Learning Partial Correlation based Deep Visual Representation for Image Classification (Supplementary Material)

Saimunur Rahman[1,2], Piotr Koniusz[*,1,3], Lei Wang[2], Luping Zhou[4], Peyman Moghadam[1], Changming Sun[1]
[1]Data61♥CSIRO, [2]University of Wollongong, [3]Australian National University, [4]University of Sydney
name.surname@data61.csiro.au, leiw@uow.edu.au, luping.zhou@sydney.edu.au

A. Datasets and the Evaluation Protocols

In this section, we provide the details of datasets and their evaluation protocols (see Section 4.1 of the main text). We perform experiments on eight widely used public image datasets, namely, MIT Indoor [67], Stanford Cars [58], Caltech-UCSD Birds (CUB 200-2011) [71], FGVC-Aircraft [65], DTD [5], iNaturalist [40], mini-ImageNet [41] and ImageNet100 [70] to demonstrate the performance of our methods. Figure 4 shows the sample images from the datasets. Further details are given below.

MIT Indoor dataset is one of the most widely used datasets in the literature for scene classification. It has a total of 15,620 images and 67 classes. Each image class contains a minimum number of 100 images. The images are collected from various types of stores (*e.g.*, grocery, bakery), private places (*e.g.*, bedroom and living room), public places (*e.g.*, prison cell, bus, library), recreational places (*e.g.*, restaurant, bar) and working environments (*e.g.*, office, studio).

Caltech-UCSD Birds or simply 'Birds' is one of the most reported datasets in fine-grained image classification (FGIC) literature. It has a total of 11,788 images and 200 image classes. There are subtle differences between these classes and they are hard to be distinguished by human observers. This dataset comes with bounding box annotations; however, we do not use any annotations in our experiments.

FGVC-Aircraft or 'Aircraft' dataset is widely used by many recent FGIC methods. It has only 10,000 images distributed among 100 aircraft classes, and each class has precisely 100 images. Similar to Birds, the classes have subtle differences between them and are hard for humans to distinguish from each other.

Stanford Cars or simply 'Cars' has a total of 16,185 images and 196 classes. The classes are organized as per the car production year, car manufacturer and car model. Cars dataset has relatively smaller objects, *i.e.*, cars, than those of the airplane dataset. Furthermore, the objects are appeared in cluttered backgrounds.

Describable Texture Dataset (DTD) has a total of 5,640 images and 47 classes. The images in all classes represents about 95% of their class attributes. For evaluation, it has 10 splits and each split has an equal number of images from each class for training, validation and test sets. The average performance across all splits is reported. We used both training and validation sets for training.

iNaturalist is a large fine-grained dataset. It has a total of 675,170 images and 5,089 classes. The classes are from 12 super-classes. The dataset is challenging due to its high class-imbalance. We use the training strategy and train-test splits specified in the original paper [40].

mini-ImageNet is a subset of ImageNet-1K dataset first proposed by Ravi *et al*. [68]. It contains a total of 60,000 images and 100 classes. We use the original 224×224 image resolution on this dataset. Since the dataset was originally proposed by the authors for few-shot learning tasks, we divide the dataset in a 90:10 ratio for training and testing. We report the average accuracy of 5 runs.

ImageNet100 is a subset of ImageNet-1K Dataset from ImageNet Large Scale Visual Recognition Challenge 2012. It contains random 100 classes proposed by Tian *et al*. [70]. ImageNet100 train and validation sets contain 1300 and 50 images per class, respectively. We use the original 224×224 image resolution on this dataset.

Table 6 gives a more concise summary of the datasets. We train our method using the respective training splits provided by the original authors of the datasets. For evaluation, we again use the respective original test splits. This also applies to all of the methods we have compared in the main text. During training, for all datasets except iNaturalist, we resize our images to 448×448 following the work of [59, 61] and use only horizontal flipping as a data augmentation. For iNaturalist, as in original paper, we resize images to 299×299.

B. Training iSICE with Different Backbones

This section provides the settings of training different backbones with iSICE mentioned in Section 4.1 of the main

*Corresponding author. Code: https://github.com/csiro-robotics/iSICE.

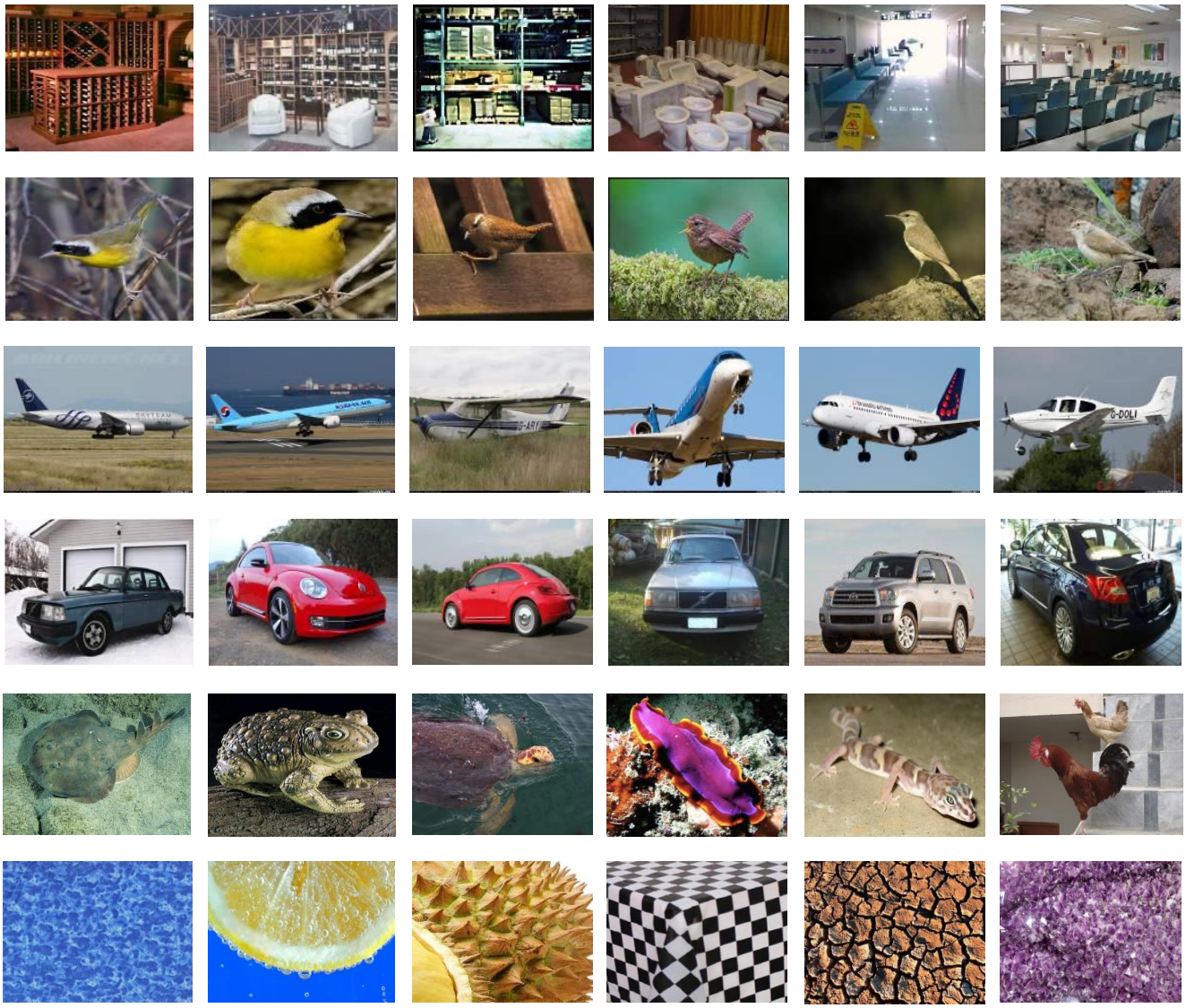

Figure 4. Sample images from the datasets used in our experiments. Rows 1, 2, 3, 4, 5 and 6 have the images from MIT Indoor, Caltech-UCSD Birds, FGVC-Aircraft, Stanford Cars, ImageNet100/mini-ImageNet and DTD datasets, respectively.

Dataset	Total classes	Total images	Predefined protocol		Major difficulty
			Training images	Testing images	
MIT Indoor	67	6,700	5,360	1,340	difficult environment
Birds	200	11,788	5,994	5,794	subtle class difference
Aircraft	100	10,000	6,600	3,400	subtle class difference
Cars	196	16,185	8,144	8,041	cluttered background
DTD	47	5,640	4512	1128	complex structure
iNaturalist	5,089	675,170	579,184	95,986	class imbalance
mini-ImageNet	100	60,000	54,000	6,000	difficult environment
ImageNet100	100	1,35,000	130,000	5,000	difficult environment

Table 6. Summary of datasets.

Dataset	Backbone	iSQRT-COV [23]	Precision $\boldsymbol{\Omega}$	iSICE							
				Sparsity constant λ							Mean ± Std. (iSICE)
				1.0	0.5	0.1	0.01	0.001	0.0001	0.00001	
MIT	VGG-16	76.12	**80.15**	77.46	78.13	78.13	78.66	78.58	78.96	78.96	78.41±0.54
	ResNet-50	78.81	80.75	78.43	80.75	80.45	80.52	80.90	80.37	**81.34**	80.39±0.93
Airplane	VGG-16	90.01	89.44	92.26	92.71	92.77	92.23	**92.83**	92.74	92.44	92.56 ±0.25
	ResNet-50	90.88	91.15	**92.89**	92.65	**92.89**	92.74	92.83	92.56	92.56	92.73±0.14
Birds	VGG-16	84.47	83.36	86.04	86.47	86.35	**86.52**	85.59	86.31	86.28	86.22±0.32
	ResNet-50	84.26	84.67	84.62	85.16	85.30	85.90	**86.05**	85.90	85.59	85.50±0.51
Cars	VGG-16	91.21	92.04	93.60	93.98	**94.06**	94.03	93.88	93.91	93.50	93.85±0.22
	ResNet-50	92.13	91.99	93.01	93.36	93.69	93.51	93.22	**93.72**	93.40	93.41±0.25

Table 7. Performance of iSICE on changing the sparsity constant λ while fixing the learning rate η and the number of iterations N to 1.0 and 5, respectively. The mean and standard deviation (std.) of the classification performance resulted by iSICE are also shown for a better understanding of sparsity constant changes in iSICE. Results are shown on multiple datasets with VGG-16 and ResNet-50 backbones. The best results in each row are highlighted with boldface.

Dataset	Backbone	iSQRT-COV [23]	Precision $\boldsymbol{\Omega}$	iSICE							
				Learning rate η							Mean ± Std. (iSICE)
				0.001	0.01	0.1	1.0	5.0	10.0	20.0	
MIT	VGG-16	76.12	80.15	78.81	79.33	77.91	78.66	78.28	**80.52**	77.76	78.75±0.95
	ResNet-50	78.81	80.75	80.82	80.82	80.75	80.52	**81.19**	79.33	78.96	80.34±0.85
Airplane	VGG-16	90.01	89.44	92.32	92.38	92.98	92.23	**93.28**	92.50	92.26	92.56±0.41
	ResNet-50	90.88	91.15	92.77	**93.01**	92.89	92.74	92.65	92.95	92.38	92.77±0.21
Birds	VGG-16	84.47	83.36	86.54	86.31	86.40	86.52	**86.73**	86.59	86.45	86.51±0.14
	ResNet-50	84.26	84.67	85.69	85.88	85.81	85.90	85.59	85.71	**85.97**	85.79±0.14
Cars	VGG-16	91.21	92.04	93.83	93.65	93.69	**94.03**	93.66	93.82	93.93	93.80±0.14
	ResNet-50	92.13	91.99	**93.60**	93.57	93.32	93.51	93.57	93.35	**93.60**	93.50±0.12

Table 8. Performance of iSICE on changing the learning rate η while fixing the sparsity constant λ and the number of iterations N to 0.01 and 5, respectively. The mean and standard deviation (std.) of the classification performance resulted by iSICE are also shown for a better understanding of learning rate changes in iSICE. Results are shown on multiple datasets with VGG-16 and ResNet-50 backbones. The best results in each row are highlighted with boldface.

text. We train iSICE with following backbones: VGG-16 [69], ResNet-50 [57], ConvNext-T [63], and Swin-T [62]. All backbones are pre-trained on ImageNet-1k [56]. We use the pre-trained weights provided by the torchvision 0.13.0 package that comes with PyTorch library [66].

All backbones produce more than 256 feature channels. In the recent literature on covariance representation [59, 60, 72], a common practice is to experiment with 256 channels for efficiency and compactness of final representation. To compare our method with the recent literature, we also conducted most of our experiments with 256 channels. Additionally, we conducted experiments with 512 channels to demonstrate the effectiveness of the proposed iSICE in working with a larger number of feature channels (provided in Table 3 in the main text). For reducing the original number of feature channels from 2048/512 to 256, we add a 1×1 convolution layer, batch normalisation and ReLU activation layers after the last convolution layer (in case of CNN models) and transformer block (in case of Swin Transformer). We then compute iSICE with the reduced feature channels, and we only use the upper-triangular entries of the symmetric matrix as a representation.

We fine-tune all backbones for 50-100 epochs with AdamW optimiser [64]. We fine-tune VGG-16 and ResNet-50 CNN backbones with an initial learning rate of 0.00012, and ConvNext-T CNN and Swin-T transformer backbones with an initial learning rate 0.00005. For all backbones, we decrease the learning rate by a factor of 10 at the 15th and

Method	Iter. N	Based on VGG-16 backbone				Based on ResNet-50 backbone			
		MIT	Airplane	Birds	Cars	MIT	Airplane	Birds	Cars
iSQRT-COV [23]	5	76.12	90.01	84.47	91.21	78.81	90.88	84.26	92.13
Precision $\mathbf{\Omega}$	7	**80.15**	89.44	83.36	92.04	80.75	91.15	84.67	91.99
	2	78.28	**92.68**	**86.66**	93.63	**80.52**	**92.89**	**93.68**	**85.92**
iSICE	5	78.66	92.23	86.52	**94.03**	**80.52**	92.74	93.51	85.90
	10	78.36	92.56	86.62	93.89	80.22	92.74	93.30	85.74
Mean±Std. (iSICE)		78.4±0.2	92.5±0.2	86.6±0.1	93.9±0.2	80.4±0.2	92.8±0.1	93.5 ±0.2	85.6 ± 0.1

Table 9. Performance of iSICE on changing number of iterations N while fixing sparsity constant λ and learning rate η to 0.01 and 1.0, respectively. The mean and standard deviation (std.) of the classification performance resulted by iSICE are also shown (only single precision is shown for ease of presentation) for a better understanding of number of iterations changes in iSICE. Results are shown on multiple datasets with VGG-16 and ResNet-50 backbones. The best results in each column (including those that surpass the performance of iSQRT-COV) are highlighted with boldface.

30th epochs. Depending on the dataset and backbones, our fine-tuning process lasts for about 3-8 hours with four P100 GPUs, 12 CPUs and 12GB memory. We provide the source code of our method as supplement material for reproducing the experiments.

C. Robustness of iSICE to Hyper-parameters

This section includes the experiments mentioned in "Robustness of iSICE on Hyper-parameter Changes" of Section 4.2 of the main text. We have conducted comprehensive experiments with the hyper-parameter range mentioned in the main text to demonstrate the robustness of iSICE with respect to hyper-parameter changes. In Tables 1, 2 and 3 of the main text, we showed the results obtained by using a consistent hyper-parameter set across different backbones and datasets to avoid overfitting. Specifically, we showed the results obtained with the median (marked with boldface) of the hyper-parameter range, *i.e.*, $\lambda = \{1.0, 0.5, 0.1, \mathbf{0.01}, 0.001, 0.0001, 0.00001\}$, $\eta = \{0.001, 0.01, 0.1, \mathbf{1.0}, 5.0, 10.0, 20.0\}$, and $N = \{1, \mathbf{5}, 10\}$.

Below we show some experiments to demonstrate the robustness of iSICE with respect to hyper-parameter changes. Specifically, we analyse the performance of iSICE when one hyper-parameter changes while the other two are fixed. For consistency, we experiment with the same hyper-parameters used for reporting the performance of iSICE across the tables of the main text, *i.e.*, $\lambda = 0.01$, $\eta = 1.0$, and $N = 5$. As mentioned above, we will fix two of them and vary the third one to observe its impact to the performance of the proposed iSICE. All of our experiments are compared with COV and Precision $\mathbf{\Omega}$ methods (please refer to the main text for details).

Robustness against sparsity constant changes. In this experiment, we change the sparsity constant λ while keeping the learning rate η and the number of iterations N fixed. Our experimental results are shown in Table 7. From the results, we can clearly see that across all datasets, the change of sparsity constants does not significantly impact the performance of iSICE. The VGG-16 based iSICE shows more robustness toward sparsity constant changes. The classification performance for the MIT dataset appears to have been more significantly impacted due to sparsity constant changes than the other three fine-grained datasets. Specifically, on the three fine-grained datasets, the standard deviation of results is significantly low, *i.e.*, less than 0.51 when compared with the mean values ranging between 85.50 to 93.85. This confirms that our method can be used for fine-grained image classification purposes with a reasonable range of sparsity constant.

Changing the learning rate. In this experiment, we change the learning rate while keeping the sparsity constant and the number of iterations fixed. Our experimental results are shown in Table 8. From the results, we can see that across all datasets, the change in learning rate does not significantly affect the performance. Two datasets, namely Birds and Cars, have shown less impact on performance, as suggested by the standard deviation of 0.14 or lower. The other two datasets also show a low standard deviation of results. The low standard deviation across a wide range of learning rates (from 0.001 to 20.0) shows that our method is robust to the changes in learning rate and a small learning rate such as 0.01 can be used for computing SICE with our method.

Changing the number of iterations. Below we change the number of iterations while keeping the sparsity constant and the learning rate fixed. Our experimental results are shown in Table 9. The results show that regardless of the CNN backbones used, the change in the number of iterations can vary the performance only up to 0.20. It is also noticeable that our method is able to give good performance even with two iterations only. This experiment shows that our method is not sensitive to the changes in the number of iterations.

D. iSICE with Learning Rate and Sparsity Modulators

This section provides additional experiments on iSICE with MLP modulators introduced in "iSICE with learning rate and sparsity modulators" of Section 4.2 of the main text. In Table 4, the CNN feature maps with average pooling were used to learn on-the-fly the learning rate and sparsity. In Table 10, we provide an additional experiment with MLP when both ∇_1 from Alg. 2 and average-pooled feature maps are used, *i.e.*, concatenated before passing them into the modulator. The combination of ∇_1 from Alg. 2 and average-pooled feature maps further improve the classification performance of iSICE.

Method	MIT	Airplane	Birds	Cars	ImageNet100
iSICE	80.5	92.7	85.9	93.5	74.8
iSICE+MLP	81.3	93.4	86.1	**93.9**	76.3
iSICE+MLP*	**81.8**	**93.8**	**86.4**	**93.9**	**77.1**

Table 10. Comparison between the classification performance of iSICE, iSICE+MLP and iSICE+MLP* (improved variant) on ResNet-50. Results of iSICE+MLP in second row use $\mathbf{X}$ to compute learning rate and sparsity with modulators. Results of iSICE+MLP* in the third row use a concatenation of both $\mathbf{X}$ and Δ_1 from Alg. 2 to compute on-the-fly learning rate and sparsity by MLP-based modulators.

E. Memory Consumption

This section describes memory consumption on iSICE mentioned. Algorithm 2 stores the $d \times d$ matrices of $\mathbf{\Sigma}$, $\mathbf{S}_i$, $\mathbf{S}_i^+$ and $\mathbf{S}_i^-$, *etc*. The memory complexity of Algorithm 2 is approximately $O(d^2(N+N_s))$, where d denotes the channel size, N is iSICE iterations, and N_s is Newton-Schulz iterations. For typical $d = 256$, $N = 5$, $N_s = 5$, iSICE uses approximately $3 \times 10 \times 8 \times 256^2 = 0.012$ GB memory which is a tiny fraction of memory that the backbone consumes.

F. Visualisation of Learned Feature Maps

Below we visualize the convolutional feature maps learned by the CNN model with different methods. We extract feature maps from the last convolution layer and perform average pooling on them. We convert the pooled feature map to a heatmap and draw it over the input image. The colour in the heatmap ranges from blue to red, blue indicates cold and red indicates hot. Fig. 5 suggests that with iSICE, the model focuses well on the key parts of car to extract features for classification. GAP (global average pooling) overly focuses on entire foreground, iSQRT focuses poorly, while iSICE lets us control the degree of 'focus' by controlling sparsity.

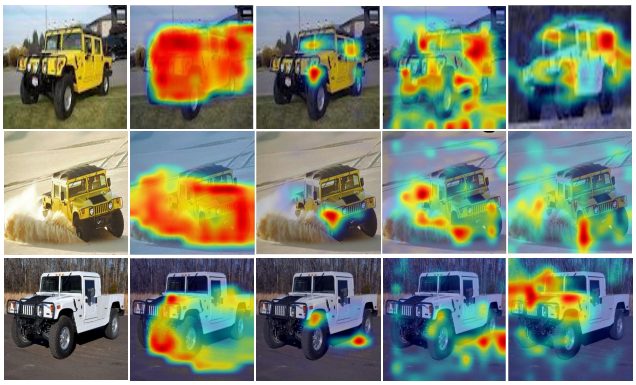

Figure 5. Visualisation of learned convolutional feature maps from Cars dataset with ResNeXt-101 backbone. From left: Input image, GAP, iSQRT-COV, Precision Matrix and iSICE.

References

[56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 13

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 13

[58] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 11

[59] Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 947–955, 2018. 11, 13

[60] Peihua Li, Jiangtao Xie, Qilong Wang, and Wangmeng Zuo. Is second-order information helpful for large-scale visual recognition? In *Proceedings of the IEEE international conference on computer vision*, pages 2070–2078, 2017. 13

[61] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015. 11

[62] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 13

[63] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 13

[64] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 13

[65] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 11

[66] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 13

[67] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE conference on computer vision and pattern recognition*, pages 413–420. IEEE, 2009. 11

[68] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017. 11

[69] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 13

[70] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European conference on computer vision*, pages 776–794. Springer, 2020. 11

[71] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 11

[72] Kaicheng Yu and Mathieu Salzmann. Statistically-motivated second-order pooling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 600–616, 2018. 13