Supplementary Material

Zhenxin Wu[1]
wuzhenxin@stu2020.jnu.edu.cn
Qingliang Chen (*corresponding author)[1,2]
tpchen@jnu.edu.cn
Yongjian Huang[3]
drwinhuang@gmail.com

[1] Department of Computer Science, Jinan University, Guangzhou, China
[2] Guangzhou Soundbox Acoustic Tech. Co., Ltd., Guangzhou, China
[3] Guangzhou Xuanyuan Research Institute Co., Ltd., Guangzhou, China

In this document, we will first supplement some statistics of benchmark datasets and detailed settings of the experiments. Then, in order to investigate the properties of our proposed RMG-PMSI, we conduct a series of ablation experiments to evaluate its key designs. Finally, we also test the impact of input size on model performance and delay.

1 Statistics of benchmark datasets

We evaluate the performance of the proposed method on three fine-grained visual classification (FGVC) datasets: CUB-200-2011 (CUB)[8], Stanford Cars (Car)[5], and FGVC-Aircraft (Air)[7]. Examples selected from the three datasets are shown in Figure 1. The detailed information of each dataset is shown in Table 1.

2 Experimental setup details

Next, we describe some implementation details of the experiments in the paper, including implementation details for MobilenetV2 and Efficientnet-B0 Baseline, and the implementation details of the SOTA FGVC methods on the MobilenetV2 backbone.

2.1 MobilenetV2 and Efficientnet-B0 baseline

For the standard MobilenetV2, Efficientnet-B0(baseline), we use the ImageNet pre-trained model, the initial learning rate of the convolutional layer is 0.001, and the learning rate of the new classifier layer is 0.01, and reduced by the cosine annealing schedule. We train them for up to 150 epochs with a batch size as 32 and use a weight decay of 0.0005 and a momentum of 0.9.

Table 1: Statistics of benchmark datasets.

Dataset	#Classes	#Train	#Test
CUB-200-2011	200	5994	5794
Standford Cars	196	8144	8041
FGVC Aircraft	100	6667	3333

Figure 1: Examples from datasets of CUB [8], Car [5] and Air [7](left to right).

2.2 FGVC SOTA Method

We compare our approach with 8 representative FGVC approaches, namely BCNN[6], HBP[10], PC[3], NTS-NET [9], DCL[1], PMG[2], API-NET[11], and Snapmix[4]. Since these works do not formally report the results of using lightweight networks as feature extractors, we implement these methods on MobilenetV2 based on published code and experiment on fine-grained datasets.

Unless otherwise specified, for the following methods, We use Momentum SGD with weight decay 1e-4, the initial learning rate of the backbone is 1e-3, and the learning of new layers (such as classification layers) is 10 times that of the backbone, i.e., 1e-2, using cosine annealing learning rate decay. batch-size=32; epochs=150.

BCNN[6]. We use the MobilenetV2 Stage4 as the output for bilinear pooling, just as mentioned in the paper. Firstly, the size of the bilinear pooling feature map is entered as (28, 28, 96), which is activated by a RELU, followed by the Bilinear pooling operation. The size of the output feature graph is (96×96). This is followed by one-dimensional Batch normalization. Finally, a full connection layer FC is used for classification.

HBP[10]. Like RMG-PT, we use the output of the last three stages as the three levels of HBP. HBP requires the feature maps of the three levels to be of the same size, so we use downsampling to map the length and width of Stage3 and Stage4 to be the same as that of Stage5, which is 14×14. As mentioned in the paper, HBP raises the channel dimension for each level to 8192. Then conduct the Bilinear pooling operation in pairs. This is followed by one-dimensional Batch normalization. Finally, a full connection layer FC is used for classification.

PMG[2]. Like RMG-PT, we use the outputs of the last three stages as the three stages of PMG. Then we use smooth conv to make the number of channels consistent across the different stages. In each stage, different granularity jigsaw patches are used as input for training. Finally, the outputs of different stages are concatenated and then run through the fully-connected layer classifier as the result of classification.

PC[3]. In addition to the cross entropy loss of MobilenetV2 itself, we add pairwise confusion loss as a regularization method according to the paper. That is, the different conditional probability distributions are brought closer together and the deep network is confused, so as to reduce the over-confidence in its prediction and improve the generalization performance. The optimization objective is to reduce the cross entropy loss of each sample in each class and reduce the Euclidean distance between the probability distributions of different

classes. Specifically, each batch of the training set is randomly divided into two parts, and then for each pair of points in the two parts, as long as the samples belonged to different classes, Euclidean confusion is added. We set the coefficient of this pairwise confusion loss as 0.5(in the experiment, we found that it is difficult to converge if the coefficient is too large), and then add the standard cross entropy to optimize the loss as a whole.

NTS-Net[9]. Resnet-50 is used as a feature extractor in the original paper, we only replace the feature extractor with MobilenetV2, and other Settings remain the same as in the original text. In other words, K=4, M=6(which means 6 regions are used to train the Navigator network for each image).

DCL[1]. We use MobilenetV2 as the feature extractor and the other hyperparameter settings refer to the original text. For example, we set Partition Granularity to 7 for CUB and Car training, and Partition Granularity=2 for Air. It is worth mentioning that in order to ensure that the model is fully trained, we add an additional 50 epochs to the 150 epochs, i.e. 200 epochs.

API-Net[11]. For each pair of images, we connect x_1 and x_2 as input to a two-layer MLP, namely FC(2560→512)FC(512→1280), as the result of interactive learning. Besides, during the training phase, we freeze the conv layers and only train the newly-added fully-connected layers in the first 8 epochs. Other super parameter Settings refer to the original text.

SnapMix[4]. SnapMix is a data augmentation approach designed for FGVC using Class Activation Maps (CAMs) and we apply it to the training standard MobilenetV2. The hyperparameter α of snapMix decides a beta distribution that is used to generate a random patch in mixing. Based on the published code, we set this hyperparameter to $\alpha = 1.0$. Other hyperparameter settings refer to the original text.

3 Supplementary ablation experiment

In this section, we present a series of ablation experiments to validate the key design of our method, including (i) the influence of stage num; (ii) the influence of each component on the model's anti-interference.

3.1 The impact of StageNum

To demonstrate the efficacy of progressive interaction training, we conduct experiments without a Recursive Mosaic Generator (RMG) on CUB dataset. We set the enlargement of the receptive field as the division between this stage and the next stage. In order to obtain the best performance and prevent excessive noise from being introduced by too shallow layer features, we use the last 5 stages as our experimental setup. The StageNum increases from 1 to 5. We use Top-1 Accuracy (Acc) as the evaluation criterion. The results of the experiment are shown in table:2, where Mix represents a mixed classification result *P*.

As the results show, when the number of stages (S) involve in interactive training is less than 4, the increase in StageNum improves the model's performance. Then, Mix Acc began to decline when the StageNum=4. This might be caused by the low-stage layer focusing on class-independent features. However, the additional supervision forces the low-stage layer (S=1,2) to focus prematurely on the features associated with classification, thus introducing too much low-stage noise to the high-stage classification through progressive interaction, resulting in a decrease in accuracy. In addition, using multi-stages of progressive interaction

Table 2: The performance of the proposed model when interacting at different stages.

	Acc(%)					
Stage(S) / StageNum	S1	S2	S3	S4	S5	Mix
{5} / 1	-	-	-	-	84.2	84.3
{5, 4} / 2	-	-	-	84.7	84.5	85.2
{5, 4, 3} / 3	-	-	82.9	84.8	84.3	**85.9**
{5, 4, 3, 2} / 4	-	80.5	83.3	84.7	83.9	85.7
{5, 4, 3, 2, 1} / 5	74.2	81.7	83.0	84.5	84.3	85.1

Table 3: Comparison of anti-interference ability of baseline and RMG-PMSI.

	CUB		Car		Air	
	Baseline	+RMG-PMSI	Baseline	+RMG-PMSI	Baseline	+RMG-PMSI
Origin	81.5	**87.1**	90.9	**93.8**	88.9	**91.5**
+ Color-Jitter	14.1 (-67.4)	**59.9 (-27.2)**	69.7 (-21.2)	**79.3 (-14.5)**	64.7 (-24.2)	**73.9 (-17.6)**
+ Gaussian-Noise	22.2 (-59.3)	**78.6 (-8.5)**	75.0 (-15.9)	**86.5 (-7.3)**	82.6 (-6.3)	**86.6 (-4.9)**

can lead to increased training costs. To sum up, we use the last three phases (S=5,4,3, StageNum=3) as the optimal choice for progressive interactive training.

3.2 Anti-interference analysis

After justifying the transferability of the RMG-PMSI on different efficient mobile backbones. We further test the anti-interference capabilities of RMG-PMSI. We use Color-Jitter to generate interference data on the testset of the three datasets. We set the Jitter coefficient to be 1, i.e., the image's brightness, contrast, and saturation will be randomly adjusted to 0% to 200% of the original image. In addition, we also generate interference images with Gaussian-Noise (mean=0, variance=0, amplitude=5) on the testset of the three datasets. And then we test these two interference data respectively on the standard MobilenetV2 (baseline) and MobilenetV2 with RMG-PMSI. The experimental results are shown in Table 3.

It can be seen that the accuracy degradation of RMG-PMSI on the two types of interference data on the three testsets is less than the baseline. Especially on CUB200, the anti-interference ability of RMG-PMSI is far better than the baseline. This may be because the distinction of birds mainly comes from some important local parts, such as eyes, feathers, and beaks. Adding interference will have a greater impact on global information, and the introduction of local information through RMG-PMSI can effectively alleviate this global interference.

Then, in order to further verify the contribution of different components of RMG-PMSI to the model's anti-interference, on the basis of the previous anti-interference experiment, we conduct experiments on the anti-interference of different components on the CUB dataset. The experimental results are shown in Table 4.

It can be seen from the experimental results that whether it is multi-stage interaction (M) or progressive training (P), the anti-interference of the model can be greatly improved. At the same time, compare to multi-stage interaction (M), the supervision imposed by progressive training (P) at different stages can help the model better obtain local and global features, so the anti-interference is stronger.

Table 4: The influence of each component on the model's anti-interference.

	Origin	+Color-Jitter	+Gaussian-Noise
Baseline	81.5	14.1 (-67.4)	22.2 (-59.3)
+M	85.1	52.9 (-32.2)	72.0 (-13.1)
+P	85.5	56.3 (-29.2)	77.7 (-7.8)
+P&M	85.9	59.4 (-26.5)	77.8 (-8.1)
+P&R	86.8	60.2 (-26.6)	78.3 (-8.5)
+P&M&R	87.1	59.9 (-27.2)	78.6 (-8.5)

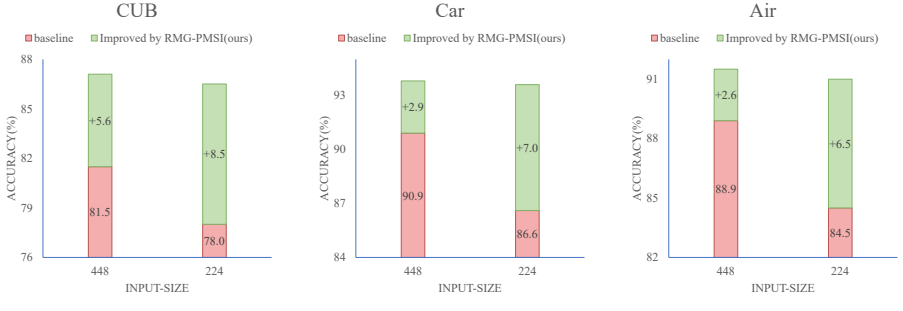
Table 5: Influence of input-size on model performance and delay.

method	Input-size	CUB	Car	Air	delay(s)
Baseline (MobilenetV2)	224	78.0	86.6	84.5	0.246
+ RMG-PMSI (ours)		**86.5 (+8.5)**	**93.6 (+7.0)**	**91.0(+6.5)**	0.287 (+0.041)
Baseline (MobilenetV2)	448	81.5	90.9	88.9	1.029
+ RMG-PMSI (ours)		**87.1 (+5.6)**	**93.8 (+2.9)**	**91.5 (+2.6)**	1.094(+0.065)

4 Influence of Input size on Model Performance and Delay

To further explore the delay of the model in real application scenarios and the influence of input size on model performance. We test our model on a Microsoft Surface Pro 7 (Intel(R)Core(TM) I5-1035G4 CPU @ 1.10GHz, 8G RAM). Specifically, we fully test the effect of two commonly used input sizes (224/448) on the model top-1 Acc(%) on three datasets. For the delay test, we randomly select 20 images on CUB/Car/Air respectively, scale them to different input sizes and put them into the model for testing, and calculate the average latency of each image. It should be emphasized that, in order to make a more intuitive and fair comparison, we do not do any optimization, and directly test the model on the Surface Pro 7 without any Conv and BN layer merging or quantization operations (e.g., INT8 quantization). Experimental results are shown in Table 4 and Fig.2.

The results are surprising. When the input size is changed from 448 to 224, the accuracy of RMG-PMSI did not decrease significantly in the three different datasets. In contrast, for standard MobilenetV2(baseline), when the input size is adjusted from 448 to 224, there is a significant decline in accuracy. This shows that RMG-PMSI can increase the robustness of the model to the input image resolution, which is very exciting.



(a) CUB. (b) Car. (c) Air.

Figure 2: Influence of input-size on model performance.

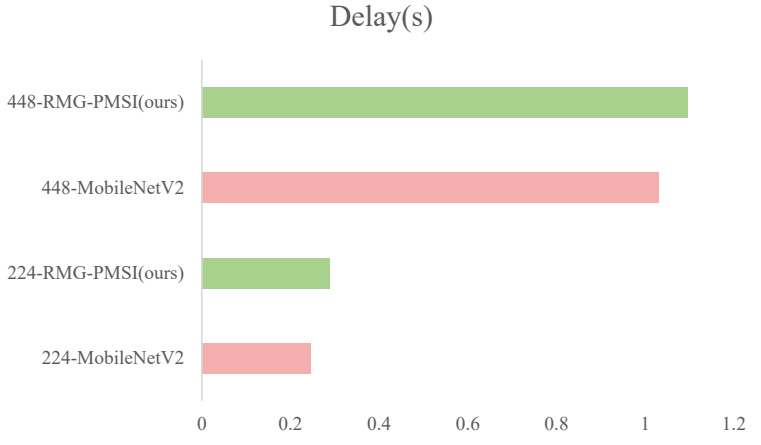


Figure 3: Delay comparison between RMG-PMSI and baseline.

In addition, it is worth emphasizing that in the inference phase, RMG-PMSI does not use RMG or involving multi-phase model. The model only computes the outputs of different stages and combines them, which can be done in parallel and used for real-time predictions. As shown in Fig.3, when the input size is set to 224, the delay of RMG-PMSI model is significantly reduced and only a slight reduction in model accuracy compared with that when the input-size is set to 448, which provides very good feasibility for the deployment and application of mobile terminals. At the same time, when input-size is fixed, compared with the baseline and RMG-PMSI, it can be found that RMG-PMSI only brings a small increase in time with significantly improved accuracy, which is totally acceptable. It also proves once again that RMG-PMSI is a novel training method designed for mobile networks on FGVC tasks, bringing a huge performance improvement with only a small increase in the computational overhead in end-to-end inference.

References

[1] Yue Chen, Yalong Bai, Wei Zhang, and Tao Mei. Destruction and construction learning for fine-grained image recognition. In *CVPR*, pages 5157–5166. Computer Vision Foundation / IEEE, 2019.

[2] Ruoyi Du, Dongliang Chang, Ayan Kumar Bhunia, Jiyang Xie, Zhanyu Ma, Yi-Zhe Song, and Jun Guo. Fine-grained visual classification via progressive multi-granularity training of jigsaw patches. In *ECCV (20)*, volume 12365 of *Lecture Notes in Computer Science*, pages 153–168. Springer, 2020.

[3] Abhimanyu Dubey, Otkrist Gupta, Pei Guo, Ramesh Raskar, Ryan Farrell, and Nikhil Naik. Pairwise confusion for fine-grained visual classification. In *ECCV (12)*, volume 11216 of *Lecture Notes in Computer Science*, pages 71–88. Springer, 2018.

[4] Shaoli Huang, Xinchao Wang, and Dacheng Tao. Snapmix: Semantically proportional

mixing for augmenting fine-grained data. In *AAAI*, pages 1628–1636. AAAI Press, 2021.

[5] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshops*, pages 554–561. IEEE Computer Society, 2013.

[6] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, pages 1449–1457. IEEE Computer Society, 2015.

[7] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *CoRR*.

[8] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[9] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. Learning to navigate for fine-grained classification. In *ECCV (14)*, volume 11218 of *Lecture Notes in Computer Science*, pages 438–454. Springer, 2018.

[10] Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. Hierarchical bilinear pooling for fine-grained visual recognition. In *ECCV (16)*, volume 11220 of *Lecture Notes in Computer Science*, pages 595–610. Springer, 2018.

[11] Peiqin Zhuang, Yali Wang, and Yu Qiao. Learning attentive pairwise interaction for fine-grained classification. In *AAAI*, pages 13130–13137. AAAI Press, 2020.