Domain-Agnostic Few-Shot Classification by Learning Disparate Modulators

Yongseok Choi Junyoung Park Subin Yi Dong-Yeon Cho

T-Brain, AI Center, SK Telecom
{yschoi, jypark, yisubin, dycho24}@sktbrain.com

Abstract

Although few-shot learning research has advanced rapidly with the help of meta-learning, its practical usefulness is still limited because most of them assumed that all meta-training and meta-testing examples came from a single domain. We propose a simple but effective way for few-shot classification in which a task distribution spans multiple domains including ones never seen during meta-training. The key idea is to build a pool of models to cover this wide task distribution and learn to select the best one for a particular task through cross-domain meta-learning. All models in the pool share a base network while each model has a separate modulator to refine the base network in its own way. This framework allows the pool to have representational diversity without losing beneficial domain-invariant features. We verify the effectiveness of the proposed algorithm through experiments on various datasets across diverse domains.

1 Introduction

Few-shot learning in the perspective of meta-learning aims to train models which can quickly solve novel tasks or adapt to new environments with limited number of examples. In case of few-shot classification, models are usually evaluated on a held-out dataset which does not have any common class with the training dataset. In the real world, however, we often face harder problems in which novel tasks arise arbitrarily from many different domains even including previously unseen ones.

In this study, we propose a more practical few-shot classification algorithm to generalize across domains beyond the common within-domain setup. Our approach to cover a complex task distribution is to construct a pool of multiple models and learn to select the best one given a novel task. This recasts task-specific adaption across domains as a simple selection problem, which makes learning more effective. Furthermore, by enforcing all models to share some of parameters, each model could keep important domain-invariant features while the pool maintains diversity as a whole.

Experimental results show that our algorithm could perform few-shot classification tasks from multiple meta-trained domains without having any domain identifiers at meta-test time. They also reveal that the proposed algorithm could be applied successfully to novel tasks from unseen domains.

2 Methods

2.1 Problem statement

Our objective is to build a domain-agnostic meta-learner beyond the common meta-learning assumptions, i.e. meta-training within a single domain and meta-testing within the same domain, presuming that one domain corresponds to one dataset. After meta-training over diverse datasets, we apply the trained meta-learner to two types of more general few-shot classification tasks which require the inter-domain generalization capability of the meta-learner.

One is few-shot classification tasks sampled from held-out classes of the datasets used during the meta-training time without knowing from which dataset each task is sampled. This could be used to evaluate whether the meta-learner is capable to adapt to a complex task distribution across multiple

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada.

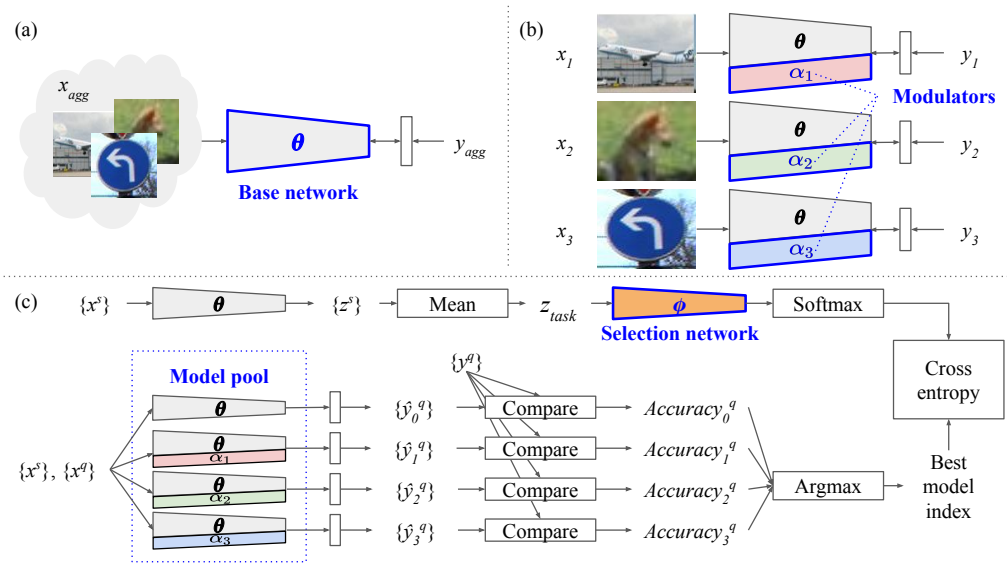


Figure 1: Training (a) a base network (θ), (b) modulators ($\{\alpha_i\}$), (c) a selection network (ϕ). (x_m^n: an example, y_m^n: a label, $\hat{y}_m^n$: a prediction, z_m^n: an embedding vector of a support (n=s) or query (n=q) set in a domain m.)

domains. We also tackle few-shot classification tasks sampled from datasets never seen during the meta-training, which requires generalization to out-of-distribution tasks in domain-level.

2.2 Building a pool of embedding models from diverse source domains

Basically, we learn an embedding model in a similar way to Prototypical Network (ProtoNet) [27] over various datatsets, each of which is considered to define its own domain. Few-shot classification can be performed across various domains by comparing distances to class prototypes from a query embedding vector on a metric space defined by the learned embedding model. However, since it is hard to map a complex task distribution spanning diverse domains into a single space, we build a pool of embedding models each of which has its own metric space. For inference, one model can be chosen as the best fit for a particular task, or we can infer by averaging outputs from all models.

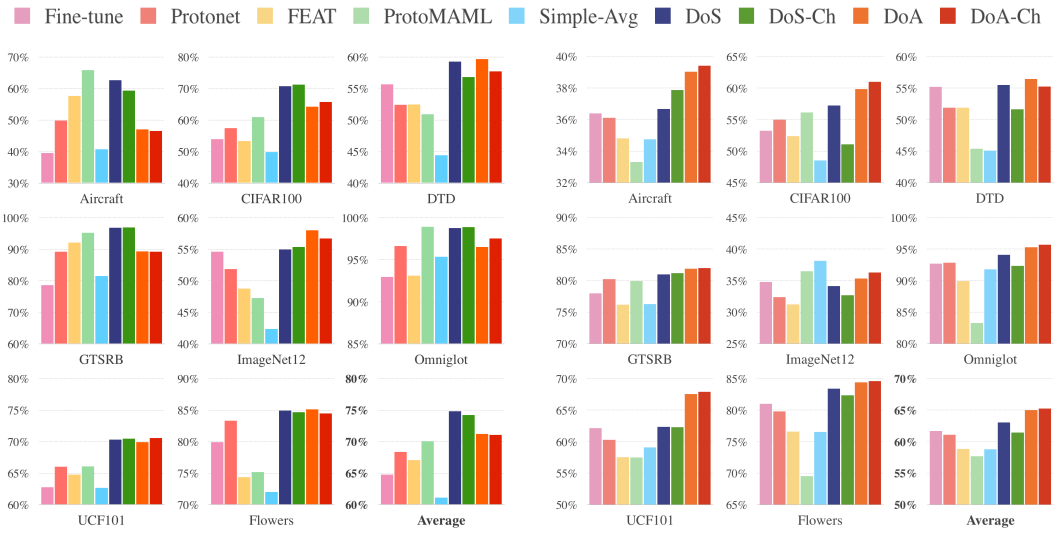
Rather than training an individual model separately, we first train a base network shared by all models over an aggregation of all source datasets following the supervised learning procedure (Figure 1(a)). Then, we build one model per source domain by adding a per-model modulator on top of the frozen base network and training each modulator on one dataset by performing metric-based meta-learning in the same way as the ProtoNet (Figure 1(b)). The modulators are inserted into the base network in a per-layer basis (Figure 3 in the supplementary material). The rationale behind this is to let the pool have diversity, which is desirable to represent a complex task distribution, with minimal parameter overhead by the modulators and capture good domain-invariant features by the base network.

The overall training procedure is summarized in Algorithm 1 in the supplementary material.

2.3 Learning to select the best model from the pool for a target task

After the construction of the pool, we learn a meta-model which predicts the best single model from the pool for a given task as described in Algorithm 1 in the supplementary material and Figure 1(c). By training this model over a number of episodes sampled from all source datasets, we expect this ability to be generalized to novel tasks from various domains including unseen domains.

This meta-model parameterized by ϕ, called a model selection network, is trained in order to map a task representation z_{task} to an index of the best model in the model pool over the sampled episodes. The task representation is obtained by passing all examples in the support set of the task through the base network and averaging all resulting embedding vectors. The index of the best model, which is the ground truth label for training the selection network, is generated by measuring the classification accuracy of all models in the pool with the query set and picking one which has the highest accuracy. This simple classification is much easier to learn than manipulating high-dimensional parameters directly, which makes our approach for task-specific adaptation more effective.



(a) On seen domains.

(b) On unseen domains.

Figure 2: 5-way 5-shot test accuracy on various datasets.

3 Experiments

3.1 Setup

3.1.1 Datasets

We use eight image classification datasets, denoted as Aircraft, CIFAR100, DTD, GTSRB, ImageNet12, Omniglot, SVHN, UCF101 and Flowers, from the Visual Decathlon dataset [22] for evaluation. All datasets are resplit for the purpose of few-shot classification. More detailed information about the datatsets can be found in Section A in the supplementary material.

3.1.2 Algorithms

We denote inference methods using the model picked by the selection network as *DoS* (Domain-generalized method by Selection) and *DoS-Ch*, which modulates the base network with convolution 1×1 and chnannel-wise modulation (i.e. scaling and biasing) respectively. We also explore inference methods, *DoA* (Domain-generalized method by Averaging) and *DoA-Ch*, to generate an output by averaging class prediction probabilities of all constituent models modulated by the above-mentioned two types of modulators. Two modulation schemes are explained further with their parameter overhead in Section B in the supplementary material.

Our algorithms are compared with *Fine-tune*, *Simple-Avg*, *ProtoNet* [27], *FEAT* [33] and *ProtoMAML* [30]. *Fine-tune* is a baseline method to add a linear classifier on top of the pre-trained base network and fine-tune it with the support set examples for 100 epochs during meta-testing. In *Simple-Avg*, we train an embedding model independently on each source domain without sharing any parameters with others in the same way as *ProtoNet* and perform inference by averaging class prediction probabilities of all these models. *FEAT* and *ProtoMAML* are the state-of-the-art algorithms focusing on single domain and cross-domain setups respectively. *TADAM* [17] was also tested but excluded from the results because its training did not converge in our multi-domain experiments.

All results are produced by our own implementations including 'further adaptation' introduced in [2]. Other details about the training are explained in Section C in the supplementary material.

3.2 Results

We compare our methods with other algorithms in 5-way 5-shot setting on both seen and unseen domains. More results including 5-way 1-shot cases are in Section D in the supplementary material.

3.2.1 Few-shot classification on seen domains

Figure 2(a) shows the test accuracy values on various seen domains. We evaluate the above-mentioned algorithms in a multi-domain test setup constructed using all available datasets. Specifically, we meta-train a model for each algorithm on all available eight datasets. Then, we meta-test the trained model for various tasks sampled from these eight datasets without knowing which dataset the task

comes from Our methods, *DoS*, *DoS-Ch*, *DoA* and *DoA-Ch*, outperform other few-shot classification methods in most cases. Although *FEAT* and *ProtoMAML* adopt their own task-specific adaptation schemes, they do not seem as effective as ours under this complex task distribution across various domains. *ProtoMAML* shows comparable results in some cases, but much inferior results in other cases. *ProtoNet* seems relatively stable, but does not produce better results than ours either.

Our selection-based methods, *DoS* and *DoS-Ch*, perform better than our averaging methods on seen domains. This implies that the learned selection network is working properly, which is highly likely to select the model with the modulator trained on the same domain as the given task even without any information about the domain at testing time. Another implication is that the best single model might be better than the averaging approach if the model from the same domain exists. It is also worth noting that *DoS-Ch* is quite competitive despite less number of additional parameters than *DoS*.

3.2.2 Few-shot classification on unseen domains

We also report the test results on unseen domains in Figure 2(b). Given a target dataset for test, we train all models using on seven other datasets. Our methods still outperform other algorithms in this more challenging setting, which reveals that our methods can be generalized to novel domains as well. Differently from the seen domain cases, our averaging methods, *DoA* and *DoA-Ch*, perform better than all other algorithms including our selection-based methods. It seems to make sense since the averaging could induce naturally synergy between beneficial models due to their similar output patterns even if we do not know which models are beneficial to a given task at testing time. However, our averaging methods outperform significantly *Simple-Avg*, which implies that our unique architecture to encourage keeping domain-invariant features, i.e. embedding models with parameter sharing, is another key factor to the high performance of the averaging methods.

4 Related works

Meta-learning is one of the most popular techniques to solve the few-shot learning problems, which includes learning a task-invariant metric space [27, 31], learning to optimize [1, 21] or learning weight initialization [5, 15]. Follow-up studies showed that metric-based meta-learning could be improved further by learning task-specific adaptation on the learned space [6, 17, 20, 26, 33].

Recent few-shot learning studies have tried to tackle challenging problems under more realistic assumptions. Some studies dealt with few-shot learning under domain shift between training and testing [8][32]. More realistic benchmark was proposed for few-shot learning to overcome limitations of the current popular benchmarks including the lack of domain divergence [30]. Similar to our approach, a few suggestions combined multiple models to benefit from their diversity [4, 11, 18]. However, they considered an ensemble with independent models unlike ours with the shared network.

Our network architecture is inspired by the parameter sharing strategies for multi-task learning [24] and multi-domain learning with domain-specific adaptation [23] because they have been known to lead to efficient parameterization and positive knowledge transfer between heterogeneous entities.

5 Conclusion and future works

We propose a new few-shot classification method generalizing to various domains including unseen domains. The core idea is to build the pool of embedding models, each of which is diversified by its own modulators while sharing most of parameters with others, and learn to select the best model for a target task through cross-domain meta-learning. Experiments show that the proposed method outperforms an ensemble of models trained separately on every different dataset and the state-of-the-art few-shot classification algorithms.

We believe that there is still a large room for improvement in this challenging task. It would be one promising extension to find the optimal way to build the pool without being confined by the policy of one model per dataset so that it can work even with a single source domain with large diversity. Soft selection or weighted averaging can be also thought as one of future research directions because a single model or uniform averaging is less likely to be optimal. We can also consider a more scalable extension to allow continual expansion of the pool only by training a modulator for an incoming source domain without re-training all existing models in the pool.

References

[1] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3981–3989, 2016.

[2] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[3] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[4] N. Dvornik, C. Schmid, and J. Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. *CoRR*, abs/1903.11341, 2019.

[5] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1126–1135, 2017.

[6] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4367–4375, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[8] B. Kang and J. Feng. Transferable meta learning across domains. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 177–187, 2018.

[9] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[10] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[11] Y. Liu, Q. Sun, A. Liu, Y. Su, B. Schiele, and T. Chua. LCC: learning to customize and combine neural networks for few-shot learning. *CoRR*, abs/1904.08479, 2019.

[12] S. Maji, E. Rahtu, J. Kannala, M. B. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013.

[13] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, Nov 2006.

[14] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

[15] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

[16] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, ICVGIP '08, pages 722–729, Washington, DC, USA, 2008. IEEE Computer Society.

[17] B. N. Oreshkin, P. R. López, and A. Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 719–729, 2018.

[18] M. Park, J. Kim, S. Kim, Y. Liu, and S. Choi. Mxml: Mixture of meta-learners for few-shot classification. *CoRR*, abs/1904.05658, 2019.

[19] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951, 2018.

[20] S. Qiao, C. Liu, W. Shen, and A. L. Yuille. Few-shot image recognition by predicting parameters from activations. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7229–7238, 2018.

[21] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[22] S. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 506–516, 2017.

[23] S. Rebuffi, H. Bilen, and A. Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8119–8127, 2018.

[24] S. Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[26] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019.

[27] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4080–4090, 2017.

[28] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[29] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323 – 332, 2012. Selected Papers from IJCNN 2011.

[30] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P. Manzagol, and H. Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *CoRR*, abs/1903.03096, 2019.

[31] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638, 2016.

[32] Y. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pages 616–634, 2016.

[33] H. Ye, H. Hu, D. Zhan, and F. Sha. Learning embedding adaptation for few-shot learning. *CoRR*, abs/1812.03664, 2018.

Supplementary Materials

A Datasets

In our experiments, we use the Visual Decathlon dataset [22] which consists of ten datasets for image classification listed below.

- FGVC-Aircraft Benchmark (Aircraft, A) [12]
- CIFAR100 (CIFAR100, C) [9]
- Daimler Mono Pedestrian Classification Benchmark (DMPCB)[13]
- Describable Texture Dataset (DTD, D) [3]
- German Traffic Sign Recognition Benchmark (GTSRB, G) [29]
- ImageNet ILSVRC12 (ImageNet12, I) [25]
- Omniglot (Omniglot, O) [10]
- Street View House Numbers (SVHN) [14]
- UCF101 (UCF101, U) [28]
- Flowers102 (Flowers, F) [16]

The categories and the number of images of each dataset are significantly different as well as the image size. All images have been resized isotropically to 72×72 pixels so that each image from various domains has the same size.

Daimler Mono Pedestrian Classification has only 2 classes, pedestrian and non-pedestrian. We excluded it from our experiments as we are considering 5-way classification tasks. SVHN was also excluded since SVHN has only 10 digit classes from 0 to 9, which were too few to split for meta-training and meta-testing. To use the remaining eight datasets for multi-domain few-shot classification, we divide the examples into roughly 70% training, 15% validation, and 15% testing classes. For ILSVRC12, we follow the split of Triantafillou *et al.* [30] to adopt class hierarchy, and we use random class splits for other datasets. The number of classes at each split is shown in Table 1. We only use train and validation sets of the Visual Decathlon because the labels of the test set is not publicly available.

Table 1: The details of datasets used in our experiments.

			SPLITS		
DATASET	# DATA	# CLASSES	TRAIN	VAL	TEST
AIRCRAFT	6667	100	70	15	15
CIFAR100	50000	100	70	15	15
DTD	3760	47	32	7	8
GTSRB	39209	43	30	6	7
IMAGENET12	1281167	1000	712	158	130
OMNIGLOT	25968	1623	1136	243	244
UCF101	9537	101	70	15	16
FLOWERS	2040	102	70	16	16

	Convolution 1x1	Channel-wise transform
Modulators $\{\alpha_i\}_{i=1}^{8}$	9,795,584	61,440
Base network θ	11,176,512	11,176,512
Selection network ϕ	66,696	66,696
Sum	21,038,792	11,304,648

Table 2: The comparison of the number of parameters for convolution 1×1 and channel-wise transform modulators. This is the case when the number of source domains is 8.

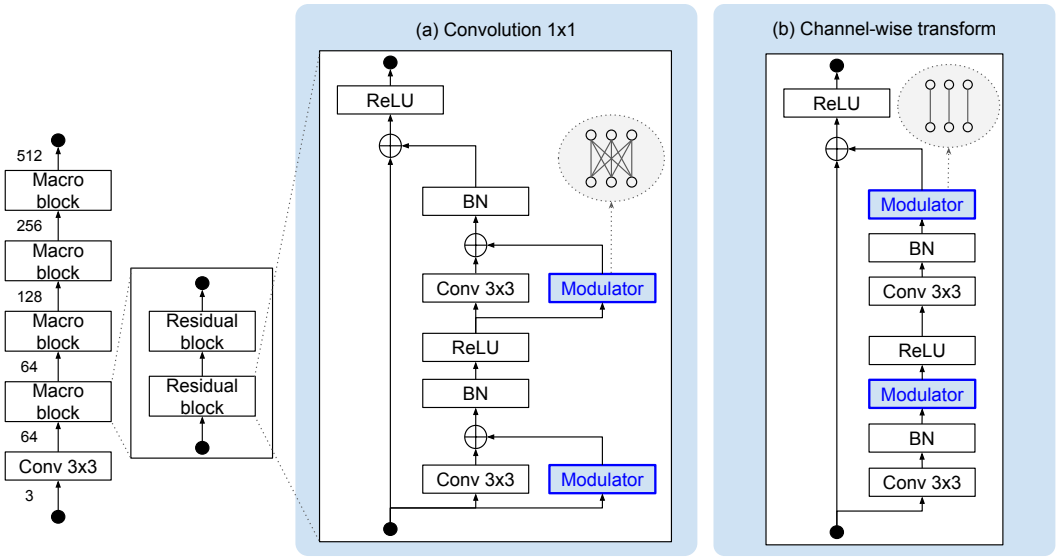


Figure 3: Architecture of the ResNet-18-based embedding network f_e.

B Architectures

Figure 3 shows the architecture of the embedding network $f_E(\cdot; \theta, \alpha_i)$, which processes an input image and produces a 512-dimensional embedding vector. The embedding network is based on the ResNet-18 architecture [7], which consists of one convolutional layer with 64 7×7 filters followed by 4 macro blocks, each having 64-128-256-512 3×3 filters. Figure 3(a) and Figure 3(b) depict how the base network is modulated by the convolution 1×1 modulator and the channel-wise transform modulator, respectively. These modulators are placed within each residual block of the macro blocks, same as the previous works in [23] and [19].

The number of parameters for two modulators are shown in Table 2. The values on the first row are the number of modulator parameters that are additionally applied to the embedding network. Note that the channel-wise transform modulator has much fewer parameters than the convolution 1×1 modulator. In particular, the channel-wise transform modulator has negligible number of parameters compared to that of the base network, i.e., ResNet-18. For each embedding model, the convolution 1×1 modulator has about 10% of the number of parameters that the base network has whereas the channel-wise transform modulator requires only less than 1%.

The selection network $f_S(\cdot; \phi)$ is a two-layered MLP (multi-layer perceptron) network, which receives an embedding vector produced by the embedding network as an input and performs the best model index prediction. Two layers are a linear layer of 512×128 and a linear layer of $128 \times (M + 1)$, where M is the number of source domains.

Algorithm 1 The overall learning procedure

Input: Training data from $D_S = \{D_{S_i}\}_{i=1}^{M}$, embedding networks $f_e(\cdot)$, a selection network $f_s(\cdot)$ ($D_{S_1}, \cdots, D_{S_M}$: source domains where M is the number of source domains, D_T: a target domain)
Output: Learned parameters θ, $\{\alpha_i\}_{i=1}^{M}$, ϕ. (α_0 means no modulation)

Step 1: Build a base network

1: Build one large classification dataset (x_{agg}, y_{agg}) by aggregating all classes from D_S.
2: Learn θ by optimizing $f_e(x; \theta, \alpha_0)$ for the aggregated dataset (α_0: no modulation).

Step 2: Add modulators through intra-domain episodic training

1: **while** not converged **do**
2: Sample one domain D_{S_i} from D_S, then sample one episode (S, Q) from D_{S_i}.
3: Learn α_i by optimizing $f_e(x; \theta, \alpha_i)$ for (S, Q) while keeping θ fixed.
4: **end while**

Step 3: Build a selection network through cross-domain episodic training

1: **while** not converged **do**
2: Sample one domain D_{S_i} from D_S, then sample one episode (S, Q) from D_{S_i}.
3: Get a task representation z_{task} by averaging embedding vectors of S from the base network.
4: Measure accuracies of $M + 1$ models $\{f_e(x; \theta, \alpha_i)\}_{i=0}^{M}$ for (S, Q). (α_0: no modulation.)
5: Set the best model index y_{sel} to the index of the model with the highest accuracy.
6: Learn ϕ by training $f_s(z_{task}; \phi)$ so as to predict y_{sel} for (S, Q).
7: **end while**

C Training details

Algorithm 1 describes the overall training procedure to construct the model pool and the selection network. Although we trained three components in a sequential manner, joint training of these components seems to make sense also.

For the fair comparison with *Fine-tune* method, we also apply algorithm-specific refinement at meta-testing time, inspired by 'further adaptation' in [2], to all other algorithms including ours. A linear classifier is placed on top of the embedding network of the *ProtoNet*, the self-attention module of the *FEAT* or the modulated embedding network of our models. During meta-testing, other parameters are fixed and the classifier is fine-tuned using the support examples for 100 iterations per episode. In case of *FEAT*, the classifier is trained for 100 epochs per query example not per episode because *FEAT* modulates a representation space for each query. We also adjust the number of adaptation of the *ProtoMAML* to 100 for the better task-adaptation as done in [2].

The hyperparameters including the learning rate are selected by grid search based on the validation accuracy. For *FEAT* and *ProtoMAML*, Adam optimizer is used for training and the learning rate and weight decay are set to be 0.0001. Other models are also trained using Adam optimizer with the learning rate 0.001 but without any regularization including the weight decay.

All reported test accuracy values are averaged over 600 test episodes with 10 queries per class.

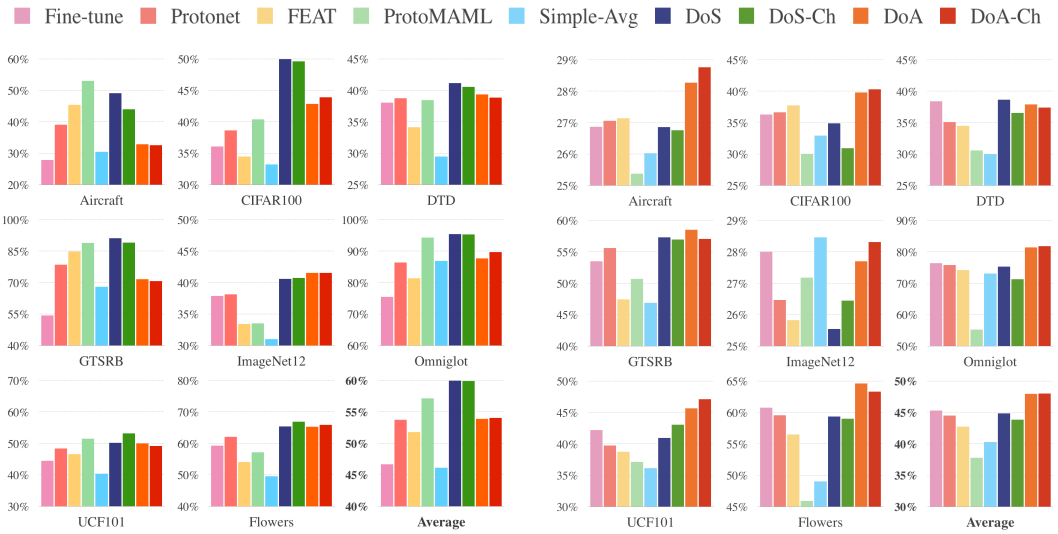
D Additional experimental results

D.1 5-way 1-shot classification

Figure 4 shows test accuracy values of 5-way 1-shot classification tasks.

D.2 Results without further adaptation

We present the experimental results when we do not apply the further adaptation scheme introduced in [2]. Specifically, *ProtoNet*, *FEAT*, and our models are tested without additional linear classifiers $f_c(\cdot; \psi)$. The number of parameter update steps in *ProtoMAML* is reduced to 3, which is not enough to have the models fine-tuned. Tables 3 and 4 show the results tested on seen and unseen domains, respectively. We can see that accuracy drops in almost all cases compared to corresponding cases



(a) On seen domains.

(b) On unseen domains.

Figure 4: 5-way 1-shot test accuracy on various datasets.

with further adaptation whose results are in Section 3 in the main text, but our models generally do better than other methods in any experimental settings.

Table 3: 5-way 5-shot classification accuracy on seen domains without further adaptation.

	Methods								
Target	F-T	Proto	FEAT	PMAML	S-Avg	DoS	DoS-Ch	DoA	DoA-Ch
A	39.53%	49.61%	58.53%	**66.39%**	37.04%	62.86%	59.39%	42.82%	42.91%
C	53.94%	57.48%	53.85%	59.37%	48.60%	69.94%	**71.57%**	62.96%	64.37%
D	55.68%	53.51%	52.91%	50.81%	42.98%	**58.65%**	56.50%	58.43%	56.54%
G	78.67%	86.21%	92.74%	95.22%	74.39%	96.49%	**96.63%**	82.93%	83.31%
I	54.64%	52.23%	49.91%	46.80%	41.69%	55.73%	55.49%	**57.83%**	57.42%
O	92.96%	96.20%	93.18%	98.98%	94.44%	98.86%	**98.80%**	95.54%	96.70%
U	62.78%	66.31%	64.76%	66.17%	59.15%	**69.78%**	69.45%	69.68%	68.93%
F	79.92%	83.46%	74.80%	75.84%	69.98%	83.60%	**84.12%**	83.63%	83.28%
Average	64.76%	68.13%	67.58%	69.95%	58.53%	**74.49%**	73.99%	69.23%	69.18%

A: Aircraft, C: CIFAR100, G: GTSRB, I: ImageNet12, O:Omniglot, U:UCF101, F:VGG-Flowers
F-T: Fine-tune, Proto: ProtoNet, PMAML: ProtoMAML, S-Avg: Simple-Avg

Table 4: 5-way 5-shot classification accuracy on unseen domains without further adaptation.

	Methods								
Target	F-T	Proto	FEAT	PMAML	S-Avg	DoS	DoS-Ch	DoA	DoA-Ch
A	36.40%	35.42%	33.30%	32.87%	33.48%	34.73%	35.63%	36.93%	**37.28%**
C	53.26%	55.17%	52.60%	58.01%	49.23%	55.79%	50.09%	58.30%	**59.22%**
D	**55.18%**	51.03%	50.37%	45.46%	45.99%	54.11%	48.81%	54.88%	54.37%
G	78.01%	76.33%	75.79%	**78.81%**	69.49%	77.03%	77.18%	77.36%	77.31%
I	34.81%	32.90%	33.50%	36.46%	**37.79%**	34.34%	32.73%	34.36%	35.27%
O	92.69%	92.64%	91.99%	83.80%	91.45%	93.68%	91.99%	**94.95%**	94.84%
U	62.16%	59.74%	58.54%	58.40%	59.06%	62.04%	62.21%	65.93%	**67.25%**
F	81.00%	79.42%	80.82%	69.47%	75.39%	81.85%	82.69%	82.71%	**83.89%**
Average	61.69%	60.33%	59.62%	57.91%	57.74%	61.70%	60.17%	63.19%	**63.68%**

A: Aircraft, C: CIFAR100, G: GTSRB, I: ImageNet12, O:Omniglot, U:UCF101, F:VGG-Flowers
F-T: Fine-tune, Proto: ProtoNet, PMAML: ProtoMAML, S-Avg: Simple-Avg

D.3 Few-shot classification on varying number of source domains

We conduct experiments with varying number of source datasets. Following the common real-world situation, we add from the largest dataset to the smallest one to our sources for meta-training. Tables 5 and 6 show the experimental results with 2, 4, and 6 source datasets on seen and unseen domains respectively.

Our selection and averaging methods outperform others consistently on seen and unseen domains similarly to the previous results. Apart from comparing between the algorithms, it is commonly observed over all algorithms that the added source often harms the performance. For example, the CIFAR100 tasks tend to work poorer as the number of source datasets increases in the seen domain case. This means that we should pay more attention to avoiding negative transfer between heterogeneous domains.

Table 5: Few-shot classification accuracy of varying number of sources on seen target domains.

		Methods							
S	T	Fine-tune	ProtoNet	FEAT	ProtoMAML	DoS	DoS-Ch	DoA	DoA-Ch
C,I	C	54.72%	65.47%	65.39%	69.51%	**73.04%**	72.63%	71.19%	69.58%
	I	57.50%	55.88%	52.53%	56.37%	57.71%	57.84%	**59.39%**	58.05%
Average		56.11%	60.68%	58.96%	62.94%	**65.38%**	65.24%	65.29%	63.82%
C,G,I,O	C	54.24%	58.31%	63.82%	68.93%	**71.11%**	70.15%	68.14%	66.67%
	G	81.21%	90.58%	95.40%	96.61%	**96.96%**	96.36%	92.94%	91.55%
	I	55.70%	52.04%	50.58%	53.72%	55.78%	56.10%	**58.04%**	57.74%
	O	94.82%	97.14%	95.10%	**99.52%**	98.94%	98.73%	97.86%	97.64%
Average		71.49%	74.52%	76.23%	79.70%	**80.70%**	80.33%	79.25%	78.40%
A,C,G, I,O,U	A	40.68%	51.92%	62.77%	**70.78%**	61.94%	57.49%	47.77%	46.86%
	C	54.03%	58.68%	56.56%	60.47%	69.54%	**70.32%**	66.48%	65.41%
	G	76.35%	89.95%	95.64%	96.50%	**97.71%**	96.78%	89.65%	89.62%
	I	53.37%	52.80%	48.62%	51.74%	54.59%	56.03%	57.34%	**58.03%**
	O	94.06%	96.94%	95.30%	**99.11%**	98.89%	98.76%	97.20%	97.92%
	U	62.01%	65.49%	62.20%	67.11%	70.44%	70.23%	69.16%	**71.42%**
Average		63.42%	69.30%	70.18%	74.29%	**75.52%**	74.94%	71.27%	71.54%

S: source datasets, T: target dataset
A: Aircraft, C: CIFAR100, G: GTSRB, I: ImageNet12, O: Omniglot, U: UCF101, F: Flowers

Table 6: Few-shot classification accuracy of varying number of sources on unseen target domains.

		Methods							
S	T	Fine-tune	ProtoNet	FEAT	ProtoMAML	DoS	DoS-Ch	DoA	DoA-ch
C,I	A	38.65%	39.38%	36.71%	35.37%	38.43%	39.27%	39.14%	**40.57%**
	D	53.89%	55.27%	52.39%	51.04%	54.80%	54.90%	**57.15%**	56.99%
	G	70.01%	81.37%	77.67%	81.25%	**83.83%**	82.65%	80.13%	79.93%
	O	92.72%	92.62%	90.61%	91.37%	93.11%	93.31%	94.00%	**94.34%**
	U	63.80%	63.25%	60.04%	59.89%	63.67%	64.81%	66.28%	**67.15%**
	F	79.28%	81.77%	78.86%	80.50%	81.26%	80.84%	**82.96%**	82.85%
Average		66.39%	68.94%	66.05%	66.63%	69.18%	69.30%	69.94%	70.31%
C,G,I,O	A	38.12%	36.88%	35.11%	34.96%	37.90%	38.04%	39.66%	**39.99%**
	D	55.04%	49.96%	49.55%	49.79%	55.85%	56.53%	**56.73%**	55.67%
	U	63.25%	56.53%	60.93%	64.12%	64.32%	64.84%	67.58%	**67.69%**
	F	79.88%	76.79%	77.67%	81.40%	80.69%	79.54%	**83.25%**	82.20%
Average		59.07%	55.04%	55.82%	57.57%	59.69%	59.74%	**61.81%**	61.39%
A,C,G I,O,U	D	53.60%	51.52%	51.86%	50.94%	55.54%	55.69%	56.71%	**57.07%**
	F	80.80%	80.58%	79.70%	79.40%	81.62%	81.66%	84.30%	**84.50%**
Average		67.20%	66.05%	65.78%	65.17%	68.58%	68.68%	70.51%	**70.79%**

S: source datasets, T: target dataset
A: Aircraft, C: CIFAR100, G: GTSRB, I: ImageNet12, O: Omniglot, U: UCF101, F: Flowers

D.4 Comparative analysis about the averaging methods

As an effort for better understanding the averaging methods, we investigate how each model in the pool contributes to the final prediction. Figures 5(a) and 5(b) show how many correct predictions are

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Model 0	41	40	32	43	42	36	37	35	41	36	43	41	36	47	45	45	41	48	37	43	40	39	47	43	40	43	40	45	37	41	38	44	37	40	45	39	40	42	39	40
Model 1	21	21	15	14	31	24	25	23	21	15	24	17	15	21	16	18	26	31	16	30	19	19	24	24	18	22	18	29	24	22	21	23	16	18	18	23	20	18	15	17
Model 2	35	39	31	33	34	34	30	26	32	33	30	25	27	37	38	38	35	36	23	41	37	30	31	37	28	36	21	37	31	33	35	30	22	34	36	29	28	37	28	30
Model 3	26	27	20	26	27	18	20	21	23	24	21	19	16	25	23	22	32	26	13	23	22	17	20	30	20	24	17	23	22	30	24	13	15	16	24	16	24	25	14	22
Model 4	22	32	17	29	30	16	20	24	21	17	28	22	23	19	32	28	26	24	17	28	23	19	29	31	26	23	24	36	28	31	26	23	21	27	28	28	21	27	17	21
Model 5	39	42	36	41	42	35	41	38	44	35	45	47	38	45	48	43	44	44	42	45	46	41	45	46	40	44	41	46	40	40	42	44	42	45	45	38	41	49	44	38
Model 6	21	13	18	8	18	20	12	14	12	12	17	14	16	12	19	13	18	18	19	19	19	15	20	21	18	16	14	17	19	16	13	15	10	11	20	16	16	12	16	16
Model 7	21	23	18	16	22	17	18	18	20	23	21	16	15	22	25	17	29	20	22	31	22	21	20	31	17	21	16	24	23	28	22	19	13	20	23	18	19	11	15	27

(a) Simple averaging (Simple-Avg)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Model 0	42	36	41	40	38	42	37	41	48	42	42	46	38	41	45	40	36	42	41	37	42	41	43	34	46	44	43	45	40	35	38	40	41	40	36	43	40	41	33	47
Model 1	42	40	41	43	36	34	34	41	43	39	37	37	31	44	44	31	36	41	45	36	42	45	41	34	44	43	41	39	36	30	37	38	43	43	39	40	41	41	32	45
Model 2	40	38	40	45	30	35	37	43	43	39	40	40	35	41	41	41	33	35	43	35	41	38	43	35	39	38	39	44	33	38	36	38	43	40	32	37	38	38	34	42
Model 3	42	40	41	47	38	40	37	41	47	41	38	39	35	45	45	35	36	42	41	36	37	44	39	33	43	42	47	47	33	38	38	40	43	42	37	39	43	42	34	48
Model 4	42	36	39	44	37	40	35	42	41	41	40	36	34	44	42	38	32	36	42	36	39	40	40	36	44	41	44	40	37	36	40	37	42	42	30	42	40	42	34	42
Model 5	42	37	41	45	41	44	36	41	46	42	36	42	35	42	45	39	36	41	43	37	45	41	43	34	44	44	45	45	43	37	37	41	41	42	42	46	43	44	34	47
Model 6	44	36	40	46	40	42	39	44	46	40	39	43	39	43	41	39	37	43	42	36	39	43	41	35	42	44	45	44	41	33	37	37	40	41	36	45	40	39	34	46
Model 7	45	41	41	46	36	40	34	45	45	40	39	40	36	41	45	39	37	43	41	38	42	43	42	38	46	43	44	45	37	42	38	40	44	44	41	45	40	41	31	48

(b) Proposed averaging (DoA)

Figure 5: Contributions of individual models in model averaging methods.

made by each model with the *Simple-Avg* and our *DoA* methods respectively given 50 queries per episode for 40 episodes.

The measured numbers show that the individual models of our *DoA* perform better than those in the *Simple-Avg*, which explains the higher performance of the proposed method partly. Additionally, we can observe that major contributors (i.e., the models with higher accuracy) tend to change every episode in our *DoA* whereas only two models seem to play dominant roles regardless of the given episode. This implies that our method for constructing the model pool provides the averaging model with more beneficial diversity.

E The loss function for the selection network

Equation (1) shows the loss ($loss_{sel}$) used to train the selection network $f_s(\cdot;\phi)$. Here, acc_i is the classification accuracy of the model with the modulator parameterized by α_i in the pool. The accuracy is measured for query examples in a given episode by making a prediction in the same way with the ProtoNet [27], where the class whose prototype is the closest to the embedding vector of a given query example is picked as the final prediction.

$$
\begin{aligned}
z_{task} &= \frac{1}{NK}\sum_{i=1}^{NK} f_e(x_i^s;\theta,\alpha_0) \\
\hat{y}_{sel} &= \text{softmax}(f_s(z_{task};\phi)) \\
y_{sel} &= \text{argmax}(\{acc_i(\{x^s,y^s\}_{i=1}^{NK},\{x^q,y^q\}_{j=1}^{T})\}_{i=0}^{M}) \\
loss_{sel} &= \text{cross\_entropy}(\hat{y}_{sel},y_{sel})
\end{aligned}
\tag{1}
$$