

Supplementary Materials

8. Training Details

Dataset splits: The CUB and FGVC-Aircraft datasets contain class lists in a canonical order (found in classes.txt and variants.txt respectively). Using this ordering we assign each class an id starting from 1 for CUB and 0 for FGVC respectively. For each dataset, we split the categories by id into 50% base, 25% validation and 25% novel classes following the rule: if $\text{id} \bmod 2 = 0$, $\text{id} \in \text{base}$, else if $\text{id} \bmod 4 = 1$, $\text{id} \in \text{validation}$, otherwise $\text{id} \in \text{novel}$.

Hyper-parameters for CUB: For each model, we use the validation set to select the best hyper-parameters as well as the best stopping iteration during the training process. Here we give all the hyper-parameters finally chosen for training on CUB. All SGD optimizers use a default 0.9 momentum. PN models involve an additional hyper-parameter, the multiplier α for L_{pose} (see formula 3 in section 4). This value is chosen as 100 for the 4-layer ConvNet and 200 for ResNet18.

For prototypical network based methods, we follow the same meta-training technique and batch sampling as [29] and set each episode to 20-way 5-shot classification with 15 query images per class. The only exception is proto+FSL, for which we follow the same setting as [29] and make each class in the episode consist of 5 refer shots for aggregating fore/background vectors, 5 shots for training for 10 shots for testing. Here we define one epoch as one pass over the whole representation set. The whole training process can then be divided into s stages, each stage containing e epochs, and the model is trained with optimizer o and weight decay β , using an initial learning rate of lr and cut by multiplying a factor γ after finishing each stage. These hyper-parameters are shown in table 4. In addition, proto+bbN chooses $\alpha=10$ and proto+MT choose the same α value as proto+PN. During the whole training process, we evaluate the model on validation set every 20 epochs and select the best one to make the final evaluation on test set.

For transfer learning based methods, following the same notation as above, we list in table 4 the hyper-parameters used for pre-training on base classes with batch size 64. For the finetuning on novel classes, we finetune the new classifier for 40 epochs with Adam, using learning rate of 0.001 and batch size of 16.

For dynamic few-shot learning based methods, we list in table 4 the hyper-parameters used for the first training stage with batch size 64. For the second training stage, in each batch, we random sample 16 fake novel and 4 base classes, each class containing 20 images. We train the weight generator for 200 epochs with Adam, using learning rate 0.001. During the second training stage, we evaluate the model on validation set every 20 epochs and select the best one to make the final evaluation on test set.

Model	4-layer ConvNet						ResNet18					
	optimizer	lr	γ	epoch	stage	weight decay	optimizer	lr	γ	epoch	stage	weight decay
transfer	SGD	0.1	0.1	200	2	5e-4	SGD	0.1	0.1	100	2	1e-3
transfer+PN	SGD	0.1	0.1	200	2	5e-4	SGD	0.1	0.1	100	2	1e-3
transfer+PN_gt	SGD	0.1	0.1	200	2	5e-4	SGD	0.1	0.1	100	2	1e-3
proto	SGD	0.1	0.1	400	2	5e-4	SGD	0.1	0.1	300	2	1e-3
proto+MT	SGD	0.1	0.1	600	2	1e-3	SGD	0.1	0.1	300	2	5e-3
proto+BP	Adam	0.001	NA	800	1	0	Adam	0.001	NA	600	1	1e-3
proto+FSL	SGD	0.01	0.1	400	2	5e-4	SGD	0.1	0.1	300	2	1e-3
proto+bbN	SGD	0.01	0.1	400	2	5e-4	Adam	0.1	0.5	160	5	0
proto+uPN	SGD	0.1	0.1	600	2	1e-3	SGD	0.1	0.1	200	2	5e-3
proto+PN	SGD	0.1	0.1	600	2	1e-3	SGD	0.1	0.1	300	2	5e-3
proto+PN_gt	SGD	0.1	0.1	400	2	5e-4	SGD	0.1	0.1	300	2	5e-3
dynamic	SGD	0.1	0.1	200	2	5e-4	SGD	0.1	0.1	100	2	1e-3
dynamic+PN	SGD	0.1	0.1	100	2	5e-4	SGD	0.1	0.1	25	3	1e-3
dynamic+PN_gt	SGD	0.1	0.1	50	2	5e-4	SGD	0.1	0.1	25	3	1e-3

Table 4. The hyper-parameters selected for training on CUB.

Hyper-parameters for FGVC-Aircraft: Same as above, we set each meta-learning episode to 20-way 5-shot classification with 15 query images per class. For each episode in PN models, in addition to calculating $L_{fewshot}$ using predicted pose heatmaps on FGVC, we also randomly sample 400 images from OID to calculate L_{pose} . In table 5 we give all hyper-parameters chosen for training. All SGD optimizers use a default 0.9 momentum. For PN models, we select $\alpha=50$. During the whole training process, we evaluate the model on validation set every 40 epochs and select the best one to make the final evaluation on test set.

4-layer ConvNet							ResNet18						
Model	optimizer	lr	γ	epoch	stage	weight decay	optimizer	lr	γ	epoch	stage	weight decay	
proto	SGD	0.1	0.1	500	2	1e-3	SGD	0.1	0.1	300	2	1e-3	
proto+PN	SGD	0.1	0.1	500	2	1e-3	SGD	0.1	0.1	300	2	5e-3	

Table 5. The hyper-parameters selected for training on FGVC-Aircraft.

9. Numerical Experiment Results

In table 6, we list all the numerical results for prototypical and dynamic based methods in all three evaluation settings.

Model	4-layer ConvNet			ResNet18		
	1-shot	5-shot	all-shot	1-shot	5-shot	all-shot
proto*	14.63	27.63	32.09	23.77	38.76	42.73
proto+MT	17.05	31.52	35.56	30.90	47.78	50.93
proto+BP	15.07	28.36	35.56	21.04	37.15	41.04
proto+FSL	18.01	34.44	39.60	26.04	42.35	47.43
proto+bbN	15.87	30.63	37.75	24.05	39.60	44.02
proto+uPN*	19.06	39.48	46.24	28.06	47.10	53.18
proto+PN*	19.69	43.05	49.56	34.90	58.64	63.44
proto+PN_gt	22.14	51.62	59.55	31.04	57.16	62.63
dynamic	16.02	28.59	35.77	24.24	38.64	43.27
dynamic+PN	27.77	47.72	54.17	33.98	54.27	60.19
dynamic+PN_gt	34.00	56.32	62.67	33.90	54.60	60.09

Table 6. All the numerical results on CUB. * indicates the results are averaged within 8 random trails for that model.

10. Training with Less Part Annotation

Here we give the training details for the experiments in section 5.4. The training settings and hyper-parameters are the same as described in section 8. Similar to training on FGVC-Aircraft, we calculate $L_{fewshot}$ over each batch using the predicted pose heatmap. At the same time, we randomly sample a subset of images with part annotations from the same 20 classes to calculate L_{pose} . Here we define the batch size as the number of images with part annotation per class in each iteration. The detailed numerical results are shown in table 7.

	percentage of training images with part annotation										
	5	10	20	30	40	50	60	70	80	90	100
batch size	1	5	7	10	15	15	15	15	17	20	20
4-layer ConvNet	40.66	44.49	46.47	47.77	47.10	47.60	50.97	49.33	49.41	51.26	50.51
ResNet18	52.78	54.17	57.53	61.41	63.51	64.52	64.35	65.57	65.87	65.99	66.33

Table 7. For different percentages of training images with part annotation, we list the number of sampled part-annotated images per class per batch, as well as the final all-shot evaluation accuracy on both shallow and deep network backbones.

11. More Examples for Nearest Neighbor

In figure 10, we show more visualization examples for the nearest neighbor experiment in section 6.1.

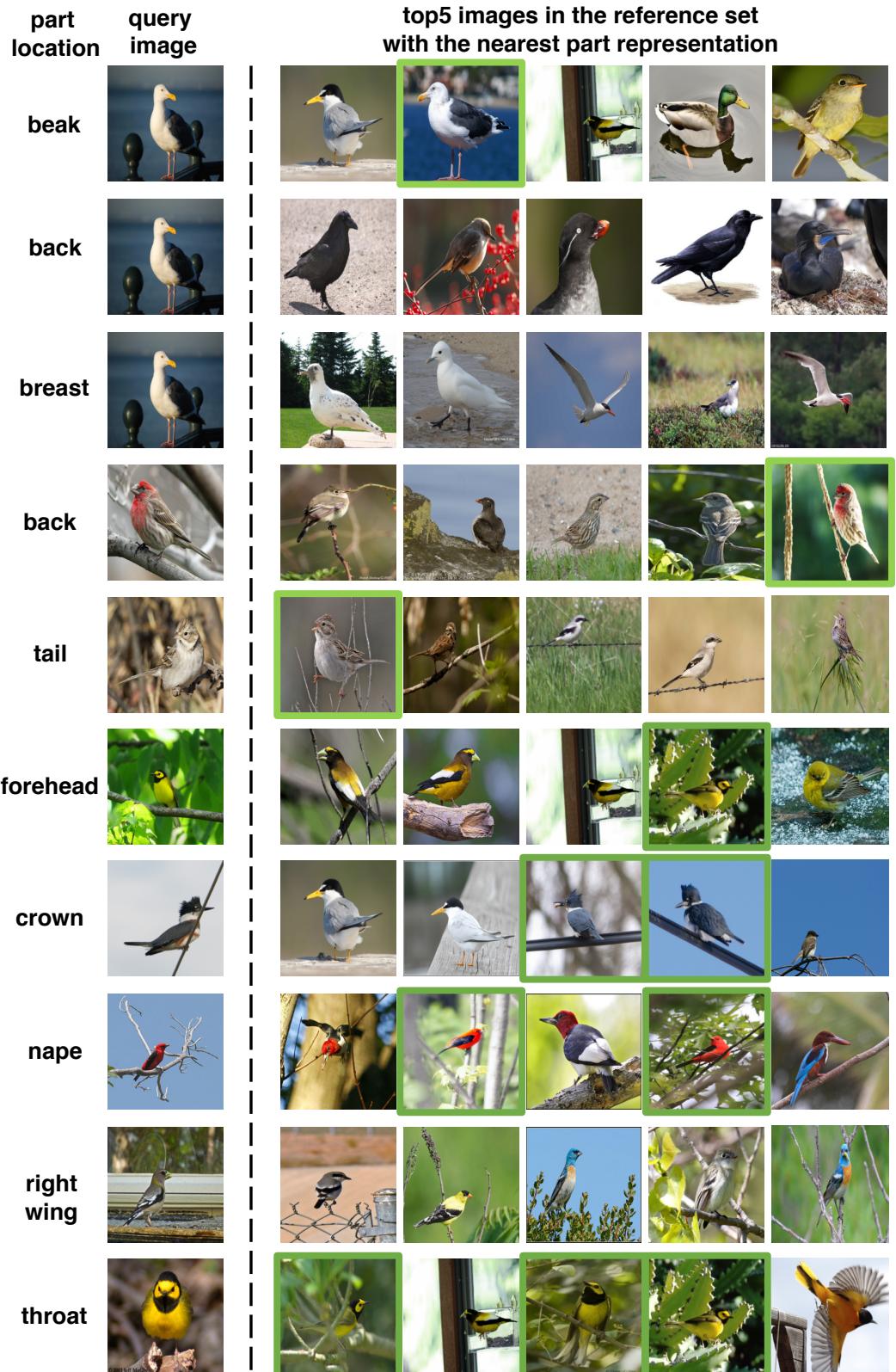


Figure 10. Images with the closest part vector to the query image, for a given part location. Image is labeled with a green box if it belongs to the same class as the query image.