A Meta Understanding of Meta-Learning

Anonymous Authors[1]

Abstract

Recent years have witnessed an abundance of new publications and approaches on meta-learning. This community-wide enthusiasm has sparked great insights, but has also created a plethora of seemingly different frameworks, which can be hard to compare and evaluate. In this paper we aim to provide a single principled, unifying framework that draws a close connection between meta-learning and traditional supervised learning. By treating pairs of task specific data sets and trained models as (feature, label) samples we can reduce most meta-learning algorithms to instances of supervised learning. This view not only unifies meta-learning into an intuitive and practical framework, but it also allows us to transfer insights from supervised learning directly to improve meta-learning. For example, we obtain a better understanding of generalization properties and we can readily transfer well-understood techniques, such as data augmentation, model ensembling, pre-training, joint training, and even nearest neighbor based methods. We provide an intuitive analogy of these methods in the context of meta-learning and show that they give rise to significant improvements in model performance.

1. Introduction

Meta-learning, or learning to learn, is the sub-field of machine learning occupied with the search for the best learning strategy as the number of tasks and learning experiences increases (Vilalta & Drissi, 2002) and has drawn significant attention recently (Finn et al., 2017a; Andrychowicz et al., 2016; Vinyals et al., 2016). Meta-learning has been developed in various areas to advanced algorithm design, including few-shot learning (Ravi & Larochelle, 2017; Snell et al., 2017), optimization (Li & Malik, 2017; Wichrowska et al., 2017), active learning (Bachman et al., 2017), transfer learning (Balaji et al., 2018; Ying et al., 2018), unsupervised learning (Metz et al., 2018), etc. Specifically, meta-learning has demonstrated the capability to generalize learned knowledge to novel tasks, which greatly reduces the need for training data and the time to optimize.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

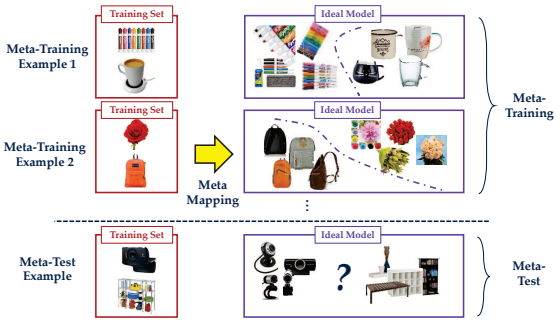


Figure 1. Meta-learning as supervised learning. By treating a pair of a training set (left) and the corresponding "ideal" model (right) as a meta labeled example, meta-learning aims to learn a shared meta mapping (from left to right) that will generalize from meta-training examples to novel meta-test examples. Take one-shot learning for instance. A training set contains one image per class, while an ideal model is a classifier trained with much more images.

Given its wide applicability and many diverse approaches, there is an increasing need for a principled, unifying meta-learning framework to facilitate future studies and development. Finn (2018); Metz et al. (2018) have made a notable step to provide a broad introduction and compare representative algorithms. In this paper we aim to push the direction forward by providing a unifying view of meta-learning that draws a close connection with supervised learning.

Generally, meta-learning can be viewed as learning a mapping that given a training set D_{tr} returns a model h. For instance, in unsupervised learning D_{tr} can be an unlabeled data set, and h a feature extractor. In one-shot learning, D_{tr} corresponds to a set of C labeled data samples and h corresponds to a C-way classifier. This observation leads to an interesting connection between meta-learning and supervised learning. Similar to supervised learning, we can train a meta-learning model on a set of meta labeled examples (task), $(D_{\text{tr}j}, h^*_j)$ pairs, and evaluate it on a test task D^{test}_{tr}. We refer to D_{tr} as a meta input (often a training set) and h^* as the corresponding ideal model (often obtained with ample training data). See Fig. 1 for an illustration.

This unifying view of meta-learning as supervised learning allows transferring ideas, experiences, and design principles of supervised learning to meta-learning, which will greatly facilitate its theoretical analysis, algorithm design, and applicability. In the following we list three notable examples. First, the unifying view offers an explanation why the learned meta mapping may or may not generalize to novel tasks: if the meta-training set contains insufficient meta examples or if the meta distribution where the meta-training set is sampled deviates from the one where novel tasks will be sampled, the meta mapping might generalize poorly. Indeed, Garg & Kalai (2018) proposed a meta-learning bound built upon supervised learning, but only for meta-learning for unsupervised learning. We expect our unifying view to inspire more theoretical analysis for meta-learning.

Second, our unifying view clearly identifies the essential components to define and design for meta-learning, providing a principled way to apply it to various areas.

Third, our unifying view broadens the scope of meta-learning algorithm design by extending that from supervised learning. We conduct extensive experiments on few-shot learning, a representative area where meta-learning is applied. We empirically show that well-known supervised learning techniques such as data augmentation, bagging (Breiman, 1996), joint training (Argyriou et al., 2007), pre-training (Yosinski et al., 2014), and non-parametric approaches (Zhang et al., 2006; Weinberger & Saul, 2009) can be applied at the task level to facilitate meta-learning.

2. A Unifying View of Meta-Learning

We first review the supervised learning framework and then define the meta-learning framework on top of it. We will use "meta (labeled) example" and "task" interchangeably.

2.1. Background: Supervised Learning

In supervised learning we collect a training set $D_{\text{tr}} = \{(x_n \in \mathcal{X}, y_n \in \mathcal{Y})\}_{n=1}^N$, composed of N i.i.d. samples from an unknown distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. We call x an input, y an output (or label), and a pair (x, y) a labeled example. We call $h : \mathcal{X} \mapsto \mathcal{Y}$ a model, which outputs a label for an input. For instance, in image classification, x is an image, y is a class name (e.g., "dog"), $\mathcal{D}$ is the distribution of real images, and h is an image classifier.

The goal of supervised learning is to search a model h given D_{tr}, so that h will work well on (x, y) sampled from $\mathcal{D}$. That is, h should achieve a small generalization loss $L_{\mathcal{D}}(h)$ according to a loss function $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$

$$L_{\mathcal{D}}(h) = E_{(x,y)\sim\mathcal{D}}[l(h(x), y)]. \tag{1}$$

To this end, we construct a hypothesis set $\mathcal{H} = \{h\}$ which contains multiple candidate models and design an algorithm $A_{\mathcal{H}}$ to search a model $\hat{h}$ from $\mathcal{H}$ by learning from D_{tr}, so that $\hat{h}$ can achieve a small $L_{\mathcal{D}}(\hat{h})$. We denote $\hat{h} = A_{\mathcal{H}}(D_{\text{tr}})$. One example of $\mathcal{H}$ is a multi-layer perceptron (MLP) with a fixed architecture but undecided weights.

One widely-used framework to design supervised learning algorithms is empirical risk minimization (ERM)

$$\begin{aligned}\hat{h} = A_{\mathcal{H}}(D_{\text{tr}}) &= \arg\min_{h\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N} l(h(x_n), y_n) \\ &= \arg\min_{h\in\mathcal{H}} L_S(h),\end{aligned} \tag{2}$$

where $L_S(h)$ is the training loss. A particular $A_{\mathcal{H}}$ is characterized by how to perform ERM.

In practice, we usually design a set of $(\mathcal{H}, A_{\mathcal{H}})$ pairs, denoted as G, and select the best pair using a held-out validation set $D_{\text{val}} = \{(x_m, y_m)\}_{m=1}^M$ sampled i.i.d. from $\mathcal{D}$

$$\begin{aligned}(\mathcal{H}, A_{\mathcal{H}})^* &= \arg\min_{(\mathcal{H},A_{\mathcal{H}})\in G} \frac{1}{M}\sum_{m=1}^{M} l(\hat{h}(x_m), y_m), \\ &= \arg\min_{(\mathcal{H},A_{\mathcal{H}})\in G} L_V(\hat{h}), \\ &\quad \text{s.t. } \hat{h} = A_{\mathcal{H}}(D_{\text{tr}}).\end{aligned} \tag{3}$$

This is called model selection and L_V is the validation error.

2.2. Meta-Learning as Supervised Learning

We provide a framework of meta-learning built upon that of supervised learning. To prevent confusion, we call the model in supervised learning a "base" model when needed.

Definition In meta-learning we collect a meta-training set $D_{\text{meta-tr}} = \{(D_{\text{tr}j} \in \mathcal{I}, h_j^* \in \mathcal{O})\}_{j=1}^{N_{\text{meta}}}$, composed of N_{meta} i.i.d. samples from an unknown meta distribution $\mathcal{D}_{\text{meta}}$ on $\mathcal{I} \times \mathcal{O}$. We call D_{tr} a training set (meta input), h^* an "ideally" base model (meta output), and a pair (D_{tr}, h^*) a meta labeled example (task). We call $g : \mathcal{I} \mapsto \mathcal{O}$ a meta model (meta mapping), which outputs a base model for a training set. For instance, in one-shot C-way learning for image classification, $D_{\text{tr}} = \{(x_n, y_n)\}_{n=1}^N$ contains N labeled images, one for each of the C classes (i.e., $N = C$). h^* is a strong classifier trained on sufficient labeled images.

The goal of meta-learning is to search a meta model g given $D_{\text{meta-tr}}$, so that g will work well on (D_{tr}, h^*) sampled from $\mathcal{D}_{\text{meta}}$. That is, g should achieve a small meta generalization loss $L_{\mathcal{D}_{\text{meta}}}(g)$ according to a meta loss $l_{\text{meta}} : \mathcal{O} \times \mathcal{O} \mapsto \mathbb{R}$

$$L_{\mathcal{D}_{\text{meta}}}(g) = E_{(D_{\text{tr}},h^*)\sim\mathcal{D}_{\text{meta}}}[l_{\text{meta}}(g(D_{\text{tr}}), h^*)]. \tag{4}$$

In one-shot learning it is to search a model predictor g that will output a strong classifier given a small training set.

Algorithm To this end, we can follow supervised learning to construct a meta hypothesis set $\mathcal{G} = \{g\}$ which contains

multiple candidate meta models and design an algorithm $B_{\mathcal{G}}$ to search a meta model $\hat{g}$ from $\mathcal{G}$ by learning from $D_{\text{meta-tr}}$, so that $\hat{g}$ can achieve a small $L_{\mathcal{D}_{\text{meta}}}(\hat{g})$. We denote $\hat{g} = B_{\mathcal{G}}(D_{\text{meta-tr}})$. We can apply ERM

$$\hat{g} = B_{\mathcal{G}}(D_{\text{meta-tr}}) = \arg\min_{g\in\mathcal{G}} \frac{1}{N_{\text{meta}}} \sum_{j=1}^{N_{\text{meta}}} l_{\text{meta}}(g(D_{\text{tr}j}), h_j^*) = \arg\min_{g\in\mathcal{G}} L_{S_{\text{meta}}}(g), \quad (5)$$

where $L_{S_{\text{meta}}}$ is the meta training error.

In practice, we can design a set of $(\mathcal{G}, B_{\mathcal{G}})$ pairs and select the best pair using a held-out meta-validation set $D_{\text{meta-val}} = \{(D_{\text{tr}m}, h_m^*)\}_{m=1}^{M_{\text{meta}}}$ sampled i.i.d. from $\mathcal{D}_{\text{meta}}$. This is called meta model selection (meta-validation).

Discussion The meta model g and the supervised learning algorithm $A_{\mathcal{H}}$ (cf. Eq. (2)) have the same forms of inputs and outputs. Therefore, g can be seen as a (supervised) learning algorithm which may involve an optimization process. Nevertheless, the generalization ability of g and $A_{\mathcal{H}}$ are drastically different. $A_{\mathcal{H}}$ is designed or selected by model selection specifically for $\mathcal{D}$, while g is learned from D_{meta} with the goal to generalize to tasks sampled from $\mathcal{D}_{\text{meta}}$.

2.3. Generalization Ability of Learned Meta Models

Our unifying view enables applying experiences of learning a base model that will generalize well to novel examples to meta-learning. For example, to increase the size of the meta-training set or to minimize the domain shift between $\mathcal{D}_{\text{meta}}$ and where the novel tasks will be sampled (Ben-David et al., 2010; Gong et al., 2012) should improve the generalization ability of the learned meta model $\hat{g}$. In the experiments we empirically show that those experiences are applicable.

2.4. A Principled Way to Apply Meta-Learning

Our view clearly indicates the components of meta-learning.

- meta labeled example (D_{tr}, h^*) and meta loss l_{meta}
- meta hypothesis set $\mathcal{G} = \{g\}$ and meta algorithm $B_{\mathcal{G}}$

Applying meta-learning to an area thus requires defining them accordingly. We discuss four examples in Section B. We further present a detailed case study on how to apply our meta-learning framework to few-shot learning in Section C.

2.5. Applications of Supervised Learning Techniques

Our unifying view enables transferring techniques and algorithms of supervised learning to meta-learning with minimum adjustment. In our experiment, we adapt widely-used techniques for improving generalization abilities or facilitating optimization, such as data augmentation, ensemble methods (Breiman, 1996; Zhou, 2012; Dietterich, 2000), joint training with related objectives (e.g., multi-task learning (Argyriou et al., 2007)), and pre-training (Yosinski et al., 2014), to meta-learning and demonstrate their effectiveness.

Besides, it is well-known that non-parametric supervised learning models (Weinberger & Saul, 2009) are able to capture local and heterogeneous structures in data. Contrast to the fact that most existing meta models are parametric, we present a non-parametric approach inspired by SVM-KNN (Zhang et al., 2006; Chao et al., 2013) in Section A.

2.6. Related Work (See Section D)

3. Experiments

We empirically validate the advantages of our unifying view of meta-learning on few-shot learning: (a) whether the number of meta-examples influences generalization; (b) whether supervised learning techniques are applicable; (c) whether our unifying view can facilitate new applications. We will use "meta (labeled) example" and "task" interchangeably.

3.1. Experimental Setups

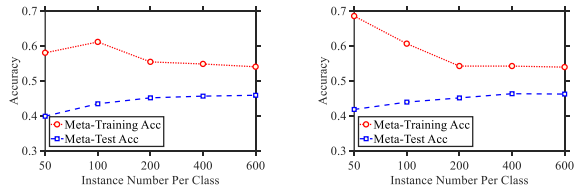
Datasets. The *Mini*ImageNet dataset (Vinyals et al., 2016) is a subset of the ImageNet (Russakovsky et al., 2015) and is widely-used to evaluate few-shot learning. There are 100 classes and 600 examples per class. We split the datasets following (Ravi & Larochelle, 2017), where there are 64, 16, 20 classes for meta-train-pool, meta-val-pool, and meta-test-pool, respectively. (See Section C.3 for the definition.) We call this the standard split (SS).

Meta Examples (Tasks) and Evaluation protocols. We follow (Rusu et al., 2019) to evaluate few-shot learning in a robust way. We draw 10,000 tasks from meta-test-pool and there are 15 validation images per class in a task, except for Table 4. That is, a meta labeled example $(D_{\text{tr}}, D_{\text{val}})$ has 15 images per class in D_{val}. We report the mean accuracy.

Baseline Methods. We investigate Model Agnostic Meta-Learning (MAML) (Finn et al., 2017a) and Prototypical Network (ProtoNet) (Snell et al., 2017). See Section E.2.

3.2. Empirical Generalization Analysis

Following our supervised view of meta-learning, the generalization ability of the learned meta model should be affected by the "effective" number of meta training examples; i.e., the number of different tasks. We empirically validate the influence with various meta-train-pool configurations on *Mini*ImageNet (SS). We keep all the 64 classes in meta-train-pool to keep the meta distribution intact. Instead, we change the number of instances in each class from 600 to 50 to construct different meta-train-pools. When there are limited instances per class in the meta-train-pool, the number of uniquely sampled tasks is constrained as well.



(*a*) MAML (*b*) ProtoNet

Figure 2. 1-shot 5-way classification accuracy on *Mini*ImageNet (SS). We show meta-training (red) and meta-test (blue) accuracy.

The change of few-shot learning performances with different meta-train-pools is summarized in Fig. 2. The trend clearly follows the supervised view: the more meta-training examples (few-shot tasks) are provided, the better generalization ability is demonstrated. Specifically, we found that learning with 50 instances per class significantly over-fits.

3.3. Supervised Learning for Meta-Learning

In this subsection, we leverage five techniques from supervised learning — joint training (multi-objective learning), pre-training, bagging, data augmentation, and non-parametric approaches — to improve meta-learning. We experiment on *Mini*ImageNet (SS), unless stated otherwise.

Multi-Objective Learning. In addition to the original objective (i.e., meta-training loss), learning jointly with related objectives (as a data-dependent regularization) has shown promising results in supervised learning. Here we add another multi-class classification objective: a 64-way classifier over all classes in meta-train-pool. This classifier shares the weights with the meta model, except the last fully-connected layer. Oreshkin et al. (2018) explored this idea, yet we provide more insights in Table 1.

Pre-Training. It is well-known that a good model initialization significantly facilitates the following model optimization for down-stream task (Erhan et al., 2010; Bengio et al., 2007). Specifically, supervised pre-training (Yosinski et al., 2014) on a large labeled dataset has become a standard in many applications. Here we investigate training a 64-way classifier with cross-entropy loss at first, and use it to initialize the meta model. Such a strategy has been explored in (Chen et al., 2019; Qiao et al., 2018; Rusu et al., 2019; Li et al., 2018b), yet we provide more insights.

The results by applying the two techniques are summarized in Table 1. Both techniques greatly and consistently improve the few-shot test accuracy for both algorithms. *However, their underlying influences are very different.* Pre-training achieves the highest meta-training and meta-validation accuracy, justifying its effectiveness to facilitate optimization. Multi-objective learning does not increase the meta-training accuracy much but improves the meta-validation and meta-test accuracy, verifying its ability to improve generalization. In summary, we show that well-known supervised learning techniques are applicable to meta-learning in the same manner they benefit supervised learning.

Table 1. Multi-objective learning (Multi-Train) and pre-training (Pre-Train) for 1-shot 5-way classification on *Mini*ImageNet (SS).

	Train Acc	Val Acc	Test Acc
ProtoNet	0.539	0.450	0.461
ProtoNet (Pre-Train)	0.602	0.487	0.500
ProtoNet (Multi-Train)	0.537	0.465	0.484
MAML	0.540	0.439	0.459
MAML (Pre-Train)	0.576	0.460	0.478
MAML (Multi-Train)	0.554	0.454	0.470

Table 2. Bagging for 1-shot 5-way classification. ProtoNet: no bagging; Average: average basic ProtoNets. Dist-Ensemble, Prob-Ensemble: two ensemble strategies.

	ProtoNet	Average	Dist-Ens	Prob-Ens
w/o Pre-Train	0.461	0.463	0.488	**0.491**
w/ Pre-Train	0.500	0.492	0.522	**0.526**

Ensemble Methods. Ensemble methods leverage the diversity among a set of basic models to construct a robust summarized model. We apply a simple ensemble method bagging (Breiman, 1996) to meta-learning, which reduces the model variances by majority voting over many classifiers. We sample 10 different sub meta-train-pools from the original one: each sub-pool contain 48 classes. Then we train 10 ProtoNets over those sub meta-train-pools and ensemble their results for final tests.

The results are in Table 2. The "ProtoNet" column is the model trained on the original meta-train-pool, the same as in Table 1. The average performance of all 10 basic models are listed in the "Average" column. (Note that, this is not ensemble yet.) We investigate two types of ensemble. The first one averages the distances computed by all models ("Dist-Ensemble") before softmax (cf. Eq. (9)) and the second one averages all prediction probabilities ("Prob-Ensemble"). We find that the average performance of basic ProtoNets is on par with that of the ProtoNet trained on the full 64-class meta-train-pool. But after we ensemble those basic models, the final performance improves a lot, which validates the effectiveness of ensemble techniques for meta-learning.

Importantly, we found that pre-training and bagging are compatible and lead to an impressive 0.526 accuracy, which is on par with state-of-the-art results (Qiao et al., 2018).

Additional Experiments. See Section E.3 and E.4.

4. Conclusion

We cast meta-learning as supervised learning, which explains the generalization among tasks and provides principles to apply meta-learning. This framework enables extending supervised learning techniques to meta-learning. We conduct empirical studies to demonstrate these advantages.

References

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *ICLR*, 2018.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

Argyriou, A., Evgeniou, T., and Pontil, M. Multi-task feature learning. In *NIPS*, 2007.

Bachman, P., Sordoni, A., and Trischler, A. Learning algorithms for active learning. In *ICML*, 2017.

Balaji, Y., Sankaranarayanan, S., and Chellappa, R. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018.

Baxter, J. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

Bello, I., Zoph, B., Vasudevan, V., and Le, Q. V. Neural optimizer search with reinforcement learning. In *ICML*, 2017.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P., and Vedaldi, A. Learning feed-forward one-shot learners. In *NIPS*, 2016.

Boney, R. and Ilin, A. Semi-supervised few-shot learning with maml. 2018.

Breiman, L. Bagging predictors. *Machine learning*, 24(2): 123–140, 1996.

Chao, W.-L., Liu, J.-Z., and Ding, J.-J. Facial age estimation based on label-sensitive learning and age-oriented regression. *Pattern Recognition*, 46(3):628–641, 2013.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. In *ICLR*, 2019.

Clavera, I., Nagabandi, A., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*, 2019.

Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15, 2000.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Duan, Y., Andrychowicz, M., Stadie, B., Ho, O. J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *NIPS*, 2017.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning? *JMLR*, 11(Feb):625–660, 2010.

Fan, Y., Tian, F., Qin, T., Li, X.-Y., and Liu, T.-Y. Learning to teach. In *ICLR*, 2018.

Finn, C. *Learning to Learn with Gradients*. PhD thesis, UC Berkeley, 2018.

Finn, C. and Levine, S. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017b.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. A bridge between hyperparameter optimization and larning-to-learn. *arXiv preprint arXiv:1712.06283*, 2017.

Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. Meta learning shared hierarchies. In *ICLR*, 2018.

Garg, V. and Kalai, A. Supervising unsupervised learning. In *NeurIPS*, 2018.

Gidaris, S. and Komodakis, N. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018.

Gong, B., Shi, Y., Sha, F., and Grauman, K. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

Hariharan, B. and Girshick, R. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017.

Hsu, K., Levine, S., and Finn, C. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018.

Huang, P.-S., Wang, C., Singh, R., Yih, W.-t., and He, X. Natural language to structured query generation via meta-learning. In *NAACL*, 2018.

Kaiser, Ł., Nachum, O., Roy, A., and Bengio, S. Learning to remember rare events. 2017.

Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on CVPR*, 2011.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition*, 2013.

Lee, Y. and Choi, S. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2018.

Lemke, C., Budka, M., and Gabrys, B. Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130, 2015.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018a.

Li, K. and Malik, J. Learning to optimize. In *ICLR*, 2017.

Li, K., Min, M. R., Bai, B., Fu, Y., and Graf, H. P. Network reparameterization for unseen class categorization. 2018b.

Li, Z., Zhou, F., Chen, F., and Li, H. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

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

Maurer, A. Algorithmic stability and meta-learning. *JMLR*, 6(Jun):967–994, 2005.

Maurer, A., Pontil, M., and Romera-Paredes, B. The benefit of multitask representation learning. *JMLR*, 17(1):2853–2884, 2016.

Metz, L., Maheswaranathan, N., Cheung, B., and Sohl-Dickstein, J. Learning unsupervised learning rules. *arXiv preprint arXiv:1804.00222*, 2018.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *ICLR*, 2018.

Munkhdalai, T. and Yu, H. Meta networks. In *ICML*, 2017.

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

Oreshkin, B. N., Lacoste, A., and Rodriguez, P. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

Pan, S. J., Yang, Q., et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Qiao, S., Liu, C., Shen, W., and Yuille, A. L. Few-shot image recognition by predicting parameters from activations. In *CVPR*, 2018.

Quattoni, A. and Torralba, A. Recognizing indoor scenes. In *CVPR*, pp. 413–420, 2009.

Ratner, A. J., Ehrenberg, H., Hussain, Z., Dunnmon, J., and Ré, C. Learning to compose domain-specific transformations for data augmentation. In *NIPS*, 2017.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Ravi, S. and Larochelle, H. Meta-learning for batch mode active learning. In *ICLR Workshop*, 2018.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

Ritter, S., Wang, J. X., Kurth-Nelson, Z., Jayakumar, S. M., Blundell, C., Pascanu, R., and Botvinick, M. Been there, done that: Meta-learning with episodic recall. In *ICML*, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F.-F. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

Salakhutdinov, R., Torralba, A., and Tenenbaum, J. Learning to share visual appearance for multiclass object detection. In *CVPR*, 2011.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Sharma, S., Jha, A., Hegde, P., and Ravindran, B. Learning to multi-task by active sampling. In *ICLR*, 2018.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

Stadie, B., Yang, G., Houthooft, R., Chen, P., Duan, Y., Wu, Y., Abbeel, P., and Sutskever, I. The importance of sampling inmeta-reinforcement learning. In *NeurIPS*, 2018.

Sudderth, E. B. and Jordan, M. I. Shared segmentation of natural scenes using dependent pitman-yor processes. In *NIPS*, 2008.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 1998.

Vanschoren, J. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

Vartak, M., Thiagarajan, A., Miranda, C., Bratman, J., and Larochelle, H. A meta-learning perspective on cold-start recommendations for items. In *NIPS*, 2017.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.

Vilalta, R. and Drissi, Y. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2): 77–95, 2002.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *NIPS*, 2016.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Wang, Y.-X. and Hebert, M. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.

Wang, Y.-X., Ramanan, D., and Hebert, M. Learning to model the tail. In *NIPS*, 2017.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10(Feb):207–244, 2009.

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Colmenarejo, S. G., Denil, M., de Freitas, N., and Sohl-Dickstein, J. Learned optimizers that scale and generalize. In *ICML*, 2017.

Ying, W., Zhang, Y., Huang, J., and Yang, Q. Transfer learning via learning to transfer. In *ICML*, 2018.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *NIPS*, 2014.

Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. In *RSS*, 2018.

Zhang, H., Berg, A. C., Maire, M., and Malik, J. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

Zhang, Y., Wei, Y., and Yang, Q. Learning to multitask. In *NeurIPS*, 2018.

Zhou, Z.-H. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

Zhu, X., Anguelov, D., and Ramanan, D. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014.

Supplementary Material

A. Meta-KNN

We present a procedure inspired by SVM-KNN (Zhang et al., 2006; Chao et al., 2013) to build a non-parametric meta model. We call it meta-KNN, as summarized in Algorithm 1. We empirically show its superior performance on meta labeled examples from heterogeneous domains (cf. Section E.3).

Algorithm 1 meta-KNN (cf. Eq. (5))

Required: $D_{\text{meta-tr}}$: a meta-training set
Required: $B_{\mathcal{G}}$: an iterative meta learning algorithm
Required: α, β, K: hyper-parameters
Required: $\hat{g} = B_{\mathcal{G}}(D_{\text{meta-tr}})$, till converge
Meta input: D_{tr} of a novel task
1: Search KNN(D_{tr}): KNN tasks from $D_{\text{meta-tr}}$
2. Copy $\tilde{g}$ from $\hat{g}$. Fine-tune $\tilde{g}$ by applying $B_{\mathcal{G}}$ for β epochs on KNN(D_{tr}) with a step size α
Meta output: $\tilde{g}(D_{\text{tr}})$

B. A Principled Way to Apply Meta-Learning

We discuss four examples as follows. Fig. 3 gives an illustration.

Unsupervised Learning Let us consider searching a feature extractor from unlabeled data that will benefits downstream applications (Boney & Ilin, 2018; Metz et al., 2018; Yu et al., 2018). In this case, $D_{\text{tr}} = \{x_n\}_{n=1}^N$, h^* is the ideal feature extractor, and l_{meta} measures the performance gap on the downstream application. $\mathcal{G}$ is a set of objective functions for unsupervised learning, while $B_{\mathcal{G}}$ can be ERM by stochastic gradient descent (SGD).

Active Learning The goal is to minimize the data labeling effort by querying N' informative examples. For classification, $D_{\text{tr}} = \{x_n\}_{n=1}^N$ is an unlabeled set, h^* is the model learned with the best N' queried examples from D_{tr}, and l_{meta} measures the performance gap on classification. g is a learning process constrained to query labels for N' examples by investigating x and the learning progress (Ravi & Larochelle, 2018; Sharma et al., 2018; Bachman et al., 2017). $B_{\mathcal{G}}$ can be ERM by SGD or by reinforcement learning algorithms (Ravi & Larochelle, 2018; Sharma et al., 2018).

Domain Generalization The goal is to train a base model on a set of source domains (SD) and apply it to a set of target domains (TD) without fine-tuning. In this case, $D_{\text{tr}} = \{(x_n, y_n)\}_{n=1}^N$ is the labeled data from SD, while h^* is the ideal base model that will work well on TD. l_{meta} is the performance gap on TD. g can be a model predictor or a learning process that considers the domain shifts (Li et al., 2018a; Balaji et al., 2018). $B_{\mathcal{G}}$ can be ERM by SGD.

Optimization One focus is to search a better updating rule in iterative optimization. In this case, D_{tr} is an objective function $L_S(\theta)$ parameterized by θ; e.g., the training loss. h^* is the optimal solution of L_S or of a related objective L_V; e.g., the validation error. l_{meta} is the gap of objective values. g is an iterative optimization algorithm with a (meta) learnable updating rule. $B_{\mathcal{G}}$ can be EMR by SGD (Andrychowicz et al., 2016; Wichrowska et al., 2017) or reinforcement learning algorithms (Li & Malik, 2017; Bello et al., 2017).

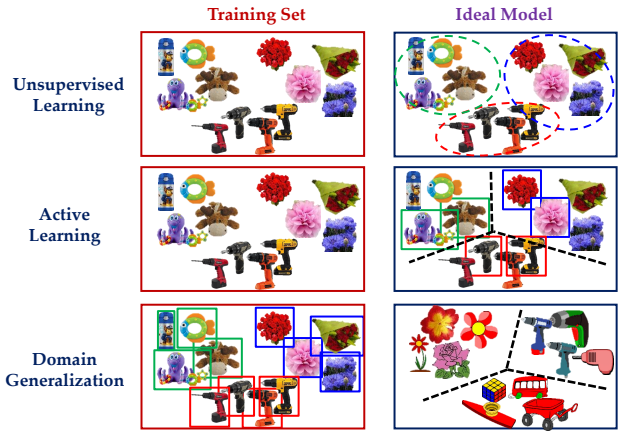


Figure 3. Meta examples in different areas. A labeled image is surrounded by a colored box that indicate its label. The ideal model in unsupervised learning is a feature extractor that facilitates clustering or classification. The ideal model in active learning is the classifier trained on the optimally queried images. The ideal model for domain generalization is a classifier for a different domain.

C. Case Study: Few-Shot Learning

We present a case study on how to apply our meta-learning framework to few-shot learning. The goal of few-shot learning is to quickly build a model for a novel task; i.e., with minimum training time and training data. Specifically, we focus on one-shot C-way learning for image classification.

C.1. Meta Labeled Example and Meta Loss

The training set D_{tr} contains one labeled image for each of the C classes; i.e., $D_{\text{tr}} = \{(x_n, y_n)\}_{n=1}^N$ ($C = N$ in this case). The ideal model h^* is an C-way classifier. The loss $l_{\text{meta}}(h, h^*)$ measures how different h compared to h^*. The later two components are flexible to design and we present two examples as follows.

Meta Loss in the Model Space h^* is trained on a larger training set. Let h be parameterized by θ, $l_{\text{meta}}(h, h') = \|\theta - \theta'\|_2^2$ (Wang & Hebert, 2016; Wang et al., 2017).

Meta Loss by Example Loss h^* will work well on $D_{\text{val}} = \{(x_m, y_m)\}_{m=1}^M$, a validation set sampled in the same way as D_{tr}. Suppose h^* minimizes the validation error L_V in Eq. (3); i.e., $h^* = \arg\min_h L_V(h)$. Then a reasonable choice of $l_{\text{meta}}(h, h^*)$ is $L_V(h) - L_V(h^*)$ which is equivalent to $L_V(h)$. In other word, we replace h^* and l_{meta} by D_{val} and l. The meta-training set thus becomes $D_{\text{meta-tr}} = \{(D_{\text{tr}j}, D_{\text{val}j})\}_{j=1}^{N_{\text{meta}}}$ and ERM in Eq. (5) can be re-written accordingly as (with constants being ignored)

$$\hat{g} = \arg\min_{g\in\mathcal{G}} \sum_{j=1}^{N_{\text{meta}}} L_{V\,j}(g(D_{\text{tr}j}))$$
$$= \arg\min_{g\in\mathcal{G}} \sum_{j=1}^{N_{\text{meta}}} \sum_{m=1}^{M} l(g(D_{\text{tr}j})(x_{jm}), y_{jm}). \quad (6)$$

$L_{V\,j}$ is the validation loss on $D_{\text{val}j} = \{(x_{jm}, y_{jm})\}_{m=1}^M$.

Eq. (6) has been applied in most of the few-shot learning algorithms (Hariharan & Girshick, 2017; Ravi & Larochelle, 2017; Gidaris & Komodakis, 2018; Qiao et al., 2018; Snell et al., 2017; Sung et al., 2018; Bertinetto et al., 2016; Vinyals et al., 2016) and we will focus on it in the next subsections.

C.2. Meta Model and Meta Algorithm

We discuss two exemplar designs of g: one views g as a supervised learning algorithm (Finn et al., 2017a) and the other views g as a feed-forward model predictor (Snell et al., 2017). Both algorithms are used in our experiments.

Meta Model as Learning Algorithm Since g outputs a classifier h given D_{tr}, it can be seen as a learning algorithm. Suppose h is parameterized by θ, let us apply ERM

$$g(D_{\text{tr}}) = \arg\min_\theta L_S(\theta) = \arg\min_\theta \frac{1}{N}\sum_{n=1}^{N} l(h_\theta(x_n), y_n).$$

Since D_{tr} is small, ERM may suffer over-fitting. One solution is to apply an iterative optimizer (e.g., gradient descent (GD)) with early stopping and a carefully chose initialization ψ (Finn et al., 2017a): early stopping limits the hypothesis set while ψ prevents under-fitting. Suppose $L_S(\theta)$ is differentiable w.r.t. θ, g with one-step GD and step size α is

$$g_\psi(D_{\text{tr}}) = \hat{\theta} = \psi - \alpha \times \nabla_\theta L_S(\psi). \quad (7)$$

The meta hypothesis set $\mathcal{G}$ thus becomes $\{g_\psi\}$ with different initializations. To search $g_{\hat{\psi}}$, we again apply ERM but on the meta-training set $D_{\text{meta-tr}}$ (cf. Eq. (6))

$$\hat{\psi} = \arg\min_\psi \sum_{j=1}^{N_{\text{meta}}} \sum_{m=1}^{M} l(g_\psi(D_{\text{tr}n})(x_{jm}), y_{jm}). \quad (8)$$

If we apply SGD for optimization then this is the one-step MAML (Finn et al., 2017a). In other word, MAML and its variants (Lee & Choi, 2018; Finn & Levine, 2018; Li et al., 2017; Rusu et al., 2019) follow our framework.

Meta Model as Model Predictor Alternatively, we can view $g(D_{\text{tr}})$ as a model predictor. For example, in Prototypical Network (ProtoNet) (Snell et al., 2017)

$$g_\psi(D_{\text{tr}})(x) = \arg\max_c \exp(-\|\psi(x) - \psi(x_c)\|_2^2), \quad (9)$$

where x_c is the image of class c, and ψ is a feature extractor. Snell et al. (2017) learns ψ via ERM (cf. Eq. (6)), essentially following our framework. Similar approaches are (Qiao et al., 2018; Sung et al., 2018; Wang & Hebert, 2016; Vinyals et al., 2016; Wang et al., 2017).

C.3. Collecting Meta-Training Set

After identifying the form of meta labeled example (D_{tr}, h^*) which is $(D_{\text{tr}}, D_{\text{val}})$ for few-shot learning, the next step is to collect the meta-training set. While collecting $(D_{\text{tr}}, D_{\text{val}})$ one-by-one is the standard way, it might not be efficient. For image classification it is easier to collect images class-by-class to first create a pool (named meta-train-pool), which may contain many classes and each has many examples. We then synthesize $(D_{\text{tr}}, D_{\text{val}})$ from the pool. This is the standard few-shot learning setting for image classification. There will be three pools of disjoint classes: meta-train pool, meta-val-pool, meta-test-pool. We draw meta labeled examples from meta-val-pool for meta model selection.

D. Related Work

There are excellent overviews and surveys (Thrun & Pratt, 1998; Vilalta & Drissi, 2002; Lemke et al., 2015; Vanschoren, 2018; Finn, 2018; Metz et al., 2018) on meta-learning. Ours is different by drawing a clear connection to supervised learning. There are also theoretical analyses on meta-learning (Baxter, 2000; Maurer et al., 2016; Maurer, 2005); however, the flexibility to design h^* (e.g., by validation data) is not taken into account. Franceschi et al. (2017) connects hyperparameter tuning and meta-learning, which aligns with the comparison of $A_{\mathcal{H}}$ and g in Section 2.2. (Garg & Kalai, 2018) also relates meta-learning to supervised learning but only for meta-unsupervised learning.

Meta-learning has also been applied to reinforcement learning and imitation learning (Stadie et al., 2018; Frans et al., 2018; Wang et al., 2016; Duan et al., 2016; 2017; Finn et al., 2017b; Yu et al., 2018), optimization (Andrychowicz et al., 2016; Wichrowska et al., 2017; Li & Malik, 2017; Bello et al., 2017), recommendation systems (Vartak et al., 2017), data augmentation (Ratner et al., 2017), natural language processing (Huang et al., 2018), neural architecture search (Elsken et al., 2018), continual learning (Riemer et al., 2019; Kaiser et al., 2017; Al-Shedivat et al., 2018; Clavera et al., 2019), transfer and multi-task learning (Ying

et al., 2018; Zhang et al., 2018), and teaching (Fan et al., 2018). Some algorithms aim for versatile purposes (Mishra et al., 2018; Munkhdalai & Yu, 2017; Ritter et al., 2018; Finn et al., 2017a; Santoro et al., 2016; Nichol et al., 2018).

E. Additional Experiments and Details

E.1. Experimental Setups

Datasets. We also consider a challenging case on the *Mini*ImageNet dataset where the splits are 30, 30, 40 classes, in which the diversity of meta-training examples is limited. We call the former standard split (SS) and the later challenging split (CS).

We further consider two set of datasets for more challenging settings. To enlarge the heterogeneity among tasks, we synthesize a "Heterogeneous" dataset from five different fine-grained classification datasets, namely AirCraft (Maji et al., 2013), Car-196 (Krause et al., 2013), Caltech-UCSD Birds (CUB) 200-2011 (Wah et al., 2011), Stanford Dog (Khosla et al., 2011), and Indoor (Quattoni & Torralba, 2009). We randomly sampled 60 classes with 50 images each from each of the 5 datasets, and equally split classes into meta-train-pool, meta-val-pool, and meta-test-pool. That is, there are 100 classes in meta-train-pool (same for the others), which includes 20 classes from each fine-grained dataset.

To investigate the applicability of meta-learning, we use the Office-Home datasets (Venkateswara et al., 2017) in a domain generalization problem. There are 65 classes and 4 domains of images per class. We test two domains, "Clipart" and "Product", which have the largest number of images.

E.2. Implementation Details

We resize images to 84×84 following (Vinyals et al., 2016). We apply the first-order MAML and tune the number of updates in Eq. (7) by meta-validation. We re-implement both algorithms and use the same C-way setting in meta-training and meta-test for consistency. That is, we disregard the trick (Snell et al., 2017) that trains with 30-way tasks while tests with 5-way tasks. Our baseline results are close to those reported in the recent overview of few-shot learning (Chen et al., 2019).

We apply the standard 4-layer ConvNets as the backbone (Vinyals et al., 2016; Snell et al., 2017). In each layer, convolution, batch normalization, ReLU, and max-pooling are concatenated sequentially. During meta-training SGD with Adam (Kingma & Ba, 2014) is employed, with a initial learning rate 2e-3.

E.3. Additional Experiments

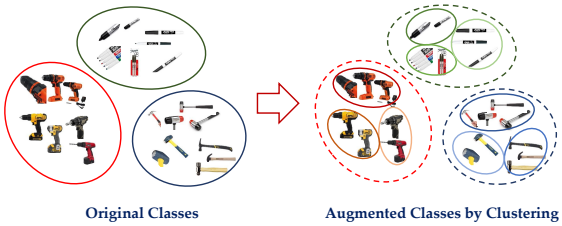


Figure 4. The class augmentation strategy by K-means.

Table 3. Test accuracy of 1-shot 30-way tasks on *Mini*ImageNet (CS). "K" means the K-means we run to augment the meta-train-pool and meta-val-pool classes. $K = 1$ means no augmentation.

$K = 1$	$K = 2$	$K = 4$	$K = 8$
0.146	0.146	0.155	0.161

Data Augmentation. Increasing the number of training examples by data augmentation is a popular technique in supervised learning to improve the generalization ability, especially when the training examples are not diverse enough.

Here we adapt the idea to meta-learning. We consider a challenging yet more realistic case on *Mini*ImageNet (CS). Specifically, we re-split the dataset into 30, 30, 40 classes for meta-train-pool, meta-val-pool, and meta-test-pool, and focus on 1-shot 30-way classification. Note that, this setting might break the applicability of meta-learning to few-shot learning: every time we will sample the same 30 classes from meta-train-pool, greatly limiting the diversity of meta labeled examples. However, this setting is indeed more realistic in practice according to the well-know long-tailed distribution (Sudderth & Jordan, 2008; Salakhutdinov et al., 2011; Zhu et al., 2014; Wang et al., 2017): there are more few-shot classes than many-shot classes.

We present a data augmentation strategy at the meta example (task) level, inspired by (Hsu et al., 2018). Specifically, we perform K-means within each class to split a class into K subcategories, resulting in $30 \times K$ augmented classes in meta-train-pool (and meta-val-pool). To ensure the quality of K-means, we pre-train a 30-way classifier on the whole meta-train-pool and use the features for K-means. We run 30 trials of K-means to compensate its randomness. We construct a meta-training example (task) by first picking a trial and sampling 30 classes from the corresponding augmented meta-train-pool. Fig. 4 gives an illustration.

The results with data augmentation are in Table 3. We cluster each class in the meta-train-pool and meta-val-pool with $K = 1, 2, 4, 8$ subcategories; 1 means no K-means. We observe a clear trend: the more classes we augment to increase task diversity, the higher accuracy we can achieve.

Non-Parametric Methods. Non-parametric supervised learning approaches are well-known to capture local and heterogeneous structures in data. Here we extend the methods

Table 4. Parametric meta models (ProtoNet) Vs. non-parametric meta models (ProtoNet-KNN). We show the test accuracy of 1-shot 5-way fine-grained classification on the Heterogeneous dataset.

	ProtoNet	ProtoNet-KNN
Test Acc	0.372	0.386

Table 5. Few-shot 5-way domain generalization on Office-Home. Two domains are investigated: Clipart (C) and Product (P).

	Meta-training		Meta-test		Test Acc	
Case	Source	Target	Source	Target	1-Shot	5-Shot
I-1	C	C	C	C	0.341	0.477
I-2	P	C	P	C	0.296	0.350
I-3	C	C	P	C	0.275	0.342
I-4	P	P	P	C	0.264	0.283
II-1	P	P	P	P	0.448	0.609
II-2	C	P	C	P	0.291	0.381
II-3	C	C	C	P	0.284	0.338
II-4	P	P	C	P	0.275	0.329

to meta-learning by applying meta-KNN (cf. Algorithm 1) to ProtoNet, named ProtoNet-KNN. In meta-learning a neighbor is a meta labeled example (i.e., a few-shot task). We characterize each task by its average feature and use the Euclidean distance to measure task similarity.

We investigate ProtoNet-KNN on the Heterogeneous dataset described in Section E.1. We focus on fine-grained classification: a test task contains 5 classes from one dataset. In meta-training, we sample a fine-grained task randomly from a dataset to train a single ProtoNet. We expect the learned ProtoNet to be equipped with dataset-agnostic fine-grained discriminative knowledge. In meta-test, we then apply ProtoNet-KNN to adapt the ProtoNet to the K nearest neighbors of a test task. To facilitate neighbor searching, we pre-sample 2,000 training tasks and set $K = 100$, $\beta = 1$, and $\alpha = 0.0002$ (cf. Algorithm 1). Table 4 shows the meta-test results (2,000 tasks). Non-parametric approaches greatly improve the performance on heterogeneous tasks. We expect a larger gap by a more sophisticated task distance.

E.4. Domain Generalization and Meta Domain Shifts

Existing few-shot learning settings assume that the training and validation examples (e.g., images) within a task are from the same domain. In practice it is sometime desirable to remove the assumption. For instance, we might want a robot to learn in a constraint environment, where examples are not diverse and are essentially few-shot, and to be able to apply the knowledge in the wild. Recall that our supervised view of meta-learning allows users to define the training set and the ideal model in a task according to the problem at hand. By defining the ideal model to be the one that will work well in a different domain, we can remove the same-domain assumption in a systematic way. We investigate this idea and apply the trick in Section C to replace the ideal model by a validation set collected in a different domain.

We experiment on the Off-Home dataset (see Section E.1), which is designed for domain adaptation. We focus on two domains: Clipart (C) and Product (P). There are 65 classes and we split them into 25 for meta-train-pool, 15 for meta-val-pool, and 25 for meta-test-pool. We work on *few-shot 5-way* tasks in which a task contains a training set D_{tr} from one (source) domain and a validation set D_{val} from the other (target) domain.

Table 5 summarizes the results. We clearly see the difficulty when the source and target domains within a task are different by comparing Case I-1 and I-2 (similarly, II-1 and II-2). We note that in each of these cases the distribution of the meta-training and meta-test examples are the same. Therefore, there is no meta-domain shift and the gap simply indicates the difficulty of tasks.

We further investigate meta-domain shifts by constructing meta-training and meta-test examples in a different way (Case I-3, I-4 and II-3, II-4). The performance in these cases are outperformed by Case I-2 and II-2 respectively where each group considers the same meta-test examples. We argue that this gap indeed results from meta-domain shifts (cf. Section 2.3) and might be resolved via domain adaptation by casting meta-learning as supervised learning.