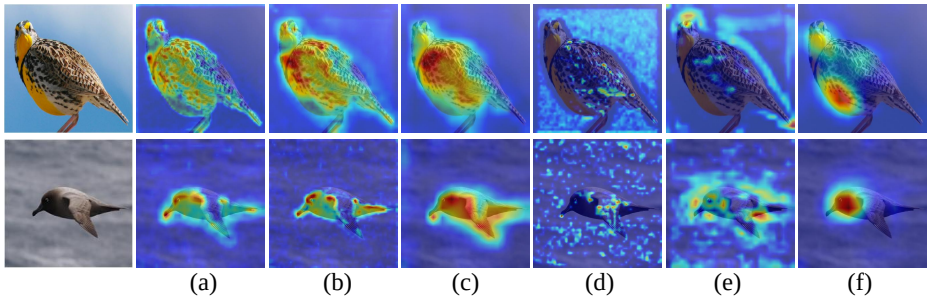

Fig. 4. Activation map of selected results on the CUB dataset with the Resnet50 as the base model. Columns (a)-(c) and (d)-(f) are visualizations of the last three stages' convolution layers of our model and the baseline model, respectively.

each stage, which indicates that jigsaw puzzle generator-processed images can force the model to learn more discriminative parts at each granularity level.

5 Conclusions

In this paper, we approached the problem of fine-grained visual classification from a rather unconventional perspective – we do not explicitly nor implicitly mine for object parts, instead we show fine-grained features can be extracted by learning across granularities and effectively fusing multi-granularity features. Our method can be trained end-to-end without additional manual annotations other than category labels, and only needs one network with one feed-forward pass during testing. We conducted experiments on three widely used fine-grained datasets, and obtained state-of-the-art performance on two of them while being competitive on the other.

Acknowledgement

This work was supported in part by the National Key R&D Program of China under Grant 2019YFF0303300 and under Subject II No. 2019YFF0303302, in part by the National Natural Science Foundation of China under Grant 61773071, 61922015, and U19B2036, in part by Beijing Academy of Artificial Intelligence (BAAI) under Grant BAAI2020ZJ0204, in part by the Beijing Nova Program Interdisciplinary Cooperation Project under Grant Z191100001119140, in part by the National Science and Technology Major Program of the Ministry of Science and Technology under Grant 2018ZX03001031, in part by the Key Program of Beijing Municipal Natural Science Foundation under Grant L172030, in part by MoE-CMCC Artifical Intelligence Project No. MCM20190701, in part by the scholarship from China Scholarship Council (CSC) under Grant CSC No. 201906470049, and in part by the BUPT Excellent Ph.D. Students Foundation No. CX2020105 and No. CX2019109.

References

1. Ahn, N., Kang, B., Sohn, K.A.: Image super-resolution via progressive cascading residual network. In: CVPR workshops (2018)
2. Berg, T., Belhumeur, P.: Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In: CVPR (2013)
3. Chang, D., Ding, Y., Xie, J., Ayan, K.B., Li, X., Ma, Z., Wu, M., Guo, J., Song, Y.Z.: The devil is in the channels: Mutual-channel loss for fine-grained image classification. IEEE Transactions on Image Processing (2020)
4. Chen, Y., Bai, Y., Zhang, W., Mei, T.: Destruction and construction learning for fine-grained image recognition. In: CVPR (2019)
5. Chen, Z., Fu, Y., Chen, K., Jiang, Y.G.: Image block augmentation for one-shot learning. In: AAAI (2019)
6. Cho, T.S., Avidan, S., Freeman, W.T.: A probabilistic image jigsaw puzzle solver. In: CVPR (2010)
7. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015)
8. Cui, Y., Zhou, F., Wang, J., Liu, X., Lin, Y., Belongie, S.: Kernel pooling for convolutional neural networks. In: CVPR (2017)
9. Ding, Y., Zhou, Y., Zhu, Y., Ye, Q., Jiao, J.: Selective sparse sampling for fine-grained image recognition. In: ICCV (2019)
10. Dubey, A., Gupta, O., Guo, P., Raskar, R., Farrell, R., Naik, N.: Pairwise confusion for fine-grained visual classification. In: ECCV (2018)
11. Fu, J., Zheng, H., Mei, T.: Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In: CVPR (2017)
12. Ge, W., Lin, X., Yu, Y.: Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In: CVPR (2019)
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation (1997)
16. Huang, S., Xu, Z., Tao, D., Zhang, Y.: Part-stacked cnn for fine-grained visual categorization. In: CVPR (2016)
17. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
19. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019)
20. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: ICCV workshops (2013)
21. Lei, J., Duan, J., Wu, F., Ling, N., Hou, C.: Fast mode decision based on grayscale similarity and inter-view correlation for depth map coding in 3d-hevc. IEEE Transactions on Circuits and Systems for Video Technology **28**(3), 706–718 (2016)
22. Li, X., Yu, L., Chang, D., Ma, Z., Cao, J.: Dual cross-entropy loss for small-sample fine-grained vehicle classification. IEEE Transactions on Vehicular Technology **68**(5), 4204–4212 (2019)
23. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: ICCV (2015)

24. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
25. Luo, W., Yang, X., Mo, X., Lu, Y., Davis, L.S., Li, J., Yang, J., Lim, S.N.: Cross-x learning for fine-grained visual categorization. In: ICCV (2019)
26. Ma, Z., Chang, D., Xie, J., Ding, Y., Wen, S., Li, X., Si, Z., Guo, J.: Fine-grained vehicle classification with channel max pooling modified cnns. IEEE Transactions on Vehicular Technology **68**(4), 3224–3233 (2019)
27. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013)
28. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
29. Shaham, T.R., Dekel, T., Michaeli, T.: Singan: Learning a generative model from a single natural image. In: ICCV (2019)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
31. Son, K., Hays, J., Cooper, D.B.: Solving square jigsaw puzzles with loop constraints. In: ECCV (2014)
32. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
33. Wang, Y., Morariu, V.I., Davis, L.S.: Learning a discriminative filter bank within a cnn for fine-grained recognition. In: CVPR (2018)
34. Wang, Y., Perazzi, F., McWilliams, B., Sorkine-Hornung, A., Sorkine-Hornung, O., Schroers, C.: A fully progressive approach to single-image super-resolution. In: CVPR workshops (2018)
35. Wei, C., Xie, L., Ren, X., Xia, Y., Su, C., Liu, J., Tian, Q., Yuille, A.L.: Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning. In: CVPR (2019)
36. Wei, K., Yang, M., Wang, H., Deng, C., Liu, X.: Adversarial fine-grained composition learning for unseen attribute-object recognition. In: ICCV (2019)
37. Xie, L., Tian, Q., Hong, R., Yan, S., Zhang, B.: Hierarchical part matching for fine-grained visual categorization. In: ICCV (2013)
38. Yang, Z., Luo, T., Wang, D., Hu, Z., Gao, J., Wang, L.: Learning to navigate for fine-grained classification. In: ECCV (2018)
39. Zhang, L., Huang, S., Liu, W., Tao, D.: Learning a mixture of granularity-specific experts for fine-grained categorization. In: ICCV (2019)
40. Zhang, N., Donahue, J., Girshick, R., Darrell, T.: Part-based r-cnns for fine-grained category detection. In: ECCV (2014)
41. Zheng, H., Fu, J., Mei, T., Luo, J.: Learning multi-attention convolutional neural network for fine-grained image recognition. In: ICCV (2017)
42. Zheng, Y., Chang, D., Xie, J., Ma, Z.: IU-Module: Intersection and union module for fine-grained visual classification. In: ICME (2020)