BatchFormer: Learning to Explore Sample Relationships for Robust Representation Learning (Appendix)

Zhi Hou[1], Baosheng Yu[1], Dacheng Tao[2,1]
[1] School of Computer Science, Faculty of Engineering, The University of Sydney, Australia
[2] JD Explore Academy, China
zhou9878@uni.sydney.edu.au, baosheng.yu@sydney.edu.au, dacheng.tao@gmail.com

A. Additional Experimental Details

In all our experiments, if not otherwise stated, we use one layer Transformer Encoder. Meanwhile, The proposed BatchFormer is inserted after the global average pooling layer in ResNet.

Long-Tailed Recognition. For Balanced Softmax [13], we directly insert BatchFormer before the classifier, and train the network with 128 batch size on 1 V100 GPU for 90 epochs. The initial learning rate for Balanced Softmax [13] is 0.05 (it is linearly decreased according to the batch size) and we set the learning rate of the weights of BatchFormer module to 0.005 to avoid overfitting. For RIDE [19], we train the network with a batch size of 400 on 4 V100 GPUs for 100 epochs with an initial learning rare of 0.1 on ImageNet-LT and 0.2 on iNaturalist 2018 respectively. For a fair comparison, we verify BatchFormer based on 3 experts RIDE. Specifically, a shared BatchFormer is incorporated in all expert branches, *i.e*. the module of BatchFormer is shared among 3 expert heads. Experimentally, we find it is not necessary to utilized shared classifier for RIDE. We thus remove shared classifier in RIDE. We think this might be because the multiple experts internally maintain the features between before and after BatchFormer. For Paco [5], we use default hyper-parameters of Paco on ImageNet-LT with four V100 GPUs, on CIFAR-100-LT with a single V100 GPU, and on Places-LT with a single V100 GPU. We simply insert BatchFormer in the query encoder, momentum encoder and prediction head, respectively. Meanwhile, the proposed BatchFormer is shared among those places. See more details in the provided code for MoCo. We find it requires 9 days to train Paco on iNaturalist18 [16] with 4 V100 GPUs. Thus, we directly evaluate BatchFormer on RIDE which only requires 32 hours.

Compositional Zero-Shot Learning. We would like to clarify some details when reproducing the results of [10] on UT-Zap50K. Specifically, we can not fully reproduce the result of [10] on UT-Zap50K due to the missing details. We also find the result with learnable feature extractor of [10] on UT-Zap50K is much better than the result with fixed feature extractor in Table 2 of [10], which is largely different from the results on other datasets. Meanwhile, the same problem has been noticed by others on github issues of [10], i.e., the result can not be unable to fully reproduced (https://github.com/ExplainableML/czsl/issues/4). Therefore, all our experiments are based on the reproduced result for a fair comparison with [10].

Domain Generalization. We mainly evaluate BatchFormer on domain generalization by using the following two baseline methods: Transfer-Learning-Library [7] and SWAD [2]. [7] provides massive traditional and recent methods for domain generalization, and we thus simply apply BatchFormer to those method and evaluate the effectiveness of BatchFormer. Specifically, we use the default setting of [7] for each method, and reproduce the result of each method with [7] for a fair comparison. All experiments are conducted three times and the results are averaged. For SWAD [2], we use the released code and default setting to reproduce the result of ResNet-18.For fair comparison, we also provide the result of ResNet-50.

Self-Supervised Learning. BatchFormer is also pluggable to contrastive learning. Specifically, with BatchFormer, we train MoCo-v2 [3] and MoCo-v3 [4] for 200 epochs and 300 epochs respectively. We follow all default training settings when comparsing with MoCo-v2 and MoCo-v3. Since it is different from supervised learning, we provide the code based on MoCo-v3. As the released code of MoCo requires 16 GPUs with 32GB memory, we conduct all contrastive learning experiments using two cluster nodes with 8 NVIDIA A100 GPUs (40GB) for each node. We keep all other experimental details same as [3] and [4] for a fair comparison.

B. Additional Experiments

To better demonstrate the effectiveness of the proposed BatchFormer, we provide additional experimental results on more tasks.

Table 1. Illustration of BatchFormer on Generalized Zero-Shot Learning based on [18]. Unseen and Seen are the Top-1 accuracies tested on unseen classes and seen classes, respectively, in GZSL.

Method	CUB [17]		
	Unseen	Seen	Harmonic mean
IZF [14]	52.7	**68.0**	59.4
TF-VAEGAN [11]	52.8	64.7	58.1
CE-GZSL [18]	63.9	66.8	65.3
CE-GZSL*(reproduced)	67.5	65.1	66.3
+ BatchFormer	**68.2**	65.8	**67.0**

Table 2. Illustration of BatchFormer for MoCo on VOC2007 [6]. Here, for a fair comparison, we use the the released code of MoCo-v2 and MoCo-v3 to run the experiments in the same setting, and obtain the baseline.

Methods	AP	AP50	AP75
MoCo-v2* [3]	56.4	82.1	63.1
+BatchFormer	**56.7**	**82.0**	**63.6**
MoCo-v3 [3]	46.6	78.2	48.9
MoCo-v3 [3]	**48.0**	**78.8**	**51.1**

B.1. Generalized Zero-Shot Learning

We also evaluate BatchFormer on generalized zero-shot learning task. Specifically, we report the accuracy of "seen", "unseen", and the harmonic mean of them (unseen and seen). We perform experiments on one of the most popular datasets for generalized zero-shot learning, CUB [17], which includes 11,788 images from 200 bird species. We build a baseline with the released code of [18] and achieve better results than [18]. As shown in Table 1, the proposed BatchFormer achieves a new state-of-the-art on Unseen and Harmonic mean.

B.2. Self-Supervised Learning

Object detection on VOC2007. We also evaluate Object Detection of MoCo on VOC2007 [6] in Table 2. Similar as MoCo-v2 [3], we use the pre-trained model to fine-tune Faster-RCNN on VOC2007 based on Detectron2 [21]. We find MoCo-v3 achieves worse result on VOC2007. However, BatchFormer consistently improves the object detection on VOC2007. Here, we train MoCo-v2 for 200 epochs, and MoCo-v3 for 100 epochs. Specifically, we think that the number of training epochs (only 100 epochs) of MoCo-v3 might limit the performance on VOC2007.

B.3. Image Recognition

Table 3 demonstrates BatchFormer for Image Classification. We find BatchFormer achieves comparable performance among ResNet50. This shows BatchFormer does not degrade the performance when the distribution of data is balanced.

Table 3. Illustration of BatchFormer for Image Recognition.

Methods	Epochs	Top-1	Top-5
ResNet50 [20]	200	78.9	
ResNet50 + BatchFormer	200	78.9	-

Table 4. Illustration of BatchFormer for Domain Generalization under different works on PACS [9]. Here, the baseline is from [7]. SWAD [2] is reproduced based on the released code of [2].

Methods	art_paint	cartoon	sketches	photo	Avg.
Baseline	81.3±0.7	76.1±0.6	75.5±2.6	**95.4** ±0.2	82.0
+BatchFormer	**82.4**±1.5	**76.4**±1.2	**75.7**±1.0	95.1±0.4	**82.4**
CORAL [15]	79.2±1.7	**75.5** ±1.1	71.4±3.1	94.7±0.3	80.2
+BatchFormer	**80.6**±0.9	74.7±1.9	**73.1**±0.3	**95.1**±0.3	**80.9**
IRM [1]	**81.0**±0.6	**71.4**±4.1	68.1±7.1	95.0±0.6	78.9
+BatchFormer	78.9±3.1	71.0±7.1	**71.5**±2.8	**96.0**±0.3	**79.4**
V-REx [8]	80.8±1.8	75.3±1.4	73.3±0.9	95.9±0.0	81.3
+ BatchFormer	**82.0**±0.3	**76.3**±0.7	**75.2**±1.7	95.3±0.1	**82.2**
MixStyle [23]	81.7±0.1	**76.8**±0.0	80.8±0.0	93.1±0.0	83.1
+BatchFormer	**84.8** ±0.4	75.3±0.0	**81.1** ±0.4	**93.6**±0.0	**83.7**
SWAD* [2]	83.1±1.5	75.9±0.9	77.1±2.4	95.6±0.6	82.9
+BatchFormer	**84.3**±0.8	**76.9** ±1.2	**78.2**±1.8	95.7±0.6	**83.9**
ResNet50					
V-REx [8]	83.8±4.8	**81.0**±0.0	**97.7**±0.4	77.7±3.1	85.0
+ BatchFormer	**87.3** ±5.0	80.2±4.6	97.1±1.7	**77.9**±4.4	**85.6**
IRM [1]	88.2±0.6	79.8±1.0	97.6±0.5	77.6±0.7	85.8
+ BatchFormer	**89.0**±0.98	**80.1** ±1.0	**98.0**±0.4	**79.8**±0.4	**86.8**
SWAD [2]	89.4±0.7	83.7±1.2	**97.7**±0.6	82.5±0.8	88.1
+BatchFormer	**90.2**±0.5	**84.0**±1.0	97.3±0.3	**83.0**±0.6	**88.6**

Table 5. Illustration of BatchFormer for Domain Generalization based on recent work [2] on OfficeHome

Methods	Art	Clipart	Product	RealWorld	Avg.
SWAD* [2]	54.5±0.8	49.4±0.1	70.9±0.1	72.7±0.2	62.1
+ BatchFormer	**57.8**±0.1	**51.0**±0.1	**73.4**±0.2	**75.1**±0.1	**64.3**
ResNet-50					
IRM [1]	66.8±0.2	54.9±0.8	77.5±0.7	80.5±0.4	69.9
+BatchFormer	**67.7** ±0.2	**55.5**±0.8	**78.4**±0.5	**81.0**±0.3	**70.6**
SWAD* [2]	65.9±0.8	58.0±0.1	78.5±0.5	80.2±0.7	70.6
+ BatchFormer	**66.7**±0.3	57.9±0.3	**79.2**±0.4	**80.6**±0.7	**71.1**

B.4. Domain Generalization

We provide more experimental results based on [7] in Table 4. Experiments on OfficeHome, VLCS, TerraIncognita are provided in Table 5, Table 6 and Table 7 respectively. The default backbone is ResNet-18.

B.5. Domain Adaption

We also demonstrate BatchFormer on Domain Adaption on VisDA2017 [12]. Table 8 shows BatchFormer effectively improves the corresponding baseline, *i.e.*, MDD [22].

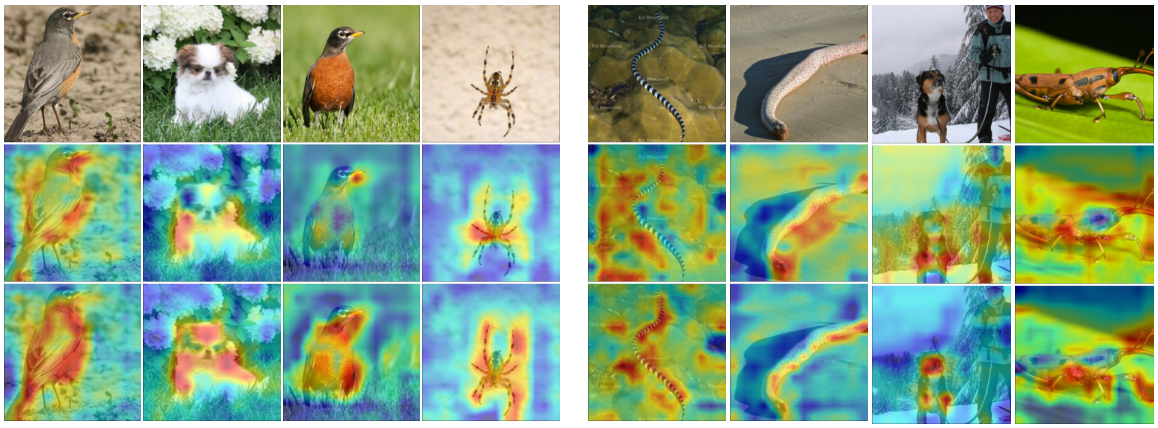

Figure 1. Grad-Cam demonstration of BatchFormer on low-shot test images based on [13]. The left images show BatchFormer enables the model pay attention on more details when the scene is simple, while the right images show BatchFormer facilitates the model ignore the spurious correlation in the image. This is clear version of Figure 5 in the paper.

Table 6. Illustration of BatchFormer for Domain Generalization based on recent work [2] (ResNet-18) on VLCS

Methods	Caltech101	LabelMe	SUN09	SUN09	Avg.
SWAD* [2]	97.2±1.4	61.4±0.1	71.2±1.7	75.5±0.8	76.3
+ BatchFormer	97.2±0.8	61.3±1.1	**71.7**±1.0	**77.4**±0.4	**76.9**

Table 7. Illustration of BatchFormer for Domain Generalization based on recent work [2] (ResNet-18) on TerraIncognita

Methods	Art	Clipart	Product	RealWorld	Avg.
SWAD* [2]	47.6±3.0	33.8±4.5	53.6±1.8	33.3±0.6	42.1
+ BatchFormer	**49.8**±1.8	**40.3**±2.0	**55.2**±1.2	**34.0**±1.1	**44.8**

Table 8. Illustration of BatchFormer for Domain Adaption on VisDA2017 [12]. The backbone is ResNet-101. Experiments are based on [7].

Methods	Synthetic $->$ Real
MDD [22]	76.8±1.5
+BatchFormer	**77.8** ±2.0

Table 9. Ablation Studies of different layers on ImageNet-LT based on Balanced Softmax [13]. The backbone is ResNet-10.

Method	All	Many	Medium	Few
BatchFormer (1 layers)	43.2	52.8	40.4	25.6
2 layers	43.2	52.8	40.3	26.0
4 layers	42.7	52.1	40.1	25.4
8 layers	43.3	53.3	40.2	26.1
16 layers	43.1	52.9	40.0	26.6

C. Ablation Studies

Number of Layers. We use one layer BatchFormer in our experiment. Table 9 demonstrates with more layers of BatchFormer, we do not observe larger improvement. We leave how to leverage more layers of BatchFormer to improve the performance in future work.

BatchFormer without PaCo loss [5]. We notice BatchFormer mainly improves PaCo on Many category on CIFAR-100-LT (imbalance ratio 100). We thus conduct additional ablation study on BatchFormer for PaCo. Here, we remove the PaCo loss with Balanced loss [5] to build the baseline. we observe consistent results in the Table 10. Noticeably, the baseline without PaCo loss is even better than the one in the main paper.

D. Visualized Comparison

Figure 1 provides clear figure of the Grad-Cam Figure in the main paper. More comparisons are included in Figure 2, Figure 3, Figure 4, where we choose the top 100 classes on ImageNet for demonstration.

Table 10. Illustration of BatchFormer without PaCo loss [5] on CIFAR-LT-100.

Method	100				200			
	All	Many	Med	Few	All	Many	Med	Few
Baseline [52]	52.0	68.1	53.2	31.6	47.31	67.8	52.6	27.3
+ BatchFormer	**52.6**	**68.7**	53.2	**33.1**	**48.13**	**68.9**	**53.1**	**28.2**

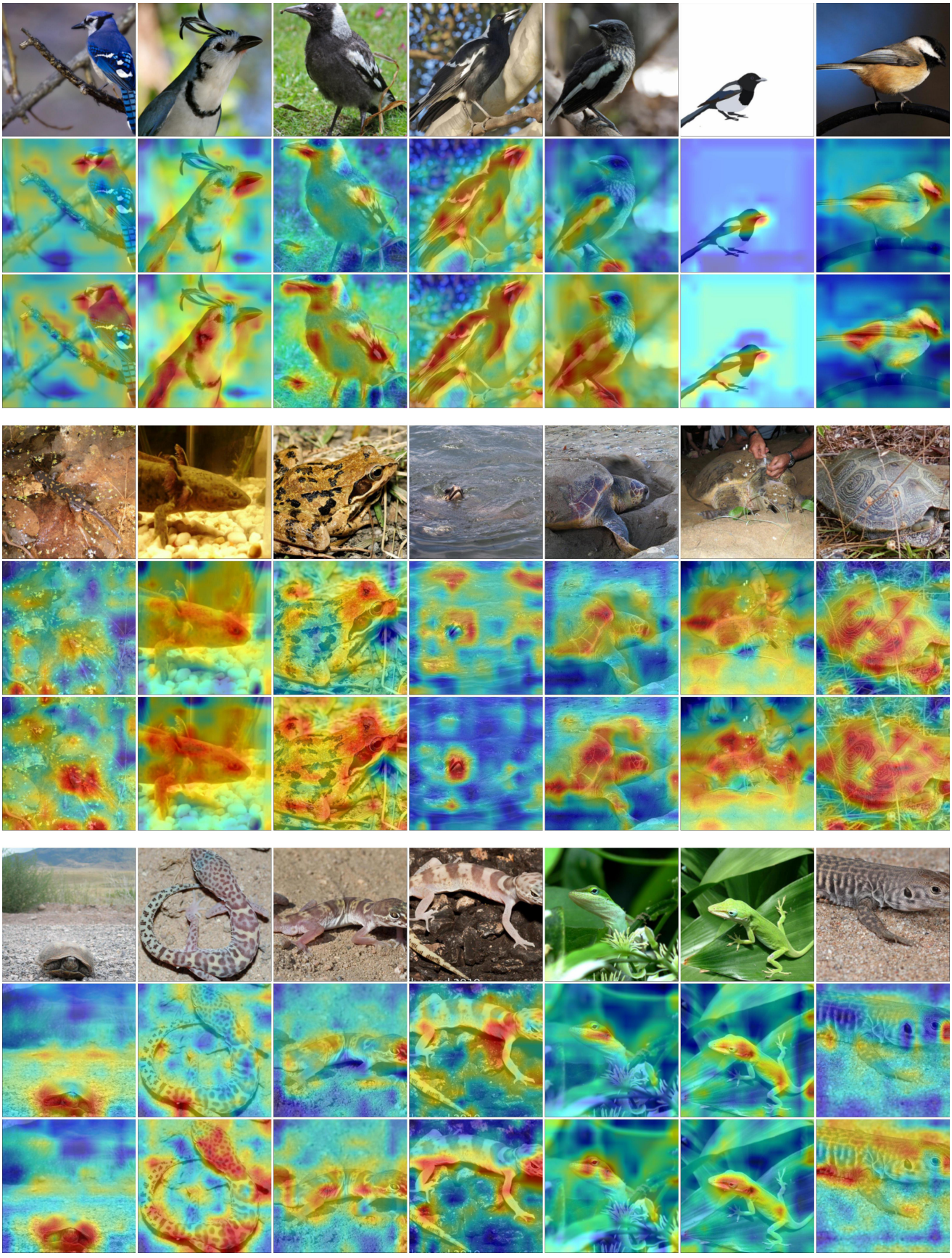

Figure 2. More Grad-Cam illustration of BatchFormer on low-shot test images based on [13]. The figures are also provided in the directory.

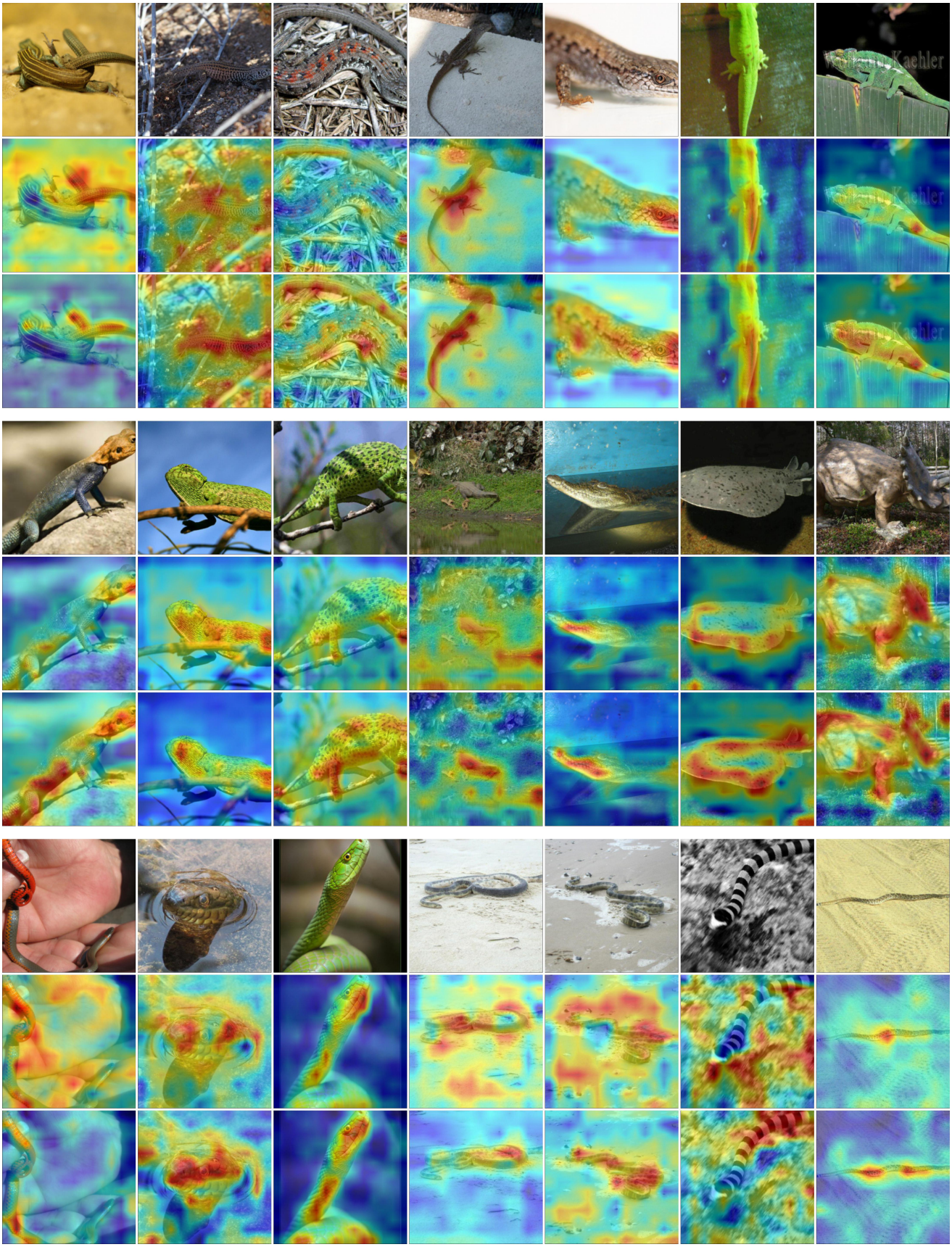

Figure 3. More Grad-Cam illustration of BatchFormer on low-shot test images based on [13]. The figures are also provided in the directory.

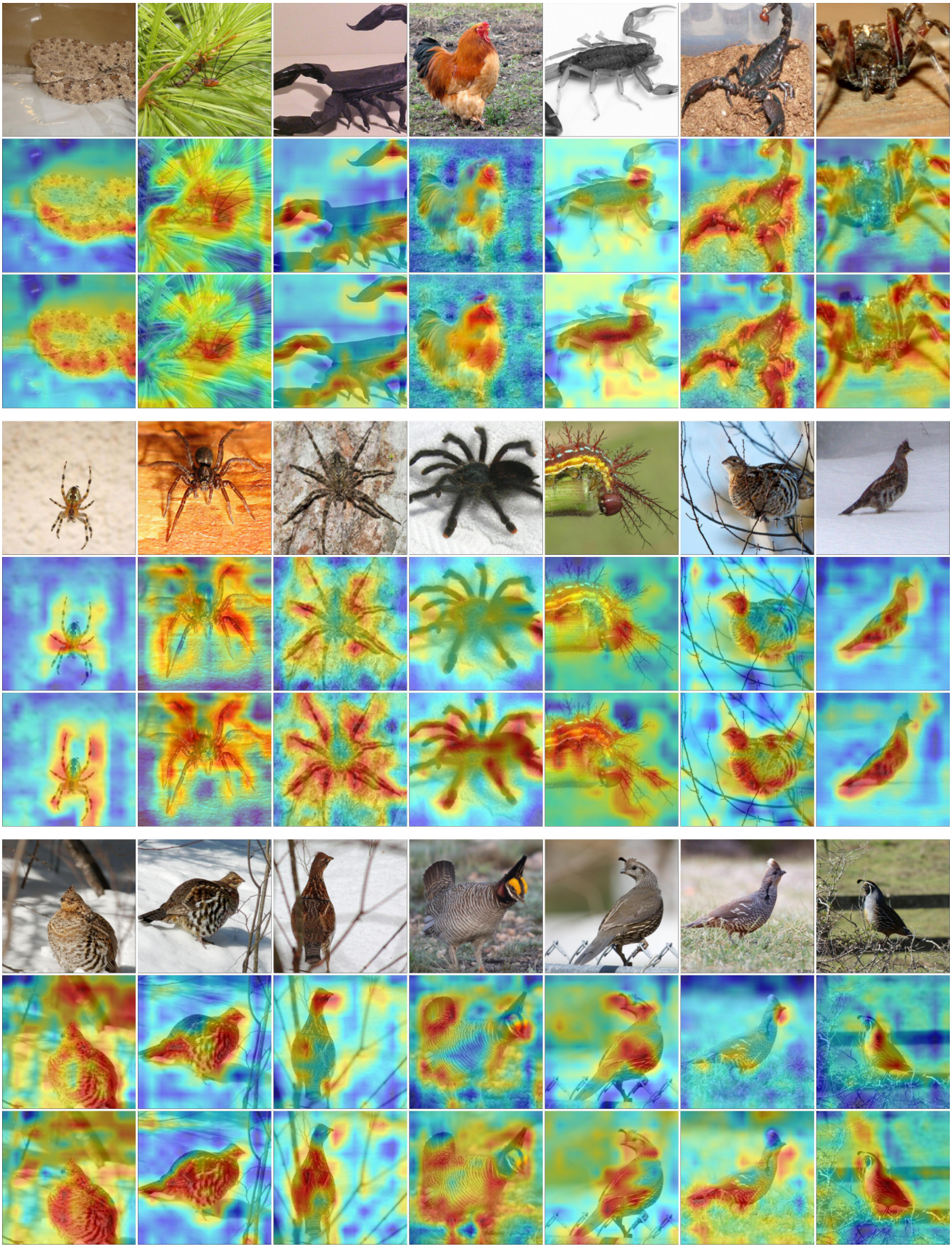

Figure 4. More Grad-Cam illustration of BatchFormer on low-shot test images based on [13]. The figures are also provided in the directory.

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 2

[2] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. In *NeurIPS*, 2021. 1, 2, 3

[3] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2

[4] Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. In *CVPR*, 2021. 1

[5] Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. Parametric contrastive learning. In *ICCV*, 2021. 1, 3

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2

[7] Junguang Jiang, Baixu Chen, Bo Fu, and Mingsheng Long. Transfer-learning-library. https://github.com/thuml/Transfer-Learning-Library, 2020. 1, 2, 3

[8] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *ICML*, 2021. 2

[9] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, pages 5542–5550, 2017. 2

[10] MF Naeem, Y Xian, F Tombari, and Zeynep Akata. Learning graph embeddings for compositional zero-shot learning. In *CVPR*. IEEE, 2021. 1

[11] Sanath Narayan, Akshita Gupta, Fahad Shahbaz Khan, Cees GM Snoek, and Ling Shao. Latent embedding feedback and discriminative features for zero-shot classification. In *ECCV*, pages 479–495. Springer, 2020. 2

[12] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017. 2, 3

[13] Jiawei Ren, Cunjun Yu, Shunan Sheng, Xiao Ma, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Balanced meta-softmax for long-tailed visual recognition. In *NeurIPS*, 2020. 1, 3, 4, 5, 6

[14] Yuming Shen, Jie Qin, Lei Huang, Li Liu, Fan Zhu, and Ling Shao. Invertible zero-shot recognition flows. In *ECCV*, pages 614–631. Springer, 2020. 2

[15] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016. 2

[16] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, pages 8769–8778, 2018. 1

[17] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2

[18] Peng Wang, Kai Han, Xiu-Shen Wei, Lei Zhang, and Lei Wang. Contrastive learning based hybrid networks for long-tailed image classification. In *CVPR*, pages 943–952, 2021. 2

[19] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella X Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*, 2021. 1

[20] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 2

[21] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 2

[22] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *ICML*, 2019. 2, 3

[23] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. *ICLR*, 2021. 2