BOIL: TOWARDS REPRESENTATION CHANGE FOR FEW-SHOT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

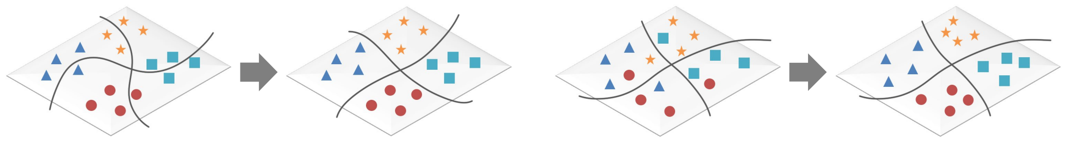
Model Agnostic Meta-Learning (MAML) is one of the most representative of gradient-based meta-learning algorithms. MAML learns new tasks with a few data samples using inner updates from a meta-initialization point and learns the meta-initialization parameters with outer updates. It has recently been hypothesized that *representation reuse*, which makes little change in efficient representations, is the dominant factor in the performance of the meta-initialized model through MAML in contrast to *representation change*, which causes a significant change in representations. In this study, we investigate the necessity of *representation change* for the ultimate goal of few-shot learning, which is solving domain-agnostic tasks. To this aim, we propose a novel meta-learning algorithm, called *BOIL* (Body Only update in Inner Loop), which updates only the body (extractor) of the model and freezes the head (classifier) during inner loop updates. BOIL leverages *representation change* rather than *representation reuse*. A frozen head cannot achieve better results than even a random guessing classifier at the initial point of new tasks, and feature vectors (representations) have to move quickly to their corresponding frozen head vectors. We visualize this property using cosine similarity, CKA, and empirical results without the head. Although the inner loop updates purely hinge on *representation change*, BOIL empirically shows significant performance improvement over MAML, particularly on cross-domain tasks. The results imply that *representation change* in gradient-based meta-learning approaches is a critical component.

1 INTRODUCTION

Meta-learning, also known as “learning to learn,” is a methodology that imitates human intelligence that can adapt quickly with even a small amount of previously unseen data through the use of previous learning experiences. To this aim, meta-learning with deep neural networks has mainly been studied using metric- and gradient-based approaches. Metric-based meta-learning (Koch, 2015; Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018) compares the distance between feature embeddings using models as a mapping function of data into an embedding space, whereas gradient-based meta-learning (Ravi & Larochelle, 2016; Finn et al., 2017; Nichol et al., 2018) quickly learns the parameters to be optimized when the models encounter new tasks.

Model-agnostic meta-learning (MAML) (Finn et al., 2017) is the most representative gradient-based meta-learning algorithm. MAML algorithm consists of two optimization loops: an inner loop and an outer loop. The inner loop learns task-specific knowledge, and the outer loop finds a universally good meta-initialized parameter allowing the inner loop to quickly learn any task from the initial point with only a few examples. This algorithm has been highly influential in the field of meta-learning, and numerous follow-up studies have been conducted (Oreshkin et al., 2018; Rusu et al., 2018; Zintgraf et al., 2018; Yoon et al., 2018; Finn et al., 2018; Triantafillou et al., 2019; Sun et al., 2019; Na et al., 2019; Tseng et al., 2020).

Very recent studies (Raghu et al., 2020; Arnold et al., 2019) have attributed the success of MAML to high-quality features before the inner updates from the meta-initialized parameters. For instance, Raghu et al. (2020) claimed that MAML learns new tasks by updating the head (the last fully connected layer) with almost the same features (the output of the penultimate layer) from the meta-initialized network. In this paper, we categorize the learning patterns as follows: A small change in



(a) MAML/ANIL. (b) BOIL.

Figure 1: **Difference in task-specific (inner) updates between MAML/ANIL and BOIL**. In the figure, the lines represent the decision boundaries defined by the head (classifier) of the network. Different shapes and colors mean different classes. (a) MAML mainly updates the head with a negligible change in body (extractor); hence, representations on the feature space are almost identical. ANIL does not change in the body during inner updates, and they are therefore identical. However, (b) BOIL updates only the body without changing the head during inner updates; hence, representations on the feature space change significantly with the fixed decision boundaries. We visualize the representations from various data sets using UMAP (Uniform Manifold Approximation and Projection for dimension reduction) (McInnes et al., 2018) in Appendix B.

the representations during task learning is named *representation reuse*, whereas a large change is named *representation change*.[1] Thus, *representation reuse* was the common belief of MAML.

Herein, we pose an intriguing question: *Is representation reuse sufficient for meta-learning?* We believe that the key to successful meta-learning is closer to *representation change* than to *representation reuse*. More importantly, *representation change* is crucial for cross-domain adaptation, which is considered the ultimate goal of meta-learning. By contrast, the MAML accomplished with *representation reuse* might be poorly trained for cross-domain adaptation since the success of *representation reuse* might rely heavily on the similarity between the source and the target domains.

To answer this question, we propose a novel meta-learning algorithm that leverages *representation change*. Our contributions can be summarized as follows:

- We emphasize the necessity of *representation change* for meta-learning through cross-domain adaptation experiments.
- We propose a simple but effective meta-learning algorithm that *learns the Body (extractor) of the model Only in the Inner Loop*(BOIL). We empirically show that BOIL improves the performance over all benchmark data sets and that this improvement is particularly noticeable in fine-grained data sets or cross-domain adaptation.
- We interpret the connection between BOIL and the algorithm using preconditioning gradients (Flennerhag et al., 2020) and show their compatibility, improving performance.
- We demonstrate that the BOIL algorithm enjoys *representation reuse* on the low-/mid-level body and *representation change* on the high-level body using the cosine similarity and the Centered Kernel Alignment (CKA). We visualize the features between before and after adaptation, and empirically analyze the effectiveness of the body of BOIL through an ablation study on eliminating the head.
- For ResNet architectures, we propose a disconnection trick that removes the back-propagation path of the last skip connection. The disconnection trick strengthens *representation change* on the high-level body.

2 Problem Setting

2.1 Meta-learning Framework (MAML)

The MAML(Model-agnostic Meta-learning) algorithm (Finn et al., 2017) attempts to meta-learn the best initialization of the parameters for a task-learner. It consists of two main optimization loops: an inner loop and an outer loop. First, we sample a batch of tasks within a data set distribution. Each

[1]In our paper, *representation reuse* and *representation change* correspond to *feature reuse* and *rapid learning* in (Raghu et al., 2020), respectively. To prevent confusion from terminology, we re-express the terms.

task τ_i consists of a support set S_{τ_i} and a query set Q_{τ_i}. When we sample a support set for each task, we first sample n labels from the label set and then sample k instances for each label. Thus, each support set contains $n \times k$ instances. For a query set, we sample instances from the same labels with the support set.

With these tasks, the MAML algorithm conducts both meta-training and meta-testing. During meta-training, we first sample a meta-batch consisting of B tasks from the meta-training data set. In the *inner loops*, we update the meta-initialized parameters θ to task-specific parameters θ_{τ_i} using the *task-specific* loss $L_{S_{\tau_i}}(f_\theta)$, where f_θ is a neural network parameterized by θ, as follows:[2]

$$\theta_{\tau_i} = \theta - \alpha \nabla_\theta L_{S_{\tau_i}}(f_\theta) \tag{1}$$

Using the query set of the corresponding task, we compute the loss $L_{Q_{\tau_i}}(f_{\theta_{\tau_i}})$ based on each inner updated parameter. By summing all these losses, the *meta-loss* of each meta-batch, $L_{meta}(\theta)$, is computed. The meta-initialized parameters are then updated using the meta-loss in the *outer loop* through a gradient descent.

$$\theta' = \theta - \beta \nabla_\theta L_{meta}(\theta), \text{ where } L_{meta}(\theta) = \sum_{i=1}^{B} L_{Q_{\tau_i}}(f_{\theta_{\tau_i}}) \tag{2}$$

In meta-testing, the inner loop, which can be interpreted as task-specific learning, is the same as in meta-training. However, the outer loop only computes the accuracy using a query set of tasks and does not perform a gradient descent; thus, it does not update the meta-initialization parameters.

2.2 Experimental setup

We used two backbone networks, **4conv network** with 64 channels from Vinyals et al. (2016) and **ResNet-12** starting with 64 channels and doubling them after every block from Oreshkin et al. (2018). For the batch normalization, we used batch statistics instead of the running statistics during meta-testing, following the original MAML (Finn et al., 2017). We trained all models for 30,000 epochs and then used *the model with the best accuracy on meta-validation data set* to verify the performance. We applied an inner update *once* for both meta-training and meta-testing. The outer learning rate was set to 0.001 and the inner learning rate was set to 0.5 for all the baselines and our models. All results were reproduced by our group and reported as the average and standard deviation of the accuracies over 5 $\times$ 1,000 tasks, and the values in parentheses in the algorithm name column of the tables are the number of shots. We validated both MAML/ANIL and BOIL on two general data sets, **miniImageNet** (Vinyals et al., 2016) and **tieredImageNet** (Ren et al., 2018), and two specific data sets, **Cars** (Krause et al., 2013) and **CUB** (Welinder et al., 2010). Note that our algorithm is not for state-of-the-art performance but for a proposal of a new learning scheme for meta-learning. Full details on the implementation and data sets are described in Appendix A.[3] In addition, the results of the other data sets at a size of 32 $\times$ 32 and using the 4conv network with 32 channels from Finn et al. (2017) (i.e., original setting) are reported in Appendix C and Appendix E, respectively.

3 BOIL (Body Only update in Inner Loop)

3.1 The ultimate goal of meta-learning: Domain-agnostic adaptation

Recently, Raghu et al. (2020) proposed two opposing hypotheses, *representation reuse* and *representation change*, and demonstrated that *representation reuse* is the dominant factor of MAML. We can discriminate two hypotheses according to which part of the neural network, body or head, is mostly updated through the inner loop. Here, the body indicates all convolutional layers, and the head indicates the remaining fully connected layer. In other words, the *representation change* hypothesis

[2]Although the inner loop(s) can be applied through one or more steps, for simplicity, we consider only the case of a single inner loop.

[3]All implementations are based on Torchmeta (Deleu et al., 2019) except for WarpGrad, and all results were reproduced according to our details. These results are not the highest for MAML/ANIL because our setting is more fitted to BOIL. However, under more suitable hyperparameters for each algorithm, the best performance of BOIL is better than that of MAML/ANIL.

attributes the capability of MAML to the updates on the body, whereas the *representation reuse* hypothesis considers that the network body is already universal to various tasks before the inner loops. To demonstrate the *representation reuse* hypothesis of MAML, the authors proposed the ANIL (Almost No Inner Loop) algorithm, which only updates the head in the inner loops during training and testing, and showed that ANIL has a performance comparable to that of MAML. This implies that the representation trained by MAML/ANIL, even before updated task-specifically, is sufficient to achieve the desired performance. Furthermore, they proposed the NIL-testing (No Inner Loop) algorithm, which removes the head and performs unseen tasks using only the distance between the representations of a support set and those of a query set during testing to identify the capability of *representation reuse*. NIL-testing of MAML also achieves a performance comparable to MAML. Based on these results, it was claimed that the success of MAML is attributed to *representation reuse*.

Here, we investigate the necessity of *representation change*. We believe that the meta-trained models should achieve a good performance in many other domains, which is referred to as *domain-agnostic adaptation* in this paper. To this end, *representation reuse* is not appropriate since *representation reuse* uses the similarity between the source and target domains. The higher the similarity, the higher the efficiency. Therefore, when there are no strong similarities between the source and target domains, good representations for the source domain could be imperfect representations for the target domain. Table 2, which lists our experimental results on cross-domain tasks, shows that the MAML enjoying *representation reuse* is worse than BOIL leveraging *representation change*, which will be discussed in detail in the next section.

3.2 BOIL ALGORITHM

Inspired by the necessity, we design an algorithm that updates only the body of the model and freezes the head of the model during the task learning to enforce *representation change* through inner updates. Because the gradients must be back-propagated to update the body, we set the learning rate of the head to zero in the inner updates during both meta-training and meta-testing. Otherwise, the learning and evaluation procedures of BOIL are the same as those of MAML. Therefore, the computational overhead does not change.

Formally speaking, with the notations used in Section 2.1, the meta-initialized parameters θ can be separated into body parameters θ_b and head parameters θ_h, that is, $\theta = \{\theta_b, \theta_h\}$. For a sample image $x \in \mathbb{R}^i$, an output can be expressed as $\hat{y} = f_\theta(x) = f_{\theta_h}(f_{\theta_b}(x)) \in \mathbb{R}^n$, where $f_{\theta_b}(x) \in \mathbb{R}^d$. The task-specific body parameters θ_{b,τ_i} and head parameters θ_{h,τ_i} through an inner loop given task τ_i are thus as follows:

$$\theta_{b,\tau_i} = \theta_b - \alpha_b \nabla_{\theta_b} L_{S_{\tau_i}}(f_\theta) \;\; \& \;\; \theta_{h,\tau_i} = \theta_h - \alpha_h \nabla_{\theta_h} L_{S_{\tau_i}}(f_\theta) \tag{3}$$

where α_b and α_h are the inner loop learning rates corresponding to the body and head, respectively. MAML usually sets $\alpha = \alpha_b = \alpha_h (\neq 0)$, ANIL sets $\alpha_b = 0$ and $\alpha_h \neq 0$, and BOIL sets $\alpha_b \neq 0$ and $\alpha_h = 0$.

These simple differences force the change in the dominant factor of task-specific updates, from the head to the body. Figure 1 shows the main difference in the inner updates between MAML/ANIL and BOIL. To solve new tasks, the head mainly or only changes in MAML/ANIL (Raghu et al., 2020), whereas in BOIL, the body changes.

3.2.1 PERFORMANCE IMPROVEMENT ON BENCHMARK DATA SETS AND CROSS-DOMAIN TASKS

Table 1: Test accuracy (%) of 4conv network on benchmark data sets. The values in parenthesis in the algorithm name column of tables are the number of shots.

Domain	General (Coarse-grained)		Specific (Fine-grained)	
Dataset	miniImageNet	tieredImageNet	Cars	CUB
MAML(1)	47.44 ± 0.23	47.44 ± 0.18	45.27 ± 0.26	56.18 ± 0.37
ANIL(1)	47.82 ± 0.20	**49.35** ± 0.26	46.81 ± 0.24	57.03 ± 0.41
BOIL(1)	**49.61** ± 0.16	48.58 ± 0.27	**56.82** ± 0.21	**61.60** ± 0.57
MAML(5)	61.75 ± 0.42	64.70 ± 0.14	53.23 ± 0.26	69.66 ± 0.03
ANIL(5)	63.04 ± 0.42	65.82 ± 0.12	61.95 ± 0.38	70.93 ± 0.28
BOIL(5)	**66.45** ± 0.37	**69.37** ± 0.12	**75.18** ± 0.21	**75.96** ± 0.17

Table 2: Test accuracy (%) of 4conv network on cross-domain adaptation.

adaptation	general to general		general to specific		specific to general		specific to specific	
meta-train	tieredImageNet	miniImageNet	miniImageNet	miniImageNet	Cars	Cars	CUB	Cars
meta-test	miniImageNet	tieredImageNet	Cars	CUB	miniImageNet	tieredImageNet	Cars	CUB
MAML(1)	47.60 ± 0.24	51.61 ± 0.20	33.57 ± 0.14	40.51 ± 0.08	26.95 ± 0.15	28.46 ± 0.18	32.22 ± 0.30	29.64 ± 0.19
ANIL(1)	49.67 ± 0.31	52.82 ± 0.29	34.77 ± 0.31	41.12 ± 0.15	28.67 ± 0.17	29.41 ± 0.19	33.07 ± 0.43	28.32 ± 0.32
BOIL(1)	**49.74** ± 0.26	**53.23** ± 0.41	**36.12** ± 0.29	**44.20** ± 0.15	**33.71** ± 0.13	**34.06** ± 0.20	**35.44** ± 0.46	**34.79** ± 0.27
MAML(5)	65.22 ± 0.20	65.76 ± 0.27	44.56 ± 0.21	53.09 ± 0.16	30.64 ± 0.19	32.62 ± 0.21	41.24 ± 0.21	32.18 ± 0.13
ANIL(5)	66.47 ± 0.16	66.52 ± 0.28	46.55 ± 0.29	55.82 ± 0.21	35.38 ± 0.10	36.94 ± 0.10	43.05 ± 0.23	37.99 ± 0.15
BOIL(5)	**69.33** ± 0.19	**69.37** ± 0.23	**50.64** ± 0.22	**60.92** ± 0.11	**44.51** ± 0.25	**46.09** ± 0.23	**47.30** ± 0.22	**45.91** ± 0.28

Table 1 and Table 2 display the superiority of BOIL on most benchmark data sets and on cross-domain adaptation tasks, where the source and target domains differ (i.e., the meta-training and meta-testing data sets are different). In Table 1, the performance improvement is particularly noticeable on the specific domain data sets CUB and Cars. The results demonstrate that *representation change* is necessary even if there is a similarity between the source and target domains. Table 2 shows that BOIL is closer to the ultimate goal of meta-learning, which is a domain-agnostic adaptation.

Recently, Guo et al. (2019) noted that existing meta-learning algorithms have weaknesses in terms of cross-domain adaptation. We divide the cross-domain adaptation into four cases: general to general, general to specific, specific to general, and specific to specific. Previous studies considered the cross-domain scenario from a general domain to a specific domain (Chen et al., 2019; Guo et al., 2019). In this paper, we also evaluate the reverse case. BOIL outperforms MAML/ANIL not only on the typical cross-domain adaptation scenario but also on the reverse one. In particular, the performance improvement, when the domain changes from birds (CUB as a meta-train set) to cars (Cars as a meta-test set), implies that the *representation change* in BOIL enables the model to adapt to an unseen target domain that is entirely different from the source domain.

3.2.2 BOIL AND PRECONDITIONING GRADIENTS

Some aspects of BOIL can be explained by preconditioning gradients (Lee & Choi, 2018; Flennerhag et al., 2020). Preconditioning gradients occur when a particular layer is shared over all tasks, warping the spaces (e.g., rotating and scaling). For instance, one might consider the frozen head of BOIL to be a warp layer of the entire body (Flennerhag et al., 2020).

Preconditioning gradients can avoid overfitting in a high-capacity model (Flennerhag et al., 2020), and such a benefit is still valid with BOIL. Indeed, many prior studies have suffered from an overfitting problem, and thus it is challenging to train the backbone network more extensively than the 4conv network with 32 filters (Finn et al., 2017). By contrast, BOIL can increase the validation accuracy with more extensive networks. The accuracy of models with 32, 64, and 128 filters continues to increase to 64.02, 66.72, and 69.23, without overfitting. In Appendix D, we report these results as well as the training and valid accuracy curves of BOIL for three different network sizes, in which the larger networks are trained well. We further hypothesize that the head is the most critical part of an overfitting problem, and BOIL can succeed in dealing with the problem by simply ignoring the head in the inner loops.

However, one essential difference between BOIL and the preconditioning gradients is whether the head is frozen. No prior studies have frozen the last fully connected layer or used any additional fully connected layer to precondition the gradients, and hence *representation reuse* is still the major factor of their training. To the best of our knowledge, BOIL is the first approach that enforces *representation change* by freezing the head in the inner loops.

Model	Accuracy
WarpGrad w/ last warp head	83.19 ± 0.79
WarpGrad w/o last warp head	83.16 ± 0.69
BOIL-WarpGrad w/ last warp conv	83.68 ± 0.82
BOIL-WarpGrad w/o last warp conv	**84.88** ± 0.42

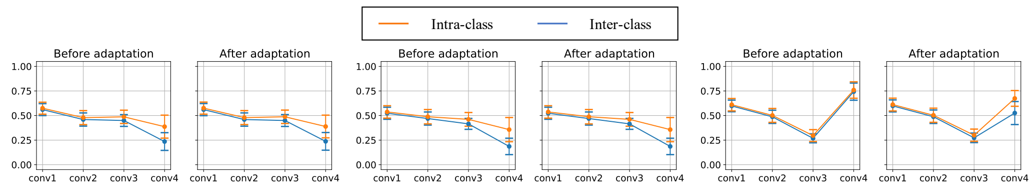
Table 3: Test accuracy(%) of WarpGrad and BOIL-WarpGrad over 5×100 tasks.

To investigate the gain from *representation change*, we adapt BOIL to WarpGrad (Flennerhag et al., 2020).[4] Four different models are tested, the architectures of which are fully described in Appendix F. Table 3 shows the test accuracy of the four models, where the BOIL-WarpGrad models freeze the fully connected layer from the corresponding WarpGrad model. It is observed that BOIL-WarpGrad improves WarpGrad and BOIL-WarpGrad without the last warp conv improves BOIL-WarpGrad with the last warp conv. The latter result indicates that, to support BOIL, the last convolution layer must not be fixed but rather updated during the inner loops.

[4]We follow the setting in https://github.com/flennerhag/warpgrad, and the details about this implementation are in Appendix F.

4 Representation change in BOIL

4.1 Representation change before/after adaptation



(a) MAML. (b) ANIL. (c) BOIL.

Figure 2: Cosine similarity of 4conv network.

To analyze whether the learning scheme of BOIL is *representation reuse* or *representation change*, we explore the layer-wise alteration of the representations before and after adaptation. We compute the cosine similarities and CKA values of the convolution layers with the meta-trained 4conv network (as detailed in Appendix A). We first investigate the cosine similarity between the representations of a query set including 5 classes and 15 samples per class from **miniImageNet** after every convolution module. In Figure 2, the orange line represents the average of the cosine similarity between the samples having the same class, and the blue line represents the average of the cosine similarity between the samples having different classes. In Figure 2, the left panel of each algorithm is before the inner loop adaptation, and the right panel is after inner loop adaptation.

The key observations from Figure 2, which are also discussed in Section 4.2 with other experiments, are as follows:

- The cosine similarities of MAML/ANIL have the similar patterns (Figure 2a and Figure 2b), supporting *representation reuse* only. Their patterns do not show any noticeable difference before or after adaptation. They make the average of the cosine similarities monotonically decrease and make the representations separable by classes when the representations reach the last convolution layer. These analyses indicate that the effectiveness of MAML/ANIL heavily leans on the meta-initialized body, not the task-specific adaptation.
- The cosine similarities of BOIL have a different pattern from those of MAML/ANIL Figure 2c), supporting *representation change*. The BOIL's pattern changes to distinguish classes before and after adaptation. Before adaptation, BOIL reduces the average cosine similarities only up to conv3, and all representations are concentrated regardless of their classes after the last convolution layer. Hence, BOIL's meta-initialized body cannot distinguish the classes. However, after adaptation, the similarity of the different classes rapidly decrease on conv4, which means that the body can distinguish the classes through adaptation.
- The reason why the change in BOIL before and after adaptation occurs only on conv4 is a peculiarity of the convolutional body, analyzed by Zeiler & Fergus (2014). Although the general and low-level features produced through the front convolution layers (e.g., colors, lines, and shapes) do not differ much from the task-specific adaptation, the discriminative representations produced through the last convolution layer (conv4) differ from class to class. The importance of the last convolutional layer on the performance in few-shot image classification tasks is also investigated by Arnold et al. (2019); Chen et al. (2020). These changes before and after the adaptation support the fact that BOIL enjoys *representation layer reuse* at the low- and mid-levels of the body and *representation layer change* in a high-level of the body.[5] Nevertheless, the degree of *representation reuse* in a low- and mid-levels in BOIL is lower than that in MAML/ANIL, which is measured using gradient norms (Appendix I).

Through these observations, we believe that MAML follows the *representation reuse* training scheme, whereas BOIL follows both *representation reuse* (before the last convolution layer) and *representation change* (at the last convolution layer) learning schemes.

[5]In the (Raghu et al., 2020), *representation reuse* and *representation change* are defined at the algorithm-level. Namely, *representation reuse* indicates that none of the representations after the convolution layers significantly change during the inner loop updates, otherwise *representation change*. We extend this concept from the algorithm-level to the layer-level. To clarify, we use *representation layer reuse/change* for layer-level

Next, we demonstrate that BOIL both enjoys *representation reuse* on the low- and mid-level and *representation change* on the high-level of the body by computing the CKA (Kornblith et al., 2019) before and after adaptation. When the CKA between two representations is close to 1, the representations are almost identical. In Figure 3, as mentioned in Raghu et al. (2020), CKA shows that the MAML/ANIL algorithms do not change the representation in the body. However, BOIL changes the representation of the last convolution layer. This result indicates that the BOIL algorithm learns rapidly through *representation change*.

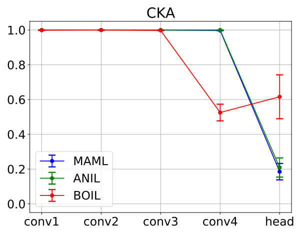


Figure 3: CKA of 4conv.

Note that the penultimate layer (i.e., conv4) of BOIL acts differently from the output layer (i.e., head) of MAML/ANIL. The penultimate layer of BOIL might seem like a pseudo-head layer, but not at all. The output layer of MAML/ANIL is adapted based on the well-represented features (i.e., features after conv4). In contrast, the penultimate layer of BOIL is adapted based on the poorly-represented features (i.e., features after conv3). It means that the head layer's role is to draw a simple decision boundary for the high-level features represented by the output of the last convolutional layer. However, the penultimate layer of BOIL acts as a non-linear transformation so that the fixed output head layer can effectively conduct a classifier role.

4.2 Empirical analysis of representation change in BOIL

Table 4: Test accuracy (%) of 4conv network according to the head's existence before/after an adaptation.

meta-train	miniImageNet							
meta-test	miniImageNet				Cars			
head	w/ head		w/o head (NIL-testing)		w/ head		w/o head (NIL-testing)	
adaptation	before	after	before	after	before	after	before	after
MAML(1)	19.96 ± 0.25	47.44 ± 0.23	48.28 ± 0.20	47.87 ± 0.14	20.05 ± 0.16	33.57 ± 0.14	34.47 ± 0.19	34.36 ± 0.16
ANIL(1)	20.09 ± 0.19	47.92 ± 0.20	**48.86** ± 0.12	**48.86** ± 0.12	20.16 ± 0.05	34.77 ± 0.31	**35.48** ± 0.24	**35.48** ± 0.24
BOIL(1)	19.94 ± 0.13	**49.61** ± 0.16	24.07 ± 0.19	46.73 ± 0.17	19.94 ± 0.06	**36.12** ± 0.29	23.30 ± 0.15	34.07 ± 0.32
MAML(5)	20.04 ± 0.17	61.75 ± 0.42	64.61 ± 0.39	64.47 ± 0.39	19.97 ± 0.18	44.56 ± 0.21	47.66 ± 0.28	47.53 ± 0.28
ANIL(5)	20.09 ± 0.13	63.04 ± 0.42	**66.11** ± 0.51	**66.11** ± 0.51	20.08 ± 0.07	46.55 ± 0.29	**49.62** ± 0.20	49.62 ± 0.20
BOIL(5)	20.04 ± 0.21	**66.45** ± 0.37	32.03 ± 0.16	64.61 ± 0.27	20.06 ± 0.16	**50.64** ± 0.22	30.33 ± 0.18	**50.40** ± 0.30

Table 4 describes the test accuracy on miniImageNet and Cars of the model meta-trained on miniImageNet before and after an inner update according to the presence of the head. To evaluate the performance in a case without a classifier, we first create a template of each class by averaging the representations from the support set. Then, the class of the sample from the query set is predicted as the class whose template has the highest cosine similarity with the representation of the sample. This is the same as NIL-testing in (Raghu et al., 2020).

The results provide some intriguing interpretations:

- **With the head for all algorithms** Before adaptation, all algorithms on the same- and cross-domain are unable to distinguish all classes (20%). This status could be considered as an optimum of meta-initialization. We also discuss it in Appendix J. In BOIL, the representations have to move quickly to their corresponding frozen head. Moreover, after adaptation, BOIL overwhelms the performance of the other algorithms. This means that *representation change* of BOIL is more effective than *representation reuse* of MAML/ANIL.
- **Without the head in MAML/ANIL** In this setting, representations from the body are evaluated before and after adaptation. Before adaptation, MAML and ANIL already generate sufficient representations to classify, and adaptation makes little or no difference. The former observation matches the cosine similarity gap between an intra- and inter-class on each left panel in Figure 2a and Figure 2b, and the latter observation matches the CKA values of close to or exactly 1 on conv4 of MAML/ANIL in Figure 3.
- **Without the head in BOIL** BOIL shows a steep performance improvement through adaptation on the same- and cross-domain. This result implies that the body of BOIL can be task-specifically updated. It is matched with Figure 2c, where the cosine similarity gap between the intra-class and inter-class on the space after conv4 is near zero before adaptation but increases after adaptation. This implies that the poor representations can be dramatically improved in BOIL. Therefore, the low CKA value on conv4 of BOIL in Figure 3 is natural.

To summarize, the meta-initialization by MAML and ANIL provides efficient representations through the body even before adaptation. By contrast, although BOIL's meta-initialization provides less efficient representations compared to MAML and ANIL, the body can extract more efficient representations through task-specific adaptation based on *representation change*. In addition, the *representation change* on the **Cars** data set is described in Appendix H.

5 BOIL TO A LARGER NETWORK

Many recent studies (Vuorio et al., 2019; Rusu et al., 2018; Sun et al., 2019) have used deeper networks such as ResNet (He et al., 2016), Wide-ResNet (Zagoruyko & Komodakis, 2016), and DenseNet (Huang et al., 2017) as a backbone network. The deeper networks, in general, use feature wiring structures to facilitate the feature propagation. We explore BOIL's applicability to a deeper network with the wiring structure, ResNet-12, and propose a simple trick to boost *representation change* by disconnecting the last skip connection. This trick is described in Section 5.1.

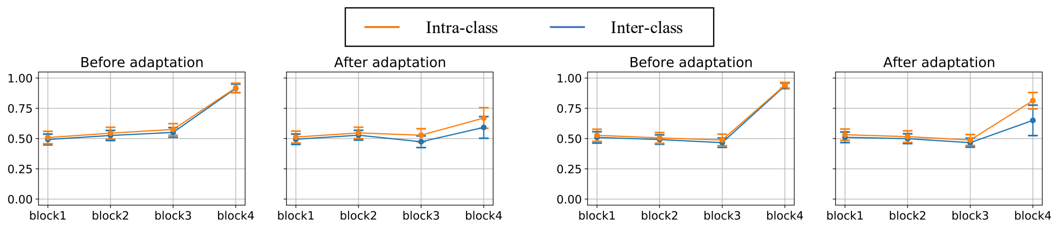
Table 5: 5-Way 5-Shot test accuracy (%) of ResNet-12. LSC means Last Skip Connection.

Meta-train	miniImageNet			Cars		
Meta-test	miniImageNet	tieredImageNet	Cars	Cars	miniImageNet	CUB
MAML w/ LSC	68.51 $\pm$ 0.39	71.67 $\pm$ 0.13	43.46 $\pm$ 0.15	75.49 $\pm$ 0.20	34.42 $\pm$ 0.06	35.87 $\pm$ 0.19
MAML w/o LSC	67.87 $\pm$ 0.22	70.31 $\pm$ 0.10	41.40 $\pm$ 0.11	73.63 $\pm$ 0.26	37.65 $\pm$ 0.11	34.77 $\pm$ 0.26
ANIL w/ LSC	68.54 $\pm$ 0.34	71.93 $\pm$ 0.11	45.13 $\pm$ 0.15	79.45 $\pm$ 0.23	35.03 $\pm$ 0.07	35.09 $\pm$ 0.19
ANIL w/o LSC	67.20 $\pm$ 0.13	69.79 $\pm$ 0.24	43.46 $\pm$ 0.18	75.32 $\pm$ 0.15	38.15 $\pm$ 0.15	36.06 $\pm$ 0.14
BOIL w/ LSC	70.50 $\pm$ 0.28	71.86 $\pm$ 0.21	49.69 $\pm$ 0.17	80.98 $\pm$ 0.14	45.89 $\pm$ 0.32	43.34 $\pm$ 0.21
BOIL w/o LSC	**71.30** $\pm$ 0.28	**74.12** $\pm$ 0.30	**49.71** $\pm$ 0.28	**83.99** $\pm$ 0.20	**48.41** $\pm$ 0.18	**44.23** $\pm$ 0.18

Table 5 shows the test accuracy results of ResNet-12, which is meta-trained and meta-tested with various data sets according to the fineness of the domains. This result indicates that BOIL can be applied to other general architectures by showing a better performance than MAML not only on standard benchmark data sets but also on cross-domain adaptation. Note that BOIL has achieved the best performance without the last skip connection in every experiment.

5.1 DISCONNECTION TRICK

Connecting the two learning schemes and ResNet's wiring structure, we propose a simple trick to eliminate the skip connection of the last residual block, which is referred to as a disconnection trick. In section 4.1, we confirmed that the model learned with BOIL applies the *representation reuse* at the low- and mid-levels of the body and *representation change* at the high-level of the body.



(a) BOIL w/ last skip connection.

(b) BOIL w/o last skip connection.

Figure 4: Cosine similarity of ResNet-12.

To investigate the effects of skip connections on a *representation change* learning scheme, we analyze the cosine similarity after every residual block in the same way as Figure 2. Figure 4a shows that ResNet with skip connections on all blocks rapidly changes not only the last block but also the other blocks. Because skip connections strengthen the gradient back-propagation, the scope of *representation change* extends to the front. Therefore, to achieve both the effective *representation reuse* and the *representation change* of BOIL, we suggest a way to weaken the gradient back-propagation from the loss function by removing the skip connection of the last block. As shown in

Figure 4b, with this simple disconnection trick, ResNet can improve the effectiveness of BOIL, as well as the *representation reuse* at the front blocks of the body and the *representation change* at the last block, and significantly improves the performance, as described in Table 5.

6 Related Work

MAML (Finn et al., 2017) is one of the most well-known algorithms in gradient-based meta-learning, achieving a competitive performance on few-shot learning benchmark data sets (Vinyals et al., 2016; Ren et al., 2018; Bertinetto et al., 2018; Oreshkin et al., 2018). To tackle the task ambiguity caused by insufficient data in few-shot learning, numerous studies have sought to extend MAML in various ways. Some studies (Oreshkin et al., 2018; Sun et al., 2019; Vuorio et al., 2019) have proposed feature modulators that make task-specific adaptation more amenable by shifting and scaling the representations extracted from the network body. In response to the lack of data for task-specific updates, there have also been attempts to incorporate additional parameters in a small number, rather than all model parameters (Zintgraf et al., 2018; Rusu et al., 2018; Lee & Choi, 2018; Flennerhag et al., 2020). With a similar approach, some studies suggested a way to update only the heads in the inner loop, which has been further improved to update the head using linear separable objectives. (Raghu et al., 2020; Bertinetto et al., 2018; Lee et al., 2019). Grant et al. (2018); Finn et al. (2018); Yoon et al. (2018); Na et al. (2019) have taken a probabilistic approach using Bayesian modeling and variational inference. In addition, Chen et al. (2020) showed that by allowing the discovered task-specific modules (i.e., a (small) subset of a network) to adapt, better performance is achieved than when allowing the whole network to adapt. Notably, in few-shot image classification tasks, the authors showed that the last convolution layer (i.e., penultimate layer) is the most important. Such results were also observed in Arnold et al. (2019).

To tackle more realistic problems, few-shot learning has recently been expanding beyond the standard n-way k-shot classification. Triantafillou et al. (2019) constructed a more scalable and realistic data set, called a meta-data set, which contains several data sets collected from different sources. In additions, Na et al. (2019) addressed n-way any-shot classification by considering the imbalanced data distribution in real-world. Furthermore, some studies (Cai & Shen, 2020; Chen et al., 2019) have recently explored few-shot learning on cross-domain adaptation, which is one of the ultimate goals of meta-learning. In addition, Guo et al. (2019) suggested a new cross-domain benchmark data set for few-shot learning and showed that the current meta-learning algorithms (Finn et al., 2017; Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018; Lee et al., 2019) underachieve compared to simple fine-tuning on cross-domain adaptation. We demonstrated that task-specific update with *representation change* is efficient for a cross-domain adaptation.

7 Conclusion

In this study, we investigated the necessity of *representation change* for solving domain-agnostic tasks and proposed the BOIL algorithm, which is designed to enforce *representation change* by learning only the body of the model in the inner loop. We connected BOIL with preconditioning gradients and showed that the effectivenesses from a connection, such as an overfitting reduction and robustness to hyperparameters change, are still valid. Furthermore, we adapt BOIL to WarpGrad, demonstrating improved performance. This result decouples the benefits of *representation change* and preconditioning gradients. Next, we demonstrated that BOIL trains a model to follow the *representation reuse* scheme on the low- and mid-levels of the body but trains it to follow the *representation change* scheme on the high-level of the body using the cosine similarity and the CKA. We validated the BOIL algorithm on various data sets, including miniImageNet, tieredImageNet, Cars, and CUB, as well as a cross-domain adaptation using a standard 4conv network and ResNet. The experimental results showed a significant improvement over MAML/ANIL, particularly cross-domain adaptation, implying that *representation change* should be considered for adaptation to unseen tasks.

We hope that our study inspires *representation change* in gradient-based meta-learning approaches. Our approach is the first to study *representation change* and focuses on classification tasks. However, we believe that our approach is also efficient in other methods or fields because our algorithm has no restrictions. Furthermore, connecting *representation change* to memorization overfitting addressed in (Yin et al., 2019; Rajendran et al., 2020) will be an interesting topic.

REFERENCES

Sébastien MR Arnold, Shariq Iqbal, and Fei Sha. Decoupling adaptation from modeling with meta-optimizers for meta learning. *arXiv preprint arXiv:1910.13603*, 2019.

Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

John Cai and Sheng Mei Shen. Cross-domain few-shot learning with meta fine-tuning. *arXiv preprint arXiv:2005.10544*, 2020.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Yutian Chen, Abram L Friesen, Feryal Behbahani, Arnaud Doucet, David Budden, Matthew Hoffman, and Nando de Freitas. Modular meta-learning with shrinkage. *Advances in Neural Information Processing Systems*, 33, 2020.

Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A Meta-Learning library for PyTorch, 2019. URL `https://arxiv.org/abs/1909.06576`. Available at: https://github.com/tristandeleu/pytorch-meta.

Rafael Rego Drumond, Lukas Brinkmeyer, Josif Grabocka, and Lars Schmidt-Thieme. Hidra: Head initialization across dynamic targets for robust architectures. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 397–405. SIAM, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pp. 9516–9527, 2018.

Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=rkeiQlBFPB`.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

Yunhui Guo, Noel CF Codella, Leonid Karlinsky, John R Smith, Tajana Rosing, and Rogerio Feris. A new benchmark for evaluation of cross-domain few-shot learning. *arXiv preprint arXiv:1912.07200*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Gregory Koch. Siamese neural networks for one-shot image recognition. 2015.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pp. 2927–2936, 2018.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29), 2018.

Donghyun Na, Hae Beom Lee, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. *arXiv preprint arXiv:1905.12917*, 2019.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 721–731, 2018.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkgMkCEtPB`.

Janarthanan Rajendran, Alexander Irpan, and Eric Jang. Meta-learning requires meta-augmentation. *Advances in Neural Information Processing Systems*, 33, 2020.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

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. *arXiv preprint arXiv:1803.00676*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2019.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. *arXiv preprint arXiv:2001.08735*, 2020.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems*, pp. 1–12, 2019.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2019.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pp. 7332–7342, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. *arXiv preprint arXiv:1810.03642*, 2018.

A Implementation Detail

A.1 n-way k-shot setting

We experimented in the 5-way 1-shot and 5-way 5-shot, and the number of shots is marked in parentheses in the algorithm name column of all tables. During meta-training, models are inner loop updated only once, and the meta-batch size for the outer loop is set to 4. During meta-testing, the number of task-specific (inner loop) updates is the same as meta-training. All models are trained for 30,000 iterations, and all the reported results are based on the model with the best validation accuracy.

A.2 model implementations

In our experiments, we employ the 4conv network and ResNet-12 for MAML and BOIL algorithms. 4conv network has of 4 convolution modules, and each module consists of a 3×3 convolution layer with 64 filters, batch normalization (Ioffe & Szegedy, 2015), a ReLU non-linearity, a 2×2 max-pool. This model is also the same as MAML except for the number of filters. However, we experiment with both MAML and BOIL with the 64 filters network for a fair comparison. ResNet-12 (He et al., 2016) has the same structure with the feature extractor of TADAM (Oreshkin et al., 2018). It has four residual blocks, and each block consists of 3 modules of convolution, batch normalization, and leaky ReLU (Xu et al., 2015). At every end of each residual block, 2×2 max-pool is applied, and the number of convolution filters is doubled from 64 on each block. Each block also has a wiring structure known as skip connection, which is a link made up of additions between the block's input and output feature for strengthening feature propagation.

Our proposed algorithms can be implemented by just dividing learning rates into for the body and the head. Table 6 shows the learning rates of each network and algorithm. α_b and α_h are the learning rates of the body and the head of the model during inner loops, and β_b and β_h are the learning rates of the body and the head of the model during outer loops.

	4conv network			ResNet-12		
	MAML	ANIL	BOIL	MAML	ANIL	BOIL
α_b	0.5	0.0	0.5	0.3	0.0	0.3
α_h	0.5	0.5	0.0	0.3	0.3	0.0
β_b	0.001	0.001	0.001	0.001	0.001	0.001
β_h	0.001	0.001	0.001	0.001	0.001	0.001

Table 6: Learning rates according to the algorithms.

A.3 Dataset

We validate the BOIL and MAML algorithms on several data sets, considering image size and fineness. Table 7 is the summarization of the used data sets.

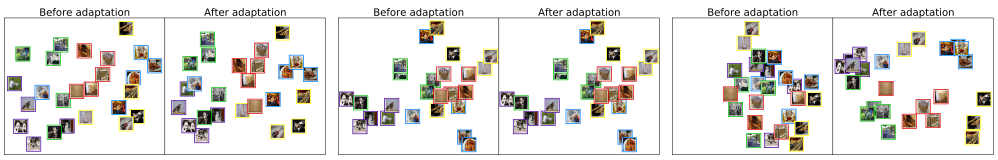
Table 7: Summary of data sets.

Data sets	miniImageNet	tieredImageNet	Cars	CUB
Source	Russakovsky et al. (2015)	Russakovsky et al. (2015)	Krause et al. (2013)	Welinder et al. (2010)
Image size	84×84	84×84	84×84	84×84
Fineness	Coarse	Coarse	Fine	Fine
# meta-training classes	64	351	98	100
# meta-validation classes	16	97	49	50
# meta-testing classes	20	160	49	50
Split setting	Vinyals et al. (2016)	Ren et al. (2018)	Tseng et al. (2020)	Hilliard et al. (2018)
Data sets	**FC100**	**CIFAR-FS**	**VGG-Flower**	**Aircraft**
Source	Krizhevsky et al. (2009)	Krizhevsky et al. (2009)	Nilsback & Zisserman (2008)	Maji et al. (2013)
Image size	32×32	32×32	32×32	32×32
Fineness	Coarse	Coarse	Fine	Fine
# meta-training classes	60	64	71	70
# meta-validation classes	20	16	16	15
# meta-testing classes	20	20	15	15
Split setting	Bertinetto et al. (2018)	Oreshkin et al. (2018)	Na et al. (2019)	Na et al. (2019)

B visualization using UMAP

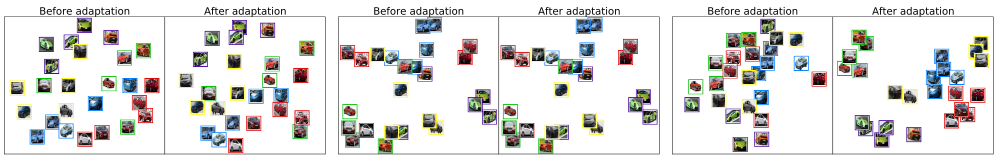
Through section 4.1, we show that conv4 in a 4conv network is the critical layer where *representation change* happens. We visualize these representations, the output of conv4, of samples from various data sets using UMAP (McInnes et al., 2018), which is an algorithm for general non-linear dimension reduction. *Samples with the same line color belong to the same class.* Many examples show the consistency with the intuition shown in Figure 1. When 1) similar instances with different classes are sampled together and 2) representations on the meta-train data set cannot capture representations on the meta-test data set, MAML/ANIL seems to be challenging to cluster samples on representation space since they are based on *representation reuse*.

B.1 Benchmark data sets



(a) MAML. (b) ANIL. (c) BOIL.

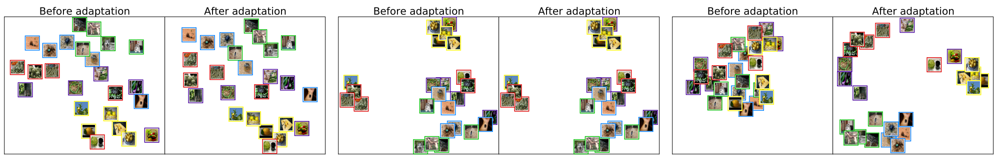
Figure 5: UMAP of samples from miniImageNet using the model meta-trained on miniImageNet.



(a) MAML. (b) ANIL. (c) BOIL.

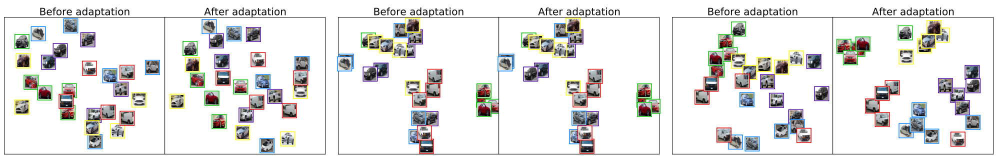
Figure 6: UMAP of samples from Cars using the model meta-trained on Cars.

B.2 Cross-domain adaptation



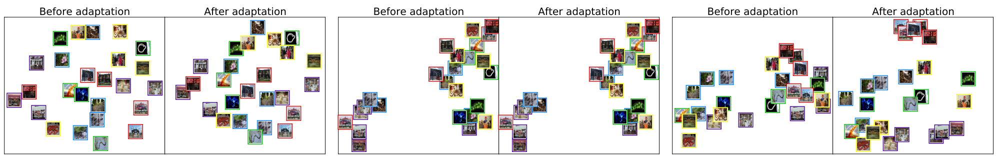
(a) MAML. (b) ANIL. (c) BOIL.

Figure 7: UMAP of samples from tieredImageNet using the model meta-trained on miniImageNet.



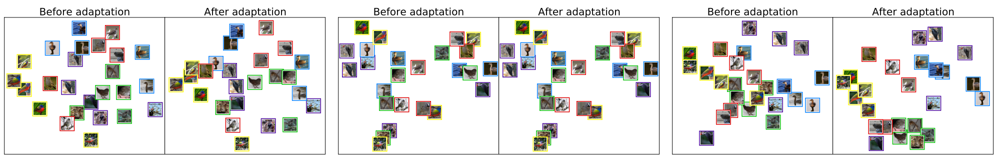
(a) MAML. (b) ANIL. (c) BOIL.

Figure 8: UMAP of samples from Cars using the model meta-trained on miniImageNet.



(a) MAML. (b) ANIL. (c) BOIL.

Figure 9: UMAP of samples from miniImageNet using the model meta-trained on Cars.



(a) MAML. (b) ANIL. (c) BOIL.

Figure 10: UMAP of samples from CUB using the model meta-trained on Cars.

C Results on Other Data sets

We applied our algorithm to other data sets with image size of 32 × 32. Similar to the analyses on section 4, these data sets can be divided into two general data sets, **CIFAR-FS** (Bertinetto et al., 2018) and **FC100** (Oreshkin et al., 2018), and two specific data sets, **VGG-Flower** (Nilsback & Zisserman, 2008) and **Aircraft** (Maji et al., 2013). Table 8, Table 9, and Table 10 generally show the superiority of BOIL even if image size is extremely tiny.

Table 8: Test accuracy (%) of 4conv network on benchmark dataset.

Domain	General (Coarse-grained)		Specific (Fine-grained)	
Dataset	CIFAR-FS	FC100	VGG-Flower	Aircraft
MAML(1)	56.55 ± 0.45	35.99 ± 0.48	60.94 ± 0.35	52.27 ± 0.23
ANIL(1)	57.13 ± 0.47	36.37 ± 0.33	63.05 ± 0.30	**54.54** ± 0.16
BOIL(1)	**58.03** ± 0.43	**38.93** ± 0.45	**65.64** ± 0.26	53.37 ± 0.29
MAML(5)	70.10 ± 0.29	47.58 ± 0.30	75.13 ± 0.43	63.44 ± 0.26
ANIL(5)	69.87 ± 0.39	45.65 ± 0.44	72.07 ± 0.48	63.21 ± 0.16
BOIL(5)	**73.61** ± 0.32	**51.66** ± 0.32	**79.81** ± 0.42	**66.03** ± 0.14

Table 9: Test accuracy (%) of 4conv network on cross-domain adaptation.

adaptation	general to general		general to specific		specific to general		specific to specific	
meta-train	FC100	CIFAR-FS	CIFAR-FS	CIFAR-FS	VGG-Flower	VGG-Flower	Aircraft	VGG-Flower
meta-test	CIFAR-FS	FC100	VGG-Flower	Aircraft	CIFAR-FS	FC100	VGG-Flower	Aircraft
MAML(1)	62.58 ± 0.35	52.81 ± 0.28	49.69 ± 0.24	27.03 ± 0.18	34.38 ± 0.19	32.45 ± 0.23	37.05 ± 0.19	25.70 ± 0.19
ANIL(1)	**63.05** ± 0.39	**55.36** ± 0.47	50.61 ± 0.29	27.39 ± 0.09	35.90 ± 0.20	33.84 ± 0.30	31.59 ± 0.22	24.55 ± 0.14
BOIL(1)	60.71 ± 0.43	54.18 ± 0.35	**56.77** ± 0.35	**29.29** ± 0.10	**39.15** ± 0.20	**34.37** ± 0.14	**49.85** ± 0.26	**29.05** ± 0.16
MAML(5)	75.32 ± 0.34	63.00 ± 0.18	64.49 ± 0.23	33.85 ± 0.25	46.81 ± 0.11	42.06 ± 0.43	47.74 ± 0.07	30.65 ± 0.19
ANIL(5)	**77.01** ± 0.51	63.89 ± 0.16	64.20 ± 0.10	33.24 ± 0.21	44.52 ± 0.25	40.51 ± 0.26	50.28 ± 0.12	28.74 ± 0.23
BOIL(5)	76.33 ± 0.30	**68.55** ± 0.20	**74.93** ± 0.11	**39.96** ± 0.11	**55.48** ± 0.21	**47.17** ± 0.38	**64.68** ± 0.23	**39.81** ± 0.25

Table 10: 5-Way 5-Shot test accuracy (%) of ResNet-12. The lsc means the last skip connection.

Meta-train	CIFAR-FS			VGG-Flower		
Meta-test	CIFAR-FS	FC100	VGG-Flower	VGG-Flower	CIFAR-FS	Aircraft
MAML w/ lsc	75.30 ± 0.19	69.34 ± 0.35	65.82 ± 0.30	74.82 ± 0.29	42.91 ± 0.20	28.50 ± 0.12
MAML w/o lsc	71.72 ± 0.19	67.60 ± 0.34	59.20 ± 0.26	72.07 ± 0.29	39.27 ± 0.23	26.94 ± 0.18
ANIL w/ lsc	74.87 ± 0.11	75.34 ± 0.45	63.72 ± 0.40	77.02 ± 0.29	45.80 ± 0.32	27.24 ± 0.13
ANIL w/o lsc	71.39 ± 0.28	69.29 ± 0.32	52.70 ± 0.24	72.13 ± 0.39	38.99 ± 0.22	26.09 ± 0.08
BOIL w/ lsc	**78.17** ± 0.14	**77.22** ± 0.45	73.90 ± 0.38	82.00 ± 0.17	50.91 ± 0.35	35.54 ± 0.25
BOIL w/o lsc	77.38 ± 0.10	70.98 ± 0.34	**73.96** ± 0.27	**83.97** ± 0.17	**55.82** ± 0.44	**37.74** ± 0.21

D Overfitting issue

We employ networks with various sizes of filters, e.g., 32, 64, 128. The best validation scores of each model are 64.01, 66.72, 69.23, and these results mean that with BOIL, the more extensive network yields higher accuracy without overfitting.

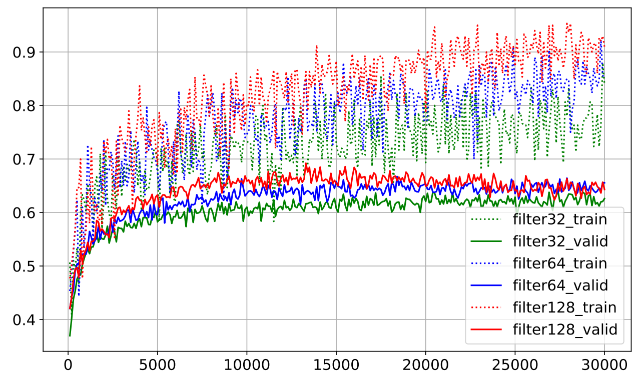


Figure 11: Training/Validation accuracy curve on miniImageNet according to filters in BOIL.

E Results under the original hyperparameters

We also evaluate our algorithm in the original setting (50 times smaller inner learning rate than ours) and confirm that BOIL is more robust to the change of hyperparameters than MAML. Such a characteristic is investigated in Lee & Choi (2018). Figure 12 shows the validation accuracy of BOIL and MAML/ANIL with the same hyperparameters optimized for MAML. It is observed that BOIL is the best or near-best, although the hyperparameters are not optimized for BOIL. Moreover, BOIL rapidly adapts and achieves considerable performance through just one adaptation.

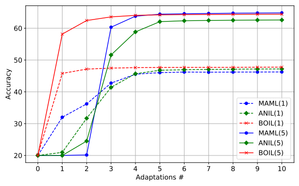


Figure 12: Accuracy on miniImageNet according to the number of adaptation(s).

Table 11: Test accuracy (%) under the same architecture, learning rate, and the number of inner updates with (Finn et al., 2017; Raghu et al., 2020).

Meta-train	miniImageNet			Cars		
Meta-test	miniImageNet	tieredImageNet	Cars	Cars	miniImageNet	CUB
MAML(1)	46.25 ± 0.18	49.45 ± 0.14	34.78 ± 0.36	46.02 ± 0.33	28.87 ± 0.11	29.92 ± 0.23
ANIL(1)	47.20 ± 0.27	50.04 ± 0.13	32.87 ± 0.39	45.31 ± 0.27	29.12 ± 0.11	30.39 ± 0.21
BOIL(1)	**47.76** ± 0.31	**51.35** ± 0.18	**34.89** ± 0.23	**50.54** ± 0.41	**32.40** ± 0.19	**32.99** ± 0.29
MAML(5)	**64.83** ± 0.30	**67.06** ± 0.25	48.25 ± 0.24	**69.27** ± 0.27	**43.52** ± 0.20	45.12 ± 0.20
ANIL(5)	62.59 ± 0.39	65.55 ± 0.16	45.44 ± 0.18	62.67 ± 0.25	36.89 ± 0.16	40.38 ± 0.19
BOIL(5)	64.40 ± 0.28	65.81 ± 0.26	**48.39** ± 0.25	68.56 ± 0.34	43.34 ± 0.21	**46.32** ± 0.11

Table 12: Test accuracy (%) according to the number of adaptation(s). Training and testing are on miniImageNet.

Adaptation #	1	2	3	4	5	6	7	8	9	10
MAML(1)	32.01 ± 0.24	36.21 ± 0.17	42.74 ± 0.20	45.54 ± 0.18	46.04 ± 0.16	46.21 ± 0.19	46.17 ± 0.18	46.20 ± 0.18	46.22 ± 0.16	46.25 ± 0.18
ANIL(1)	20.97 ± 0.03	31.68 ± 0.25	41.41 ± 0.26	45.69 ± 0.20	46.78 ± 0.26	46.95 ± 0.30	47.05 ± 0.31	47.10 ± 0.30	47.17 ± 0.28	47.20 ± 0.27
BOIL(1)	**45.79** ± 0.45	**47.15** ± 0.30	**47.46** ± 0.34	**47.61** ± 0.34	**47.67** ± 0.31	**47.70** ± 0.32	**47.70** ± 0.33	**47.71** ± 0.34	**47.74** ± 0.32	**47.76** ± 0.31
MAML(5)	20.02 ± 0.00	20.15 ± 0.02	60.31 ± 0.34	63.78 ± 0.34	**64.41** ± 0.35	**64.55** ± 0.33	**64.64** ± 0.31	**64.72** ± 0.31	**64.77** ± 0.30	**64.83** ± 0.40
ANIL(5)	20.00 ± 0.00	24.52 ± 0.19	51.59 ± 0.17	58.84 ± 0.46	62.06 ± 0.37	62.34 ± 0.36	62.45 ± 0.38	62.51 ± 0.37	62.55 ± 0.38	62.59 ± 0.39
BOIL(5)	**58.15** ± 0.23	**62.42** ± 0.33	**63.56** ± 0.26	**64.04** ± 0.30	64.21 ± 0.28	64.27 ± 0.30	64.32 ± 0.30	64.35 ± 0.29	64.38 ± 0.28	64.40 ± 0.28

F WarpGrad and BOIL-WarpGrad

F.1 Implementation Detail

We follow the default setting of the public code except for meta train steps and the number of filters, following Flennerhag et al. (2020). We change meta train steps to 100 and the number of filters to 128. The task is 20way-5shot(in expectation) on Omniglot. Here, "in expectation" means that 100 samples are used for task-specific updates, but the number of samples per class is not the same.

F.2 Architecture

Here are the architectures of WarpGrad and BOIL-WarpGrad. The WarpGrad w/o last warp head model is the default one in the original code.

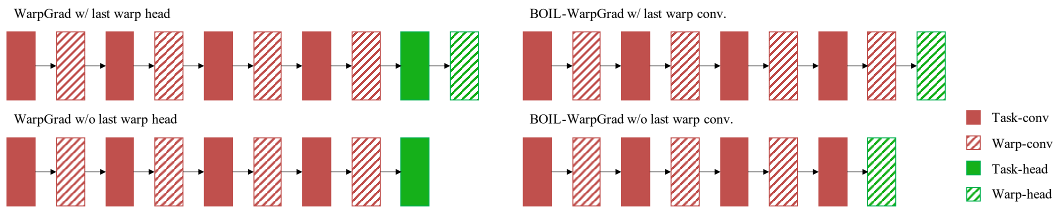
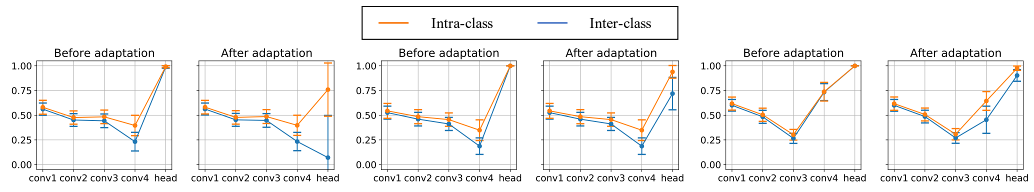

WarpGrad w/ last warp head
BOIL-WarpGrad w/ last warp conv.
WarpGrad w/o last warp head
BOIL-WarpGrad w/o last warp conv.
Task-conv
Warp-conv
Task-head
Warp-head

Figure 13: Architectures of WarpGrad and BOIL-WarpGrad.

G Cosine similarities including head

Figure 14 is an extended version of Figure 2.

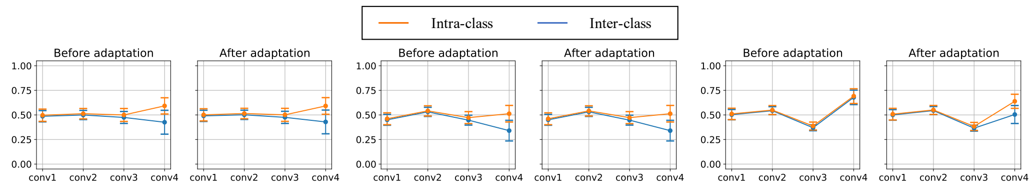


(a) MAML. (b) ANIL. (c) BOIL.

Figure 14: Cosine similarity of 4conv network including head.

H Representation change in BOIL on Cars

This section describes *representation change* in BOIL on Cars. The structure of this section is the same as that of section 4.



(a) MAML. (b) ANIL. (c) BOIL.

Figure 15: Cosine similarity of 4conv network on Cars.

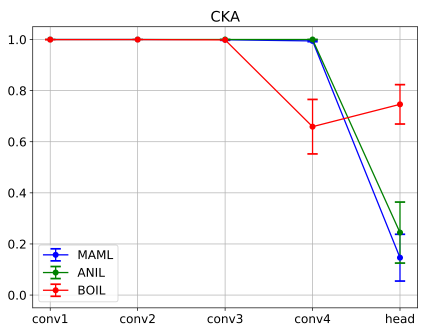


Figure 16: CKA of 4conv on Cars.

Table 13: Test accuracy (%) of 4conv network according to the head's existence before/after an adaptation.

meta-train	Cars							
meta-test	Cars				CUB			
head	w/ head		w/o head (NIL-testing)		w/ head		w/o head (NIL-testing)	
adaptation	before	after	before	after	before	after	before	after
MAML(1)	20.03 ± 0.25	45.27 ± 0.26	47.87 ± 0.18	47.23 ± 0.24	20.01 ± 0.08	29.64 ± 0.19	**31.01** ± 0.26	31.15 ± 0.23
ANIL(1)	20.01 ± 0.18	46.81 ± 0.24	**49.45** ± 0.18	49.45 ± 0.18	20.02 ± 0.08	28.32 ± 0.32	29.72 ± 0.27	29.72 ± 0.27
BOIL(1)	20.19 ± 0.19	**56.82** ± 0.21	25.46 ± 0.29	**52.36** ± 0.13	19.96 ± 0.12	**34.79** ± 0.27	22.93 ± 0.17	**34.51** ± 0.21
MAML(5)	20.11 ± 0.16	53.23 ± 0.26	59.67 ± 0.22	59.38 ± 0.23	20.00 ± 0.22	32.18 ± 0.13	36.12 ± 0.24	36.61 ± 0.19
ANIL(5)	20.09 ± 0.17	61.95 ± 0.38	**67.03** ± 0.36	67.03 ± 0.36	19.99 ± 0.18	37.99 ± 0.15	**43.27** ± 0.31	43.27 ± 0.31
BOIL(5)	20.04 ± 0.08	**75.18** ± 0.21	36.65 ± 0.11	**71.52** ± 0.27	20.02 ± 0.05	**45.91** ± 0.28	29.04 ± 0.18	**47.02** ± 0.25

I Gradient norm

We calculated the norm of gradients caused by an inner loop according to the algorithm. Although the norm of gradients on the head of BOIL is not really zero, we marked 0 because the learning rate on the head is zero. The norm of gradients of biases is negligible (about 10^{-8}), and thus it was omitted. MAML/ANIL has an extremely small norm or no norms on all convolutional layers. It implies that representations little or no change. On the other hand, BOIL has a large norm on the conv4 layer. It implies that representations change significantly. In addition, from the analysis of cosine similarity and CKA, we mentioned that BOIL enjoys *representation reuse* in a low- and mid-level of body. Nevertheless, Figure 17 shows that the amount of *representation change* in a low- and mid-level in BOIL is larger than that in MAML/ANIL.

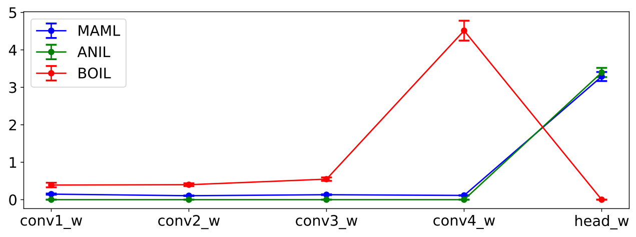


Figure 17: Gradient norm.

J Additional considerations of the head of BOIL

We additionally discuss what the ideal meta-initialization is. Because the few-shot classification tasks are constructed with sampled classes each time, every task consists of different classes. Since the class indices are randomly assigned at the beginning of each task learning, the meta-initialized parameters cannot contain any prior information on the class indices. For instance, it is not allowed that the meta-initialized parameters encode class similarities between class i and class j. Any biased initial guess could hinder the task learning. The meta-initialized parameters should be in-between (local) optimal points of tasks as depicted in Figure 18 so that the network can adapt to each task with few task-specific updates.[6]

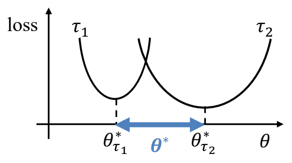
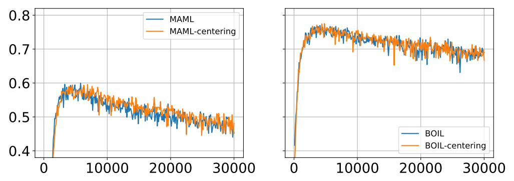
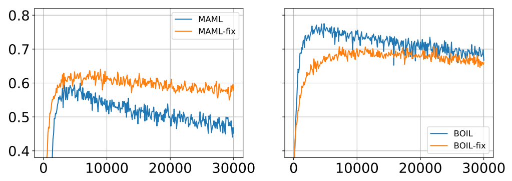


Figure 18: Ideal meta-initialization.



(a) Comparison with *centering* algorithm.



(b) Comparison with *fix* algorithm.

Figure 19: Valid accuracy curves of (a) centering algorithm and (b) fix algorithm on Cars.

When the head parameters $\theta_h = [\theta_{h,1}, ..., \theta_{h,n}]^\top \in \mathbb{R}^{n \times d}$ have orthonormal rows (i.e.,$\|\theta_{h,i}\|_2 = 1$ for all i and $\theta_{h,i}^\top \theta_{h,j} = 0$ for all $i \neq j$), the meta-initialized model can have the unbiased classifier. Here, $a^\top$ denotes the transpose of a and $\| \cdot \|_2$ denotes the Euclidean norm. With the orthonormal rows, therefore, each logit value $\theta_{h,j}{}^\top f_{\theta_b}(x)$ can be controlled independently of other logit values. Recall that the softmax probability p_j for class j of sample x is computed as follows:

$$p_j(x) = \frac{e^{\theta_{h,j}{}^\top f_{\theta_b}(x)}}{\sum_{i=1}^{n} e^{\theta_{h,i}{}^\top f_{\theta_b}(x)}} = \frac{1}{\sum_{i=1}^{n} e^{(\theta_{h,i} - \theta_{h,j})^\top f_{\theta_b}(x)}}. \tag{4}$$

[6]The similar consideration is discussed in (Drumond et al., 2020).

In Equation 4, indeed, the softmax probability only depends on the differences of the rows of the head parameters $\theta_{h,i} - \theta_{h,j}$. Adding a vector to all the rows (i.e., $\theta_{h,i} \leftarrow \theta_{h,i} + c$ for all i) does not change the softmax vector. So, we can expect the same nice meta-initialized model, when a parallel shift of the rows of the head parameters can make orthonormal rows. To support this experimentally, we design the *centering* algorithm that operates a parallel shift of θ_h by subtracting the average of the row vectors of θ_h after every outer update on both MAML and BOIL, i.e., $[\theta_{h,1} - \bar{\theta_h}, ..., \theta_{h,n} - \bar{\theta_h}]^\top$ where $\bar{\theta_h} = \frac{1}{n}\sum_{i=1}^{n}\theta_{h,i}$. Figure 19a shows that this parallel shift operations does not affect the performance of two algorithms on Cars.

Next, we investigate the cosine similarity between ${\theta_{h,i}}^\top - {\theta_{h,k}}^\top$ and ${\theta_{h,j}}^\top - {\theta_{h,k}}^\top$ for all different i, j, and fixed k. From the training procedures of MAML and BOIL, it is observed that the average of cosine similarities between the two gaps keeps near 0.5 during meta-training (Figure 20). Note that 0.5 is the cosine similarity between ${\theta_{h,i}}^\top - {\theta_{h,k}}^\top$ and ${\theta_{h,j}}^\top - {\theta_{h,k}}^\top$ when ${\theta_{h,i}}^\top$, ${\theta_{h,j}}^\top$, and ${\theta_{h,k}}^\top$ are orthonormal. From the results, we evidence that the orthonormality of θ_h is important for the meta-initialization and meta learning algorithms naturally keep the orthonormality.

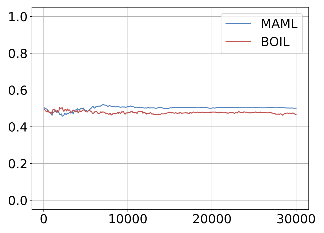


Figure 20: Average of cosine similarities between gaps.

From the above observation, we design the *fix* algorithm that fixes θ_h to be orthonormal for the meta-initialized model. Namely, MAML-fix updates θ_h in inner loops only, and BOIL-fix does not update θ_h. The *fix* algorithm can be easily implemented by initializing θ_h to be orthonormal through the Gram-Schmidt method from a random matrix and setting the learning rate for the head of the model during the outer loop to zero.

Figure 19b depicts the valid accuracy curves of the *fix* algorithm on Cars. The experiments substantiate that orthonormal rows of θ_h are important and that BOIL improves the performance. (1) Comparing MAML to MAML-fix (the left panel of Figure 19b), MAML-fix outperforms MAML. It means that the outer loop calculated through the task-specific head following MAML is detrimental because the outer loop adds unnecessary task-specific information to the model. (2) Comparing vanilla models to fix models (both panels of Figure 19b), a fixed meta-initialized head with orthonormality is less over-fitted. (3) Comparing BOIL to BOIL-fix (the right panel of Figure 19b), although BOIL-fix can achieve almost the same performance with BOIL with sufficient iterations, BOIL converges faster to a better local optimum. This is because θ_h is trained so that the inner loop can easily adapt $f_{\theta_b}(x)$ to each class.

K Representation change in ResNet-12

Figure 21 shows the CKA of ResNet according to the algorithm. Like the 4conv network, MAML/ANIL algorithms change the values only in the logit space, i.e., the space after head, regardless of the last skip connection. However, the BOIL algorithm changes the values in the representation spaces. By disconnecting the last skip connection, *representation change* is concentrated on the high-level representation space, i.e., the CKA of BOIL w/o LSC is smaller than that of BOIL w/ LSC after block4. Table 14 shows empirical results of ResNet-12.

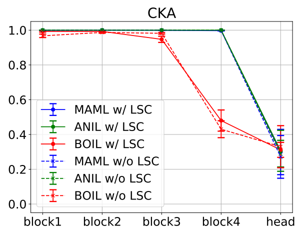
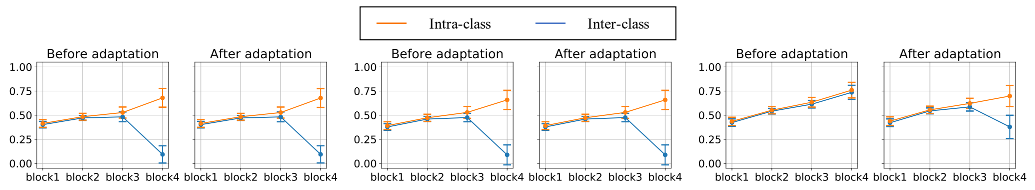


Figure 21: CKA of ResNet-12 on miniImageNet.

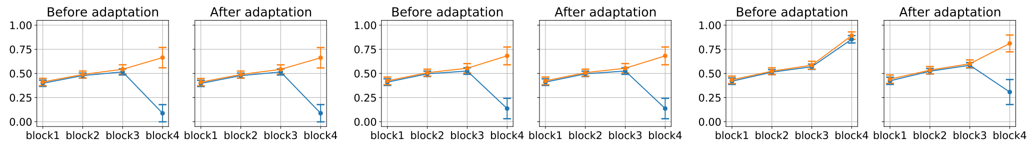
Table 14: 5-Way 5-Shot test accuracy (%) of ResNet-12 meta-trained on miniImageNet according to the head's existence before/after an adaptation.

meta-train	miniImageNet							
meta-test	miniImageNet				Cars			
head	w/ head		w/o head (NIL-testing)		w/ head		w/o head (NIL-testing)	
adaptation	before	after	before	after	before	after	before	after
MAML w/ LSC	20.02 ± 0.21	68.51 ± 0.39	**70.44** ± 0.30	70.37 ± 0.32	20.06 ± 0.07	43.46 ± 0.15	46.08 ± 0.22	46.05 ± 0.19
MAML w/o LSC	20.04 ± 0.36	67.87 ± 0.22	69.35 ± 0.15	69.28 ± 0.14	19.98 ± 0.16	41.40 ± 0.11	43.55 ± 0.17	43.56 ± 0.16
ANIL w/ LSC	19.85 ± 0.19	68.54 ± 0.34	70.31 ± 0.34	70.31 ± 0.34	19.99 ± 0.15	45.13 ± 0.15	**47.16** ± 0.20	47.16 ± 0.20
ANIL w/o LSC	19.97 ± 0.21	67.20 ± 0.13	68.47 ± 0.21	68.47 ± 0.21	20.03 ± 0.23	43.46 ± 0.18	45.16 ± 0.11	45.16 ± 0.11
BOIL w/ LSC	20.01 ± 0.18	70.50 ± 0.28	44.65 ± 0.49	70.34 ± 0.31	20.02 ± 0.08	49.69 ± 0.17	38.73 ± 0.16	49.54 ± 0.23
BOIL w/o LSC	20.00 ± 0.00	**71.30** ± 0.28	40.04 ± 0.33	**71.18** ± 0.29	20.00 ± 0.00	**49.71** ± 0.28	32.53 ± 0.29	**51.41** ± 0.32

Furthermore, we identified *representation change* of ResNet-12 meta-trained on Cars in BOIL.



(a) MAML w/ LSC. (b) ANIL w/ LSC. (c) BOIL w/ LSC.



(d) MAML w/o LSC. (e) ANIL w/o LSC. (f) BOIL w/o LSC.

Figure 22: Cosine similarity of ResNet-12 on Cars.

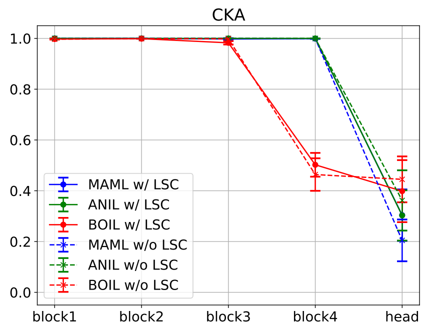


Figure 23: CKA of ResNet-12 on Cars.

Table 15: 5-Way 5-Shot test accuracy (%) of ResNet-12 meta-trained on Cars according to the head's existence before/after an adaptation.

meta-train	Cars							
meta-test	Cars				CUB			
head	w/ head		w/o head (NIL-testing)		w/ head		w/o head (NIL-testing)	
adaptation	before	after	before	after	before	after	before	after
MAML w/ LSC	20.11 ± 0.22	75.49 ± 0.20	77.01 ± 0.14	76.76 ± 0.17	20.13 ± 0.10	35.87 ± 0.19	36.57 ± 0.20	36.62 ± 0.17
MAML w/o LSC	20.04 ± 0.08	73.63 ± 0.26	75.81 ± 0.20	75.61 ± 0.25	20.03 ± 0.14	34.77 ± 0.26	36.26 ± 0.21	36.04 ± 0.27
ANIL w/ LSC	20.06 ± 0.35	79.45 ± 0.23	**80.88** ± 0.19	80.88 ± 0.19	20.12 ± 0.14	35.09 ± 0.19	35.80 ± 0.16	35.80 ± 0.16
ANIL w/o LSC	20.20 ± 0.33	75.32 ± 0.15	76.90 ± 0.16	76.90 ± 0.16	20.01 ± 0.18	36.06 ± 0.14	**37.34** ± 0.13	37.34 ± 0.13
BOIL w/ LSC	20.00 ± 0.00	80.98 ± 0.14	40.54 ± 0.11	81.11 ± 0.24	20.00 ± 0.00	43.84 ± 0.21	37.27 ± 0.12	**45.44** ± 0.23
BOIL w/o LSC	20.00 ± 0.00	**83.99** ± 0.20	50.42 ± 0.23	**83.60** ± 0.18	20.00 ± 0.00	**44.23** ± 0.18	31.69 ± 0.14	44.19 ± 0.17

L OUTPUT LAYER OF MAML/ANIL AND PENULTIMATE LAYER OF BOIL

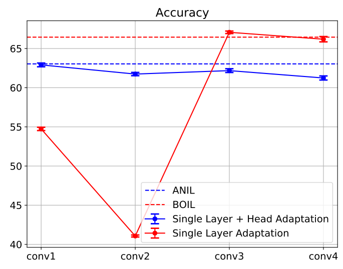
Table 16: Test accuracy (%) of NIL-testing on conv3 and conv4 of 4conv network before/after an adaptation.

meta-train	miniImageNet							
meta-test	miniImageNet				Cars			
head	NIL-testing on conv3		NIL-testing on conv4		NIL-testing on conv3		NIL-testing on conv4	
adaptation	before	after	before	after	before	after	before	after
MAML(1)	32.56 ± 0.14	32.73 ± 0.14	48.28 ± 0.20	47.87 ± 0.14	30.86 ± 0.09	31.03 ± 0.08	34.47 ± 0.19	34.36 ± 0.16
ANIL(1)	**34.34** ± 0.10	**34.34** ± 0.10	**48.86** ± 0.12	**48.86** ± 0.12	**31.09** ± 0.08	**31.09** ± 0.08	**35.48** ± 0.24	**35.48** ± 0.24
BOIL(1)	30.65 ± 0.07	31.09 ± 0.12	24.07 ± 0.19	46.73 ± 0.17	27.53 ± 0.16	27.80 ± 0.15	23.30 ± 0.15	34.07 ± 0.32
MAML(5)	51.03 ± 0.08	53.29 ± 0.10	64.61 ± 0.39	64.47 ± 0.39	44.54 ± 0.21	45.19 ± 0.21	47.66 ± 0.28	47.53 ± 0.28
ANIL(5)	**55.55** ± 0.14	**55.55** ± 0.14	**66.11** ± 0.51	**66.11** ± 0.51	**45.81** ± 0.14	**45.81** ± 0.14	**49.62** ± 0.20	49.62 ± 0.20
BOIL(5)	49.42 ± 0.12	50.12 ± 0.12	32.03 ± 0.16	64.61 ± 0.27	43.93 ± 0.32	44.52 ± 0.18	30.33 ± 0.18	**50.40** ± 0.30

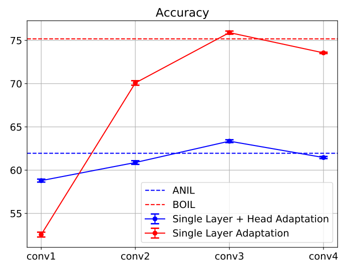
To investigate the role of the penultimate layer (i.e., the last convolutional layer) of BOIL, we evaluated MAML/ANIL and BOIL through NIL-testing on conv3 in the same way with NIL-testing on conv4 (Table 4), except for the position of representations. Table 16 shows that the input of the penultimate layer (i.e., features after conv3) cannot be simply classified (i.e., the desired performance cannot be achieved), and these representation capacities are similar for all MAML, ANIL, and BOIL. Therefore, It is thought that MAML/ANIL and BOIL acts similarly until conv3 and BOIL is not simply the shifted version of MAML/ANIL.

Furthermore, the input of the output layer before adaptation (i.e., features after the penultimate layer) is enough to achieve the desired performance in MAML/ANIL in advance. From this result, we believe the head layer's role of MAML/ANIL is to draw a simple decision boundary for the high-level features represented by the output of the last convolutional layer. However, the penultimate layer of BOIL acts as a non-linear transformation so that the fixed output head layer can effectively conduct a classifier role.

M ABLATION STUDY ON THE LEARNING LAYER



(a) miniImageNet.



(b) Cars.

Figure 24: Test accuracy according to the learning layer in the body.

In this section, we investigate whether training any representation layer during inner updates is better than training the output layer during inner updates by learning only one convolutional layer. Figure 24 shows the test accuracy according to a single learning layer in the body. For instance, conv1 plotted in red color indicates that the algorithm updates only the conv1 layer during inner updates. In contrast, conv1 plotted in blue color is the case that the algorithm updates the conv1 layer and the head layer during inner updates.

On both miniImageNet and Cars, it is observed that training a higher-level representation layer (conv3, conv4) without updating the head (red line) performs better than training any single conv layer with the output layer (blue line). We also observe that learning only a lower-level representation layer (conv1, conv2) can significantly decrease accuracy. The results reassure that *representation change* in higher-level layers of the body boosts the performance discussed by the layer-wise analyses in section 4. However, training only a lower-level layer behaves badly since lower-level layers retain general representations (a related discussion is in Section 4.1, e.g., the third key observation).

Table 17: Test accuracy (%) according to the learning layer(s). Standard deviation is omitted.

Learning layer	1 layer					2 layers				3 layers			4 layers		all
conv1	✓					✓				✓			✓		✓
conv2		✓				✓	✓			✓	✓		✓	✓	✓
conv3			✓				✓	✓		✓	✓	✓	✓	✓	✓
conv4				✓				✓	✓		✓	✓	✓	✓	✓
head					✓				✓			✓		✓	✓
Algorithm					ANIL								BOIL		MAML
miniImageNet	54.74	41.07	**67.07**	66.19	63.04	60.79	**67.44**	65.86	61.24	67.18	**67.40**	62.99	**66.45**	61.11	61.75
Cars	52.55	70.08	**75.90**	73.56	61.95	68.70	**78.21**	72.99	61.46	73.59	**74.80**	64.58	**75.18**	63.97	53.23

We expand this ablation study to training multiple consecutive layers with and without the head. The results are reported in Table 17. We consistently observe that learning with the head is far from the best accuracy. All the combinations having nice performances do not train the head in the inner loop update. We also find several settings skipping the lower-level layers in the inner loop that perform slightly better than BOIL. We believe each neural network architecture and data set pair has its own best layer combination. When it is allowed to search for the best combination using huge computing power, we can further improve BOIL. However, the most important design policy is that the inner loop update should freeze the head and encourage to learn higher-level representation features but to reuse lower-level representation features. BOIL follows the design rule by simply freezing the head in the inner loop that is already almost the best approach in most cases.

N Ablation study on the learning rate of the head

Learning rate	miniImageNet	Cars
0.00 (BOIL)	66.45 ± 0.37	75.18 ± 0.21
0.05	38.81 ± 0.21	68.67 ± 0.21
0.10	49.49 ± 0.16	68.86 ± 0.30
0.50 (MAML in ours)	61.75 ± 0.42	53.23 ± 0.26

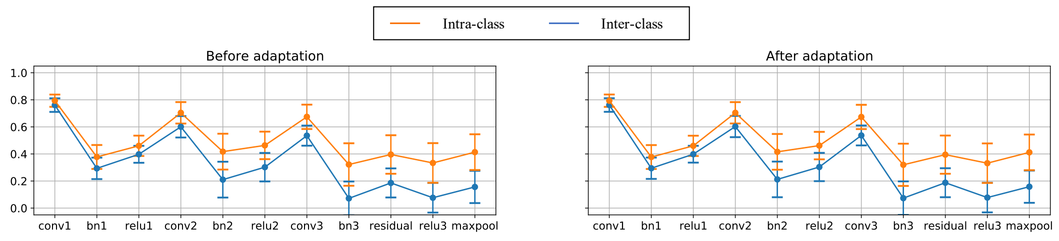
Table 18: Test accuracy according to the learning rate of the head.

In this section, we control the inner loop update learning rate of the head to verify the effect of training the head to the performance. The results are depicted in Table 18. When the learning rate is 0 (BOIL), we can observe the best performance. However, the accuracy rapidly decreases as the learning rate of the head grows. Even with $1/10\times$ head learning rate compared to other layers, the test accuracies are significantly degraded. Based on this result, we think the freezing head is crucial.

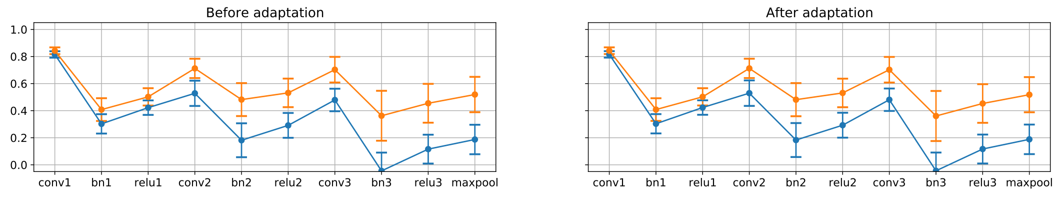
O Representation layer change in the last block of ResNet-12

In this section, we explore the *representation layer reuse* and *representation layer change* in the last block of a deeper architecture network. Figure 25 and Figure 26 show the cosine similarities between representations after all layers in the last block (i.e., block4) on miniImageNet and Cars. In the case of MAML/ANIL, there is little or no *representation layer change* in all layers of the last block. In contrast, in the case of BOIL, the gap between intra-class similarity and inter-class similarity is enlarged through adaptation in some or all layers of the last block, which indicates that *representation layer reuse* in low-level layers of the block and *representation layer change* in high-level layers of the block are mixed in the last block.

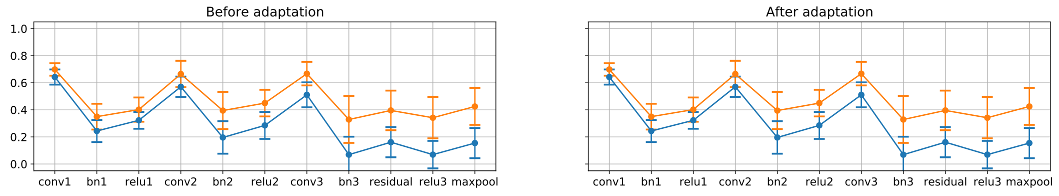
Furthermore, it is observed that representation at high-level layers in the last block changes more when the last skip connection does not exist (e.g., Figure 25f and Figure 26f) than when the last skip connection exists (e.g., Figure 25e and Figure 26e). This result confirms that the disconnection trick strengthens *representation layer change* at high-level layers by not directly propagating general representations from prior blocks.



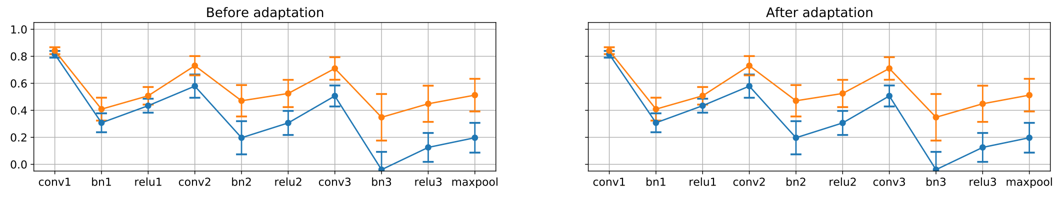
(a) MAML w/ LSC.



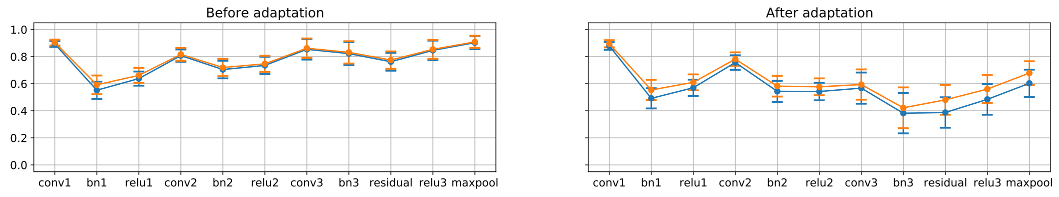
(b) MAML w/o LSC.



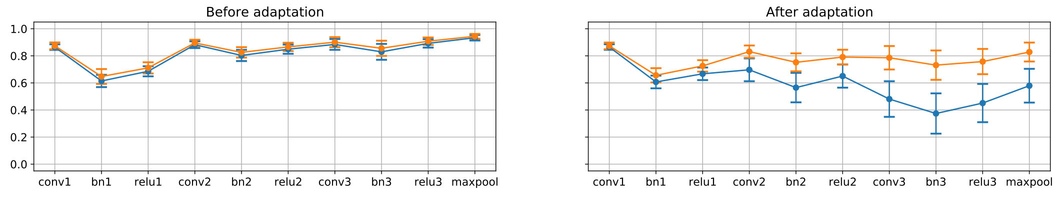
(c) ANIL w/ LSC.



(d) ANIL w/o LSC.

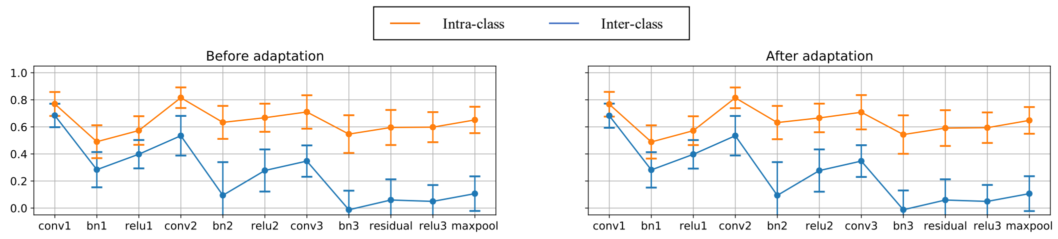


(e) BOIL w/ LSC.

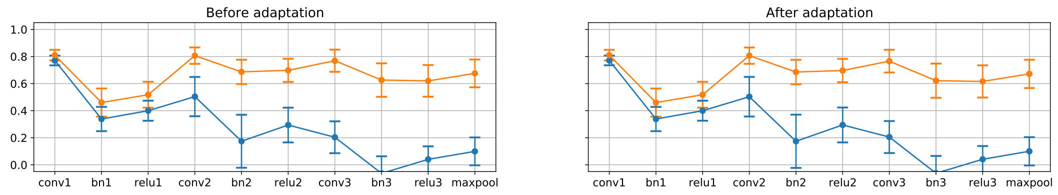


(f) BOIL w/o LSC.

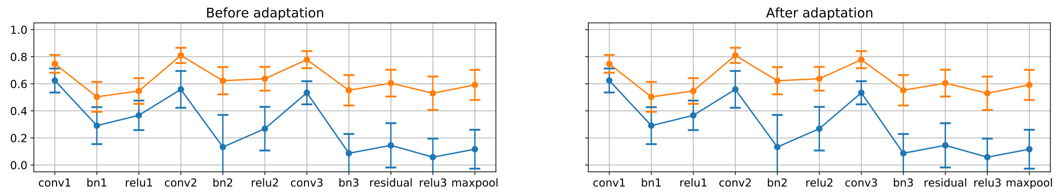
Figure 25: Cosine similarity in block4 (the last block) of ResNet-12 on miniImagenet.



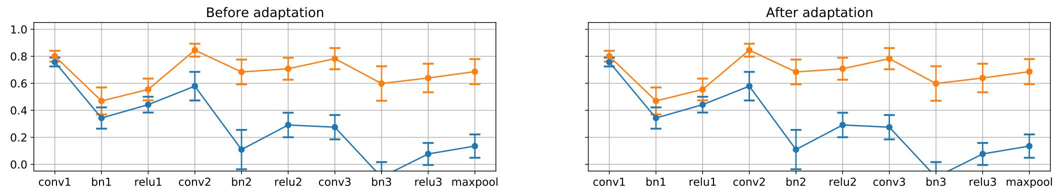
(a) MAML w/ LSC.



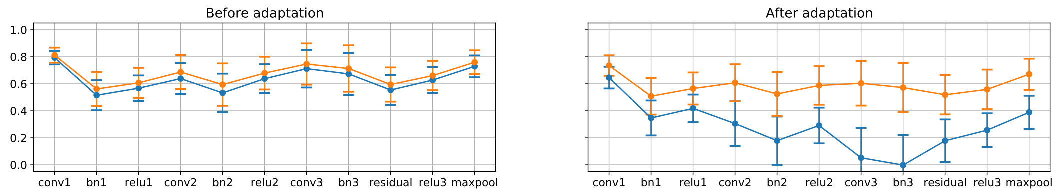
(b) MAML w/o LSC.



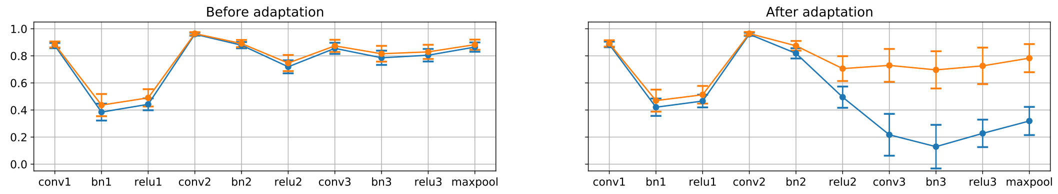
(c) ANIL w/ LSC.



(d) ANIL w/o LSC.



(e) BOIL w/ LSC.



(f) BOIL w/o LSC.

Figure 26: Cosine similarity in block4 (the last block) of ResNet-12 on Cars.