Supplementary Material for "Joint distribution Matters: Deep Brownian Distance Covariance for Few-Shot Learning"

Jiangtao Xie[1,*], Fei Long[1,*], Jiaming Lv[1], Qilong Wang[2], Peihua Li[1,†]
[1]Dalian University of Technology, China [2]Tianjin University, China

In this supplement, we present details on the implementations of our DeepBDC and the counterparts. Besides, we conduct experiments providing additional ablation study and comparison. We finally show BDC's ability to characterize non-linear dependence. The source code is available at http://www.peihuali.org/DeepBDC.

S1 Implementations

S1-1 Benchmarks

***mini*ImageNet** The *mini*ImageNet [S-22] is a few-shot benchmark constructed from ImageNet [S-6] for general object recognition. It consists of 100 classes each of which contains 600 images. Following previous works [S-4, S-25, S-26], we use the splits provided by [S-18], which involves 64 classes for meta-training, 16 classes for meta-validation and the remaining 20 classes for meta-testing.

***tiered*ImageNet** The *tiered*ImageNet [S-19] is also a few-shot benchmark for general object recognition. It is constructed from ImageNet [S-6] as well, which, different from *mini*ImageNet, considers the hierarchical structure of ImageNet. This dataset contains 608 classes from 34 super-classes and a total of 779,165 images. Among these classes, 20 super-classes (351 classes) are used for meta-training, 6 super-classes (97 classes) for meta-validation and the remaining 8 super-classes (160 classes) for meta-testing.

CUB Caltech-**UCSD** **B**irds-200-2011 [S-23] (CUB) dataset is a widely used fine-grained categorization benchmark. This dataset contains 200 bird classes with 11,788 images in total. We use the splits of [S-4], in which the total classes are divided into 100/50/50 for meta-training/validation/testing. Following [S-4, S-14, S-1], we conduct experiments on CUB with the original raw images, instead of cropped images via annotated bounding boxes [S-26, S-27].

***mini*ImageNet → CUB** Chen *et al*. [S-4] build the cross-domain task for assessing the domain transfer ability of the models. In this setting, all 100 classes of *mini*ImageNet are used for meta-training, while the models are evaluated on the meta-testing set (50 classes) of CUB.

***mini*ImageNet → Aircraft** Aircraft [S-15] contains 10,000 images from 100 classes. We perform meta-training on the whole *mini*ImageNet; we adopt the splits on Aircraft proposed by [S-25], where 25 classes are used for meta-validation and 25 classes are for meta-testing. Same as [S-25], we conduct experiments with the images cropped by using the bounding box annotations.

***mini*ImageNet → Cars** Stanford Cars [S-10] (Cars) contains 196 classes and a total 16,185 images. We follow the splits of [S-13] to build the meta-validation set (17 classes) and meta-testing set (49 classes). Similar to the other cross-domain benchmarks, the full *mini*ImageNet is used as meta-training set.

S1-2 Architectures

For fair comparisons with previous methods, we use two kinds of networks as backbones, i.e., ResNet-12 and ResNet-18. Following previous practice, the input resolution of images is 84×84 for ResNet-12, and 224×224 for ResNet-18, respectively. Besides, we use higher capacity models, i.e., ResNet-34 with input images of 224×224 and its variant fit for input images of 84×84.

As in [S-21, S-11, S-16, S-25], ResNet-12 consists of four stages each of which contains one residual block. The widths of the residual blocks of the four stages are [64, 160, 320, 640]. Each residual block has three 3×3 convolution layers with a batch normalization and a 0.1 leaky ReLU. Right after every block except the last one, a 2×2 max pooling layer is used to down-sample the feature maps. We adopt ResNet-18 and ResNet-34 proposed in [S-8] with the last down-sampling being removed. The architecture of the variant of ResNet-34 is same as that of ResNet-12, except that the numbers of residual blocks of the four stages are [2, 3, 4, 2], instead of [1, 1, 1, 1] in ResNet-12.

*Equal contribution. †Corresponding author, peihuali@dlut.edu.cn. The work was supported by National Natural Science Foundation of China (61971086, 61806140), and CCF-Baidu Open Fund (2021PP15002000).

Method	Hyper-params	*mini*ImageNet	*tiered*ImageNet	CUB	*mini*ImageNet → CUB/Aircraft/Cars
Meta DeepBDC	Initial LR	1e-4	5e-5	1e-3	5e-5
	LR decay	[40, 80]*0.1	[70]*0.1	[40, 80]*0.1	[70]*0.1
	Epochs	100	100	100	100
STL DeepBDC	Initial LR	5e-2	5e-2	5e-2	5e-2
	LR decay	[100,150]*0.1	[40,70]*0.1	[120,170]*0.1	[100,150]*0.1
	Epochs	180	100	220	180

Table S-1. Hyperparameter settings of our DeepBDC.

S1-3 Training and Evaluation Protocols

Training During training, following [S-4, S-25, S-27], we use standard data augmentation including random resized crop, color jittering and random horizontal flip on all benchmarks. We adopt the SGD algorithm with a momentum of 0.9 and a weight decay of 5e-4 to train our proposed DeepBDC networks. For ResNet-12, we apply DropBlock [S-7] regularization during training as in [S-26, S-11, S-5]. We tune the number of epochs and the scheduling of learning rate (LR) on different benchmarks.

For our *Meta DeepBDC* which is based on meta-learning framework, we train the model by uniformly sampling episodes (tasks) from the meta-training set. Following previous works [S-5, S-25, S-27], right before performing the episodic training, we pre-train the networks on the full meta-training set whose weights are used as initialization. To sample a 5-way 1-shot task, we randomly pick up 5 classes each with 1 support image and 16 query images selected at random. Similarly, every 5-way 5-shot task contains 5 support images and 16 query images. Tab. S-1 (upper part) shows the hyper-parameter settings for training Meta DeepBDC on all benchmarks. Let us take *mini*ImageNet as an example: the initial learning rate (LR) is 1e-4, which is divided by 10 at epoch 40 and 80, respectively, and training proceeds until epoch 100. We adopt the model achieving highest accuracy on the meta-validation for evaluation on the meta-testing set.

Our *STL DeepBDC* is based on Good-Embed [S-21], which falls into simple transfer learning framework, not requiring episodic training. In practice, we train the network for a common multi-way classification task, using a softmax classifier via a standard cross-entropy loss on the whole meta-training set spanning all classes. We set the batch size to 64 across all benchmarks. Furthermore, following [S-21], we conduct sequential self-distillation to distill knowledge from the trained model. The networks thus obtained are used as embedding models for extracting features (i.e., outputs of the last convolution layer of one network). The hyper-parameter settings for STL DeepBDC are shown in Tab. S-1 (bottom part).

Evaluation We uniformly sample episodes (tasks) from the meta-testing set to evaluate the models' performance. Following [S-4], we build 5-way 1-shot or 5-shot setting, respectively, both with 15 query images. We report mean accuracy of 2,000 episodes with 95% confidence intervals. Our Meta DeepBDC does not require additional training, so we directly performing testing. Our STL DeepBDC needs to train a linear classifier for each episode using the trained model for feature extraction. We implement the logistic regression via L-BFGS-B algorithm and linear SVM via LIBSVM based on scikit-learn software package [S-17]. We perform L2 normalization for the features in logistic regression and SVM; furthermore, we standardize the normalized features before fed to SVM. Implementation of the soft-max classifier is based on PyTorch, where we adopt SGD algorithm with a batch size of 4, a momentum of 0.9, a weight decay of 1e-3 and a learning rate of 1e-2.

S2 Implementation of the Counterparts

ProtoNet We use implementation of Chen *et al*. [S-4] which is public available [1].

ADM We use the code released by the authors [S-12] [2]. Differently, we add one 1×1 convolution for dimension reduction before computing mean vectors and covariance matrices. As the original method performs unsatisfactorily, we use a different one fusing ADM and DN4 [S-13]. Specifically, we attach independently the ADM branch and DN4 branch to the backbone, both with individually learnable scaling parameters. Next, the negative KL-divergence score and Image-to-Class (I2C) score are separately fed to softmax and are then added for the final cross entropy loss. We combine Meta DeepBDC and DN4 similarly.

CovNet We mainly follow implementation of the authors [S-24] [3]. Practically, we have two differences: (1) we introduce a 1×1 convolution for dimension reduction, and (2) for 5-way 1-shot classification, we use the inner product as the metric, rather than the Frobenious norm which produces poor results.

[1] https://github.com/wyharveychen/CloserLookFewShot
[2] https://github.com/WenbinLee/ADM
[3] https://github.com/daviswer/fewshotlocal

Method	ProtoNet†		Good-Embed†		Meta DeepBDC		STL DeepBDC	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ResNet-12	62.11	80.77	64.98	82.10	67.34	84.46	67.83	85.45
ResNet-34	64.56	81.16	66.14	82.39	68.20	84.97	**68.66**	**85.47**
Δ	2.45	0.39	1.16	0.29	0.86	0.51	0.83	0.02

Table S-2. Comparison of different capacity models on *mini*ImageNet with input images of 84×84. The ResNet-34 model is a variant of [S-8] which is described in Sec. S1-2. † Reproduced with our setting.

Method	ProtoNet†		Good-Embed†		Meta DeepBDC		STL DeepBDC	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ResNet-18	80.90	89.81	77.92	89.94	83.55	93.82	84.01	94.02
ResNet-34	80.58	90.11	79.33	90.10	**85.25**	94.31	84.69	**94.33**
Δ	-0.32	0.30	1.41	0.16	1.70	0.49	0.68	0.31

Table S-3. Comparsion of different capacity models on CUB with input images of 224×224. Here we use ResNet-34 proposed in [S-8]. † Reproduced with our setting.

For all of the aforementioned methods, we remove the last down-sampling of the backbones. Following the previous practice [S-25, S-5, S-28], we employ the weights of pre-trained models as initialization before performing episodic training.

S3 Additional Experiments

This section introduces additional experiments to ablate our DeepBDC and to compare with the counterparts.

S3-1 On prototype in Meta DeepBDC

In Meta DeepBDC, for the 5-shot setting, the prototype of a support class is computed as the average of BDC matrices of 5 support images belonging to this class. Here, we evaluate two other options for computing the prototype. (1) We average features of 5 support images, and then the averaged features are used to compute the BDC matrix as the prototype. (2) We concatenate features of 5 support images for computing the BDC matrix as the prototype. These two methods achieve accuracies (%) of 82.36 and 83.74 on *mini*ImageNet, respectively, which are lower than the accuracy obtained by averaging 5 support BDC matrices (84.46).

S3-2 Effect of Higher Capacity Models

To evaluate the effect of higher capacity models, we conduct experiments on CUB using the ResNet-34 with 224×224 input images, and on *mini*ImageNet using the variant of ResNet-34 with input images of 84×84. We compare our Meta DeepBDC and STL DeepBDC with their respective baselines, i.e., ProtoNet and Good-Embed.

The comparison on *mini*ImageNet is shown in Tab. S-2. It can be seen that, for every method with either setting, the accuracy obtained by using high-capacity ResNet-34 is higher than that using ResNet-12. Among them, the improvements of ProtoNet and Good-Embed for 1-shot task are significant (over 1 percentage points). Despite the improvements, our Meta DeepBDC and STL DeepBDC outperform their corresponding baselines by large margins. According to Tab. S-3 which presents results on CUB, we can see that overall all methods improve when using higher-capacity ResNet-34, while the improvements of ProtoNet are not significant. Again, we observe that our methods are significantly superior to their baselines.

Method	5×5		10×10		21×21	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet [S-20] †	61.81	79.62	62.11	80.77	61.45	80.00
Good-Embed [S-21] †	65.73	83.08	64.98	82.10	64.68	81.85
Meta DeepBDC	66.74	83.83	67.34	84.46	66.83	84.20
STL DeepBDC	67.76	85.39	67.83	85.45	67.44	85.44

Table S-4. Accuracy (%) against number of features on *mini*ImageNet for 5-way classification. † Reproduced with our setting.

S3-3 Effect of Feature Number on DeepBDC

To obtain more convolutional features, CTX [S-3] removes the last down-sampling in the backbone networks, while ADM [S-12] removes the last two. Similar to them, we also remove down-sampling. On *mini*ImageNet, we evaluate the effect of down-sampling on our DeepBDC. As the input resolution is 84×84 for ResNet-12, the spatial size of feature maps outputted by the original backbone is 5×5 and thus we have a total of 25 features; the spatial sizes become 10×10 and 21×21 if the last down-sampling and the last two are eliminated, respectively.

Tab. S-4 summarize the results. We first notice that variation of feature number has minor effect on our STL DeepBDC for either 1-shot or 5-shot task. Regarding our Meta DeepBDC, it can be seen that for both 1-shot and 5-shot tasks, the accuracy increases slightly when the number of features is 100, but then decreases when provided with 441 features. At last, we note that ProtoNet and Good-Embed achieves individual best results when the feature number is 100 and 25, respectively. Throughout the main paper, we report results with removal of the last down-sampling.

S3-4 Comparison of Latency of Meta-training Task

In the main paper, we compare the latency of meta-testing task. Here, we give additional comparison for meta-training task, which is also important in practice. The latency is measured on *mini*ImageNet with the backbone of ResNet-12. Following the setting of DeepEMD [S-27],

Method	Meta-training		Meta-testing		Accuracy	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet [S-20] †	304	365	115	143	62.11	80.77
ADM [S-12] †	908	967	199	221	65.87	82.05
CovNet [S-24] †	310	374	120	144	64.59	82.02
DeepEMD [S-27]	>80K	$>10^6$	457	12,617	65.91	82.41
Meta DeepBDC	505	623	161	198	67.34	84.46
STL DeepBDC	–		184	245	67.83	85.45

Table S-5. Comparison of latency (ms) for 5-way classification on *mini*ImageNet. † Reproduced with our setting.

we adopt QPTH solver [S-2] in meta-training and OpenCV solver in meta-testing, using the code released by the authors[4].

Tab. S-5 shows comparison results for 5-way classification; for reference, we also include latency of meta-testing task and recognition accuracies which have been discussed in the primary paper. As regards the meta-training, we find that ProtoNet and CovNet are fastest, while our Meta DeepBDC is somewhat slower than them. Though the meta-testing speed of ADM is comparable to that of our method, its meta-training latency is much larger than ours; the reason is that backpropagation of ADM involves GPU-unfriendly matrix inversions. Notably, DeepEMD is at least 80 times slower for 1-shot and 1000 times slower for 5-shot than the other methods; we mention that FRN also observes the big latency of DeepEMD [S-25].

S3-5 Effect of Channel Number on DeepBDC and the Counterparts

As described in the main paper, both of CovNet and ADM need to estimate second moments, leading to quadratic increase of representations in channel number d. Therefore, for fair comparison, we also add a 1×1 convolution for them to reduce the number of channels. Dimension reduction is hurtful for ProtoNet and DeepEMD, so for them we leave the original channel as it is.

Fig. S-1 plots the curves of accuracies as a function of d. In light of the curves, we can draw several conclusions as follows. (1) The channel number d has non-trivial effect on ADM and CovNet. The accuracies (%) of ADM and CovNet reach the highest values when $d = 196$ (82.05) and $d = 256$ (82.02), respectively. Their accuracies drop gradually when d becomes larger, and when $d = 640$, they achieve accuracies only slightly higher than ProtoNet. (2) Across all values of d, both instantiations of our DeepBDC clearly perform better than the competing methods.

We mention that our re-implementation non-trivially improves performance of CovNet and ADM, providing fair and competitive baselines. Besides, these results show that dimension reduction plays an important role for second moment-based methods.

[4] https://github.com/icoz69/DeepEMD

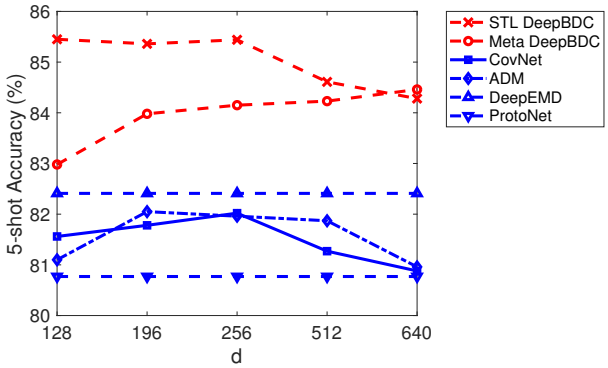


Figure S-1. Accuracy as a function of channel number d for 5-way 5-shot classification on *mini*ImageNet.

S4 Linear and Non-linear Relation Modeling

One of the favorable properties of Brownian Distance Covariance (BDC) is the ability to model both linear and non-linear dependency between random variables X and Y. In contrast, traditional covariance can only model linear relations. To facilitate visual understanding, we consider five simulated examples of bivariate distributions [S-9], i.e., "W-shape", "Diamond", "Parabola", "Two parabolas" and "Circle", and two examples we developed, i.e., "Butterfly" and "Heart", respectively. In these examples, two random variables X and Y have different kinds of non-linear relationships. Also, we simulate seven kinds of linear relations based on HHG package [5]. For each set of observation pairs, we compute the classical correlation

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sqrt{\mathrm{cov}(X,X)}\sqrt{\mathrm{cov}(Y,Y)}} \tag{S-1}$$

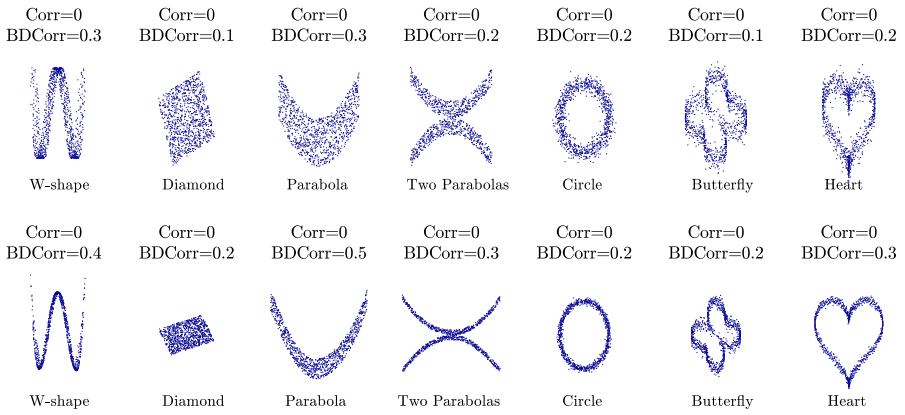
and Brownian distance correlation

$$\mathrm{BDCorr}(X,Y) = \frac{\mathrm{BDC}(X,Y)}{\sqrt{\mathrm{BDC}(X,X)}\sqrt{\mathrm{BDC}(Y,Y)}} \tag{S-2}$$

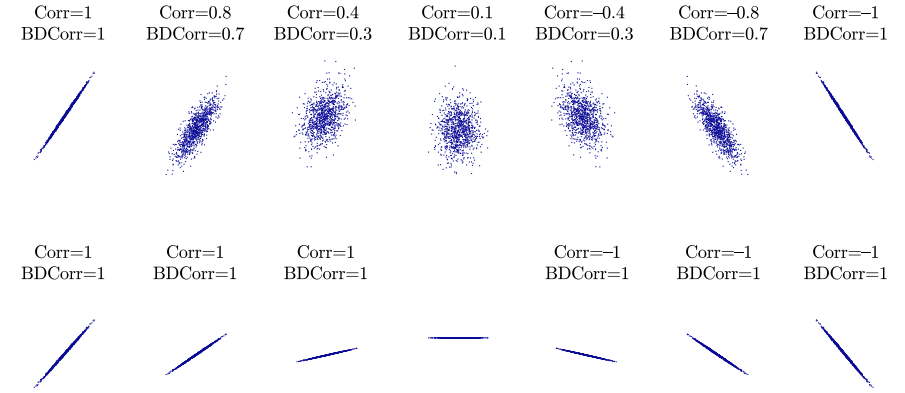
Here $\mathrm{cov}(X,Y)$ and $\mathrm{BDC}(X,Y)$ respectively denote the covariance and Brownian distance covariance. Naturally, $\mathrm{cov}(X,X)$ and $\mathrm{BDC}(X,X)$ denote variance and Brownian distance variance of X, respectively.

Fig. S-2 shows the scatter plots of the simulated examples together with the values of correlation and Brownian distance correlation. From Fig. S-2a, we can see that for all non-linear relations $\mathrm{Corr}(X,Y) = 0$, indicating that classical correlation fails to model such complex relations; on the contrary, Brownian distance correlation can characterize

[5] https://cran.r-project.org/web/packages/HHG/index.html



(a) Non-linear relations.



(b) Linear relations.

Figure S-2. Comparison of linear and non-linear relation modeling between classical Correlation (Corr) and Brownian Distance Correlation (BDCorr). This illustration was inspired by D. Boigelot.

the non-linear dependencies. As shown in Fig. S-2b, compared to correlation, Brownian distance correlation has similar capability to model linear relations, except that it cannot distinguish the orientation as it is always non-negative; besides, both of them cannot reflect the slope of linear relations.

References

[S-1] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagn'e. Associative alignment for few-shot image classification. In *ECCV*, 2020.

[S-2] Brandon Amos and J. Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *ICML*, 03 2017.

[S-3] Ankush Gupta Carl Doersch and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *NIPS*, 2020.

[S-4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019.

[S-5] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *CVPR*, 2021.

[S-6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[S-7] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Dropblock: A regularization method for convolutional networks. In *NIPS*, 2018.

[S-8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[S-9] Ruth Heller, Yair Heller, and Malka Gorfine. A consistent multivariate test of association based on ranks of distances. *Biometrika*, 100(2):503–510, 2013.

[S-10] Jonathan Krause, Michael Stark, Deng Jia, and Fei Fei Li. 3D Object representations for fine-grained categorization. In *ICCV Workshop*, 2013.

[S-11] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

[S-12] Wenbin Li, Lei Wang, Jing Huo, Yinghuan Shi, Yang Gao, and Jiebo Luo. Asymmetric distribution measure for few-shot learning. *IJCAI*, 2020.

[S-13] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*, 2019.

[S-14] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *ECCV*, 2020.

[S-15] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[S-16] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *NIPS*, 2018.

[S-17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[S-18] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[S-19] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

[S-20] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[S-21] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *ECCV*, 2020.

[S-22] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[S-23] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[S-24] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *CVPR*, 2019.

[S-25] Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In *CVPR*, 2021.

[S-26] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, 2020.

[S-27] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *CVPR*, 2020.

[S-28] Ziqi Zhou, Xi Qiu, Jiangtao Xie, Jianan Wu, and Chi Zhang. Binocular mutual learning for improving few-shot classification. In *ICCV*, 2021.